Higher Secondary Course

COMPUTER APPLICATIONS

(Commerce)

CLASS - XII

Government of Kerala

DEPARTMENT OF EDUCATION

State Council of Educational Research and Training (SCERT), Kerala

2015

THE NATIONAL ANTHEM

Jana-gana-mana adhinayaka, jaya he
Bharatha-bhagya-vidhata.
Punjab-Sindh-Gujarat-Maratha
Dravida-Utkala-Banga
Vindhya-Himachala-Yamuna-Ganga
Uchchala-Jaladhi-taranga
Tava subha name jage,
Tava subha asisa mage,
Gahe tava jaya gatha.
Jana-gana-mangala-dayaka jaya he
Bharatha-bhagya-vidhata.
Jaya he, jaya he, jaya he,
Jaya jaya jaya, jaya he!

PLEDGE

India is my country. All Indians are my brothers and sisters.

I love my country, and I am proud of its rich and varied heritage. I shall always strive to be worthy of it.

I shall give my parents, teachers and all elders respect, and treat everyone with courtesy.

To my country and my people, I pledge my devotion. In their well-being and prosperity alone lies my happiness.

Prepared by:

State Council of Educational Research and Training (SCERT)
Poojappura, Thiruvananthapuram 695012, Kerala
Website : www.scertkerala.gov.in *e-mail* : scertkerala@gmail.com
Phone : 0471 - 2341883, Fax : 0471 - 2341869
Typesetting and Layout : SCERT

To be printed in quality paper - 80gsm map litho (snow-white)

Dear learners,

The syllabus of Computer Applications for the Commerce discipline at Higher Secondary Level has been revised and implemented in view of the fact that computer is used currently as a tool for various applications, especially in the field of e-Banking, e-Commerce, e-Governance, etc. Higher studies and placements, at present, greatly demand human resources with adequate knowledge in computer applications and information technology. In Class XI, the syllabus focussed on the fundamentals of computer, hardware and software components, computer network and Intenet. Emphasis is also given to put a strong foundation to develop problem solving skills and create computer programs.

The syllabus of Class XII Computer Application (Commerce) is a continuation to that of Class XI. Hence the text book, designed in accordance with the syllabus, begins with some advanced features of C++ programming language to solve complex problems. Since we are in the age of Internet and most of us are users of web applications, an introduction to develop the same is also provided. The concept of database and facilities of information retrieval are included. A chapter is dedicated to present a brief idea about Enterprise Resource Planning. An exclusive section on the new trends and issues in the field of Information Communication Technology is also incorporated. Considering the increase in the use of Internet, an awareness about Cyber laws is presented to safeguard against Cyber crimes.

I hope this book will meet all the requirements for stepping to levels of higher education and pave the way to the peak of success.

Dr S. Raveendran Nair
Director
SCERT, Kerala

Textbook Development Team

Joy John
HSST, St. Joseph's HSS,
Thiruvananthapuram

A. N. Sathian
HSST, GM HSS, Koyilandy,
Kozhikode

Mustafa Shamsul Haq K. K.
HSST, GHSS Kuthuparamba,
Kannur

A. S. Ismael
HSST, PJMS GHSS, Kandassankadavu,
Thrissur

Vinod V.
HSST, NSS HSS, Prakkulam,
Kollam

Prasanth P. M.
HSST, St. Joseph's Boys' HSS,
Kozhikode

Sunil Kariyatan
HSST, Govt. Brennen HSS,
Thalassery

Asees V.
HSST, Govt. HSS Velliyode,
Kozhikode

Rajamohan C.
HSST, Nava Mukunda HSS,
Thirunavaya, Malappuram

Najeeb P. P.
HSST, Himayathul Islam HSS,
Kozhikode

Experts

Dr Lajish V. L.
Assistant Professor, Dept. of Computer Science, University of Calicut

Madhu V. T.
Director, Computer Centre, University of Calicut

Dr Sushil Kumar R.
Associate Professor, Dept. of English, D. B. College, Sasthamcotta

Vinayakumaran Nair N.
Assistant Commandant, Hi-Tech Cell, Police Head Quarters, Trivandrum

Dr Kabeer V.
Asst. Prof & Head, Dept. of Computer Science, Farook College, Kozhikode

Dr Madhu S. Nair
Assistant Professor, Dept. of Computer Science, University of Kerala

Dr Binu P. Chacko
Associate Professor, Dept. of Computer Science, Prajyoti Niketan College, Pudukad, Thrissur

Dr Deepa L. C.
Assistant Professor, Dept. of English, Govt. Women's College, Trivandrum

Maheswaran Nair V.
Sub Divisional Engineer, Regional Telecom Training Centre, Kaimanom, Tvpm.

Artists

Sudheer Y. **Vineeth V.**

Academic Co-ordinator

Dr Meena S.
Research Officer, SCERT

Contents

kkkk

Icons used in this textbook

Let us do

Know your progress

Information box

Let us practice

Let us conclude

1 Review of C++ Programming

Significant Learning Outcomes

After the completion of this chapter the learner

- uses input statements in programs to enter data into the computer.
- uses output statements in programs to display various forms of output.
- applies various forms of if statements to make decisions while solving problems.
- compares else if ladder and switch statement.
- distinguishes different looping statements of C++.
- selects appropriate loop in programs for solving problems.
- uses the concept of nested loop in problem solving and predicts the output.
- identifies the effect of break and continue statements in loops by explaining their effect on the program flow.

The basic concepts of C++ language were discussed in Class XI. A good understanding of these concepts is very essential to attain the significant learning outcomes envisaged in the following chapters. This chapter is a quick tour to refresh the concepts and skills you acquired in C++ language in Class XI. Each concept will be presented with necessary details only. The most important aspects like selection statements and looping statements are explained with the help of programs. Some advanced features like nested loops and the effect of `break` and `continue` statements in loops are also introduced in this chapter.

Since we use GNU Compiler Collection (GCC) with Geany IDE for developing C++ programs, we should be aware of the structure of program, format of specifying the header file, size of the data types, etc.

1.1 Basics of C++

C++ being a programming language, we started by learning its character set followed by tokens, expressions and statements. We also discussed data types and their modifiers. While constructing expressions, we identified the need of data type conversions. Table 1.1 provides a brief idea of these elements.

An overview of C++	
Character set	Fundamental unit of C++ language. Classified into letters (a - z, A - Z), digits (0 - 9), special characters (# , ; : > { + etc.), white spaces (space bar, tab, new line) and some other characters whose ASCII code fall in the range from 0 to 255.
Tokens	Basic building blocks of C++ programs. Constituted by one or more characters. Classified into keywords, identifiers, literals, punctuators and operators.
Keywords	Reserved words that convey specific meaning to the language compiler.
Identifiers	User-defined words to identify memory locations, statements, functions, data types, etc. Certain rules are to be followed to ensure the validity of identifiers. Identifiers include variables, labels, function names, etc.
Literals	Tokens that do not change their value during the program run. They are also known as constants. Classified into integer constants, floating point constants, character constants and string constants. Integer constant is constituted by digits only with an optional plus (+) or minus (-) sign as the first character. Floating point constant is expressed in fractional form and exponential form. Character constant is a single character of C++ enclosed within single quotes. There are some special character constants, called escape sequences. They represent some non-printable or non-graphic characters like new line ('\n'), tab space ('\t') and punctuation marks like single quote ('\' '), double quotes ('\"'), question mark ('\?') etc. String constant is a sequence of characters enclosed by a pair of double quotes.
Operators	Symbols that tell the compiler about some operations. Each of them actually triggers a specific operation. Based on the number of operands (data on which operation is carried out), operators are classified into unary, binary and ternary. Another classification is based on the type of operation

	performed. They consist of arithmetic (+, -, *, /, %), relational (<, <=, >, >=, ==, !=) and logical (&&, ||, !) operators. These operators give some value as the result of the operation. There are some special operators named *get from* (>>) for input, *put to* (<<) for output and assignment (=) for setting a value in a variable. Another category of operators implicitly performs an assignment operation after an arithmetic operation. They include increment (++), decrement (--) and arithmetic assignment (+=, -=, *=, /=, %=) operators.
Punctuators	Special characters like comma (,), semi colon (;), hash (#), braces ({}) etc. used for the perfection of syntax of various constructs of the language used in programs. They have semantic and syntactic meaning to the compiler.
Data types	These are means to identify the type of data and associated operations handling these data. Data types are classified into fundamental and user-defined data types. Fundamental data types represent atomic values and they include `int`, `char`, `float`, `double` and `void`. Each data type excluding void has its own size and range for the values they represent. Data type void represents an empty set of data and hence its size is zero.
Type modifiers	The keyword `signed`, `unsigned`, `short` and `long` are the type modifiers. They are used with data types to modify the size of memory space and range of data supported by the basic data types.
Expressions	Expressions are constituted by operators and required operands to perform an operation. Based on the operators used, they are classified into arithmetic expressions, relational expressions and logical expressions. Arithmetic expression is divided into integer expression and real expression. Integer expression consists only of integer data as operands and it returns integer value. In real expressions, the operands and the return-value are floating point data.

An overview of C++	
Type conversion	Relational expressions consist of numeric or character data as operands and they return True or False as outputs. Logical expressions use relational expressions as operands in practice and return True or False value as results. When different types of operands are involved in an arithmetic expression, type conversion takes place. It is the process of converting the current data type of a value into another type. It may be implicitly and explicitly converted. In implicit type conversion, complier is responsible for the conversion. It always converts a lower type into higher one and hence it is also known as type promotion. In explicit conversion, user is responsible for the conversion. Here the user determines the destination data type and hence it is known as type casting.

Table 1.1: Basic elements of C++ language

1.1.1 Various statements in a C++ program

Usually every program begins with pre-processor directives. We use `#include` statement, the pre-processor directive to attach a header file to provide information about predefined identifiers and functions used in the program. The pre-processor directive statements are followed by `using namespace` statement. Usually we use the predefined namespace `std` to specify the scope of identifiers `cin` and `cout`. Then the `main()` function appears. It is an essential function in a C++ program, where the program execution starts and ends. It consists of declaration statements and a set of executable statements required for solving the problem. Let us have a close look at these statements.

Declaration statement

Variables are identifiers of memory locations and are used in programs to refer to data. They should be declared prior to their use in the program and data types are required for this. The following statements are examples for variable declaration:

```
int n, sum;
float rad, area;
signed int a,b,c;
```

Values can be provided to the variables along with the declaration as shown below:

```
int n=10;
```

This kind of statement is known as variable initialisation statement. The value assigned to the variable can be replaced by some other value later in the program. But the following statement does not allow changing the value of the variable.

```
const int n=10;
```

Here the variable initialisation begins with the access modifier `const`, a keyword that restricts a change in the value of the variable.

Input statement

C++ provides the operator **>>**, called extraction operator or get from operator. It is a binary operator and hence it requires two operands. The first operand is the pre-defined identifier `cin` that identifies keyboard as input object. The second operand is strictly a variable. We can use more than one variable in the same statement to receive more than one input. The following are valid examples:

```
cin>>rad;
cin>>a>>b>>c;
```

Output statement

To perform output operation, C++ gives the operator **<<**, called insertion operator or put to operator. It is also a binary operator. The first operand is the pre-defined identifier `cout` that identifies monitor as the output object. The second operand may be a constant, a variable or an expression. The following are some examples for output statements:

```
cout<< "hello";
cout<< area;
cout<< 25;
cout<< a+b+c;
cout<< "Sum of " << n << "numbers = " << sum;
```

Assignment statement

A specific data is stored in memory locations (variables) using assignment operator (**=**). The statement that contains = is known as assignment statement. It is also a binary operator and the operand to the left of = should be a variable. The operand after = may be a numeric constant, a variable or an expression of numeric type. The following assignment statements are valid:

```
n = 253;
area = 3.14*rad*rad;
a = b = c;
```

There are special assignment operators, called arithmetic assignment operators. They are **+=**, **–=**, ***=**, **/=** and **%=**. These are all binary operators and the operand to left of these operators should be variables. Their operations may be illustrated as follows:

```
n+=2;          // It is equivalent to n=n+2;
a*=b;          // It is equivalent to a=a*b;
sum-=n%10;     // It is equivalent to sum=sum-n%10;
```

The operators **++** and **––** are special operators of C++, which are, in a way, similar to assignment statements. These are unary operators and the operand should be a variable. The following statements illustrate the operations associated with them:

```
n++;           // It is equivalent to n=n+1;
a--;           // It is equivalent to a=a-1;
```

There are two versions for these operators: postfix form and prefix form. `a++;` and `a--;` are the postfix form of increment and decrement operators respectively. `++a;` and `--a;` are the prefix form. Whatever be the form, ++ operator adds 1 to the content of the operand variable and the result is stored in it.

The two versions differ when used with an assignment or output statement. Assume that **a** is an integer variable with value 5 and **b** is another integer variable. After the execution of the statement: `b=a++;` the value of `b` will be 5 and that of `a` will be 6. That is, `b=a++;` is equivalent to the statement sequence: `b=a;  a=a+1;`. So this method of incrementing is known as use and change method.

But the statement, `b=++a;` is equivalent to the statement sequence: `a=a+1; b=a;`. So the values of both `a` and `b` will be 6. This method of incrementing is known as change and use method.

Similarly, the statement `cout<<a--;` will display 5, but the value of `a` will be 4. It is equivalent to the statement sequence `cout<<a;  a=a-1;`. But the statement `cout<<--a;` is equivalent to `a=a-1;  cout<<a;`.

The input, output and assignment operators (>>, << and =) may appear more than once in the respective statements. It is known as cascading. The following are examples in each category for cascading of input, output and assignment operators respectively.

```
cin >> a >> b >> c;
cout << "Sum of " << n << "numbers = " << sum;
a = b = c;
```

1.1.2 Structure of a C++ program

Program 1.1 shows the basic structure of a C++ program. It accepts the length and breadth of a rectangle and computes its area and perimeter.

Program 1.1: To find the area and perimeter of a rectangle

```
#include <iostream>
using namespace std;
int main()
{
    float length, breadth, peri, area;
    cout << "Enter the length and breadth of rectangle: ";
    cin >> length >> breadth;
    peri = 2*(length + breadth);
    area = length * breadth;
    cout << "Perimeter = " << peri << endl;
    cout << "Area = " << area << endl;
    return 0;
}
```

Program 1.1 uses the header file `iostream`, since the identifiers `cin` and cout are used. The second line is also essential to use `cin` and `cout` independently. The `using namespace` statement uses `std` to make `cin` and `cout` available in `main()`. In GCC, the function name `main()` is preceded by the data type `int`. Variables are declared using `float` data type. Cascading of input operator and output operators are utilised. Formulae are used in the assignment statements to solve the problem. The `endl` is used instead of `'\n'` to print a new line after each of the results. Figure 1.1(a) shows the screenshot of Program 1.1 in Geany IDE and Figure 1.1(b) shows the output on execution.

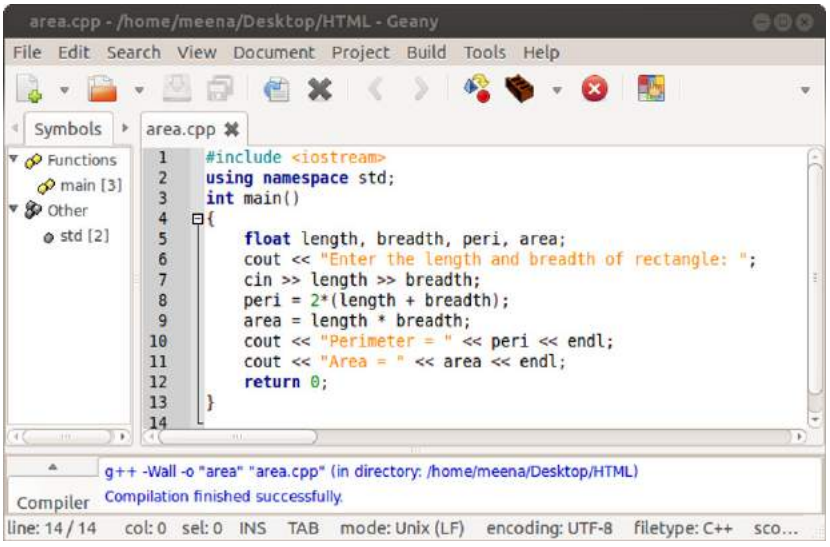


Fig. 1.1(a): Program 1.1.in Geany IDE

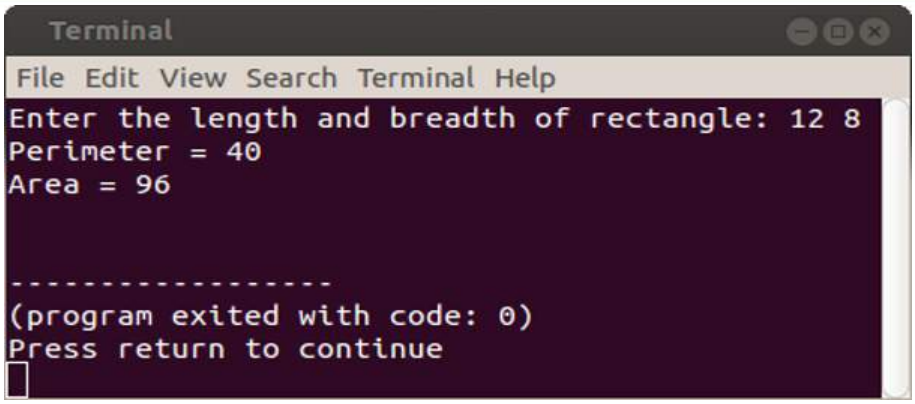


Fig. 1.1(b): Output of Program 1.1 in the terminal window of Geany IDE

This program follows a sequential structure. That is, all statements in the program are executed in a sequential fashion.

While writing a C++ program, we use the statement "**`using namespace std;`**". Why?

A program cannot have the same name for more than one identifier (variables or functions) in the same scope. In our home two or more persons (or even living beings) will not have the same name. If there are, it will surely make conflicts in the identity within the home. So, within the scope of our home, a name should be unique. But our neighbouring home may have a person (or any living being) with the same name as that of one of us. It will not make any confusion of identity within the respective scopes. But an outsider cannot access a particular person by simply using the name; but the house name is also to be mentioned.

The concept of namespace is similar to a house name. Different identifiers are associated to a particular namespace. It is actually a group name in which each item is unique in its name. User is allowed to create own namespaces for variables and functions. The keyword `using` technically tells the compiler about a namespace where it should search for the elements used in the program. In C++, `std` is an abbreviation of 'standard' and it is the standard namespace in which `cout`, `cin` and a lot of other things are defined. So, when we want to use them in a program, we need to follow the format `std::cout` and `std::cin`. This kind of explicit referencing can be avoided with the statement `using namespace std;` in the program. In such a case, the compiler searches this namespace for the elements `cin`, `cout`, `endl`, etc. So whenever the computer comes across `cin`, `cout`, `endl` or anything of that matter in the program, it will read it as `std::cout`, `std::cin` or `std::endl`.

The statement `using namespace std;` doesn't really add a function, it is the include `<iostream>` that "loads" `cin`, `cout`, `endl` and all the like.

Know your progress

1. What is meant by token in C++?
2. Read the following tokens and identify the type of token to which each of these belongs:

   i. number ii. 23.98 iii. "\0" iv. cin v. ++
   vi. void vii. '\\' viii. ; ix. = x. a
3. What is the role of #include statement in C++ program?
4. What is wrong in the statement: cin>>25; ?
5. List the type modifiers of C++.

1.2 Control statements

The sequential flow of execution in a program may need to be altered while solving problems. It may be in the form of selection, skipping or repeated execution of one or more statements. Usually this decision will be based on some condition(s). C++ provides statements to facilitate this requirement with the help of control transfer statements. These are classified into two: (i) decision making/selection statements and (ii) iteration statements. Let us see how these statements help problem solving.

1.2.1 Selection statements

C++ provides two statements to select a task from the alternatives based on a condition. They are **`if`** statement and **`switch`** statements. `if` statement has different versions: simple if, `if - else` and `else if` ladder. The following program illustrates the mode of execution of these statements. This program accepts the CE scores of Computer Applications in three terms and finds the highest as the final CE score.

Program 1.2: To find the best CE score from the three given scores

```
#include <iostream>
using namespace std;
int main()
{
    short int ce1, ce2, ce3, final_ce;
    cout<<"Enter three CE scores: ";
    cin>>ce1>>ce2>>ce3;
    if (ce1>ce2)
      final_ce=ce1;//Will be executed, if the condition is true
    else
      final_ce=ce2;//Will be executed, if the condition is false
```

```
    if (ce3>final_ce) final_ce=c3;//No else block for this if
    cout<<"Final CE Score is "<<final_ce;
    return 0;
}
```

Program 1.2 uses `short int` data type for the variables. In GCC, `int` takes 4 bytes of memory, whereas `short` takes only 2 bytes. In this problem we need only the values within the range of `short`. The program also uses `if-else` statement and a simple `if` statement. Initially, the test expression `ce1>ce2` will be evaluated. If it evaluates to true, the value of `ce1` will be assigned to `final_ce`, otherwise that of `ce2` will be assigned. After that, the third score stored in `ce3` is compared with the content of `final_ce`. If the score in `ce3` is higher, it will be stored in `final_ce`, otherwise `final_ce` will not be changed.

We learned that `else if` ladder is a multi-branching statement. Program 1.3 illustrates the working of `else if` ladder. It accepts a character and prints whether it is an alphabet, digit or any other character.

Program 1.3: To check whether the character is uppercase letter, lowercase letter, digit or other characters

```
#include <iostream>
using namespace std;
int main()
{
    char ch;
    cout<<"Enter a character: ";
    cin>>ch;
    if (ch>='A' && ch<='Z')
        cout<<"Uppercase letter";
            else if (ch>='a' && ch<='z')
                cout<<"Lowercase letter";
                    else if (ch>='0' && ch<='9')
                        cout<<"Digit";
                            else
                                cout<<"Other character";
    return 0;
}
```

Program 1.3 uses `else if` ladder or `else if` staircase to select a statement from four different alternatives. Three conditions are provided for selecting one among the three actions. The fourth alternative will be selected when all the three conditions are evaluated to false.

In some cases, where integer equality conditions are used for decision making, `switch` statement can replace `else if` ladder. Since `char` type data are treated as numeric, they are also used in equality checking. Program 1.4 illustrates this concept. This program accepts any one of the four letters a, b, c and d. If 'a' is the input, the word 'Abacus' will be displayed. Similarly, 'Binary' for 'b', 'Computer' for 'c' and 'Debugging' for 'd' will be displayed. For the given problem, actually the `default` statement is not required. In that case, there will not be any response from the program for an input other than the four specified characters. So, to make the program user-friendly, `default` case is mentioned in this program.

Program 1.4: To display a word for a given character

```
#include <iostream>
using namespace std;
int main()
{
    char ch;
    cout<<"Enter a, b, c or d: ";
    cin>>ch;
    switch(ch)
    {
      case 'a': cout<<"Abacus";
                break;
      case 'b': cout<<"Binary";
                break;
      case 'c': cout<<"Computer";
                break;
      case 'd': cout<<"Debugging";
                break;
      default : cout<<"Invalid input!!";
    }
    return 0;
}
```

Program 1.4 also uses the concept of multi branching. Different cases are given, out of which only one will be executed. Selection will be based on the match between the value of the expression provided with switch and the constant attached with any one case. If none of the constants is matched with the value of `ch`, the `default` case will be executed.

Let us do

Replace the switch statement used in Program 1.4 with else if ladder.

In Program 1.4, if break; statements are removed what will be the output?

The else if ladder used in Program 1.3 cannot be replaced with switch. Why?

Conditional operator (? :)

It is a ternary operator of C++ and it requires three operands. It can substitute if - else statement. The `if – else` statement used in Program 1.2 can be replaced by the following:

```
final_ce = (ce1>ce2) ? ce1 : ce2;
```

If we have to find the largest among three scores, nesting of conditional operator can be used as follows:

```
final_ce = (ce1>ce2) ? ((ce1>ce3)?ce1:ce3) :
                                    ((ce2>ce3)?ce2:ce3);
```

1.2.2 Looping statements

C++ provides three looping statements: **`while`**, **`for`** and **`do-while`**. A looping statement has four components: initialisation expression, test expression, update expression and loop-body. The loop-body is the set of statements for repeated execution. The execution is continued as long as the test expression (condition) is true. The variable used in the test expression, called loop control variable, gets its initial value through the initialisation expression (or statement). Update expression changes the value of the loop control variable. Usually, it takes place after each execution of the loop-body.

Looping statements, also called iteration statements, are classified into two: entry-controlled and exit-controlled. In entry-controlled loops, test expression is evaluated before the execution of the loop-body. Program control enters the loop-body only if the condition is true. `while` and `for` are examples of entry-controlled loops. But in exit-controlled loop, condition is checked only after executing the loop-body. So it is certain that the loop-body will be executed at least once in the case of exit-controlled loop. `do-while` statement belongs to this category.

All the three expressions (initialization, test and update) are placed together in for loop. But in the case of `while` and `do-while`, initialisation expression has to be given before the loop and update expression within the loop-body. Test expression appear along with the word `while`. Let us see some programs to understand the working of these loops.

Program 1.5: To find the sum of digits of a number

```
#include <iostream>
using namespace std;
int main()
{
    int num, sum=0, dig;
    cout<<"Enter a number: ";
    cin>>num;
    while (num>0)
    {
        dig=num%10;
        sum=sum+dig;
        num=num/10;
    }
    cout<<"Sum of the digits of the input number = "<<sum;
    return 0;
}
```

In Program 1.5, `num` is the loop control variable as it is involved in the test expression. The initialisation of this variable is through an input statement. Note that the updation expression is within the loop-body. If the input is not a positive number, the body will never be executed.

Let us see another program that uses `for` statement to constitute a loop.

Program 1.6: To find the sum of the first N natural numbers

```
#include <iostream>
using namespace std;
int main()
{
    int n, sum=0;
    cout<<"Enter the limit: ";
    cin>>n;
    for(int i=1; i<=n; i++)
        sum=sum+i;
    cout<<"Sum of the first "<<n<<" natural numbers = "<<sum;
    return 0;
}
```

In Program 1.6, the variable `i` is the loop control variable and is initialised with an assignment statement. The updation is done with increment operation. The program needs a loop based on counting. The for statement is more appropriate in such situations.

As mentioned earlier, `do-while` loop ensures the execution of loop-body at least once. Its application is illustrated in the following program.

Program 1.7: To find the average height of a group of students

```
#include <iostream>
using namespace std;
int main()
{
    float hgt, sum=0, avg_hgt;
    short n=0;
    char ch;
    do
    {
        cout<<"Enter the height: ";
        cin>>hgt;
        n++;
        sum=sum+hgt;
        cout<<"Any more student (Y/N)? ";
        cin>>ch;
    }while (ch=='Y' || ch=='y');
    avg_hgt=sum/n;
    cout<<"Average Height = "<<avg_hgt;
    return 0;
}
```

The execution of the loop-body in Program 1.7 is repeated as long as user responds by inputting 'Y' or 'y'.

Let us do

Rewrite the looping segment of Program 1.5 using `for` and `do-while` statements.

Rewrite the looping segment of Program 1.6 using `while` and `do-while` statements.

Rewrite the looping segment of Program 1.7 using `for` and `while` statements.

Know your progress

1. What are the selection statements of C++?
2. What are the four components of a loop?
3. Give examples for entry-controlled loop.
4. Which control transfer statement is equivalent to the conditional operator (`?:`)?
5. All `switch` statements can be replaced by any form of `if` statement. State whether it is true or false.

1.3 Nested Loops

Placing a loop inside the body of another loop is called nesting of a loop. In a nested loop, the inner loop statement will be executed repeatedly as long as the condition of the outer loop is true. Here the loop control variables for the two loops should be different.

Let us observe how a nested loop works. Take the case of a minute-hand and second-hand of a clock. Have you noticed the working of a clock? While the minute-hand stands still at a position, the second-hand moves to complete one full rotation (say 1 to 60). The minute hand moves to the next position (that is, the next minute) only after the second hand completes one full rotation. Then the second-hand again completes another full rotation corresponding to the minute-hand's current position. For each position of the minute-hand, the second-hand completes one full rotation and the process goes on. Here the second hand movement can be treated as the execution of the inner loop and the minute-hand's movement can be treated as the execution of the outer loop. Figure 1.2 shows the concept of nested loops with the help of digital watch.

Fig. 1.2: Concept of nested loop using digital watch

As in Figure 1.2, the value of second changes from 0 to 59 keeping the minute fixed. Once the value of seconds reached 59, the next change will be in minutes. Once the minute is changed, the second resets its value to 0.

All types of loops in C++ allow nesting. An example is given to show the working procedure of a nested for loop.

```
for( i=1; i<=2; ++i)
{
    for(j=1; j<=3; ++j)
    {
        cout<< "\n" << i << " and " << j;
    }
}
```

Outer loop

Inner loop

Initially value 1 is assigned to the outer loop variable `i`. Its test expression is evaluated to be True and hence the body of the loop is executed. The body contains the inner loop with the control variable `j` and it begins to execute by assigning the initial value 1 to `j`. The inner loop is executed 3 times, for `j=1`, `j=2`, `j=3`. Each time it evaluates the test expression `j<=3` and displays the output since it is True.

```
1 and 1
1 and 2
1 and 3
```

The first 1 is of `i` and the second 1 is of `j`

When the test expression `j<=3` is False, the program control comes out of the inner loop. Now the update statement of the outer loop is executed which makes `i=2`. Then the test expression `i<=2` is evaluated to True and once again the loop body (i.e. the inner loop) is executed. Inner loop is again executed 3 times, for `j=1`, `j=2`, `j=3` and displays the output.

```
2 and 1
2 and 2
2 and 3
```

After completing the execution of the inner loop, the control again goes back to the update expression of the outer loop. Value of `i` is incremented by 1 (now `i=3`) and the test expression `i<=2` is now evaluated to be False. Hence the loop terminates its execution. Table 1.2 illustrates the execution of the above given program segment:

Iterations	Outer loop (i)	Inner loop (j)	Output
1	1	1	1 and 1
2	1	2	1 and 2
3	1	3	1 and 3
4	2	1	2 and 1
5	2	2	2 and 2
6	2	3	2 and 3

Table 1.2: Execution of a nested loop

When working with nested loops, the control variable of the outer loop changes its value only after the inner loop is terminated. Let us write a program to display the given triangle using nested loop.

```
*
* *
* * *
* * * *
* * * * *
```

Program 1.8: To display a triangle of stars

```
#include<iostream>
using namespace std;
int main()
{
    short int i, j;
    for(i=1; i<=5; ++i)     //outer loop
    {
        cout<< "\n" ;
        for(j=1; j<=i; ++j) // inner loop
            cout<< '*';
    }
    return 0;
}
```

Let us do

In Program 1.8, what will the output be if we use the statement `cout<<i;` instead of `cout<<'*';`?

Similarly, what will the output be if we use the statement `cout<<j;`?

1.4 Jump statements

The statements that facilitate the transfer of program control from one place to another are called jump statements. C++ provides four jump statements that perform unconditional control transfer in a program. They are `return`, `goto`, `break` and `continue` statements. All of these are keywords. In addition, C++ provides a standard library function `exit()` that helps us to terminate a program.

The `return` statement is used to transfer control back to the calling program or to come out of a function. It will be explained in detail later in Chapter 3. Now, we will discuss the other jump statements.

1.4.1 goto statement

The goto statement can transfer the program control to anywhere in the function. The target destination of a goto statement is marked by a label, which is an identifier. The syntax of goto statement is:

```
            goto  label;
            ............;
            ............;
label:      ............;
            ............;
```

where the `label` can appear in the program either before or after `goto` statement. The label is followed by a colon (:) symbol. For example, consider the following code fragment which prints numbers from 1 to 50.

```
    int i=1;
start:
    cout<<i;
    ++i;
    if (i<=50)
        goto start;
```

Here, the `cout` statement prints the value 1. After that `i` is incremented by 1 (now `i=2`), then the test expression `i<=50` is evaluated. Since it is True the control is transferred to the statement marked with the label `start`. When the test expression evaluates to False, the process terminates and transfers the program control following the `if` statement. It is to be noted that the usage of `goto` is not encouraged in structured programming.

1.4.2 break statement

We have already discussed the effect of `break` in `switch` statement. It transfers the program control outside the `switch` block. When a `break` statement is encountered in a loop (`for`, `while`, `do-while`), it takes the program control outside the immediate enclosing loop. Execution continues from the statement immediately after the control structure. Let us see how it affects the execution of loops. Consider the following two program segments.

Code segment 1:

```
i=1;
while(i<=10)
{
    cin>>num;
    if (num==0)
        break;
    cout<<"Entered number is: "<<num;
    cout<<"\nInside the loop";
    ++i;
}
cout<<"\nComes out of the loop";
```

The above code fragment allows to input 10 different numbers. During the input if any number happens to be 0, the program control comes out of the loop by skipping the rest of the statements within the loop-body and displays the message "Comes out of the loop" on the screen. Let us consider another code segment that uses break within a nested loop.

Code segment 2:

```
for(i=1; i<=5; ++i)     //outer loop
{
    cout<<"\n";
    for(j=1; j<=i; ++j) //inner loop
    {
        cout<<"* ";
        if (j==3) break;
    }
}
```

```
*
* *
* * *
* * *
* * *
```

This code segment will display the given pattern:

The nested loop executes normally for the value of i=1, i=2, i=3. For each value of i, the variable j takes values from 1 to i. When the value of i becomes 4, the inner loop executes for the value of j=1, j=2, j=3 and comes out from the inner loop on executing the break statement.

1.4.3 continue statement

The statement continue is another jump statement used for skipping over a part of the code within the loop-body and forcing the next iteration. The break statement forces termination of the loop, but the continue statement forces next iteration of the loop. The following program segment explains the working of a continue statement:

```
for (i=1; i<=10; ++i)
{
    if (i==6)
        continue;
    cout<<i<<"\t";
}
```

This code gives the following output:

1 2 3 4 5 7 8 9 10

Note that 6 is not in the list. When the value of i becomes 6 the continue statement is executed. As a result, the output statement is skipped and program control goes to the update expression for next iteration.

A `break` statement inside a loop will abort the loop and transfer control to the statement following the loop. A `continue` statement will just abandon the current iteration and let the loop continue with next iteration. When a `continue` statement is used within `while` and `do-while` loops, care should be taken to avoid infinite execution.

Let us do

Let us compare `break` and `continue` statements. Table 1.3 is designed to show the comparison aspects. Some of the entries are filled. The remaining entries are left for you to fill with proper comparison points.

continue	break
•	• Used only with loops
• Takes the control outside the loop by skipping the remaining part of the loop	• ....
•	• Program control goes outside only when the test expression of the loop returns false

Table 1.3: Comparison between `break` *and* `continue`

Let us write a program that requires the use of nested loops. Program 1.9 can display all prime numbers below 100.

Program 1.9: To display all prime numbers below 100

```
#include<iostream>
using namespace std;
int main()
{   short int n, i, flag;
    cout<<"Prime numbers below 100 are...\n";
    for(n=2; n<=100; n++)  //Outer loop
    {   flag=1;
        for(i=2; i<=n/2; i++)     //Inner loop
            if(n%i==0)
            {   flag=0;
                break;//Takes the control outside the iiner loop
            }
        if(flag==1)  cout<<n<<'\t';
    }
    return 0;
}
```

In Program 1.8, the variable `n` is given the values from 2 to 100 through the outer loop. Each value is checked for prime using the inner loop. If any of the values from 2 to `n/2` is found to be a factor of `n`, inner loop is terminated by changing the value of `flag` from 1 to 0. The value of `n`, for which `flag` remains 1 after the termination of the inner loop, is prime and is printed.

Let us conclude

We have refreshed ourselves with the basic concepts of C++ language. Character set, tokens, data types, type modifiers, expressions and type conversions are presented in capsule form. Different types of C++ statements are recollected with the help of examples. Various control transfer statements are briefly explained with programs. Nested loops and the two jump statements `break` and `continue` are introduced as the new concepts. A clear-cut idea about these topics is very much essential to learn the concepts covered in Chapters 2 and 3 of this book.

Let us practice

1. Write a C++ program to display all palindrome numbers between 100 and 200.
2. Write a C++ program to display all Armstrong numbers below 1000.
3. Write a C++ program to display all perfect numbers below 1000.
4. Write a C++ program to display the multiplication table of a number.
5. Write a C++ program to prepare electricity bill for a group of consumers. The previous meter reading and current reading are the inputs. The payable amount is to be calculated using the following criteria:

   Up to 300 units : Rs. 5.00/- per unit
   Up to 350 units : Rs. 5.70/- per unit
   Up to 400 units : Rs. 6.10/- per unit
   Up to 500 units : Rs. 6.70/- per unit
   Above 500 units : Rs. 7.50/- per unit

   The program should provide facility to input the details of any number of consumers as the user wants.

Let us assess

1. What is wrong with the following code segment?

```
for (short i=1; i<5; ++i)
    for (i=5; i>0; --i)
        cout<<i<<"\t";
```

2. Distinguish between `break` and `continue` statements.

3. The `default` case in `switch` is equivalent to the `else` block of `else if` ladder. Justify this statement with the help of example.
4. What will be the output of the following code fragment?

```
for (outer=10; outer>5; --outer)
    for (inner=1; inner<4; ++inner)
        cout<<outer<<"\t"<<inner<<endl;
```

5. Write a program to produce the following output using nested loop:

```
A
A   B
A   B   C
A   B   C   D
A   B   C   D   E
```

6. Read the following C++ code snippet:

```
for (n=1; n<5; ++n)
{
    cout<<i;
    if (i==2) continue;
    if (i%3==0) break;
    cout<<"Hello";
}
```

Choose the correct output of this code from the following:

a. 1Hello2Hello3Hello4Hello b. 1Hello2Hello3

c. 1Hello23 d. 1Hello23Hello

7. Read the following C++ code segment and replace it with a looping statement:

```
        cin>>n;
loop:   r=n%10;
        s=s*10+r;
        n=n/10;
        if (n!=0) goto loop;
        cout<<s;
```

8. Which of the following is not a character constant in C++?

a. `'\t'` b. `'a'` c. `'9'` d. `'9a'`

9. Some of the following identifiers are invalid. Identify them and give reason for the invalidity.

a. `unsigned` b. `cpp` c. `2num` d. `cout`

10. What happens if `break` is not used in `switch` statement?

2 Arrays

Significant Learning Outcomes

After the completion of this chapter, the learner

- identifies the scenarios where an array can be used.
- uses arrays to refer to a group of data.
- familiarises with the memory allocation for arrays.
- accesses any element in an array while solving problems.
- solves problems in which large amount of data is to be processed.
- represnts strings using character arrays.
- carries out various word processing operations using character arrays.

We use variables to store data in programs. But if the quantity of data is large, more variables are to be used. This will cause difficulty in accessing the required data. We have revised the concept of C++ data types in Chapter 1 and we used basic data types to declare variables and perform type conversion. In this chapter, a derived data type in C++, named 'array' is introduced. The word 'array' is not a data type name, rather it is a kind of data type derived from fundamental data types to handle large number of data easily. We will discuss the creation and initialisation of arrays, and operations like traversal.

2.1 Array and its need

An **array** is a collection of elements of the same type placed in contiguous memory locations. Arrays are used to store a set of values of the same type under a single variable name. Each element in an array can be accessed using its position in the list, called index number or subscript.

Why do we need arrays? We will illustrate this with the help of an example. Let us consider a situation where we need to store the scores of 20 students in a class and has to find their class

average. If we try to solve this problem by making use of variables, we will need 20 variables to store students' scores. Remembering and managing these 20 variables is not an easy task and the program may become complex and difficult to understand.

```
int  a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t;
float  avg;
cin>>a>>b>>c>>d>>e>>f>>g>>h>>i>>j>>k>>l>>m>>n>>o>>p>>q>>r>>s>>t;
avg  =  (a+b+c+d+e+f+g+h+i+j+k+l+m+n+o+p+q+r+s+t)/20.0;
```

As it is, this code is fine. However, if we want to modify it to deal with the scores of a large number of students, say 1000, we have a very long and repetitive task at hand. We have to find a way to reduce the complexity of this task.

The concept of array comes as a boon in such situations. As it is a collection of elements, memory locations are to be allocated. We know that a declaration statement is needed for memory allocation. Let us see how arrays are declared and used.

2.1.1 Declaring arrays

Just like the ordinary variable, the array is to be declared properly before it is used. The syntax for declaring an array in C++ is as follows.

```
data_type  array_name[size];
```

In the syntax, `data_type` is the type of data that the array variable can store, `array_name` is an identifier for naming the array and the `size` is a positive integer number that specifies the number of elements in the array. The following is an example:

```
int  num[10];
```

The above statement declares an array named `num` that can store 10 integer numbers. Each item in an array is called an `element` of the array. The elements in the array are stored sequentially as shown in Figure 2.1. The first element is stored in the first location; the second element is stored in the second location and so on.

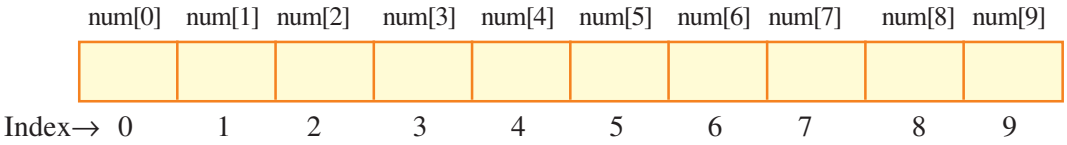


Fig. 2.1: Arrangement of elements in an array

Since the elements in the array are stored sequentially, any element can be accessed by giving the array's name and the element's position. This position is called the **index** or **subscript** value. In C++, the array index starts with zero. If an array is

declared as `int num[10];` then the possible index values are from 0 to 9. In this array, the first element can be referenced as `num[0]` and the last element as `num[9]`. The subscripted variable, `num[0]`, is read as "num of zero" or "num zero". It's a shortened way of saying "the num array subscripted by zero". So, the problem of referring the scores of 1000 students can be resolved by the following statement:

```
int score[1000];
```

The array, named `score`, can store the scores of 1000 students. The score of the first student is referenced by `score[0]` and that of the last by `score[999]`.

2.1.2 Memory allocation for arrays

The amount of storage required to hold an array is directly related to its type and size. Figure 2.2 shows the memory allocation for the first five elements of array **num**, assuming 1000 as the address of the first element. Since `num` is an integer type array, size of each element is 4 bytes (in a system with 32 bit integer representation using GCC) and it will be represented in memory as shown in Figure 2.2.

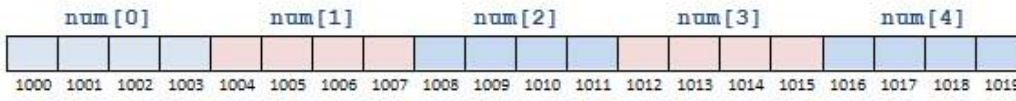


Fig. 2.2: Memory allocation for an integer array

The memory space allocated for an array can be computed using the following formula:

total_bytes = `sizeof`(array_type) × size_of_array

For example, total bytes allocated for the array declared as `float a[10];` will be 4 × 10 = 40 bytes.

2.1.3 Array initialisation

Array elements can be initialised in their declaration statements in the same manner as in the case of variables, except that the values must be included in braces, as shown in the following examples:

```
int score[5] = {98, 87, 92, 79, 85};
char code[6] = {'s', 'a', 'm', 'p', 'l', 'e'};
float wgpa[7] = {9.60, 6.43, 8.50, 8.65, 5.89, 7.56, 8.22};
```

Initial values are stored in the order they are written, with the first value used to initialize element 0, the second value used to initialize element 1, and so on. In the first example, `score[0]` is initialized to 98, `score[1]` is initialized to 87, `score[2]` is initialized to 92, `score[3]` is initialized to 79, and `score[4]` is initialized to 85.

If the number of initial values is less than the size of the array, they will be stored in the elements starting from the first position and the remaining positions will be initialized with zero, in the case of numeric data types. For `char` type array, such positions will be initialised with `' '` (space bar) character. When an array is initialized with values, the size can be omitted. For example, the following declaration statement will reserve memory for five elements:

```
int num[] = {16, 12, 10, 14, 11};
```

2. 1.4 Accessing elements of arrays

Array elements can be used anywhere in a program as we do in the case of normal variables. We have seen that array is accessed element-wise. That is, only one element can be accessed at a time. The element is specified by the array name with the subscript. The following are some examples of using the elements of the array named score:

```
score[0] = 95;
score[1] = score[0] - 11;
cin >> score[2];
score[3] = 79;
cout << score[2];
sum = score[0] + score[1] + score[2] + score[3] + score[4];
```

The subscript in brackets can be a variable, a constant or an expression that evaluates to an integer. In each case, the value of the expression must be within the valid subscript range of the array. An important advantage of using variable and integer expressions as subscripts is that, it allows sequencing through an array by using a loop. This makes statements more structured keeping away from the inappropriate usage as follows:

```
sum = score[0] + score[1] + score[2] + score[3] + score[4];
```

The subscript values in the above statement can be replaced by the control variable of `for` loop to access each element in the array sequentially. The following code segment illustrates this concept:

```
sum = 0;
for (i=0; i<5; i++)
     sum = sum + score[i];
```

An array element can be assigned a value interactively by using an input statement, as shown below:

```
for(int i=0; i<5; i++)
    cin>>score[i];
```

When this loop is executed, the first value read is stored in the array element `score[0]`, the second in `score[1]` and the last in `score[4]`.

Program 2.1 shows how to read 5 numbers and display them in the reverse order. The program includes two `for` loops. The first one allows the user to input array values. After five values have been entered, the second `for` loop is used to display the stored values from the last to the first.

Program 2.1: To input the scores of 5 students and display them in reverse order

```
#include <iostream>
using namespace std;
int main()
{
    int i, score[5];
    for(i=0; i<5; i++)   //  Reads the scores
    {
        cout<<"Enter a score: ";
        cin>>score[i];
    }
    for(i=4; i>=0; i--)   //  Prints the scores
        cout<<"score[" << i << "] is " << score[i]<<endl;
    return 0;
}
```

The following is a sample output of program 2.1:

```
Enter a score: 55
Enter a score: 80
Enter a score: 78
Enter a score: 75
Enter a score: 92
score[4] is 92
score[3] is 75
score[2] is 78
score[1] is 80
score[0] is 55
```

Accessing each element of an array at least once to perform any operation is known as ***traversal*** operation. Displaying all the elements of an array is an example of traversal. If any operation is performed on all the elements in an array, it is a case of traversal. Program 2.2 shows how traversal is performed in an array.

Program 2.2: Traversal of an array

```
#include <iostream>
using namespace std;
int main()
{
    int a[5], i;
    cout<<"Enter the elements of the array :";
    for(i=0; i<5; i++)
        cin >> a[i];  //Reading the elements
    for(i=0; i<5; i++)
        a[i] = a[i] + 1; // A case of traversal
    cout<<"\nNow value of elements in the array are...\n";
    for(i=0; i<5; i++)
        cout<< a[i]<< "\t";  // Another case of traversal
    return 0;
}
```

The following is a sample output of program 2.2:

```
Enter the elements of the array : 12  3   6   1   8
Now value of elements in the are...
13  4  7  2  9
```

Let us do

1. Write array declarations for the following:
   a. Scores of 100 students
   b. English letters
   c. A list of 10 years
   d. A list of 30 real numbers
2. Write array initialization statements for the following:
   a. An array of 10 scores: 89, 75, 82, 93, 78, 95, 81, 88, 77, and 82
   b. A list of five amounts: 10.62, 13.98, 18.45, 12.68, and 14.76
   c. A list of 100 interest rates, with the first six rates being 6.29, 6.95, 7.25, 7.35, 7.40 and 7.42.
   d. An array of 10 marks with value 0.
   e. An array with the letters of VIBGYOR.
   f. An array with number of days in each month.
3. Write a C++ code to input values into the array: `int ar[50];`
4. Write a C++ code fragment to display the elements in the even positions of the array: `float val[100];`

Let us solve another problem that requires traversal operation. Program 2.3 accepts five numbers from a user and finds the sum of these numbers.

Program 2.3: To find the sum of the elements of an array

```
#include <iostream>
using namespace std;
int main()
{
    int a[5], i, sum;
    cout<<"Enter the elements of the array :";
    for(i=0; i<5; i++)
        cin >> a[i]; //Reading the elements
    sum = 0;
    for(i=0; i<5; i++)
        sum = sum + a[i]; // A case of traversal
    cout<<"\nSum of the elements of the array is "<< sum;
    return 0;
}
```

The following is a sample output of Program 2.3:

```
Enter the elements of the array : 12  3   6   1   8
Sum of the elements of the array is 30
```

Program 2.4 illustrates another case of traversal to find the largest element in an array. In this program a variable `big` is used to hold the largest value. Initially it is assigned with the value in the first location. Then it is compared with the remaining elements. Whenever a larger value is found, it repalces the value of `big`.

Program 2.4: To find the largest element in an array

```
#include <iostream>
using namespace std;
int main()
{
    int a[5], i, big;
    cout<<"Enter the elements of the array :";
    for(i=0; i<5; i++)
        cin >> a[i];
    big = a[0];
    for(i=1; i<5; i++)
        if (a[i] > big)  // A case of traversal
            big = a[i];
```

```
    cout<<"\nThe biggest element is " << big;
    return 0;
}
```

The following is a sample output of program 2.4:

```
Enter the elements of the array : 12  3    6    1    8
The biggest element is 12
```

2.2 String handling using arrays

We know that string is a kind of literal in C++ language. It appears in programs as a sequence of characters within a pair of double quotes. Imagine that you are asked to write a program to store your name and display it. We have learned that variables are required to store data. Let us take the identifier `my_name` as the variable. Remember that in C++, a variable is to be declared before it is used. A declaration statement is required for this and it begins with a data type. Which data type should be used to declare a variable to hold string data? There is no basic data type to represent string data. We may think of **`char`** data type. But note that the variable declared using `char` can hold only one character. Here we have to input string which is a sequence of characters.

Let us consider a name "Niketh". It is a string consisting of six characters. So it cannot be stored in a variable of `char` type. But we know that an array of `char` type can hold more than one character. So, we declare an array as follows:

```
char my_name[10];
```

It is sure that ten contiguous locations, each with one byte size, will be allocated for the array named `my_name`. If we follow the usual array initialization method, we can store the characters in the string "Niketh" as follows:

```
char my_name[10]={'N', 'i', 'k', 'e', 't', 'h'};
```

Figure 2.3 shows the memory allocation for the above declared character array. Note that, we store the letters in the string separated by commas. If we want to input the same data, the following C++ statement can be used:

```
for (int i=0; i<6; i++)
    cin >> my_name[i];
```

During the execution of this statement, we have to input six letters of "Niketh" one after the other separated by **Space bar**, **Tab** or **Enter** key. The memory allocation in both of these cases will be as shown in Figure 2.3.

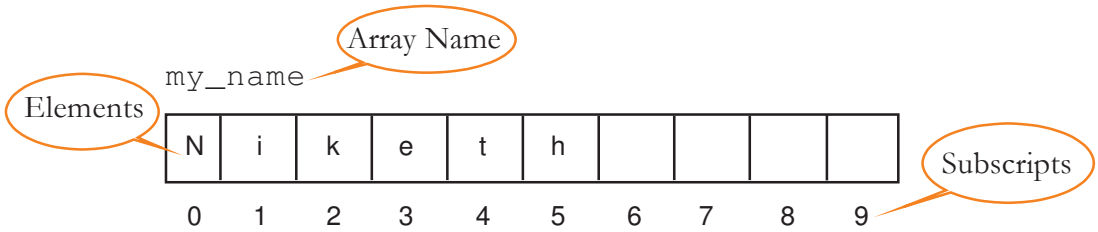


Fig. 2.3 : Memory allocation for the character array

So, let us conclude that a character array can be used to store a string, since it is a sequence of characters. However, it is true that we do not get the feel of inputting a string. Instead, we input the characters constituting the string one by one.

In C++, character arrays have some privileges over other arrays. Once we declare a character array, the array name can be considered as an ordinary variable that can hold string data. Let's say that a character array name is equivalent to a string variable. Thus your name can be stored in the variable `my_name` (the array name) using the following statement:

```
cin >> my_name;
```

It is important to note that this kind of usage is wrong in the case of arrays of other data types. Now let us complete the program. It will be like the one given in Program 2.5.

Program 2.5 To input a string and display

```
#include <iostream>
using namespace std;
int main()
{
     char my_name[10];
     cout << "Enter your name: ";
     cin >> my_name;
     cout << "Hello " << my_name;
     return 0;
}
```

On executing this program we will get the output as shown below.

```
Enter your name: Niketh
Hello Niketh
```

Note that the string constant is not `"Hello"`, but `"Hello "` (a white space is given after the letter `'o'`).

Let us do

Run Program 2.5 and input your full name by expanding the initials if any, and check whether the output is correct or not. If your name contains more than 10 characters, increase the size of the array as needed.

2.3 Memory allocation for strings

We have seen how memory is allocated for an array of characters. As Figure 2.3 shows, the memory required depends upon the number of characters stored. But if we input a string in a character array, the scene will be different. If we run Program 2.5 and input the string `Niketh`, the memory allocation will be as shown in Figure 2.4.

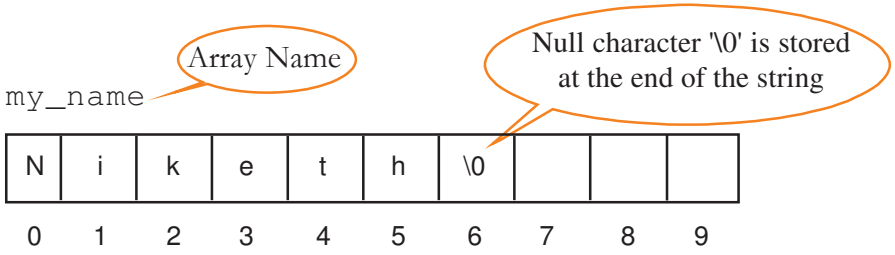


Fig. 2.4 : Memory allocation for the character array

Note that a null character `'\0'` is stored at the end of the string. This character is used as the string terminator and added at the end automatically. Thus we can say that memory required to store a string will be equal to the number of characters in the string plus one byte for null character. In the above case, the memory used to store the string `Niketh` is seven bytes, but the number of characters in the string is only six.

As in the case of variable initialization, we can initialize a character array with a string as follows:

```
char my_name[10] = "Niketh";
char str[] = "Hello World";
```

In the first statement 10 memory locations will be allocated and the string will be stored with null character as the delimiter. The last three bytes will be left unused. But, for the second statement, size of the array is not specified and hence only 12 bytes will be allocated (11 bytes for the string and 1 for `'\0'`).

2.4 Input/Output operations on strings

Program 2.5 contains input and output statements for string data. Let us modify the declaration statement by changing the size of the array to 20. If we run the program by entering the name `Maya Mohan`, the output will be as follows:

```
Enter your name: Maya Mohan
Hello Maya
```

Note that though there is enough size for the array, we get only the word "Maya" as the output. Why does this happen?

Let us have a close look at the input statement: `cin>>my_name;`. We have experienced that only one data item can be input using this statement. A white space is treated as a separator of data. Thus, the input `Maya Mohan` is treated as two data items, `Maya` and `Mohan` separated by white space. Since there is only one input operator (**>>**) followed by a variable, the first data (i.e., `Maya`) is stored. The white space after "Maya" is treated as the delimiter.

So, the problem is that we are unable to input strings containing white spaces. C++ language gives a solution to this problem by a function, named **`gets()`**. The function `gets()` is a console input function used to accept a string of characters including white spaces from the standard input device (keyboard) and store it in a character array.

The string variable (character array name) should be provided to this function as shown below:

```
gets(character_array_name);
```

When we use this function, we have to include the library file `cstdio.h` in the program. Let us modify Program 2.5, by including the statement `#include<cstdio>`, and replacing the statement `cin>>my_name;` by `gets(my_name);` After executing the modified program, the output will be as follows:

```
Enter your name: Maya Mohan
Hello Maya Mohan
```

The output shows the entire string that we input. See the difference between `gets()` and `cin`.

Though we do not use the concept of subscripted variable for the input and output of strings, any element in the array can be accessed by specifying its subscript along with the array name. We can access the first character of the string by `my_name[0]`, fifth character by `my_name[4]` and so on. We can even access the null character (`'\0'`) by its subscript. Program 2.6 illustrates this idea.

Program 2.6: To input a string and count the vowels in a string

```
#include <iostream>
#include <cstdio>    //To use gets() function
using namespace std;
int main()
{
    char str[20];
```

```
    int vow=0;
    cout<<"Enter a string: ";
    gets(str);
    for(int i=0; str[i]!='\0'; i++)
        switch(str[i])
        {   case 'a':
            case 'e':
            case 'i':
            case 'o':
            case 'u': vow++;
        }
    cout<<"No. of vowels in the string "<<str<<" is "<<vow;
    return 0;
}
```

If we run Program 2.6 by inputting the string "hello guys", the following output can be seen:

```
Enter a string: hello guys
No. of vowels in the string hello guys is 3
```

Now, let us analyse the program and see how it works to give this output.

- In the beginning, the `gets()` function is used and so we can input the string `"hello guys"`.
- The body of the `for` loop will be executed as long as the element in the array, referenced by the subscript `i`, is not the null character (`'\0'`). That is, the body of the loop will be executed till the null character is referenced.
- The body of the loop contains only a `switch` statement. Note that, no statements are given against the first four cases of the `switch`. In the last case, the variable `vow` is incremented by 1. You may think that this is required for all the cases. Yes, you are right. But, you should use the `break` statement for each case to exit the `switch` after a match. In this program the action for all the cases are the same and that is why we use this style of code.
- While the `for` loop iterates, the characters will be retrieved one by one for matching against the constants attached to the cases. Whenever a match is found, the variable `vow` is incremented by 1.
- As per the input string, matches occur when the value of `i` becomes 1, 4 and 7. Thus, the variable `vow` is incremented by 1 three times and we get the correct output.

We have seen how `gets()` function facilitates input of strings. Just like the other side of a coin, C++ gives a console function named **`puts()`** to output string data.

The function `puts()` is a console output function used to display a string data on the standard output device (monitor). Its syntax is:

```
puts(string_data);
```

The string constant or variable (character array name) to be displayed should be provided to this function. Observe the following C++ code fragment:

```
char str[10] = "friends";
puts("hello");
puts(str);
```

The output of the above code will be as follows:

```
hello
friends
```

Note that the string `"friends"` in the character array `str[10]` is displayed in a new line. Try this code using `cout<<"hello";` and `cout<<str;` instead of the `puts()` functions and see the difference. The output will be in the same line without a space in between them in the case of `cout` statement.

Let us do

Predict the output, if the input is "HELLO GUYS" in Program 2.6. Execute the program with this input and check whether you get the correct output. Find out the reason for difference in output. Modify the program to get the correct output for any given string.

Let us conclude

We have discussed array as a data type to refer to a group of same type of data. Memory allocation for arrays is explained with the help of schematic diagrams. The use of looping statements, especially `for` loop in manipulating the elements of an array are also illustrated through programs. We have also seen that how arrays help to handle strings effectively in programs.

Let us practice

1. Write a C++ program to input the amount of sales for 12 months into an array named SalesAmt. After all the input, find the total and average amount of sales.
2. Write a C++ program to create an array of N numbers, find the average and display those numbers greater than the average.

3. Write a C++ program to swap the first and the last elements of an integer array.
4. Write a C++ program to input 10 integer numbers into an array and determine the maximum and minimum values among them.
5. Write a C++ program to input a string and find the number of uppercase letters, lowercase letters, digits, special characters and white spaces.
6. Write a C++ program to count the number of words in a sentence.
7. Write a C++ program to find the length of a string.

Let us assess

1. The elements of an array with ten elements are numbered from ____ to ____.
2. An array element is accessed using _____.
3. If `AR` is an array, which element will be referenced using `AR[7]`?
4. Consider the array declaration `int a[3]={2,3,4};` What is the value of `a[1]`?
5. Consider the array declaration `int a[]={1,2,4};` What is the value of `a[1]`?
6. Printing all the elements of an array is an example for _____ operation.
7. Write down the output of the following code segment:

```
puts("hello");
puts("friends");
```

8. Write the initialisation statement to store the string "GCC".
9. Define an Array.
10. What does the declaration `int studlist[1000];` mean?
11. How is memory allocated for a single dimensional array?
12. Write C++ statements to accept an array of 10 elements and display the count of even and odd numbers in it.
13. Read the following statements:

```
char name[20];
cin>>name;
cout<<name;
```

What will be the output if you input the string "Sachin Tendulkar"? Justify your answer.

14. Write C++ statements to accept two single dimensional arrays of equal length and find the difference between corresponding elements.
15. Write a program to check whether a string is a palindrome or not.

3 Functions

Significant Learning Outcomes

After the completion of this chapter, the learner

- identifies the merits of modular programming in problem solving.
- classifies various input output functions for character and string data.
- compares character input functions.
- uses appropriate character and string functions for the I/O operations.
- applies mathematical functions for solving problems.
- uses string functions for the manipulation of string data.
- manipulates character data with predefined character functions.
- implements modular programming by creating functions.
- identifies the role of arguments and compares different methods of function calling.
- recognises the scope of variables and functions in a program.

We discussed some simple programs in the previous chapters. But to solve complex problems, larger programs running into thousands of lines of code are required. Complex problems are divided into smaller ones and programs to solve each of these sub problems are written. In other words, we break the larger programs into smaller sub programs.. In C++, function is a way to divide large programs into smaller sub programs. We are familiar with the function `main()`. We know that it is the essential function in a C++ program. The statements required to solve a problem are given within a pair of braces `{` and `}` after the header `int main()`. We have not broken a problem into sub problems so far and hence the entire task is assisgned to `main()` function. But there are some functions readily available for use, which makes programming simpler. Each of them is assigned with a specific task and they are stored in header files. So, these functions are known as built-in functions or predefined functions. Besides such functions, we can define functions for a specific task. These are called user-defined functions. In this chapter we will discuss some important predefined

functions and learn how to define our own functions. Before going into these, let us familiarise ourselves with a style of programming called modular programming.

3.1 Concept of modular programming

Let us consider the case of a school management software. It is a very large and complex software which may contain many programs for different tasks. The complex task of school management can be divided into smaller tasks or modules and developed in parallel, and later integrated to build the complete software as shown in Figure 3.1.

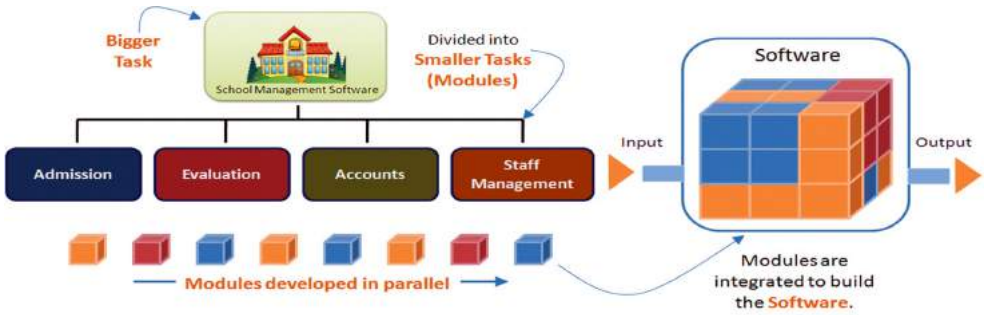


Fig. 31: Modular programming style

In programming, the entire problem will be divided into small sub problems that can be solved by writing separate programs. This kind of approach is known as modular programming. Each sub task will be considered as a module and we write programs for each module. The process of breaking large programs into smaller sub programs is called **modularization**. Computer programming languages have different methods to implement modularization. The sub programs are generally called functions. C++ also facilitates modular programming with functions.

3.1.1 Merits of modular programming

The modular style of programming has many advantages. It reduces the complexity and size of the program, makes the program more readable, improves re-usability and makes the debugging process easy. Let us discuss these features in detail:

Reduces the size of the program: In some cases, certain instructions in a program may be repeated at different points of the program. Consider the expression $\frac{x^5 + y^7}{\sqrt{x} + \sqrt{y}}$. To evaluate this expression for given values of x and y, we have to use instructions for the following:

1. find the 5th power of x.
2. find the 7th power of y.
3. add the results obtained in steps 1 and 2.
4. find the square root of x.
5. find the square root of y.
6. add the result obtained in steps 4 and 5.
7. divide the result obtained in step 3 by that in step 6.

We know that separate loops are needed to find the results of steps 1 and 2. Can you imagine the complexity of the logic to find the square root of a number? It is clear that the program requires the same instructions to process different data at different points. The modular approach helps to isolate the repeating task and write instructions for this. We can assign a name to this set of instructions and this can be invoked by using that name. Thus program size is reduced.

Less chance of error occurrence: When the size of program is reduced, naturally syntax errors will be less in number. The chance of logical error will also be minimized because in a modularized program, we need to concentrate only on one module at a time.

Reduces programming complexity: The net result of the two advantages discovered above is reducing programming complexity. If we properly divide the problem into smaller conceptual units, the development of logic for the solution will be simpler.

Improves reusability: A function written once may be used later in many other programs, instead of starting from scratch. This reduces the time taken for program development.

3.1.2 Demerits of modular programming

Though there are significant merits in modular programming, proper breaking down of the problem is a challenging task. Each sub problem must be independent of others. Utmost care should be given while setting the hierarchy of the execution of the modules.

3.2 Functions in C++

Let us consider the case of a coffee making machine and discuss its function based on Figure 3.2. Water, milk, sugar and coffee powder are supplied to the machine. The machine

Fig. 3.2 : Function of coffee making machine

processes it according to a set of predefined instructions stored in it and returns coffee which is collected in a cup. The instruction-set may be as follows:

1. Get 60 ml milk, 120 ml water, 5 gm coffee powder and 20 gm sugar from the storage of the machine.
2. Boil the mixture
3. Pass it to the outlet.

Usually there will be a button in the machine to invoke this procedure. Let us name the button with the word "MakeCoffee". Symbolically we can represent the invocation as:

Cup = MakeCoffee (Water, Milk, Sugar, Coffee Powder)

We can compare all these aspects with functions in programs. The word "MakeCoffee" is the **name** of the function, "Water", "milk", "sugar" and "coffee powder" are **parameters** for the function and "coffee" is the result **returned**. It is **stored** in a "Cup". Instead of cup, we can use a glass, tumbler or any other container.

Similarly, a C++ function accepts parameters, processes it and returns the result. Figure 3.3 can be viewed as a function **Add** that accepts 3, 5, 2 and 6 as parameters, adds them and returns the value which is stored in the variable **C**. It can be written as:

C = Add (3, 5, 2, 6)

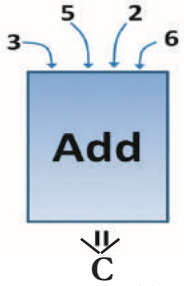


Fig. 3.3. Addition function

We can say that ***function*** is a named unit of statements in a program to perform a specific task as part of the solution. It is not necessary that all the functions require some parameters and all of them return some value. C++ provides a rich collection of functions ready to be used for various tasks. The tasks to be performed by each of these are already written, debugged and compiled, their definitions alone are grouped and stored in files called header files. Such ready-to-use sub programs are called ***predefined functions*** or ***built-in functions***.

While writing large programs, the predefined functions may not suffice to apply modularization. C++ provides the facility to create our own functions for some specific tasks. Everything related to a function such as the task to be carried out, the name and data required are decided by the user and hence they are known as ***user-defined functions***.

What is the role of **`main()`** function then? It may be considered as user-defined in the sense that the task will be defined by the user. We have learnt that it is an essential function in a C++ program because the execution of a program begins in `main()`.

Without `main()`, C++ program will not run. All other functions will be executed when they are called or invoked (or used) in a statement.

3.3 Predefined functions

C++ provides a number of functions for various tasks. We will discuss only the most commonly used functions. While using these functions, some of them require data for performing the task assigned to it. We call them ***parameters*** or ***arguments*** and are provided within the pair of parentheses of the function name. There are certain functions which give results after performing the task. This result is known as ***return-value*** of the function. Some functions do not return any value, rather they perform the specified task. In the following sections, we discuss functions for manipulating strings, performing mathematical operations and processing character data. While using these functions the concerned header files are to be included in the program.

3.3.1 Console functions for character I/O

We have discussed functions for input/output operations on strings. C++ also provides some functions for performing input/ouput operations on characters. In Chapter 2, we discussed `gets()` function and its advantage in string input. Now let us see some functions to input and output character data. These functions require the inclusion of header file **`cstdio`** (stdio.h in Turbo C++ IDE) in the program.

`getchar()`

This function returns the character that is input through the keyboard. The character can be stored in a variable as shown in the example given below:

```
char ch = getchar();
```

In the previous chapter we have seen `puts()` function and its advantage in string output. Let us have a look at a function used to output character data.

`putchar()`

This function displays the character given as the argument on the standard output unit (monitor). The argument may be a character constant or a variable. If an integer value is given as the argument, it will be considered as an ASCII value and the corresponding character will be displayed. The following code segment illustrates the use of `putchar()` function.

```
char ch = 'B';      //assigns 'B' to the variable ch
putchar(ch);        //displays 'B' on the screen
putchar('c');       //displays 'c' on the screen
putchar(97);        //displays 'a' on the screen
```

Program 3.1 illustrates the working of these functions. This program allows inputting a string and a character to be searched.

Program 3.1: To search for a character in a string using console functions

```
#include <iostream>
#include <cstdio>
using namespace std;
int main()
{
    char str[20], ch;
    int i, num=0;
    puts("Enter a string:"); //To print '\n' after the string
    gets(str);//To accept a string with white spaces
    cout<<"\nEnter the character to be searched: ";
    ch=getchar();//To input the character to be searched
    /* A loop to search for the character and count its
       occurrences in the string. Search will be
       terminated when null character is found   */
    for(i=0; str[i]!='\0'; i++)
        if (str[i]==ch)
            num++;
    cout<<"\nThe number of occurences of the character ";
    putchar(ch);
    cout<<" is : "<<num;
    return 0;
}
```

Sample output of the above program is given below

```
Enter a string:
examination
Enter the character to be searched: a
The number of occurences of the character a is : 2
```

3.3.2 Stream functions for I/O operations

C++ provides another facility to perform input/output operations on character and strings. It is in the form of functions that are available in the header file **iostream**. These functions are generally called stream functions since they allow a stream of bytes (data) to flow between memory and objects. Devices like the keyboard and the monitor are referenced as objects in C++. Let us discuss some of these functions.

A. Input functions

These functions allow the input of character and string data. The input functions such as **get()** and **getline()** allow a stream of bytes to flow from input object into the memory. The object **cin** is used to refer to keyboard and hence whenever we input data using keyboard, these functions are called or invoked using this object as **cin.get()** and **cin.getline()**. Note that a period symbol (.), called ***dot operator*** is used between the object `cin` and the function.

i. get()

It can accept a single character or multiple characters (string) through the keyboard. To accept a string, an array name and size are to be given as arguments. Following code segment illustrates the usage of this function.

```
char ch, str[10];
ch=cin.get(ch); //accepts a character and stores in ch
cin.get(ch);    //equivalent to the above statement
cin.get(str,10); //accepts a string of maximum 10 characters
```

ii. getline()

It accepts a string through the keyboard. The delimiter will be Enter key, the number of characters or a specified character. This function uses two syntaxes as shown in the code segment given below.

```
char ch, str[10];
int len;
cin.getline(str,len);        // With 2 arguments
cin.getline(str,len,ch);     // With 3 arguments
```

In the first usage, `getline()` function has two arguments - a character array (here it is, `str`) and an integer (`len`) that represents maximum number of characters that can be stored. In the second usage, a delimiting character (content of `ch`) can also be given along with the number of characters. While inputting the string, only (`len-1`) characters, or characters upto the specified delimiting character, whichever occurs first will be stored in the array.

B. Output functions

Output functions like **put()** and **write()** allow a stream of bytes to flow from memory into an output object. The object **cout** is used with these functions since we use the monitor for the output.

i. put()

It is used to display a character constant or the content of a character variable given as argument.

```
char ch='c';
cout.put(ch);      //character 'c' is displayed
cout.put('B');     //character 'B' is printed
cout.put(65);      //character 'A' is printed
```

ii. write()

This function displays the string contained in the argument. For illustration see the example given below.

```
char str[10]="hello";
cout.write(str,10);
```

The above code segment will dispaly the string `hello` followed by 5 white spaces, since the second argument is 10 and the number of characters in the string is 5.

Program 3.2: To illustrate the working of stream input/output functions

```
#include <iostream>
#include <cstring> //To use strlen() function
using namespace std;
int main()
{
    char ch, str[20];
    cout<<"Enter a character: ";
    cin.get(ch); //To input a character to the variable ch
    cout<<"Enter a string: ";
    cin.getline(str,10,'.');     //To input the string
    cout<<"Entered character is:\t";
    cout.put(ch);           //To display the character
    cout.write("\nEntered string is:",20);
    cout.write(str,strlen(str));
    return 0;
}
```

On executing Program 3.2, the following output will be obtained:

```
Enter a character: p
Enter a string: hello world
Entered character is:     p
Entered string is:
hello wo
```

Let us discuss what happens when the program is executed. In the beginning, `get()` function allows to input a character, say `p`. When the function `getline()` is executed, we can input a string, say `hello world`. The `put()` function is then executed to

display the character p. Observe that the `write()` function displays only `hello wo` in a new line. In the `getline()` function, we specified the integer 10 as the maximum number of characters to be stored in the array `str`. Usually 9 characters will be stored, since one byte is reserved for `'\0'` character as the string terminator. But the output shows only 8 characters including white space. This is because, the Enter key followed by the character input (p) for the `get()` function, is stored as the **`'\n'`** character in the first location of `str`. That is why, the string, `hello wo` is displayed in a new line.

If we run the program, by giving the input `hello.world`, the output will be as follows: Observe the change in the content of `str`.

```
Enter a character: a
Enter a string: hello.world
Entered character is:      a
Entered string is:
hello
```

The change has occurred because the `getline()` function accepts only the characters that appear before the dot symbol.

3.3.3 String functions

While solving problems string data may be involved for processing. C++ provides string functions for their manipulation. The header file **cstring** (string.h in Turbo C++) is to be included in the program to use these functions.

i. strlen()

This function is used to find the length of a string. Length of a string means the number of characters in the string. Its syntax is:

```
int strlen(string);
```

This function takes a string as the argument and gives the length of the string as the result. The following code segment illustrates this.

```
char str[] = "Welcome";
int n;
n = strlen(str);
cout << n;
```

Here, the argument for the `strlen()` function is a string variable and it returns the number of characters in the string, i.e. 7 to the variable `n`. Hence the program code will display 7 as the value of the variable `n`. The output will be the same even though the array declaration is as follows.

```
char str[10] = "Welcome";
```

Note that the array size is specified in the declaration. The argument may be a string constant as shown below:

```
n = strlen("Computer");
```

The above statement returns 8 and will be stored in n.

ii. strcpy()

This function is used to copy one string into another. The syntax of the function is:

```
strcpy(string1, string2);
```

The function will copy string2 to string1. Here string1 is an array of characters and string2 is an array of characters or a string constants. These are the arguments for the execution of the function. The following code illustrates its working:

```
char s1[10], s2[10] = "Welcome";
strcpy(s1,s2);
cout << s1;
```

The string "Welcome" contained in the string variable s1 will be displayed on the screen. The second argument may be a string constant as follows:

```
char str[10]
strcpy(str,"Welcome");
```

Here, the string constant "Welcome" will be stored in the variable str. The assignment statement, str = "Welcome"; is wrong. But we can directly assign value to a character array at the time of declaration as:

```
char str[10] = "Welcome";
```

iii. strcat()

This function is used to append one string to another string. The length of the resultant string is the total length of the two strings. The syntax of the functions is:

```
strcat(string1, string2);
```

Here string1 and string2 are array of characters or string constants. string2 is appended to string1. So, the size of the first argument should be able to accommodate both the strings together. Let us see an example showing the usage of this function:

```
char s1[20] = "Welcome", s2[10] = " to C++";
strcat(s1,s2);
cout << s1;
```

The above program code will display "Welcome to C++" as the value of the variable s1. Note that the string in s2 begins with a white space.

iv. strcmp()

This function is used to compare two strings. In this comparison, the alphabetical order of characters in the strings is considered. The syntax of the function is:

```
strcmp(string1, string2)
```

The function returns any of the following values in three different situations.

- Returns 0 if `string1` and `string2` are same.
- Returns a –ve value if `string1` is alphabetically lower than `string2`.
- Returns a +ve value if `string1` is alphabetically higher than `string2`.

The following code fragment shows the working of this function.

```
char s1[]="Deepthi", s2[]="Divya";
int n;
n = strcmp(s1,s2);
if(n==0)
    cout<<"Both the strings are same";
else if(n < 0)
    cout<<"s1 < s2";
else
    cout<<"s1 > s2";
```

It is clear that the above program code will display `"s1 < s2"` as the output.

v. strcmpi()

This function is used to compare two strings ignoring cases. That is, the function will treat both the upper case and lower case letters as same for comparison. The syntax and working of the function are the same as that of `strcmp()` except that `strcmpi()` is not case sensitive. This function also returns values as in the case of `strcmp()`. Consider the following code segment:

```
char s1[]="SANIL", s2[]="sanil";
int n;
n = strcmpi(s1,s2);
if(n==0)
    cout<<"strings are same";
else if(n < 0)
    cout<<"s1 < s2";
else
    cout<<"s1 > s2";
```

The above program code will display `"strings are same"` as the output, because the uppercase and lowercase letters will be treated as same during the comparison.

Program 3.3 compares and concatenates two strings. The length of the newly formed string is also displayed.

Program 3.3 : To combine two strings if they are different and find its length

```
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
    char s1[15], s2[15], s3[30];
    cout<<"Enter two strings: ";
    cin>>s1>>s2;
    int n;
    n=strcmp(s1,s2);
    if (n==0)
        cout<<"\nThe input strings are same";
    else
    {
        cout<<"\nThe input strings are not same";
        strcpy(s3,s1);//Copies the string in s1 into s3
        strcat(s3,s2);//Appends the string in s2 to that in s3
        cout<<"String after concatenation is: "<<s3;
        cout<<"\nLength of the new string is: "
             <<strlen(s3);
    }
    return 0;
}
```

Header file essential for using string manipulating functions

Sample Output:

```
Enter two strings:india
kerala
The input strings are not same
String after concatenation is:indiakerala
Length of the new string is: 11
```

3.3.4 Mathematical functions

Now, let us discuss the commonly used mathematical functions available in C++. We should include the header file **cmath** (math.h in Turbo C++) to use these functions in the program.

i. abs()

It is used to find the absolute value of an integer. It takes an integer as the argument (+ve or –ve) and returns the absolute value. Its syntax is:

```
int abs(int)
```

The following is an example to show the output of this function:

```
int n = -25;
cout << abs(n);
```

The above program code will display 25. If we want to find the absolute value of a floating point number, we can use **fabs()** function as used above. It will return the floating point value.

ii. sqrt()

It is used to find the square root of a number. The argument to this function can be of type `int`, `float` or `double`. The function returns the non-negative square root of the argument. Its syntax is:

```
double sqrt(double)
```

The following code snippet is an example. This code will display 5.

```
int n = 25;
float b = sqrt(n);
cout << b;
```

If the value of n is 25.4, then the answer will be 5.03841

iii. pow()

This function is used to find the power of a number. It takes two arguments **x** and **y**. The argument **x** and **y** are of type `int`, `float` or `double`. The function returns the value of $\mathbf{x}^{\mathbf{y}}$. Its syntax is:

```
double pow(double, double)
```

The following example shows the working of this function.

```
int x = 5, y = 4, z;
z = pow(x, y);
cout << z;
```

The above program code will display 625.

x^y means 5^4 which is 5*5*5*5 i.e. 625

Program 3.4 : To find the area of a triangle and a circle using mathematical functions

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
    const float pi=22.0/7;
    int a,b,c, radius;
    float s, area1, area2;
    cout<<"Enter the three sides of the triangle: ";
    cin>>a>>b>>c;
    s = (a+b+c)/2.0;
    area1 = sqrt(s*(s-a)*(s-b)*(s-c));
    cout<<"The Area of the Triangle is: "<<area1;
    cout<<"\nEnter the radius of the circle: ";
    cin>>radius;
    area2 = pi*pow(radius,2);
    cout<<"Area of the Circle is: "<<area2;
    return 0;
}
```

Header file essential for using mathematical functions

The following is a sample output of the above program:

```
Enter the three sides of the triangle: 5    7    9
The Area of the Triangle is: 17.4123
Enter the radius of the circle: 2.5
Area of the Circle is: 12.5714
```

3.3.5 Character functions

These functions are used to perform various operations on characters. Following are the various character functions available in C++. The header file **cctype** (ctype.h in Turbo C++) is to be included to use these functions in a program.

i. isupper()

This function is used to check whether a character is in upper case (capital letter) or not. The syntax of the function is:

```
int isupper(char c)
```

The function returns 1 if the given character is in uppercase, and 0 otherwise.

The following statement assigns 0 to the variable n.

```
int n = isupper('x');
```

Consider the following statements:

```
char c = 'A';
int n = isupper(c);
```

The value of the variable n, after the execution of the above statements will be 1, since the given character is in upper case.

ii. islower()

This function is used to check whether a character is in lower case (small letter) or not. The syntax of the function is:

```
int islower(char c)
```

The function returns 1 if the given character is lower case, and 0 otherwise.

After executing the following statements, the value of the variable n will be 1 since the given character is in lower case.

```
char ch = 'x';
int n = islower(ch);
```

But the statement given below assigns 0 to the variable n, since the given character is in the uppercase.

```
int n = islower('A');
```

iii. isalpha()

This function is used to check whether the given character is an alphabet or not. The syntax of the function is:

```
int isalpha(char c)
```

The function returns 1 if the given character is an alphabet, and 0 otherwise.

The following statement assigns 0 to the variable n, since the given character is not an alphabet.

```
int n = isalpha('3');
```

But the statement given below displays 1, since the given character is an alphabet.

```
cout << isalpha('a');
```

iv. isdigit()

This function is used to check whether the given character is a digit or not. The syntax of the function is:

```
int isdigit(char c)
```

The function returns 1 if the given character is a digit, and 0 otherwise.

After executing the following statement, the value of the variable n will be 1 since the given character is a digit.

```
n = isdigit('3');
```

When the following statements are executed, the value of the variable n will be 0, since the given character is not a digit.

```
char c = 'b';
int n = isdigit(c);
```

v. isalnum()

This function is used to check whether a character is alphanumeric or not. The syntax of the function is:

```
int isalnum (char c)
```

The function returns 1 if the given character is alphanumeric, and 0 otherwise.

Each of the following statements returns 1 after the execution.

```
n = isalnum('3');
cout << isalnum('A');
```

But the statements given below assigns 0 to the variable n, since the given character is neither alphabet nor digit.

```
char c = '-';
int n = isalnum(c);
```

vi. toupper()

This function is used to convert the given character into its uppercase. The syntax of the function is:

```
char toupper(char c)
```

The function returns the upper case of the given character. If the given character is in upper case, the output will be the same.

The following statement assigns the character constant 'A' to the variable c.

```
char c = toupper('a');
```

But the output of the statement given below will be 'A' itself.

```
cout << (char)toupper('A');
```

Note that type conversion using (char) is used in this statement. If conversion method is not used, the output will be 65, which is the ASCII code of 'A'.

vii. tolower()

This function is used to convert the given character into its lower case. The syntax of the function is:

```
char tolower(char c)
```

The function returns the lower case of the given character. If the given character is in lowercase, the output will be the same.

Consider the statement: `c = tolower('A');`

After executing the above statement, the value of the variable `c` will be `'a'`. But when the following statements will be executed, the value of the variable `c` will be `'a'` itself.

```
char x = 'a';
char c = tolower(x) ;
```

In the case of functions `tolower()` and `toupper()`, if the argument is other than alphabet, the given character itself will be returned on execution.

Program 3.5 illustrates the use of character functions. This program accepts a line of text and counts the lowercase letters, uppercase letters and digits in the string. It also displays the entire string both in uppercase and lowercase.

Program 3.5: To count different types of characters in the given string

```
#include <iostream>
#include <cstdio>
#include <cctype>
using namespace std;
int main()
{
    char text[80];
    int Ucase=0, Lcase=0, Digit=0;
    cout << "Enter a line of text: ";
    gets(text);
    for(int i=0; text[i]!='\0'; i++)
        if (isupper(text[i]))
            Ucase++;
        else if (islower(text[i]))
            Lcase++;
        else if (isdigit(text[i]))
            Digit++;
```

Loop will be terminated when the value of `i` point to the null character

```
    cout << "\nNo. of uppercase letters = " << Ucase;
    cout << "\nNo. of lowercase letters = " << Lcase;
    cout << "\nNo. of digits = " << Digit;
    cout << "\nThe string in uppercase form is\n";
    i=0;
    while (text[i]!='\0')
    {
        putchar(toupper(text[i]));
        i++;
    }
    cout << "\nThe string in lowercase form is\n";
    i=0;
    do
    {
        putchar(tolower(text[i]));
        i++;
    }while(text[i]!='\0');
    return 0;
}
```

If `cout<<` is used instead of `putchar()`, the ASCII code of the characters will be displayed

A sample output is given below:

```
Enter a line of text : The vehicle ID is KL01 AB101
No. of uppercase letters = 7
No. of lowercase letters = 11
No. of digits = 5
The string in uppercase form is
THE VEHICLE ID IS KL01 AB101
The string in lowercase form is
the vehicle id is kl01 ab101
```

Let us do

Prepare a chart in the following format and fill up the columns with all predefined functions we have discussed so far.

Function	Usage	Syntax	Example	Output

Know your progress

1. What is modular programming?
2. What is meant by a function in C++?
3. Name the header file required for using character functions.
4. Predict the output of `cout<<sqrt(49)`.
5. Pick the odd one out and give reason:
   a. `strlen()` b. `pow()` c. `strcpy()` d. `strcat( )`
6. Name the header file needed for the function pow().
7. Predict the output when we compare the strings "HELLO" and "hello" using the function `strcmpi()`.
8. What will be output of the statement
   `cout<<strlen("smoking kills");` ?
9. Which function converts the alphabet 'P' to 'p'?
10. Identify the name of the function which tests whether the argument is an aphabet or number.

3.4 User-defined functions

All the programs that we have discussed so far contain a function named **`main()`**. We know that a C++ program begins with preprocessor directive statements followed by `using namespace` statement. The remaining part is actually the definition of a function. The `int main()` in the programs is called ***function header*** (or function heading) of the function and the statements within the pair of braces immediately after the header is called its ***body***.

The syntax of a function definition is given below:

```
data_type  function_name(argument_list)
{
     statements in the body;
}
```

The `data_type` is any valid data type of C++. The `function_name` is a user-defined word (identifier). The `argument_list`, which is optional, is a list of parameters, i.e. a list of variables preceded by data types and separated by commas. The body comprises of C++ statements required to perform the task assigned to the function. Once we have decided to create a function, we have to answer certain questions.

(i) Which data type will be used in the function header?

(ii) How many arguments are required and what should be the preceding data type of each?

Let us recollect how we have used the predefined functions `strcpy()` and `sqrt()`. We have seen that these functions will be executed when they are called (or used) in a C++ statement. The function `getchar()` takes no arguments in the parentheses. But for `strcpy()`, two strings are provided as arguments or parameters. Without these arguments this function will not work, because it is defined with two string (character array) arguments. In the case of `sqrt()`, it requires a numeric data as argument and gives a result (of `double` type) after performing the predefined operations on the given argument. This result, as mentioned earlier, is called the **return-value** of the function. The data type of a function depends on this value. In other words we can say that the function should return a value which is of the same data type of the function. So, the data type of a function is also known as the **return type** of the function. Note that we use `return 0;` statement in `main()` function, since it is defined with `int` data type as per the requirement of GCC.

The number and type of arguments depend upon the data required by the function for processing. Note that a function can return only one value. But some functions like `puts()` and `gets()` do not return any value. The header of such functions uses `void` as the return type. A function either returns one value or nothing at all.

3.4.1 Creating user-defined functions

Based on the syntax discussed above, let us create some functions. The following is a function to display a message.

```
void saywelcome()
{
    cout<<"Welcome to the world of functions";
}
```

The name of the function is `saywelcome()`, its data type (return type) is `void` and it does not have any argument. The body contains only one statement.

Now, let us define a function to find the sum of two numbers. Four different types of definitions are given for the same task, but they vary in definition style and hence the usage of each is different from others.

Function 1	*Function 2*
`void sum1()` `{` `  int a, b, s;` `  cout<<"Enter 2 numbers: ";` `  cin>>a>>b;` `  s=a+b;` `  cout<<"Sum="<<s;` `}`	`int sum2()` `{` `  int a, b, s;` `  cout<<"Enter 2 numbers: ";` `  cin>>a>>b;` `  s=a+b;` `  return s;` `}`

Function 3	*Function 4*
`void sum3(int a, int b)` `{` `  int s;` `  s=a+b;` `  cout<<"Sum="<<s;` `}`	`int sum4(int a, int b)` `{` `  int s;` `  s=a+b;` `  return s;` `}`

Let us analyse these functions and see how they are different. The task is the same in all these functions, but they differ in the number of parameters and return type.

Table 3.1 shows that the function defined with a data type other than **`void`** should return a value in accordance with the data type. The **`return`** statement is used for this purpose (Refer to functions 2 and 4). The **`return`** statement returns a value to the calling function and transfers the program control back to the calling function. So, remember that if a `return` statement is executed in a function, the remaining statements within that function will not be executed.

Name	Arguments	Return value
sum1()	No arguments	Does not return any value
sum2()	No arguments	Returns an integer value
sum3()	Two integer arguments	Does not return any value
sum4()	Two integer arguments	Returns an integer value

Table 3.1 : Analysis of functions

In most of the functions, `return` is placed at the end of the function. The functions defined with void data type may not have a `return` statement within the body. But if we use `return` statement, we cannot provide any value to it.

Now, let us see how these functions are to be called and how they are executed. We know that no function other than `main()` is executed automatically. The sub functions, either predefined or user-defined, will be executed only when they are called from `main()` function or other user-defined function. The code segments within the rectangles of the following program shows the function calls. Here the `main()` is the calling function and `sum1()`, `sum2()`, `sum3()`, and `sum4()` are the called functions.

```
int main()
{
    int x, y, z=5, result;
    cout << "\nCalling the first function\n";
    sum1();
    cout << "\nCalling the second function\n";
    result = sum2();
    cout << "Sum given by function 2 is " << result;
    cout << "\nEnter values for x and y : ";
    cin >> x >> y;
    cout << "\nCalling the third function\n";
    sum3(x,y);
    cout << "\nCalling the fourth function\n";
    result = sum4(z,12);
    cout << "Sum given by function 4 is " << result;
    cout << "\nEnd of main function";
    return 0;
}
```

The output of the program will be as follows:

```
Calling the first function
Enter 2 numbers: 10  25
Sum=35
Calling the second function
Enter 2 numbers: 5   7
Sum given by function 2 is 12
Enter values for x and y : 8   13
Calling the third function
Sum=21
Calling the fourth function
Sum given by function 4 is 17
End of main function
```

Input for a and b of sum1()

Input for a and b of sum2()

Input for x and y of main()

Function 4 requires two numbers for the task assigned and hence we provide two arguments. The function performs some calculations and gives a result. As there is only one result, it can be returned. Comparatively this function is a better option to find the sum of any two numbers.

Now, let us write a complete C++ program to find the product of two numbers. We will write the program using a user-defined function. But where do we write the

user-defined function in a C++ program? The following table shows two styles to place the user-defined function:

Program 3.6 - Product of two numbers - Program 3.7

Function before main()

```
#include <iostream>
using namespace std;
int product(int a, int b)
{
   int p;
   p = a * b;
   return p;
}
//Definition above main()
int main()
{
   int ans, num1,num2;
   cout<<"Enter 2 Numbers:";
   cin>>num1>>num2;
   ans = product(num1,num2);
   cout<<"Product = "<<ans;
   return 0;
}
```

Function after main()

```
#include <iostream>
using namespace std;
int main()
{
   int ans, num1,num2;
   cout<<"Enter 2 Numbers:";
   cin>>num1>>num2;
   ans = product(num1,num2);
   cout<<"Product = "<<ans;
   return 0;
}
//Definition below main()
int product(int a, int b)
{
   int p;
   p = a * b;
   return p;
}
```

When we compile Program 3.6, there will be no error. But if we compile Program 3.7, there will be an error `'product was not declared in this scope'`. Let us see what this error means.

3.4.2 Prototype of functions

We have seen that a C++ program can contain any number of functions. But it must have a `main()` function to begin the execution. We can write the definitions of functions in any order as we wish. We can define the `main()` function first and all other functions after that or vice versa. Program 3.6 contains the `main()` function after the user-defined function, but in Program 3.7, the `main()` is defined before the user-defined function. When we compile this program, it will give an error - "`product was not declared in this scope`". This is because the function `product()` is called in the program, before it is defined. During the compilation of the `main()` function, when the compiler encounters the function call `product()`, it is not aware of such a function. Compiler is unable to check whether there is such a function, whether its usage is correct or not, and whether it is accessible or not. So

it reports an error. This error can be removed by giving a declaration statement about the function and this statement is known as prototype. A **function prototype** is the declaration of a function by which compiler is provided with the information about the function such as the name of the function, its return type, the number and type of arguments, and its accessibility. This information is essential for the compiler to verify the correctness of the function call in the program. This information is available in the function header and hence the header alone will be written as a statement before the function call. The following is the format:

```
data_type  function_name(argument_list);
```

In the prototype, the argument names need not be specified. Number and type of arguments must be specified.

So, the error in Program 3.7 can be rectified by inserting the following statement before the function call in the `main()` function.

```
int product(int, int);
```

Like a variable declaration, a function must be declared before it is used in the program. If a function is defined before it is used in the program, there is no need to declare the function separately. The declaration statement may be given outside the `main()` function. The position of the prototype differs in the accessibility of the function. We will discuss this later in this chapter. Wherever be the position of the function definition, execution of the program begins in `main()`.

3.4.3 Arguments of functions

We have seen that functions have arguments or parameters for getting data for processing. Let us see the role of arguments in function call. Consider the function given below:

```
float SimpleInterest(int P, int N, float R)
{
    float amt;
    amt = P * N * R / 100;
    return amt;
}
```

This function gives the simple interest of a given principal amount for a given period at a given rate of interest. The following code segment illustrates different function calls:

```
cout << SimpleInterest(1000,3,2);//Function call 1
int x, y; float z=3.5, a;
cin >> x >> y;
a = SimpleInterest(x, y, z); //Function call 2
```

When the first statement is executed, the values 1000, 3 and 2 are passed to the argument list in the function definition. The arguments P, N and R get the values 1000, 3 and 2, respectively. Similarly, when the last statement is executed, the values of the variables x, y and z are passed to the arguments P, N and R.

The variables x, y and z are called actual (original) arguments or actual parameters since they are the actual data passed to the function for processing. The variables P, N and R used in the function header are known as formal arguments or formal parameters. These arguments are intended to receive data from the calling function. **Arguments or parameters** are the means to pass values from the calling function to the called function. The variables used in the function definition as arguments are known as **formal arguments**. The constants, variables or expressions used in the function call are known as **actual (original) arguments**. If variables are used in function prototype, they are known as dummy arguments.

Now, let us write a program that uses a function `fact()` that returns the factorial of a given number to find the value of nCr. As we know, factorial of a number N is the product of the first N natural numbers. The value of nCr is calculated by the formula $\frac{n!}{r!(n-r)!}$, where n! denotes the factorial of n.

Program 3.8: To find the value of nCr

```
#include<iostream>
using namespace std;
int fact(int);   //Function prototype
int main()
{
    int n,r, ncr;
    cout<<"Enter the values of n and r : ";
    cin>>n>>r;
    ncr=fact(n)/(fact(r)*fact(n-r));
    cout<<n<<"C"<<r<<" = "<<ncr;
    return 0;
}
int fact(int N)  //Function header
{
    int f;
    for(f=1; N>0; N--)
        f=f*N;
    return f;
}
```

Actual arguments

Function call according to the formula

Formal argument

Factorial is returned

The following is a sample output:

```
Enter the values of n and r : 5     2
5C2 = 10
```

User inputs

3.4.4 Functions with default arguments

Let us consider a function `TimeSec()` with the argument list as follows. This function accepts three numbers that represent time in hours, minutes and seconds. The function converts this into seconds.

```
int TimeSec(int H, int M=0, int S=0)
{
    int sec = H * 3600 + M * 60 + S;
    return sec;
}
```

Note that the two arguments `M` and `S` are given default value 0. So, this function can be invoked in the following ways.

```
long s1 = TimeSec(2,10,40);
long s2 = TimeSec(1,30);
long s3 = TimeSec(3);
```

It is important to note that all the default arguments must be placed from the right to the left in the argument list. When a function is called, actual arguments are passed to the formal arguments from left onwards.

When the first statement is executed, the function is called by passing the values 2, 10 and 40 to the formal parameters `H`, `M` and `S`, respectively. The initial values of `M` and `S` are over-written by the actual arguments. During the function call in the second statement, `H` and `M` get values from actual arguments, but `S` works with its default value 0. Similarly, when the third statement is executed, `H` gets the value from calling function, but `M` and `S` use the default values. So, after the function calls, the values of `s1`, `s2` and `s3` will be 7840, 5400 and 10800, respectively.

We have seen that functions can be defined with arguments assigned with initial values. The initialized formal arguments are called **default arguments** which allow the programmer to call a function with different number of arguments. That is, we can call the function with or without giving values to the default arguments.

3.4.5 Methods of calling functions

Suppose your teacher asks you to prepare invitation letters for the parents of all students in your class, requesting them to attend a function in your school. The teacher can give you a blank format of the invitation letter and also a list containing

the names of all the parents. The teacher can give you the name list in two ways. She can take a photo copy of the name list and give it to you. Otherwise, she can give the original name list itself. What difference would you feel in getting the name list in these two ways? If the teacher gives the original name list, you will be careful not to make any marks or writing in the name list because the teacher may want the same name list for future use. But if you are given a photo copy of the name list, you can make any marking in the list, because the change will not affect the original name list.

Let us consider the task of preparing the invitation letter as a function. The name list is an argument for the function. The argument can be passed to the function in two ways. One is to pass a copy of the name list and the other is to pass the original name list. If the original name list is passed, the changes made in the name list, while preparing the invitation letter, will affect the original name list. Similarly, in C++, an argument can be passed to a function in two ways. Based on the method of passing the arguments, the function calling methods can be classified as Call by Value method and Call by Reference method. The following section describes the methods of argument passing in detail.

a. Call by value (Pass by value) method

In this method, the value contained in the actual argument is passed to the formal argument. In other words, a copy of the actual argument is passed to the function. Hence, if the formal argument is modified within the function, the change is not reflected in the actual argument at the calling place. In all previous functions, we passed the argument by value only. See the following example:

```
void change(int n)
{
    n = n + 1;
    cout << "n = " << n << '\n';
}
void main()
{
    int x = 20;
    change(x);
    cout << "x = " << x;
}
```

The parameter n has its own memory location to store the value 20 in change().

The value of x is passed to n in change()

When we pass an argument as specified in the above program segement, only a copy of the variable x is passed to the function. In other words, we can say that only the value of the variable x is passed to the function. Thus the formal parameter n in

the function will get the value 20. When we increase the value of n, it will not affect the value of the variable x. The following will be the output of the above code:

```
n = 21
x = 20
```

Table 3.2 shows what happens to the arguments when a function is called using call-by-value method:

Before function call	*After function call*	*After function execution*
main() { .... } x 20 change(int n) { ..... } n	main() { .... } x 20 change(int n) { ..... } n 20	main() { .... } x 20 change(int n) { ..... } n 21

Table 3.2: Call by value procedure

b. Call by reference (Pass by reference) method

When an argument is passed by reference, the reference of the actual argument is passed to the function. As a result, the memory location allocated to the actual argument will be shared by the formal argument. So, if the formal argument is modified within the function, the change will be reflected in the actual argument at the calling place. In C++, to pass an argument by reference we use reference variable as formal parameter. A **reference variable** is an alias name of another variable. An ampersand symbol (**&**) is placed in between the data type and the variable in the function header. Reference variables will not be allocated memory exclusively like the other variables. Instead, it will share the memory allocated to the actual arguments. The following function uses reference variable as formal parameter and hence call by reference method is implemented for function call.

```
void change(int & n)
{
    n = n + 1;
    cout << "n = " << n << '\n';
}
```

The parameter n is a reference variable and hence there is no exclusive memory allocation for it

```
void main()
{
    int x=20;
    change(x);
    cout << "x = " << x;
}
```

The reference of x will be passed to n of the change() function, which results into the sharing of memory.

Note that the only change in the `change()` function is in the function header. The **&** symbol in the declaration of the parameter `n` means that the argument is a reference variable and hence the function will be called by passing reference. Hence when the argument `x` is passed to the `change()` function, the variable `n` gets the address of `x` so that the location will be shared. In other words, the variables `x` and `n` refer to the same memory location. We use the name `x` in the `main()` function, and the name `n` in the `change()` function to refer the same storage location. So, when we change the value of `n`, we are actually changing the value of `x`. If we run the above program, we will get the following output:

```
n = 21
x = 21
```

Table 3.3 depicts the changes in the arguments when call-by-reference is applied for the function call.

Before function call	*After function call*	*After function execution*
main() { .... } x 20 change(int &n) { ..... }	main() { .... } x 20 n change(int &n) { ..... }	main() { .... } x 21 n change(int &n) { ..... }

Table 3.3: Call by value reference procedure

We have discussed the difference between the two methods of function calling. Basically the methods differ in the mode of passing arguments. Let us consolidate the difference between these two methods of function calling. Table 3.4 gives the clear picture.

Call by Value Method	Call by Reference Method
• Ordinary variables are used as formal parameters. • Actual parameters may be constants, variables or expressions. • The changes made in the formal arguments do not reflect in actual arguments. • Exclusive memory allocation is required for the formal arguments.	• Reference variables are used as formal parameters. • Actual parameters will be variables only. • The changes made in the formal arguments do reflect in actual arguments. • Memory of actual arguments is shared by formal arguments.

Table 3.4 : Call by value v/s Call by reference

Let us discuss a typical example for call by reference method. This program uses a function that can be called by reference method for exchanging the values of the two variables in the `main()` function. The process of exchanging values of two variables is known as swapping.

Program 3.9: To swap the values of two variables

```
#include <iostream>
using namespace std;
void swap(int & x, int & y)
{
    int t = x;
    x = y;
    y = t;
}
int main()
{
    int m, n;
    m = 10;
    n = 20;
    cout<<"Before swapping m= "<< m <<" and n= "<<n;
    swap(m, n);
    cout<<"\nAfter swapping m= "<< m <<" and n= "<<n;
    return 0;
}
```

Let us go through the statements in Program 3.9. The actual arguments `m` and `n` are passed to the function by reference. Within the `swap()` function, the values of

x and y are interchanged. When the values of x and y are changed, actually the change takes place in m and n. Therefore the output of the above program code is:

```
Before swapping m= 10 and n= 20
After swapping m= 20 and n= 10
```

Modify the above program by replacing the formal arguments with ordinary variables and predict the output. Check your answer by executing the code in the lab.

Know your progress

1. Identify the most essential function in C++ programs.
2. List the three elements of a function header.
3. What is function prototype?
4. Which component is used for data transfer from calling function to called function?
5. What are the two parameter passing techniques used in C++?
6. Identify the name of the function call where & symbol is used along with formal arguments.

3.5 Scope and life of variables and functions

We have discussed C++ programs consisting of more than one function. Predefined functions are used by including the header file concerned. User-defined functions are placed before or after the `main()` function. We have seen the relevance of function prototypes while defining functions. We have also used variables in the function body and as arguments. Now, let us discuss the availability or accessibility of the variables and functions throughout the program. Program 3.10 illustrates the accessibility of local variables in a program.

Program 3.10: To illustrate the scope and life of variables

```
#include <iostream>
using namespace std;
int cube(int n)
{
    int cb;
    cout<< "The value of x passed to n is " << x;
    cb = n * n * n;
    return cb;
}
```

This is an error because the variable x is declared within the `main()` function. So it cannot be used in other functions.

```
int main()
{
    int x, result;
    cout << "Enter a number : ";
    cin >> x;
    result = cube(x);
    cout << "Cube = " << result;
    cout << "\nCube = "<< cb;
    return 0;
}
```

This is an error because the variable cb is declared within the cube() function. So it cannot be used in other functions.

When we compile the program, there will be two errors because of the reasons shown in the call-outs. The concept of availability or accessibility of variables and functions is termed as their scope and life time. **Scope** of a variable is that part of the program in which it is used. In the above program, scope of the variable cb is in the cube() function because it is declared within that function. Hence this variable cannot be used outside the function. This scope is known as **local scope**. On completing the execution of a function, memory allocated for all the variables within a function is freed. In other words, we can say that the life of a variable, declared within a function, ends with the execution of the last instruction of the function. So, if we use a variable n within the main(), that will be different from the argument n of a called function or a variable n within the called function. The variables used as formal arguments and/or declared within a function have local scope.

Just like variables, functions also have scope. A function can be used within the function where it is declared. That is the function is said to have local scope. If it is declared before the main() function and not within any other function, the scope of the function is the entire program. That is the function can be used at any place in the program. This scope is known as **global scope**. Variables can also be declared with global scope. Such declarations will be outside all the functions in the program.

Look at Program 3.11 to get more clarity on the scope and life of variables and functions throughout the program.

Program 3.11 : To illustrate the scope and life of variables and functions

```
#include <iostream>
using namespace std;
int cb;   //global variable
void test()//global function since defined above other functions
{
    int cube(int n);  //It is a local function
    cb=cube(x); //Invalid call. x is local to main()
    cout<<cb;
}

int main() // beginning of main() function
{
    int x=5;  //local variable
    test();   //valid call since test() is a global function
    cb=cube(x);  //Invalid call. cube() is local to test()
    cout<<cb;
}
int cube(int n)//Argument n is local variable
{
    int val= n*n*n; //val is local variable
    return val;
}
```

The given comments explain the scope and life of functions. A function which is declared inside the function body of another function is called a ***local function*** as it can be used within that function only. A function declared outside the function body of any other function is called a ***global function***. A global function can be used throughout the program. In other words, the scope of a global function is the entire program and that of a local function is only the function where it is declared. Table 3.5 summarises the scope and life time of variables and functions.

Scope & life	Local	Global
Variables	• Declared within a function or a block of statements. • Available only within that function or block. • Memory is allocated when the function or block is active and freed when the execution of the function or block is completed.	• Declared outside all the functions. • Available to all functions in the program. • Memory is allocated just before the execution of the program and freed when the program stops execution.
Functions	• Declared within a function or a block of statements and defined after the calling function. • Accessible only within that function or the block.	• Declared or defined outside all other functions. • Accessible by all functions in the program

Table 3.5: Scope and life of variables and functions

Let us conclude

Modular programming is an approach to make programming simpler. C++ facilitates modularization with functions. Function is a named unit of program to perform a specific task. There are two types of functions in C++: predefined functions and user-defined functions. Predefined functions can be used in a program only if we include the header file concerned in the program. User-defined functions may need to be declared if the definition appears after the calling function. During function call, data may be transferred from calling function to the called function through arguments. Arguments may be classified as actual arguments and formal arguments. Either call by value method or call by reference method can be used to invoke functions.

Lab activity

1. Define a function to accept a number and return 1 if it is odd, 0 otherwise. Using this function write a program to display all odd numbers between 10 and 50.
2. Write a program to find the product of three integer numbers using a user defined function. Invoke the function using call by value and call by reference methods. Verify the results.
3. Write a program to find the smallest of three or two given numbers using a function (use the concept of default arguments).
4. With the help of a user-defined function find the sum of digits of a number. That is if the given number is 345 then the result should be 3+4+5 = 12.
5. Using a function, write a program to find the area of a circle.
6. Write a program to check whether a number is positive, negative or zero. Use a user defined function for checking.

Let us assess

1. What is a meant by a function in C++?
2. The built-in function to find the length of a string is _____.
3. Write down the role of header files in C++ programs.
4. When will you use void data type for function definition?
5. Distinguish between actual parameters and formal parameters.
6. Construct the function prototypes for the following functions
   a. `Total()` takes two double arguments and returns a double
   b. `Math()` takes no arguments and has no return value
7. Discuss the scope of global and local variables with examples.
8. Distinguish between Call-by-value method and Call-by-reference method used for function calls.
9. In C++, function can be invoked without specifying all its arguments. How?
10. How a local function differs from a global function?
11. Identify the built-in functions needed for the following cases
    a. To convert the letter 'c' to 'C'
    b. To check whether a given character is alphabet or not.
    c. To combine strings "comp" and "ter" to make "computer"
    d. To find the square root of 25
    e. To return the number 10 from -10

12. Look at the following functions:

```
int sum(int a,int b=0,int c=0)
{
    return (a + b + c);
}
```

a. What is the speciality of the function regarding the parameter list?

b. Give the outputs of the following function calls by explaining its working and give reason if the function call is wrong.

i. `cout << sum (1, 2, 3);` ii. `cout << sum(5, 2);`

iii. `cout << sum();` iv. `cout << sum(0);`

13. The prototype of a function is: `int fun(int, int);`

The following function calls are invalid. Give reason for each.

a. `fun("hello",4);` b. `cout<<fun();` c. `val = fun(2.5, 3.3);`

d. `cin>>fun(a, b);` e. `z=fun(3);`

14. Consider the following program and predict the output if the radius is 5. Also write the reason for the output.

```
#include<iostream>
using namespace std;
float area(float &);
int main()
{
    float r, ans;
    cout<<"Enter radius :";
    cin>>r;
    ans = area(r);
    cout<<area;
    cout<<r;
    return 0;
}
float area(float &p)
{
    float q;
    q = 3.14 * p * p;
    p++;
    return q;
}
```

15. Modify the program given in question 14 by applying call by value method to call the function and write the possible difference in the output.

4 Web Technology

Significant Learning Outcomes

After the completion of this chapter, the learner

- explains how secure communication is brought about on the web.
- describes the use of a web server and the concept of web hosting.
- differentiates static and dynamic pages.
- identifies the differences between programming languages and scripts.
- compares the different types of scripting languages.
- explains the need for cascading style sheet.
- identifies the basic HTML elements to create web pages.
- lists fundamental HTML tags and their important attributes.
- classifies the HTML tags.
- uses formatting tags appropriately in an HTML document to make the web page attractive.
- identifies the similarities and differences among the formatting tags.
- observes the use of the tags <PRE> and <DIV>.
- chooses the tag for moving objects/ contents in a document.
- uses the <FONT> tag to format the text content effectively.
- utilises the facility of comments in an HTML document.
- inserts images into HTML document using <IMG> tag.

We are all living in an era of the Internet and might have accessed it for seeking information. You might have searched for your Class XI results and viewed it from a website. People rely on different websites for various purposes like submitting online applications, viewing web contents, watching movies, banking transactions, purchase of goods through online transaction, and so on. We know that a website is a collection of web pages. A web page may contain texts, graphics, sounds, animations, and movies. A website consisting of several web pages are designed to give information about an organisation, product, service, etc. How is this website made available on the Internet? These web pages are to be stored in web servers connected to the Internet, to be made available to others. This chapter presents an overview of communication over the Internet and the role of web servers in it. The different tools and technologies that are available for developing websites are introduced here. The concept of dynamic web pages and an overview of the scripting languages used to make web pages dynamic are also discussed. Web pages are developed with the help of a language called Hyper Text Markup Language

(HTML). HTML is also known as the language of the Internet. HTML tells the browser how to display the contents on a browser window. In this chapter, we will also familiarise ourselves with the fundamentals of creating web pages using HTML.

4.1 Communication on the web

We have learned the actions that occur when a web browser (client) tries to access a website from the Internet in Chapter 9 of Class XI. First, the URL (Uniform Resource Locator) is sent to a Domain Name System (DNS) server to obtain its corresponding IP (Internet Protocol) address and then the browser connects to this server using the IP address. The web server software installed in the server computer processes this request and the web page to be displayed is then sent to the client. The client browser formats and displays this web page.

In order to communicate on the web, computers/devices need to understand each other. This is made possible by making all devices follow the same protocol, namely the Transmission Control Protocol/Internet Protocol (TCP/IP). We have discussed TCP/IP protocol and its working in detail in Chapter 8, Computer Networks of Class XI. The data to be sent is broken down into small data packets along with the address of the recipient computer by the TCP protocol. The devices called routers, route and transport these data packets to their destination computers using Internet Protocol. Figure 4.1 shows the route of a data packet from a sender to a recipient.

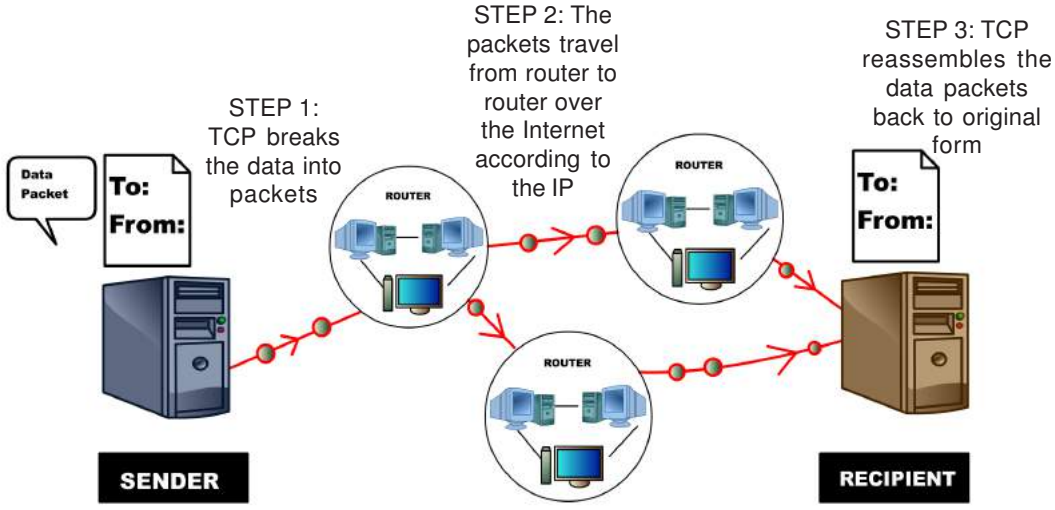


Fig. 4.1: Routing of a data packet from a sender to a recipient

In the Internet, there are several types of communication like, accessing websites, sending e-mails, etc. We have already learned in Chapter 9, Internet of Class XI that websites are accessed using HTTP (Hyper Text Transfer Protocol) and e-mail

communication happens using SMTP (Simple Mail Transfer Protocol). Both these protocols work on top of lower level protocol called Internet Protocol. Internet Protocol provides the basics of communication over Internet. The use of a single protocol - Internet Protocol - for all communication over the Internet has several advantages. The routers need not be programmed separately to handle different types of data. They do not need to know about the data they are transporting and are concerned only about the address to which the packet is to be delivered. This openness of the data part of the packet gives the freedom to design new protocols for the Internet. It is this openness and flexibility of TCP/IP that led to the development of protocols for social media websites to handle messages, content websites to handle video and audio, banking websites to handle money transfer securely, etc. This turned out to be a deciding factor for the economic success of the Internet.

Communication on the web can be categorised as (i) client (browser) to web server and (ii) web server to web server communication. Authentication and security are essential for communication over the Internet. Authentication on the web is the process of determining whether a computer/server is the computer that it claims to be. Security should be provided to communication over Internet so that the messages are not intercepted and modified by hackers.

4.1.1 Client to web server communication

Client to web server communication does not usually require authentication. But in the case of web based banking applications/e-mail services, user names and passwords are required to be sent to the server. This information cannot be sent to the server in plain text due to security reasons. The hackers may steal usernames and passwords, if it is communicated and shared as plain text. In such cases we use HTTPS (Hyper Text Transfer Protocol Secure) technology to encrypt the username and password, and then send it to the server. HTTPS works using Secure Sockets Layer (SSL) which provides a standard security technology for establishing an encrypted connection between computers on Internet. SSL provides security capabilities to HTTP. The SSL protocol not only ensures privacy, but also ensures that no other website can impersonate the user's login account nor alter the information sent.

When the browser requests for a secure web page, the server first returns its SSL certificate. The browser then verifies whether this certificate is valid by checking it with the corresponding certification authority. A certification authority certifies whether an SSL certificate given to it is a valid one. This is an assurance that the particular website is of the organisation it claims to be. Verisign Inc. is a certification

authority. If an SSL certificate is valid, the browser starts an encrypted session. During this session, the browser and the server encrypts all the transmitted data. This process is displayed in Figure 4.2(a). In India, Information Technology Act makes it mandatory for websites that provide banking transactions to use HTTPS to accept confidential information from clients. You can click on the lock symbol in the address bar and view the certificate as shown in Figure 4.2(b).

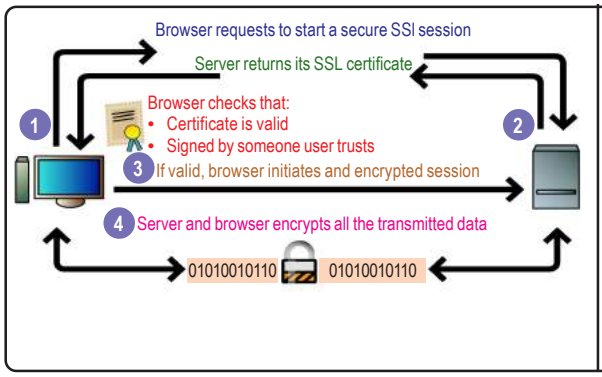


Fig. 4.2(a) : Client - server authentication process

Fig. 4.2(b): https authentication of SBI website

4.1.2 Web server to web server communication

Web server to web server communication also may be required in certain web applications. For example, the web server of an online shopping website (seller/ merchant on Internet) needs to send confidential information to a bank web server and vice versa. In such cases the web servers of the merchant and the bank are to be authenticated. Digital certificates help to verify whether the data received is from

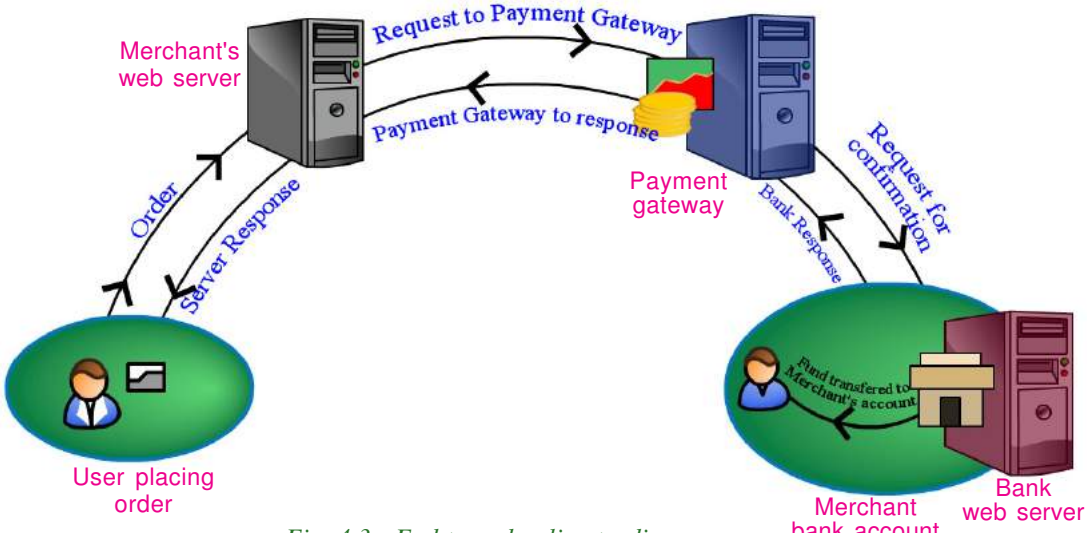


Fig. 4.3 : End to end online trading process

the actual server itself. Once the servers are authenticated, the servers communicate using encrypted data. Payment gateway is a server that acts as a bridge between merchant server and bank server and transfers money in an encrypted format whenever an online payment/money transfer is made. This process is shown in Figure 4.3.

4.2 Web server technologies

Let us see what happens internally when we visit the website www.dhsekerala.gov.in, the official website of the Directorate of Higher Secondary Education (DHSE). At first, the home page (the first page displayed when a website will be accessed) of the website will be transferred to our computer (client) from the web server of DHSE. The client is usually a computer or a mobile device which has a browser software that requests for web pages. The web server uses the web server software to store the pages of the websites and delivers it on request from the client. In the following sections, we will discuss the features of a web server and the requirements for setting up a web server.

4.2.1 Web server

The term web server is often used for referring to the server computer that hosts websites. It is also used to refer to a web server software that is installed in a server computer in order to make it a web server. In this section we will discuss the features of a web server computer and a web server software.

A web server enables us to deliver web pages or services like e-mail, blog, etc. to users on the Internet. A web server consists of a server computer that runs a server operating system and a web server software installed on it for providing services like www, e-mail, etc. over the Internet.

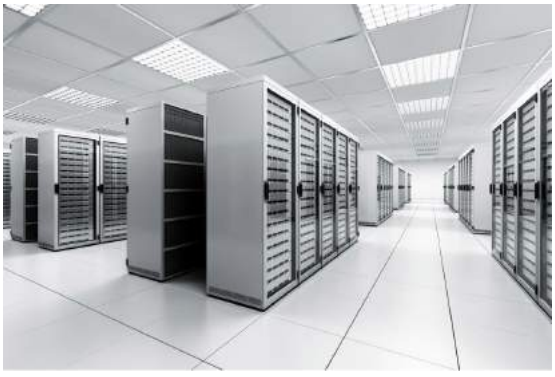


Fig. 4.4 : A data center

A web server is a powerful computer which is always switched on and connected to a high bandwidth Internet connection. This will facilitate Internet users around the world to access the websites and services hosted by it at any point of time. Depending on the requirements, a web server can have single or multiple processors, fast access RAM, high performance hard disks, Ethernet cards that supports fast communication, etc. To ensure faster Internet connectivity and redundant power supply, a web server is usually installed in a data center.

A data center is a dedicated physical location where organisations house their servers and networking systems. Data centers are used for storing, processing and serving large amounts of mission-critical data to their clients. A data center requires extensive backup power supply systems, cooling systems, high speed networking connections and security systems. A typical data center facility is shown in Figure 4.4. In a data center several servers may be mounted into special racks that offer good ventilation and easy maintenance as shown in Figure 4.5.

Fig. 4.5 : Rack mounted servers

Popular server operating systems include various Linux distributions (Redhat, openSUSE, Debian, Ubuntu, etc.), Microsoft Windows Server, FreeBSD, Oracle Solaris, etc.

After setting up a server, a web server software has to be installed in it and configured according to the given operating system. The web server software is a program that uses the client-server model and the Hypertext Transfer Protocol (HTTP) to ensure timely distribution of files stored in it to the users. These files are sent as webpages to the user on request using the web server software and can be accessed using a web browser. Some of the preferred web server packages are Apache Server, Microsoft Internet Information Server (IIS), Google Web Server (GWS) and nginx (pronounced as engine-x).

After the installation and configuration of a web server software, software packages like FTP (File Transfer Protocol), e-mail, DNS, database, etc. can be added as well. The main features provided by the web server software are the provisions to create multiple websites, configure website security, etc.

4.2.2 Software ports

We have discussed hardware ports in Chapter 2, Components of the Computer System of Class XI. Hardware ports are used to connect external devices to the computer. These devices communicate with the computer using these ports, i.e., VGA ports are used to connect monitors, PS/2 ports for keyboard/mouse, etc. Similarly, a software port is used to connect a client computer to a server to access its services like HTTP, FTP, SMTP, etc. To distinguish the ports, the software ports are given unique numbers. The purpose of software ports is to identify different services like e-mail, file transfer, etc. running on a single server computer. Each service available on the server can be set and accessed using a different port number.

Port number is a 16-bit number which when added to a computer's IP address/ URL, can be used for communicating with a particular service available on that server. A service in a website can be accessed in the format given below:

http://google.co.in:80

Here http is the protocol, google.co.in is the domain name and 80 is the port number. Some well-known ports and the associated services are listed in Table 4.1.

Default Port No.	Service
20 & 21	File Transfer Protocol (FTP)
22	Secure Shell (SSH)
25	Simple Mail Transfer Protocol (SMTP)
53	Domain Name System (DNS) service
80	Hypertext Transfer Protocol (HTTP)
110	Post Office Protocol (POP3)
443	HTTP Secure (HTTPS)

Table 4.1: Ports and their services

4.2.3 DNS servers

In Chapter 8, Computer Networks of Class XI, we learned about Domain Name System (DNS). A DNS server runs a special purpose networking software that contains a database of domain names and their IP addresses. The Domain Name System (DNS) runs on a DNS server and returns the IP address of a domain name requested by the client computer.

A Domain Name System contains several DNS server computers. The DNS servers are arranged in a hierarchy. At the top level there are 13 root servers that contain name server information for all the generic top-level domains such as .com and .org as well as country-specific domain addresses such as .in or .uk. Several copies of these root servers are placed in different locations around the globe as shown in Figure 4.6. All other DNS servers are installed in the lower level of hierarchy.

Fig. 4.6 : Root servers around the globe

Let us see how the DNS searches and locates the IP address of a domain name. Suppose we are visiting the website of the Police department of the Government of Kerala. Let us type the domain name for the police department, namely www.keralapolice.org in our browser. The following steps illustrate how the DNS resolves the IP address. See Figure 4.7.

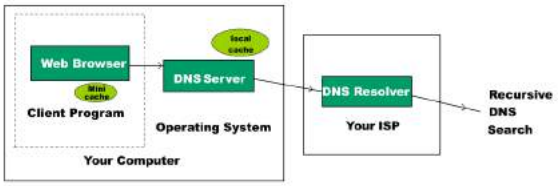


Fig. 4.7 : DNS search

1. All browsers store the IP addresses of the recently visited websites in its cache. Therefore, the browser first searches its local memory (mini cache) to see whether it has the IP address of this domain. If found, the browser uses it.
2. If it is not found in the browser cache, it checks the operating system's local cache for the IP address.
3. If it is not found there, it searches the DNS server of the local ISP.
4. In the absence of the domain name in the ISP's DNS server, the ISP's DNS server initiates a recursive search starting from the root server till it receives the IP address.
5. The ISP's DNS server returns this IP address to the browser.
6. The browser connects to the web server using the IP address of www.keralapolice.org and the webpage is displayed in the browser window. If the IP address is not found, it returns the message 'Server not found' in the browser window.

The 13 root servers around the globe manage the entire database of domain names and their corresponding IP addresses. Each of these root servers are a network of servers installed in many countries around the world. They are named as A, B, C, D, E, F, G, H, I, J, K, L and M. These servers are maintained by various agencies like ICANN, NASA (National Aeronautics and Space Administration), University of Maryland, VeriSign Inc., etc. ICANN's (Internet Corporation for Assigned Names and Numbers) Root Server System Advisory Council is comprised of the organisations that manage the root servers. They are responsible for advising ICANN on matters relating to the operation, administration, security and integrity of the Internet's Root Server System. In India, NIXI (National Internet Exchange of India) has sponsored three root servers at Mumbai (I Root), Delhi (K Root) and Chennai (F Root).

In large organisations like educational institutions, government departments, software firms, etc. where there are hundreds of computers or devices connected to the Internet, a local DNS server is hosted inside the intranet of the organisation.

This local DNS server contains a list of domain names and their IP addresses which the users in the organisation regularly access. This list can be updated periodically to include new domain names. Whenever a computer in the intranet tries to access a website in the Internet, it first searches for the domain name in the local DNS server and finds the IP address. This speeds up the Internet access in the organisation. Only if the domain name is not found in the local DNS server, it searches the DNS of ISP.

Instead of directing the computer to search for IP address in the ISP's DNS server, we can direct it to search the public DNS operated by Google. Google Public DNS is a free Domain Name System (DNS) resolution service that can be used as an alternative to our current DNS provider. The IP addresses for Google's public DNS are 8.8.8.8 and 8.8.4.4. We can configure our network settings to direct to Google's public DNS using either of the IP addresses.

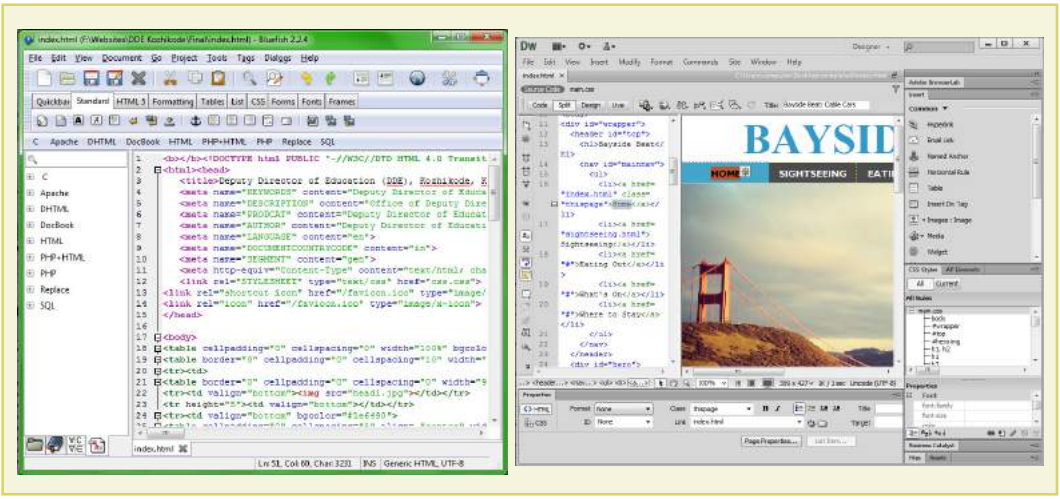
Know your progress

1. Name a protocol that works on top of the Internet Protocol (IP).
2. Expand HTTPS.
3. What are the advantages of installing web servers in data centers?
4. State whether true or false.
   a. The web server software works based on a client-server model.
   b. The web server consists of server operating system and web server software.
5. The number of bits in a software port number is _____.
   a. 8 b. 16 c. 32 d. 64
6. A Domain Name System returns the _____ of a domain name.

4.3 Web designing

The first step in setting up a website is planning the web pages for the website. Suppose we are going to develop a website for our school. After deciding the pages and the links, we need to design these web pages. Any text editor can be used to create a home page, a page for displaying the courses in the school, facilities available, contact address, etc. and link them using a menu to form an attractive website.

There are many web designing softwares available. You can also employ the features available in any of the web designing tools that helps you to create a web page with the ease of preparing a document in a word processor. These software also provide features to design web pages and link them together to form a website. Facilities to transfer files to the server using FTP protocol is also built into such software. Popular web designing softwares are Bluefish, Bootstrap, Adobe Dreamweaver, Microsoft Expression Web, etc. Figure 4.8 shows the Integrated Development Environment (IDE) of popular web designing softwares.



Bluefish *Dreamweaver*

Fig. 4.8 : IDE of web designing software

You have already learned the basics of creating web pages in high school classes. HTML consists of tags and its attributes used to format web pages. In this chapter we will also get familiar with the different HTML tags and attributes.

4.4 Static and dynamic web pages

You might have noticed that the web pages in the website of some small business/ organisations, school website, etc. remain the same whenever you visit it. These websites are called static websites. There are some other websites like the website that displays your mark in SSLC or Higher Secondary Examination (HSE), that changes when different register numbers are given to them. They are called dynamic web pages.

Static web pages are web pages that remain the same all the time until their code is changed manually. Initially, web pages were created with HTML only and such web pages are static web pages. Later, the emergence of scripting languages like JavaScript, VBScript, etc. brought animations into the web page. The colour and style of a portion of the web page changes when the mouse is kept over it, images are loaded

and displayed one after another in a sequence - all these are designed using scripting languages. The web pages that contain all these features are also classified as static web pages.

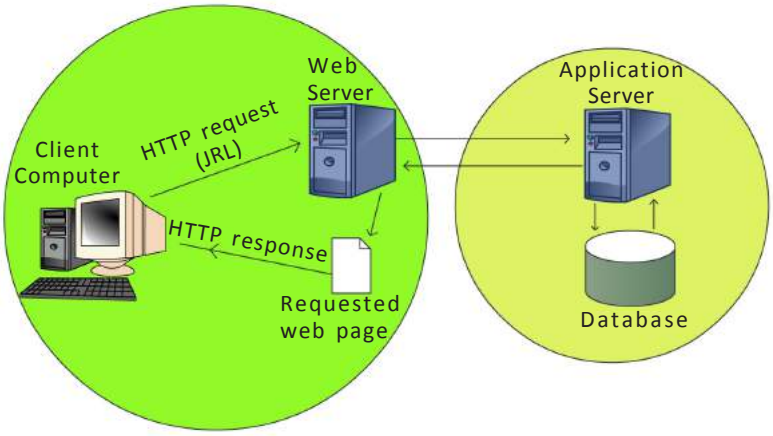


Fig. 4.9 : Static and dynamic web page requests

The web pages that contain server side code which creates a web page each time it is accessed are called dynamic web pages. Program code that runs on the server is called server side code. Dynamic web pages are executed using a server side application program that is installed on a web server. The script in the web page is executed on the web server and the resulting HTML page is sent to the client browser. In most cases data is accessed from databases to create web pages. The web pages that display SSLC, HSE results, bus, train and flight booking, banking/financial transactions, etc. display dynamic content, and are considered as dynamic web pages. Technologies like PHP, ASP, JSP, etc. are used to develop dynamic web pages. The working of static and dynamic web pages is represented using Figure 4.9. A comparison of static and dynamic web pages is given in Table 4.2.

Static web page	Dynamic web page
The content and layout of a web page is fixed.	The content and layout may change during run time.
Static web pages never use databases.	Database is used to generate dynamic content through queries.
Static web pages directly run on the browser and do not require any server side application program.	Dynamic web page runs on the server side application program and displays the results.
Static web pages are easy to develop.	Dynamic web page development requires programming skills.

Table 4.2: Comparison of static and dynamic web pages

4.5 Scripts

Scripts are program codes written inside HTML pages. They are written using a text editor like notepad. Scripting languages like JavaScript, VB script, PHP, Perl, etc. are used to create dynamic web pages.

Traditional programming languages are set of instructions carried out by the computer hardware with the help of an operating system, whereas scripting languages are interpreted by a web browser or by a web server software. Today most of the standalone programs are being replaced by web based programs. Earlier, the software used in banks were installed in the branches of the bank itself. Almost all banks have their banking software available on the bank's web server and the bank employees access them using the Internet. Web based software like banking software, higher secondary admission software, etc. use scripting languages for their development.

In an HTML page, a script is written inside **<SCRIPT>** and **</SCRIPT>** tags. `<SCRIPT>` is used to embed or refer to an executable script within an HTML document. It has the attributes **Type** and **Src**. `Type` attribute is used to identify the scripting language code embedded within the script tag. We will see more about HTML tags and attributes later in this chapter.

`<SCRIPT Type="text/javascript">` is used to insert JavaScript code in the HTML code.

Scripts can be written as an external file and can be linked to an HTML file. `Src` attribute is used to specify the URL of an external file containing scripting code to be linked to a web page.

`<SCRIPT Type="text/javascript" Src="scriptcode.js">` is used to insert JavaScript code inside the file `scriptcode.js` into an HTML file.

4.5.1 Types of scripting languages

Let us consider a login page of a website in the Internet. Usually the page will prompt the user to enter the user name and password. User will click the 'Login' button after entering the user name and password. What will happen if the user clicks the 'Login' button without entering the user name? Naturally, the computer will tell the user that he/she has not entered the user name. Remember that the web page we are viewing is controlled by two computers - the client computer where the web page is viewed and the server computer where the web page comes from.

Where will we check whether the user has entered a user name or not - in the client computer or in the server computer? The server computer is thousand times busier

than the client computer, because a large number of people may be visiting the same web site and the server is the only one computer to handle all those requests. Therefore, all the tasks that can be done at the client side must be done at the client side itself. This will reduce the workload of the server.

It should also be noted that, if this type of checking is done at the server, when the user clicks the 'Login' button, the entered data has to be sent to the server from the client through the Internet. The data has to travel a long distance through the Internet to reach the server. When the data reaches the server, it will be placed in a queue because a large number of clients may be sending the same request to the server. The sent data will wait in the queue till it gets the chance to be processed by the server. The server will check whether any user name or password is entered by the client or not. If not, it will send a message back to the client specifying that the user name is not given as displayed in Figure 4.10. Again, the message has to travel a long distance back from the server to the client through the Internet. In short, if the user clicks the submit button, without entering the user name, he/she has to wait a few seconds till he/she gets the message that he/she has not given the user name. Besides this, the data has to travel from client to server and back from server to client, which unnecessarily makes network traffic busy.

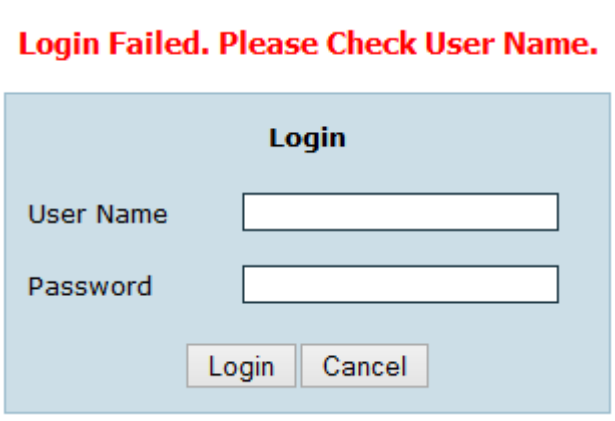


Fig. 4.10 : Response from the server

If this checking is done at the client side itself, when the user clicks the submit button, the script code in the client side can check whether the user has given some entry as the user name and password. If not, a message can be displayed. During this process, the data does not travel across the Internet to the server, nor does it disturb the server for this simple task. When the user clicks the submit button, the user gets the message that he has not given the user name, within a second as shown in Figure 4.11. It also does not engage network resources unnecessarily.

Now let us consider another situation. Suppose a user enters a wrong user name and password. The client computer may be able to check whether there is any entry with such a user name and password. However it cannot check whether the user name and password are correct or not. This is because only the server computer has the details of all user names and corresponding passwords. Therefore, whether

the user name and password match each other, can be checked at the server side only. This is a situation where you have to use the server side scripting for the validation. When you need to perform some kind of validation of data that makes use of information from the server, it must be done at the server side itself.

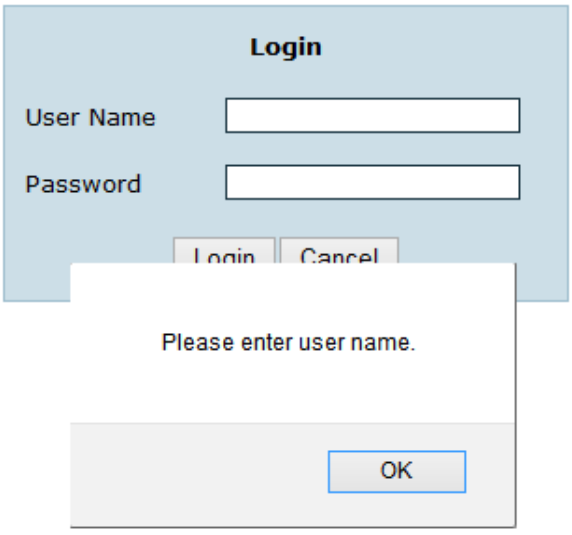


Fig. 4.11 : Response from client browser

Isn't it now clear that scripting languages are classified into client side scripts and server side scripts? Client side scripting is used to perform any task at the client side and is executed in the browser. Scripts that are executed in the server are called server side scripts. The output produced after execution of the server side scripts is in the form of an HTML page which is sent to the client.

A. Client side scripting

In client side scripting, the script code for validation is downloaded along with the HTML code to our browser. When we click the submit/save button this client side script is executed in our browser. If there is an error, it displays the message. It will send the data to the web server only if the validations are correct.

Since the script code is executed on the client browser, it provides a quicker response to users. This type of scripting will allow the client browser to share some of the burden on the web server while running a web application. The disadvantage of client side scripting is that there are browsers that do not support scripts. In some cases users may turn off the execution of scripts in the browser. In such cases client side scripting may not work.

Popular client side scripting technologies are JavaScript and VB script. Client side scripting is mostly used for validations and also for performing simple calculations at the client side before sending the data to the web server.

B. Server side scripting

We discussed dynamic web pages in the previous section. Server side scripts are used to create dynamic web pages. Let us discuss another example for server side scripting. Consider the website that displays the results of SSLC examination. When we enter the SSLC register number of a student, the website displays the scores of

that particular student. This is the same for each student who has appeared for the SSLC examination. We know that it is not practical to design a web page for each of the several lakhs of students who have written the SSLC examination. If so, how is this done? The results of these several lakhs of students are stored in a database in the web server. Server side scripts are used to access the result of a particular student from the database whose register number is entered by the user. The server side script then uses this result to create a simple HTML web page. This web page is then sent to the client browser and the browser displays it. This way the server side script creates a web page for each student who has appeared for SSLC examination as shown in Figure 4.12. The rapid growth of web based applications has increased the use of server side scripting.

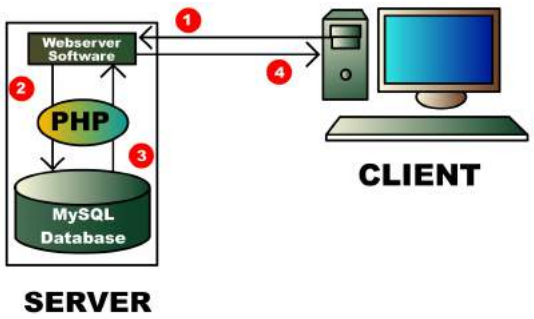


Fig. 4.12 : Working of server side scripts

Server side scripting is a technology in which the web page containing server side scripts requested by the user is executed in the server and the result, which is an HTML code, is sent to the client browser. Server side scripting creates web pages dynamically. Since the scripts are executed at the server, the type and version of the browser or operating system on the client's computer does not affect its execution. Since the scripts are executed on the server, it consumes server resources. Popular server side scripting languages are Perl, PHP, ASP, JSP, etc.

A comparison of the classifications of scripting languages is given in Table 4.3.

Client side scripting	Server side scripting
Script is copied to the client browser	Script remains in the web server
Script is executed in the client browser	Script is executed in the web server and the web page produced is returned to the client browser
Client side scripts are mainly used for validation of data at the client.	Server side scripts are usually used to connect to databases and return data from the web server
Users can block client side scripting	Server side scripting cannot be blocked by a user
The type and version of the web browser affects the working of a client side script	The features of the web browser does not affect the working of server side script

Table 4.3 : Comparison of client side and server side scripting

We have seen that the client side scripts are mainly used for validations at the user's browser and that it reduces the load on the server and network traffic. Therefore, the entire validation checking scripts are moved to the client side. Now the data sent to the web server is free from all errors and can be directly saved to the database. But if the client's browser does not support scripts or the user has turned off the scripting in the browser for security reasons, the data will be sent to the server without validation. This causes invalid data to be stored in the database. In order to protect the validity of the data saved in the database, it is better to have a validation check at the server side also.

4.5.2 Scripting languages

We have learned the different classifications of scripting languages. Let us now discuss the features of some of the popular scripting languages.

A. JavaScript

JavaScript is a client side scripting language used to make web pages interactive. JavaScript was developed by Brendan Eich (Figure 4.13) while he was working for Netscape Communications Corporation. On the client side, JavaScript is implemented as an interpreted language. Any text editor such as Geany IDE, Notepad, etc. can be used to write JavaScript code. It is popular as a client side scripting tool and works in almost every web browser. JavaScript can be inserted inside HTML code or can be written as an external file and then included inside HTML file. If a JavaScript code is written as an external file, it is common to use the extension .js. This identifies the file as a JavaScript file. JavaScript is popular as a tool for validation of forms in the client side. It can also be used for performing simple calculations and also for bringing animations to a web page. We will discuss JavaScript in detail in Chapter 6.

Fig. 4.13: Brendan Eich (1961 -)

The popularity of JavaScript has led to more developments in client side scripting. While applying online for Higher Secondary Plus One admissions, immediately after entering your SSLC register number, you might have noticed that your name, date of birth and other details appear in the text boxes below. This is done without reloading the entire web page. The data is taken from the server and filled in the text boxes without refreshing the web page. Ajax is the technology used here. Ajax improves the interactivity of the browsers. Ajax is Asynchronous JavaScript and Extensible Markup Language (XML). XML is a markup language which helps users to create new tags. After implementing Ajax on a website, it does not require the entire page to be reloaded for displaying dynamic content on web pages. Ajax

allows web pages to be updated by exchanging small amounts of data between the client and the server behind the scenes. This means that it is possible to update parts of a web page, without reloading the entire web page. However, since Ajax relies more on JavaScript, if the browser is not capable of handling JavaScript or the user has turned off JavaScript functionality in the browser, the Ajax application will not work.

B. VB Script

VBScript is a scripting language developed by Microsoft Corporation based on the popular programming language Visual Basic. VBScript was developed to use either as a client side scripting language for the Microsoft Internet Explorer or as a server side scripting language with the Microsoft Internet Information Server (IIS). Unfortunately, browsers other than Internet Explorer may not be able to correctly interpret and display the VBScript code. Therefore, it is less popular as a client side scripting tool. Since Windows operating system is popular as a server based operating system, VBScript is popular for server side scripting. With the introduction of .NET framework - a library of usable program code, Microsoft has taken the decision to incorporate VBScript as a part of ASP.NET in .NET framework.

C. PHP

PHP stands for 'PHP: Hypertext Preprocessor'. PHP is an open source general-purpose scripting language that is suited for web development and can be embedded into HTML code. It is a server side scripting tool and its code is similar to Java, C and Perl. The main objective of PHP is to develop dynamic web pages at ease. PHP was originally created by Rasmus Lerdorf (Figure 4.14) in 1994 but it is now developed by The PHP Group. The web page files that contain PHP code have the extension .php.

Fig. 4.14: Rasmus Lerdorf (1968 -)

PHP code is inserted inside HTML code and when the user requests for a PHP web page, it is interpreted and executed on the web server. To process the PHP code on the web server, a PHP interpreter has to be installed on the web server. After execution of the PHP code in the web server, an HTML page is created, which is sent to the client browser. One of the strongest and most significant features in PHP is its support for database programming. The most common database used with PHP is MySQL. PHP interpreter is available for all operating systems. Linux platforms commonly use LAMP (Linux, Apache, MySQL and PHP) server software which is freely downloadable. LAMP uses Linux as the server operating system, Apache as the web server, MySQL as the database and PHP for server side scripting. Windows operating systems use WAMP server software which is also available for free download.

D. Active Server Pages

Microsoft Active Server Pages (ASP) is a server-side scripting environment that can be used to create and run interactive web applications. ASP contains HTML and a scripting language code. The scripting language can be VBScript or JavaScript. ASP files have the extension .asp. These files are compiled using a feature built in Microsoft's web server software, Internet Information Server (IIS). ASP files are executed only on Windows operating systems. After execution at the server, the resultant HTML web page is sent to the client browser. It is a very powerful scripting language that provides support to a variety of databases. The introduction of ASP.NET stopped Microsoft from releasing further versions of ASP. ASP.NET provides features like reduced coding, availability of a variety of buttons, text boxes etc. which help in creating web based applications.

E. Java Server Pages

Java Server Pages (JSP) technology provides a simple and fast way to create dynamic web content. It is a server side scripting language that was developed by Sun Microsystems in 1999. JSP is similar to PHP, but uses Java as programming language. JSP files have the extension .jsp. To run JSP, Apache Tomcat web server is required. The JSP code consisting of HTML and Java is executed on the web server and the resulting HTML code is sent to the browser. JSP is an integral part of Java 2 Platform Enterprise Edition (J2EE) which is used for developing and running large scale web based softwares.

4.6 Cascading Style Sheet

Cascading Style Sheets (CSS) is a style sheet language used for describing the formatting of a document written in HTML. Using CSS, we can control the colour of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, borders and its colours, what background images or colours are used, as well as a variety of other effects in a web page. A CSS file allows us to separate HTML content from its style. CSS can be implemented in three different ways - inline, embedded and linked.

- In inline style, the CSS style is applied to each tag separately using the style attribute in the body part of the web page.
- Embedded CSS codes are placed within the **<HEAD>** part of the web page.
- Linked CSS implementation is done using an external file with the file extension .css that contains only CSS code and is linked with the web page.

The advantage of CSS is that we can reuse the same code for all the pages. If the CSS styles for a website is implemented as a linked external file, then the modification

of a style, modifies the way the tags are presented in all the web pages of the website. Since CSS styles are written in a common place, it separates CSS and HTML, which makes it easy for maintenance. Moreover, the tags in web pages are well organised with the style specifications and therefore it is easy to understand. This also reduces the size of the web page thereby providing faster downloads for web pages.

CSS allows adapting the presentation of a web page to devices with different screen sizes such as desktop monitors, tablets or mobiles. It is considered that CSS along with JavaScript will be used in the next version of HTML called HTML5, to bring animations and interaction to web pages.

Know your progress

1. The web pages that remain the same until their code is changed manually are called _____.
2. Name two technologies that can be used to develop dynamic web pages.
3. The tag used to embed scripts is _____.
4. Write any one of the uses of client side scripting.
5. A JavaScript file has the extension _____.
6. What is the advantage of using Ajax?
7. Classify the following scripting languages into client side and server side.

   Javascript, PHP, ASP, VBScript
8. .asp files are compiled using the web server software _____.
9. List the different ways of implementing CSS.

4.7 Basic concepts of HTML documents

HTML is the most widely used language to write web pages. Every web page is actually an HTML file. Each HTML file is a plain text that defines a set of commands for creating hypertext documents. These commands are known as ***HTML tags***. While using these tags, some keywords may be attached to them, which make the instruction more specific. These words are known as attributes. Therefore, an ***HTML document*** is made up of tags and attributes which work together to decide how the contents of the web page have to be displayed on the browser. Actually, the study of HTML means the study of tags and their attributes. Before going into the details of tags and attributes, let us have a look at the basic structure of an HTML document.

4.7.1 Basic structure of an HTML document

The basic structure of an HTML document is shown in Example 4.1.

Example 4.1: A sample HTML document to illustrate the structure of a web page

```
<HTML>
    <HEAD>
      <TITLE> This is the title of web page </TITLE>
    </HEAD>
    <BODY>
      Hello, Welcome to the world of web pages!
    </BODY>
</HTML>
```

You can see some of the words in the upper case within a pair of angle brackets **<** and **>**. These are HTML tags. It is not necessary that tags be written in the upper case. HTML is not case sensitive. We can use either the upper or lower case or even a mix of the two. In this book, we follow the style of using the upper case for HTML tags and sentence case (i.e., the first letter is capital) for attributes to distinguish them from other words or text.

As shown in Example 4.1, all HTML pages begin with the tag **<HTML>** and end with tag **</HTML>**. There are mainly two sections in an HTML document namely head section and body section. The **<HEAD>** tag is used to define the head section. The head section contains the information about the document, including the title of the web page. The **<TITLE>** tag is used to define the title of the page, which will be displayed on the title bar of the browser window. The **<BODY>** tag is used to define the body section. The body section contains the contents to be displayed in the web page. If we open this document in a web browser, it will appear as shown in Figure 4.15.

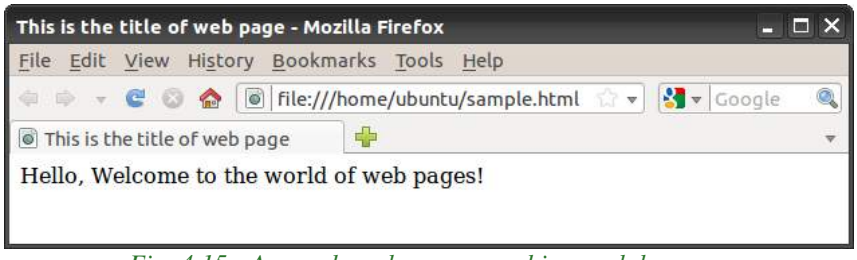


Fig. 4.15 : A sample web page opend in a web browser

Versions in HTML: HTML was created by Tim Berners Lee in late 1991 but "HTML 2.0" was the first standard HTML specification which was published in 1995. HTML 4.01 was a major version of HTML and it was published in late 1999. The latest version, HTML 5 was released in 2012. But it is still being modified for adding more multimedia integration.

4.7.2 Tags in HTML document

As mentioned earlier, tags are the commands used in the HTML document that tell web browsers how to format and organise our web pages to show the contents. Every tag consists of a tag name enclosed between the angle brackets '**<**' and '**>**'. HTML tags are not case sensitive. Therefore the tags `<HTML>`, `<html>`, `<Html>`, `<HtmL>`, etc. have the same meaning.

4.7.3 Container tags and empty tags

Most tags are used in pairs - an opening tag and a closing tag. For example `<HTML>` is the opening tag and `</HTML>` is the closing tag. Note that the closing tag has the same text as the opening tag, but has an additional forward slash (**/**) character after the first angle bracket. Tags that require opening tag as well as closing tag are known as container tags. Container tags are applicable for a section. The opening tag is given at the beginning of a section and closing tag is placed at the end of the section. For example, `<HTML>` and `</HTML>` forms the opening and closing tag pairs for an HTML document.

Some tags are given an exemption to this rule, and these tags do not require closing tag. Such tags are known as empty tags. This kind of tag does not span over a section. The tags `<BR>`, `<HR>`, `<IMG>`, etc. are examples of empty tags. We will see these tags in the forthcoming section of this chapter.

4.7.4 Attributes of tags

Certain parameters are frequently included within the opening tag to provide additional information such as colour, measurement, location, alignment or other appearances to the web browser. These parameters are called attributes. Most of the attributes require a value. In HTML, the value can be given in single quotes or double quotes (i.e., `attribute='value'` or `attribute="value"`). Each tag has a standard set of attributes and we can use them as per the requirement. If an attribute is used, normally it appears after the tag name separated by a space. If more than one attribute is used, their order of appearance is not important.

For example, to change the background colour of the web page to yellow, we can write the code as `<BODY Bgcolor = "Yellow">`. Here, Bgcolor is the attribute and Yellow is the value of this attribute. The other attributes of `<BODY>` tag and other tags will be discussed in the forthcoming section.

4.7.5 HTML Elements

A pair of tags and the content enclosed between these tags are known as an element. Figure 4.16 shows that the Body element contains the opening tag `<BODY>` with

attributes if any, the closing tag </BODY>, and the contents in between these two tags.

Fig. 4.16 : An HTML element

The basic structure of an HTML document contains four sets of HTML tags.

They are:

```
<HTML>    </HTML>
<HEAD>    </HEAD>
<TITLE>   </TITLE>
<BODY>    </BODY>
```

4.8 Creating an HTML document

Now, let us create a simple web page using the HTML code given in Example 4.1. Text editors like Geany, Gedit, TextPad, Notepad, Notepad++, etc. can be used to create HTML documents. The file is to be saved with a name with an extension **.html** or **.htm** (for example, sample.html). Figure 4.17 shows an HTML code in Geany editor, saved as **sample.html**.

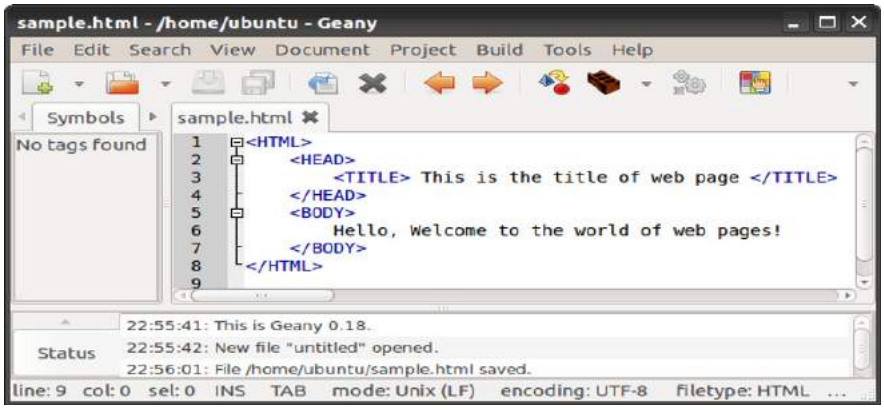


Fig. 4.17: HTML code in Geany Editor

Viewing an HTML document in a Browser

Once the HTML document is prepared, it can be viewed in a browser by opening the document with the browser. There are many browsers like Mozilla Firefox, Google Chrome, Internet Explorer, Netscape navigator, etc. Screen shots given in this book, are obtained by opening the web page using Mozilla Firefox. The output of the HTML code created using Geany can also be obtained by clicking the Execute button in the tool bar of Geany.

4.9 Essential HTML tags

Let us have a detailed discussion on the essential tags required to create web pages. The tags, their use, associated attributes and their values, and their appearance in the browser window will be illustrated in this section.

4.9.1 <HTML> - Starting an HTML page

The entire HTML document is bounded by a pair of <HTML> and </HTML> tags. The <HTML> tag identifies the document as an HTML document. In general <HTML> is always the first tag in an HTML page and the </HTML> is the last tag. Everything else in the web page is in between these two tags. That is, the Head section and the Body section lie inside the <HTML> and </HTML> tags. It is a container tag pair. The main attributes of the <HTML> tag are **Dir** and **Lang**.

Dir

The Dir attribute of <HTML> tag specifies the direction of the text to be displayed on the web page. This attribute can have values either **ltr** (left-to-right) or **rtl** (right-to-left). By default, the value of this attribute is ltr. The value rtl is used when languages like Arabic, Hebrew, Chinese etc. are used for content presentation. For example, <HTML Dir = "rtl"> specifies that the document is to be read from right-to-left.

Sl. No.	Code	Language
1	En	English
2	Fr	French
3	De	German
4	It	Italian
5	El	Greak
6	Es	Spanish
7	Ar	Arabic
8	Ja	Japanese
9	Hi	Hindi
10	Ru	Russian

Table 4.3 : Some common language codes

Lang

The **Lang** attribute of <HTML> tag specifies the language we have generally used within the document. The value "en" is used for English language and "it" is used to specify Italian language.

For example, the code <HTML Lang = "ar"> specifies that the language used in the HTML document is Arabic language. Some common language codes used with Lang attribute are given in Table 4.3.

4.9.2 <HEAD> - Creating head

It contains the head of an HTML document, which holds information about the document such as its title, scripts used, style definitions, etc. The tag pair <HEAD> and </HEAD> declares the head section. It is also a container tag pair.

4.9.3 <TITLE> - Creating a title

It is a container tag pair that contains the title of the HTML document, which will appear in the web browser's title bar. The search engine uses the Title to identify the page. The tag pair <TITLE> and </TITLE> is used inside the tag pair <HEAD> and </HEAD> to mention the document title.

4.9.4 <BODY> - Creating a body

The body tag pair <BODY> and </BODY> specifies the document body section. This section contains the content to be displayed in the browser window. Hence, all other tags, which define the document content are given in the body section. Before discussing these tags, let us discuss various attributes of <BODY> tag.

Background

This attribute sets an image as background for the documents body. This attribute of <BODY> tag makes the page more attractive. The general format is:

```
<BODY Background = "URL of the picture">
```

The HTML code given in Example 4.2 shows the sky as the background image of a web page.

Example 4.2: To set an image as background for a web page

```
<HTML>
<HEAD>
   <TITLE> Background Image </TITLE>
</HEAD >
<BODY Background = "Sky.jpg">
   Hello, Welcome to the world of Web Pages!........
</BODY>
</HTML>
```

Here, the HTML code in Example 4.1 is modified by prioviding an attribute Background with the value "Sky.jpg" in the <BODY> tag as <BODY Background = "Sky.jpg">. Before opening the page, we have to place the image file in the current working directory. The web page is displayed as shown in Figure 4.18.

Fig. 4.18 : An image as background

Bgcolor

This attribute specifies a colour for the background of the document body. For example, `<BODY Bgcolor = "grey">` will display the background in grey colour.

The value of `Bgcolor` attribute can be given in two ways.

- ***Color_name*** - specifies the background colour with a colour name (like "red", "grey" etc.)
- ***Hex_number*** - specifies the background colour with a hexadecimal code (like "#ff6080", "#303030" etc.). Each hexadecimal code will be preceded with a hash sign #.

  The six digit number and letter combinations represent colours by giving their RGB (Red, Green, Blue) value. Of the six digits, the first two digits represent the amount of red, the second two digits represent the amount of green, and the last two digits represent the amount of blue as a hexadecimal value in the range 00 - FF. For example, #000000 is black, #FF0000 is bright red, #00FF00 is bright green, and #FFFFFF is white (fully saturated with all the three colours). We can try various colour combinations according to our choice of hex number. Table 4.4 shows a few colours with their Names and Hex values.

Colour	Colour Name	Colour HEX
	Black	#000000
	Red	#FF0000
	Green	#00FF00
	Blue	#0000FF
	Yellow	#FFFF00
	Aqua	#00FFFF
	Grey	#C0C0C0
	White	#FFFFFF

Table 4.4: List of colours with their Name and Hexadecimal value

Text

This attribute specifies the colour of the text content in the page. By default the browser displays the text in black colour on a white/grey background. We have already discussed how to change the background colour using `Bgcolor` attribute. Simillarly the colour of the text can be changed using the attribute `Text`. For example, `<BODY Text = "yellow">` shows the text in yellow colour. Like `Bgcolor`, the value of `Text` attribute can be given as colour name or hexadecimal code. For example, `Text = "Blue"` or `Text = "#00FFDD"` etc.

Link, Alink and Vlink

A hyperlink is an element, a text or an image that we can click on, and jump into another document or another section of the same document. A hyperlink points to a whole document or to a specific element within a document. We will discuss hyperlink in detail later in this chapter.

Link: This attribute specifies the colour of the hyperlinks that are not visited by the viewer. The default colour for `Link` attribute is blue.

Alink: It specifies the colour of the active hyperlink. The link remains active only for the moment the mouse is clicked on it. Hence at the time of selection the colour of the link will be changed to `Alink` value. The default `Alink` colour is green.

Vlink: It specifies the colour of the hyperlink which is already visited by the viewer. The default colour for `Vlink` is purple.

Leftmargin and Topmargin

The margin refers to the blank area left from the edge of the page. `Leftmargin` attribute is used to leave some blank area on the left side of the document and `Topmargin` refers to the blank area at the top edge of the document window. The value is specified in pixels.

For example, `<BODY Leftmargin = "60" Topmargin = "70">` will make the body text indent 60 pixels away from left edge of the page and 70 pixels away from the top edge of the page.

The code in Example 4.3 is an illustration to the attributes `Bgcolor`, `Text`, `Topmargin`, and `Leftmargin` of `<BODY>` tag. Figure 4.19 shows the corresponding web page.

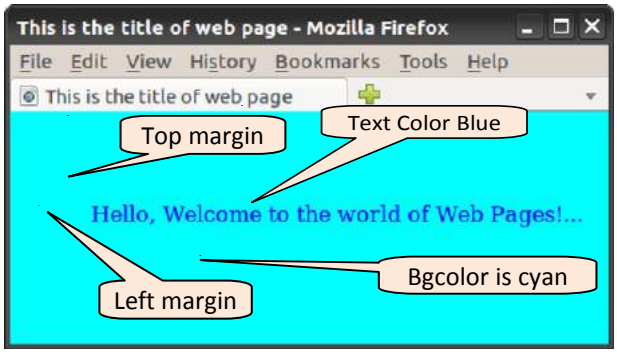


Fig. 4.19: Use of attributes with BODY tag

Example 4.3: To set a colour in the background of a web page

```
<HTML>
<HEAD>
     <TITLE> This is the title of web page </TITLE>
</HEAD>
<BODY Bgcolor= "cyan" Text= " Blue"
```

```
      Topmargin= "70" Leftmargin= "60">
      Hello, Welcome to the world of Web Pages!........
</BODY>
</HTML>
```

Know your progress

1. HTML stands for _____.
2. What is a container tag?
3. The type of tag that requires only a starting tag but not an ending tag is called _____.
4. State true or false
   a. Tags are case sensitive.
   b. `Bgcolor` is an attribute of `<BODY>` tag.
   c. `<TITLE>` tag is an empty tag.
   d. `Dir` is an attribute of `<HEAD>` tag.
5. Name the attributes of `<HTML>` tag.
6. What is the use of attributes in a tag?
7. List the different attributes of `<BODY>` tag.

Let us do

Create an HTML document to display the name of your school with an image of the school as background of the web page. Then modify the page by changing the colour of the text and background.

4.10 Some common tags

We have discussed the basic tags and their attributes needed for an HTML document. There are several other tags, with which web page contents can be made more attractive. Some of these tags are used for formatting the text contents in the body section of the HTML document and therefore they are called formatting tags. Now let us see some of them, which are essential for the layout of the body content.

4.10.1 <H1>, <H2>, <H3>, <H4>, <H5> and <H6> - Heading tags

A heading is a word, phrase, or sentence given at the beginning of a written passage that explains what it is about. Headings are typically displayed in larger and/or bolder fonts than normal body texts. HTML has six levels of headings from **<H1>**

to **<H6>**. Here <H1> creates the biggest text and <H6> the smallest. While displaying any heading, browser adds one line before and one line after that heading. The main attribute of this tag is **Align** and the possible values are,

Left : Text is aligned to the left margin.

Right : Text is aligned to the right margin.

Center : Text is aligned to the centre of the page.

Example 4.4 shows different heading types and alignments and Figure 4.20 shows the corresponding web page.

Example 4.4: To illustrate different heading styles

```
<HTML>
<HEAD>
   <TITLE> Heading Tags </TITLE>
</HEAD >
<BODY Bgcolor= "#FFEFD5">
   <H1 Align= "left"> This is a Heading type 1 </H1>
   <H2 Align= "center"> This is a Heading type 2 </H2>
     <H3 Align= "right"> This is a Heading type 3 </H3>
     <H4> This is a Heading type 4 </H4>
     <H5> This is a Heading type 5 </H5>
     <H6> This is a Heading type 6 </H6>
</BODY>
</HTML>
```

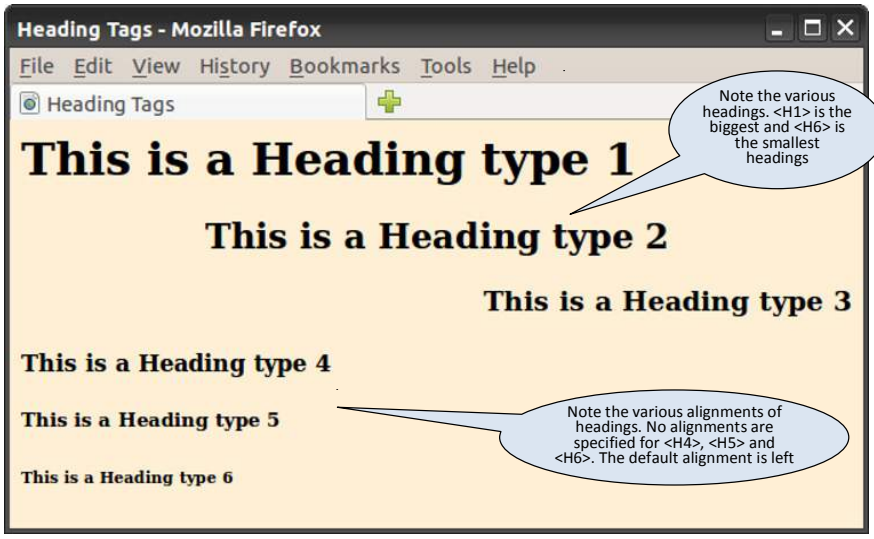


Fig. 4.20 : Setting different heading tags with different alignments

4.10.2 <P> tag - Creating paragraphs

The **<P>** tag enables us to organise the text within the <BODY> tag into paragraphs. It indicates a new paragraph and instructs the browser to add a blank line before the paragraph. Paragraphs in HTML acts much like the paragraphs in any word processor. The paragraph element begins with the <P> tag and ends with </P> tag. The **Align** attribute sets the alignment of the text in the paragraph with the values left, right, center or justify.

The code given in Example 4.5 shows how paragraphs are designed with different alignments. Figure 4.21 shows the resultant web page.

Example 4.5: To design paragraphs with different alignments

```
<HTML>
<HEAD>
      <TITLE> Paragraph Tags </TITLE>
</HEAD>
<BODY>
      <P>
      This paragraph          contains
      a lot of lines in the source    code,
            but the browser ignores it.
      </P>
      <P Align= "right">
      This paragraph       contains a lot of spaces
      in the source code,but the browser ignores it.
      </P>
      <P Align= "justify">
      The number of lines in a paragraph depends on
      the size of the browser window.
      If you resize the browser window, the number of lines
            in this paragraph will change.
      When we align a paragraph as justify, the space between
      the words in a line will be adjusted
      so that the lines will be both left and right aligned
      at the same time.
    </P>
</BODY>
</HTML>
```

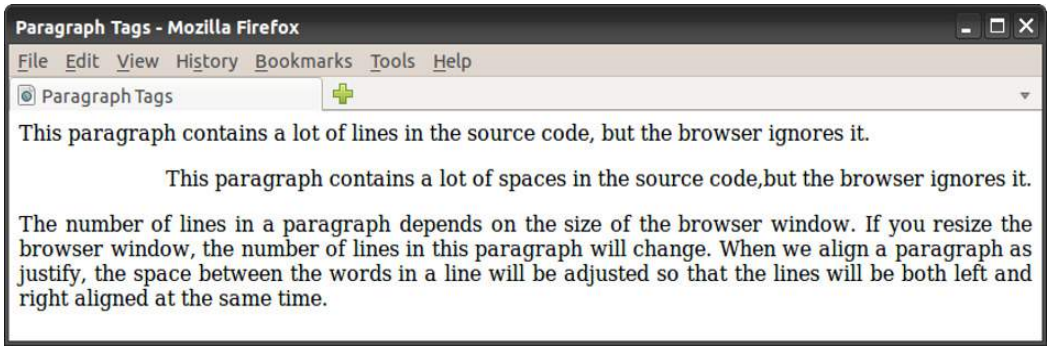


Fig. 4.21: Use of <P> tag with Align attribute

In Example 4.5, we can see that the source code contains three lines in the first paragraph and two lines in the second paragraph with extra spaces. But the web page of this code shown in Figure 4.21 gives a single line for the first two paragraphs. That is, the browser will remove extra spaces and extra lines when the page is displayed. Note that the second paragraph is right aligned and the third paragraph is aligned as justified. Any number of spaces and any number of lines count as only one space. Therefore with HTML, we cannot change the output by adding extra spaces or extra lines in the HTML code. But this is possible in HTML. Now let us discuss how an extra line can be added to the text content using **
** tag. Note that the web pages obtained in large or small screens and resized windows may not match with the one given in Figure 4.21.

4.10.3
 tag - Inserting line break

The purpose of BR element is that it creates a line break within a block of text in a web page. The **
** tag is used to break the current line of text and continue from the beginning of the next line. The
 tag is an empty tag, which means that it has no ending (closing) tag.

The HTML code in Example 4.6 displays our National Pledge and Figure 4.22 shows the resultant page in which we can see the effect of
 tag and the difference between <P> tag and
 tag.

Example 4.6: To show the National Pledge with line breaks

```
<HTML>
<HEAD>
   <TITLE> Line Breaks </TITLE>
</HEAD >
<BODY Bgcolor = "#FFEFC5">
   <H1 Align = "center"> Our National Pledge </H1>
   <P>India is my country and all Indians
```

```
      are my brothers and sisters.<BR>
      I love my country and I am proud of its
      rich and varied heritage.<BR>
      I shall always strive to be worthy of it.<BR>
      I shall give respect to my parents, teachers and all
      elders and treat everyone with courtesy.<BR>
      To my country and my people, I pledge my
      devotion. <BR>In theirwell-being and prosperity alone
      lies my happiness.
    </P>
</BODY>
</HTML>
```

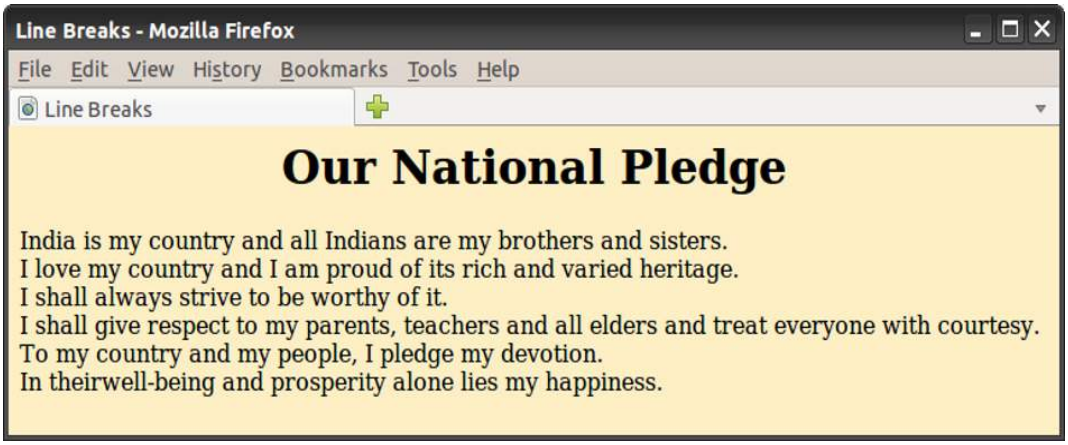


*Fig. 4.22 : Output showing the use of
tag*

Let us do

Referring to Example 4.6 and Figure 4.24, fill in Table 4.5 with appropriate points to distinguish between <P> tag and
 tag.

<P> tag	 tag
	Breaks the current line and continues to the next line.
Container tag	

*Table 4.5: <P> tag Vs
 tag*

4.10.4 <HR> tag - creating horizontal line

The **<HR>** tag produces a horizontal line (rule) spread across the width of the browser window. We can change the size (thickness) and width (length) of the line using its attributes **Size** and **Width**. The value of Size is given in pixels; and the value of

Width attribute is given in pixels or in percentage of total width of the document window (72 pixels = 1 Inch). The other important attributes of <HR> tag are Noshade and Color. The Noshade attribute has no value. The Color attribute sets the colour of the line (rule). Remember that <HR> is an empty tag. Obviously Align is another attribute which makes the alignment of the ruler left, center or right.

The code given in Example 4.7 creates a web page as shown in Figure 4.23 to show the effect of <HR> tag and its attributes.

Example 4.7: To draw various types of lines

```
<HTML>
<HEAD>
   <TITLE> Horizontal Rules </TITLE>
</HEAD >
<BODY Bgcolor= "#BDB76B">
<H1 Align= "center"> Horizontal Rules </H1>
Line 1<HR Width= "50%" Align= "center"> <BR>
Line 2<HR Width= "40%" Align= "center" Noshade> <BR>
Line 3<HR Size= "10" Width= "30%" Align= "center"> <BR>
Line 4<HR Size= "10" Width= "30%" Align= "center" Noshade><BR>
Line 5<HR Size= "10" Width= "20%" Align= "center" Noshade
          Color= "gold"> <BR>
Line 6<HR Size="10" Width="20%" Align="center" Color="Aqua">
</BODY>
</HTML>
```

Examining this HTML document and its output, can you identify the similarities and the differences in the shapes of lines? Here, the first two lines are drawn without the Size attribute and the first line hasn't any Noshade atribute. Line 3 and Line 4 are of size 10 but the third one does not have any Noshade attribute. The last two lines have an additional attribute Color. Note the different shapes of these lines in each case. You may modify this code with all possible values for different attributes and see the changes in the web page, during your lab activity.

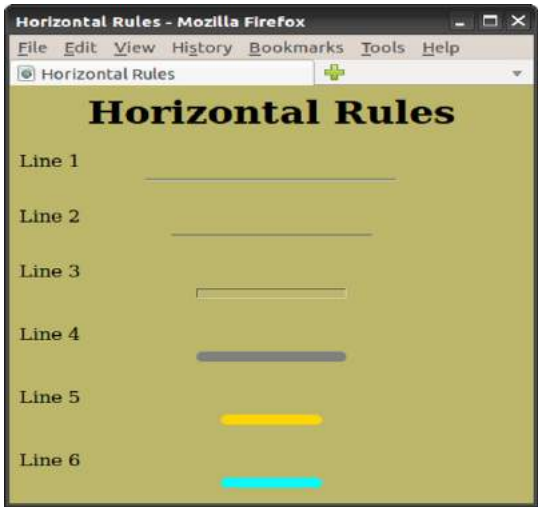


Fig. 4.23 : Different types of horizontal rules

4.10.5 <CENTER> tag - Centering the content

The **<CENTER>** tag brings the content to the centre of a web page horizontally. The content may be text, image, table, etc. This is a container tag and the content enclosed between <CENTER> and </CENTER> tag pair will be centred in the browser window. There is no attribute for this tag.

4.10.6 Text formatting tags

The text in the web page can be formatted, as we often do in word processors. The importance of a text is usually specified with features such as bold, italics, underline, etc. Let us discuss the HTML tags available for this purpose.

<B> - Making text bold

This tag sets the text style to bold. The <B> element displays the content enclosed in bold typeface.

<I> - Italicising the text

It sets the text style to italics. The content enclosed between <I> and </I> tags become italicised.

<U> - Underlining the text

The <U> tag is used to underline a text in HTML. The contents enclosed between <U> and </U> tags become underlined. The formatting tags <U>, <B> and <I> can be combined, so that the content will become bold, italicized and underlined.

<S> and <STRIKE> - Striking through the text

The <S> and <STRIKE> tags are used for the same effect. They display the text in strike through style. For example, ~~Thank you all~~ is a strike through text.

<BIG> - Making the text bigsized

The <BIG> tag is used to make the content bigger in size than the normal text size. It is often used to emphasise the word or lines in the document. Normally the font size of <BIG> tag is one size bigger than the current font size.

<SMALL> - Making the text smallsized

The <SMALL> tag is used to make the size of the text smaller than the current size. Normally the font size of <SMALL> tag is one size smaller than the current font size.

<STRONG> - Making bold text

The <STRONG> tag is a phrase tag. It defines an important text. The <STRONG> text is usually rendered in bold face. It is just the same as <B> tag. The strong element is used to emphasize a phrase of text content.

The code given in Example 4.8 illustrates the application of these tags. For instance, let us quote the words of Mahatma Gandhi. Figure 4.24 shows the resultant web page of this code.

Example 4.8: To illustrate the text formatting tags

```
<HTML>
<HEAD>
   <TITLE> Formatting Tags </TITLE>
</HEAD>
<BODY>
   <P>
   <CENTER><B>Mahatma Gandhi </B>is the <I> <U> Father of
               our Nation.</U> </I> </CENTER>
   Every student must learn the inspiring words (quotes)
   of Gandhiji. It will inspire everyone. Let us have a
   look at it.
   </P><BR>
   <SMALL> The weak can never forgive </SMALL>.
   <BIG> Forgiveness is an attribute of the strong</BIG>
   <BR> Live as if you were to die tomorrow.
   <STRONG> Learn as if you were to live forever.</STRONG>
</BODY>
</HTML>
```

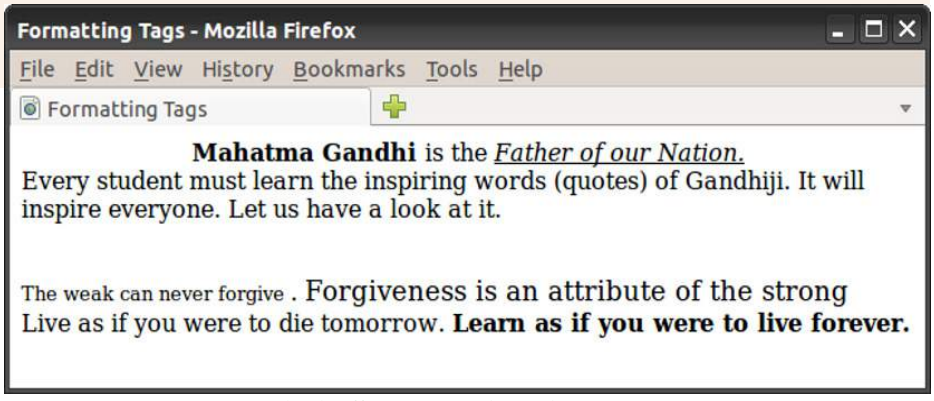


Fig. 4.24: Illustration of text formatting tags

Let us see some more tags which give the text some special appearance.

<EM> - Emphasising the text

The <EM> tag is used to emphasise the text. In practice, the element is usually rendered in italics. The effect of using <EM> tag is the same as that of <I> tag.

<SUB> and <SUP> tags- Creating subscripts and superscripts

We have studied the molecular formula of water, sulfuric acid, etc. in high school classes. Obviously they are H_2O and H_2SO_4 respectively. How can we represent

such notations in HTML? We can see that the figures are written in subscript form. The <SUB> tag is used to create subscripts in a web page. We can display the text H_2O with the code H₂O.

Similarly, the superscripts as in algebraic expressions like $(a+b)^2 = a^2 + 2ab + b^2$ can be represented by the tag <SUP>. The above expression can be written as

```
(a+b)<SUP>2</SUP> = a<SUP>2</SUP> + 2ab + b<SUP>2</SUP>
```

<BLOCKQUOTE> and <Q> tags - Indenting a quotation

The <BLOCKQUOTE> tag is used to indent the content enclosed in these tags. The HTML <Q> tag (Quote tag) is used to indicate the text enclosed in double quotation marks with an indent. This tag is intended for short quotations that do not require paragraph breaks, whereas <BLOCKQUOTE> is used for long quotations.

The illustration of the tags mentioned above is done in Example 4.9. Let us remember that World Environment Day is celebrated on June 5. Let us create thoughts for this day and create a web page using these tags. The corresponding web page is shown in Figure 4.25.

Example 4.9: To illustrate <SUP>, <BLOCKQUOTE> and <Q> tags

```
<HTML>
<HEAD>
   <TITLE> BlockQuote and Q tags  </TITLE>
</HEAD>
<BODY Bgcolor= "#98FB98" Text= "#008000">
   Every year we celebrate World Environment Day on 5<SUP>
   th</SUP> June. Let us have a message to all on this
   occassion.
   <BLOCKQUOTE> <B>June 5<SUP>th</SUP> is World Environment
     Day. </B>
     Mother nature too needs care and protection. Show her
     your care by caring for her trees. Love trees and love
     nature. And work for a greener environment because
     generations have to come... The future depends on us...
   </BLOCKQUOTE>
   <Q>Keep your world clean and green. Save trees, Save the
      environment!!
   </Q>
</BODY>
</HTML>
```

From Figure 4.25, it can be seen that the first paragraph is a normal paragraph, which starts from the left edge of the browser window. The superscripting is also applied in it. The second paragraph starts with an indent due to the use of <BLOCKQUOTE> tag. Finally, the third paragraph is within double quotes. It is the content enclosed by the <Q> tag pair. Observe the alignment of these paragraphs also. Note that, the <BLOCKQUOTE> tag may include more than one paragraph.

Fig. 4.25: Output of Example 4.9

Know your progress

1. Name some of the text formatting tags.
2. List the different attributes of <HR> tag.
3. How many levels of heading tags are available in HTML?
4. Write an HTML code segment to display x^3+y^3.
5. State True or False.
   a.
 tag is an empty tag.
   b. <EM> and <I> tags have same usage in HTML document.
   c. <U> and <I> tags are not allowed to be used together.
6. What is the use of <STRONG> tag?
7. Which tag performs the same function as that of <STRONG> tag?
8. Pick the odd one out from the following:
   a. HTML b. ALIGN c. HEAD d. CENTER

4.10.7 <PRE> - Displaying preformatted text

Suppose we want to display the content as we entered in the text editor. The **<PRE>** tag can facilitate this purpose. Normally the browser delimitted the white spaces, new line characters, the tab spaces, etc. Therfore, we can turn off the automatic formatting applied by the browser with the help of <PRE> tag. This tag tells the browser that the enclosed text is preformatted and should not be reformatted again; i.e., it tells the browser to display the text exactly in its original form.

Example 4.10 and Figure 4.26 give us an idea of this tag. Let us create a web page that contains some slogans on World Environment Day.

Example 4.10: To illustrate <PRE> tag

```
<HTML>
<HEAD>
   <TITLE> Pre Formatting tags </TITLE>
</HEAD>
<BODY Bgcolor = "#eee8aa" Text = "#b22222">
   <PRE>
     Don't Pollute Water,
            Don't Pollute Air,
     Don't Pollute Environment,
            And Don't Pollute Yourself,
     Celebrate World Environment Day ...
   </PRE>
</BODY>
</HTML>
```

The web page in Figure 4.26 shows that anything written within <PRE> and </PRE> tags will be displayed as it is in the HTML document.

Fig. 4.26: Illustration of <PRE> tag

4.10.8 <ADDRESS> - Displaying the address

The **<ADDRESS>** tag defines the contact information for the author/owner of a document or an article. The content of this tag can include name, phone numbers, PIN numbers, e-mail addresses, etc. Most of the browsers display the texts in italics.

The code given in Example 4.11 illustrates the <ADDRESS> tag. The appearance of this page is shown in Figure 4.27.

Example 4.11: To illustrate <ADDRESS> tag

```
<HTML>
<HEAD>
   <TITLE> Address tag </TITLE>
</HEAD>
   <BODY Bgcolor= "#DDA0DD">
```

```
      The contact details of the "SCERT" is the following:
    <ADDRESS>
      State Council of Educational Research and Training
      (SCERT),<BR>
      Poojappura,<BR>
      Thiruvananthapuram,<BR>
      PIN: 695012, KERALA.<BR>
      Tel : 0471 - 2341883
    </ADDRESS>
</BODY>
</HTML>
```

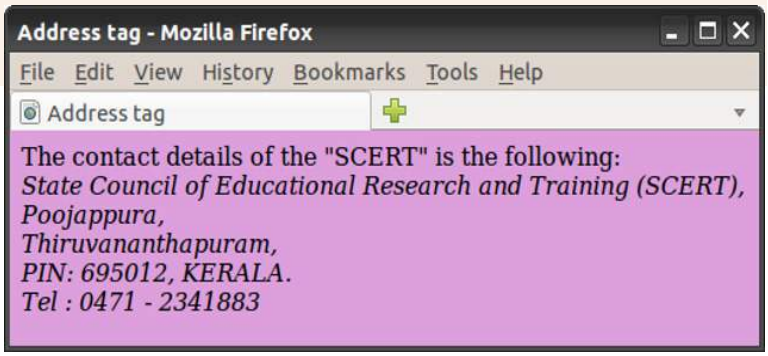


Fig. 4.27 : Web page containing <ADDRESS> Tag

The <ADDRESS> tag is usually used to describe a postal address, when it is considered as a part of the contact information. Although the address element displays the text with the same default styling as that of <I> or <EM> elements, it is suitable for use while dealing with contact information. Typically an <ADDRESS> element can be placed inside the footer which contains information about the author, copyright, etc. of the current section, if any.

4.10.9 <MARQUEE> - Displaying text in a scrolling Marquee

So far we discussed the tags in HTML that just display the contents in the browser. But there is a tag called **<MARQUEE>**, which displays a piece of text or image scrolling horizontally or vertically in the web page.

Given below is the list of important attributes that are used with <MARQUEE> tag.

- **Height**: Sets the height of the marquee in pixels or in percentage of browser window height.
- **Width**: This specifies the width of the marquee in pixels or in percentage of browser window's width value.
- **Direction**: This specifies the direction in which marquee should scroll. This can have a value like up, down, left or right.
- **Behaviour**: This specifies the type of scrolling of the marquee. This can have a value like scroll, slide and alternate.

- **Scrolldelay**: This specifies time delay between each jump. This will have value in seconds like 10, 15, etc.
- **Scrollamount**: This specifies the speed of the marquee text.
- **Loop**: This specifies how many times the marquee element should scroll on the screen. The default value is `Infinte`, which means that the marquee scrolls endlessly.
- **Bgcolor**: This specifies background colour in terms of colour name or colour hex value.
- **Hspace**: This specifies horizontal space around the marquee. This can be a value in pixels or percentage value.
- **Vspace**: This specifies vertical space around the marquee. This can be a value in pixels or percentage value.

The code given in Example 4.12 illustrates the use of `<MARQUEE>` tag and Figure 4.28 shows the corresponding web page.

Example 4.12: To illustrate <MARQUEE> tag

```
<HTML>
<HEAD>
    <TITLE> HTML marquee Tag </TITLE>
</HEAD>
<BODY>
    <MARQUEE Width= "50%"> This  will take only 50% width of
      Browser Window</MARQUEE>
    <MARQUEE Height= "100" Hspace= "100" Bgcolor= "#44BB22"
           Direction= "up"> Scrolling up </MARQUEE>
    <MARQUEE Height= "20" Vspace= "30" Bgcolor= "#FFBB00"
      Direction= "right"> This will scroll from left to right
    </MARQUEE>
</BODY>
</HTML>
```

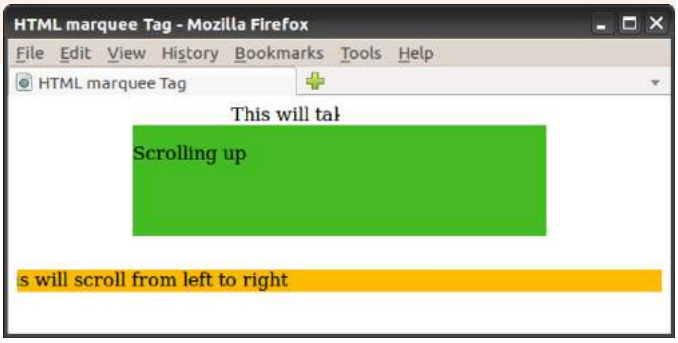


Fig. 4.28 : Marquee tag with different attributes

The first marquee starts from the middle and scrolls towards the left of the window since the width is 50%. The second marquee is in the green coloured portion and 100 pixels height is provided for scrolling. The

scroll area is set 100 pixels horizontally away from the left margin. The third marquee is filled with the background colour "#FFBB00" and placed 30 pixels vertically below the previous marquee. The scroll window has a height of 40 pixels and it scrolls from left to right. Instead of words or phrases, we can also use images as the marquee content.

4.10.10 <DIV> - Formatting a block of text

The **<DIV>** tag is used for defining a section or a block in the document. With the <DIV> tag, we can group large sections of HTML documents together and format them. This section may contain paragraphs, tables, etc. Most browsers place a line break before and after <DIV> elements. The attributes of <DIV> tag are:

Align : sets the horizontal alignment with values `left`, `right`, `center`, and `justify`.

Id : assigns a unique identifier for the tag.

Style : indicates how to render the content in terms of colour, font, etc.

In Example 4.13, we use <DIV> tag to apply `Align` and `Style`. Figure 4.29 shows the resultant web page.

Example 4.13: To illustrate <DIV> tag

```
<HTML>
<HEAD>
   <TITLE> DIV Tag </TITLE>
</HEAD >
<BODY Bgcolor= "#ddffff">
   <H2 Align= "center"> Success Story </H2>
   <DIV Align= "Center" Style= "Color:#0000FF"> One day a
   partially deaf four - year old child came home with a
   note in his pocket from his teacher, "Your Tommy is too
   stupid to learn, get him out of the school."
   <P>His mother read the note and answered, "My Tommy is
      not too stupid to learn, I will teach him myself." And
      that Tommy grew up to be the great Thomas Alwa Edison.
   </P> He had only three months of formal schooling.
   </DIV>
   All success stories are stories of great failures.
   You learn from your failure and move forward.
</BODY>
</HTML>
```

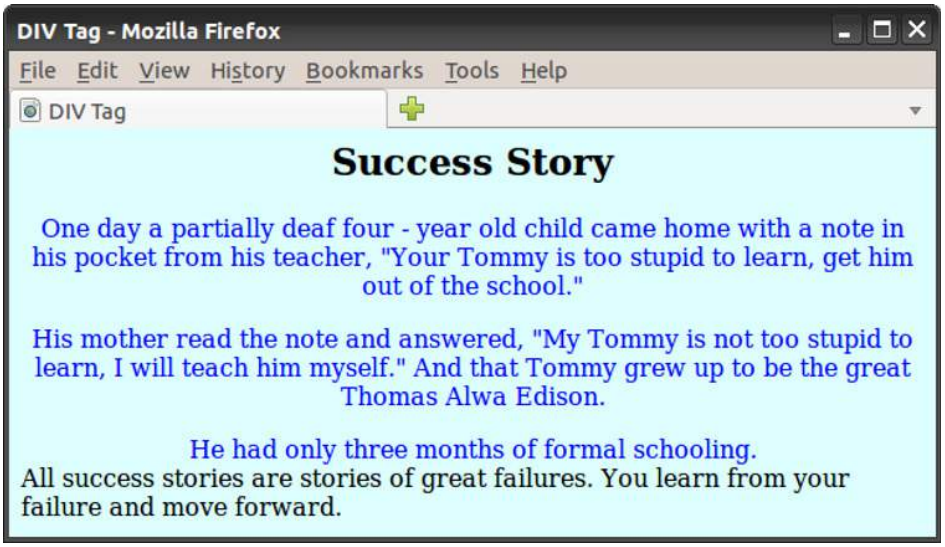


Fig. 4.29 : Application of <DIV> tag

4.10.11 <FONT> - Specifying font characteristics

The **<FONT>** tag allows us to change the size, style and colour of the text enclosed within <FONT> and </FONT> tags. It is generally used for changing the appearance of a short segment of the document. The attributes of <FONT> tag are:

Color : This attribute sets the text colour using either a ColorName or a colour in the Hexadecimal format.

Face : This attribute specifies the font face. If no face attribute is mentioned, the document text in the default style is used in the first font face that the browser supports.

Size : This attribute specifies the font size whose value ranges from 1 to 7, with default value 3.

The code given in Example 4.14 illustrates the usage of <FONT> tag and Figure 4.30 shows the corresponding web page.

Example 4.14: To illustrate <FONT> tag

```
<HTML>
<HEAD> <TITLE> Font tags </TITLE> </HEAD>
<BODY Bgcolor= "#eee8aa">
   Every success story is also a story of great failure.
   The only difference is that every time they failed,
   they bounced back. <BR><BR>
```

```
    <FONT Size="6" Face="Courier New" Color="#B22222">
    Successful people don&apos;t do great things,
    they only do small things in a great way.
    </FONT>
</BODY>
</HTML>
```

In the above code, we marked a portion `'` in green colour. What do you mean by this? From the output, we can understand that this is for getting the single quote in the word **don't**. There are several other characters like this. They are discussed in the following section.

Fig. 4.30: Illustration of <FONT> tag

4.11 HTML entities for reserved characters

In HTML, the symbols like **<**, **>**, **&**, etc. have special meaning and cannot be used in the HTML documents as part of the text content. The browser treats these symbols as the punctuation marks of HTML words like tags and entities. For example, the angle brackets **<** and **>** are used to express tags. So, when we want to display these symbols as part of the text in the web page, we must use HTML entities. Table 4.6 shows a list of a few special characters and their equivalent entities.

In order to display the text A < B & A > C in a web document, we have to specify it in the HTML document as: **`A < B & A > C`**.

Character	Entity	Description
		Non Breaking Space
"	"	Double quotation mark
'	'	Single quotation mark
&	&	Ampersand
<	<	Less than
>	>	Greater than
©	©	Copyright Symbol
™	™	Trademark Symbol
®	®	Registered Symbol

Table 4.6 : List of entities and their description

4.12 Adding comments in HTML document

It is a good practice to add comments in HTML documents, especially in complex documents. Comments help us to understand the code and it increases the code readability. HTML provides comment tag to insert comments in the source code.

Comments are not displayed in the browser window. HTML comments are placed within **<!-- -->** tag. So any content placed within <!-- --> tag will be treated as a comment and will be completely ignored by the browser. In Geany editor the comments are displayed in red colour. The code in Example 4.15 illustrates the use of comments and Figure 4.31 shows the resultant message.

Example 4.15: To illustrate the use of comment tag

```
<HTML>
<HEAD> <!-- Document Header Starts -->
    <TITLE> Comment Tags </TITLE>
</HEAD>
<BODY Bgcolor= "#D8D8D8">
    <!-- This is a comment -->
    <p>The paragraph starts here. Comment statements are
    not displayed in the browser window</p>
    <!-- Comments are not displayed in the browser -->
</BODY>
</HTML>
```

Fig. 4.31: Comments in HTML

Know your progress

1. How are special characters represented in HTML?
2. `Face` attribute is used with _____ tag.
3. List the attributes of `<FONT>` tag.
4. What is the use of `<PRE>` tag?
5. For scrolling a text, we use _____ tag.
6. What are the main attributes of `<MARQUEE>` tag?
7. What is the use of `<ADDRESS>` tag?
8. What is the normal font size in `<FONT>` tag?
9. Name the main attributes of `<DIV>` tag.

4.13 Inserting images

Images always make content presentation more attractive and communicative. Now a days, most of the websites are rich in visuals. New versions of HTML come with many web development features, but the code required for adding images is simple. HTML provides a tag **<IMG>** to insert images in HTML pages. The following is the simple way of using this tag:

```
<IMG Src = "picture1.jpg">
```

The <IMG> tag is an empty tag and it has many attributes. Src is the main attribute and it specifies the file name of the image to be inserted. We can use JPEG, PNG or GIF image files based on our needs, but make sure that the correct filename with the extension is specified using Src attribute. If the image file is not in the current working directory, we have to specify the path of the file or the URL where the file is available.

Setting space for the image

We can set the space in the web page for the image by specifying the values for the **Width** and **Height** attributes. The values are given in terms of either pixels or percentage of its actual size. If these attributes are not specified, the browser will display the image in its actual size.

Now let us discuss how to set space between images. We know that there are two types of space between the images when they are placed in a window - vertical space and horizontal space. HTML offers two attributes **Vspace** and **Hspace**, for providing vertical spacing and horizontal spacing between the images in the web page.

The HTML code in Example 4.16 demonstrates the two types of spacing discussed above and Figure 4.32 shows the web page obtained from the code.

Example 4.16: To provide different types of spacing for images

```
<HTML>
<HEAD>
   <TITLE> Inserting Images </TITLE>
</HEAD>
<BODY Bgcolor="#E0FFFF">
   <H2 Align="center">Inserting vertical and horizontal
                     spacing between images</H2>
   Here the images are placed with <B><I><U> Vspace and
   Hspace </U></I></B> attributes <BR>
```

```
    <IMG Src= "book3.jpg" Height= "50" Width= "70"
         Vspace= "10" Hspace= "10">
    <IMG Src= "book3.jpg" Height= "50" Width= "70"
         Vspace= "10" Hspace= "10"> <BR>
    <IMG Src= "book3.jpg" Height= "50" Width= "70"
         Vspace= "10" Hspace= "10">
    <IMG Src= "book3.jpg" Height= "50" Width= "70"
         Vspace= "10" Hspace= "10"> <BR>
</BODY>
</HTML>
```

Figure 4.32 shows that the images are placed within the given width and height, and the distance between the images is according to the specified horizontal and vertical spacing.

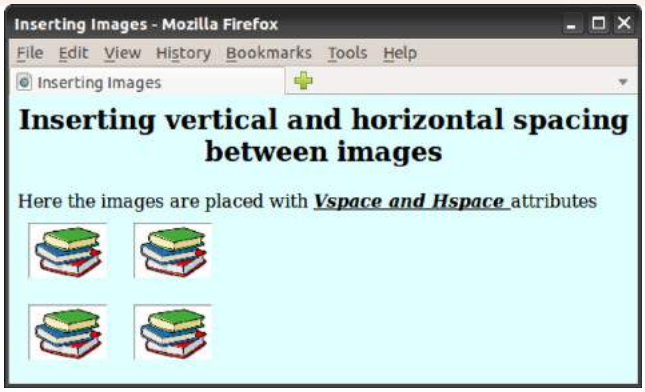


Fig. 4.32: Images within the specified size and with horizontal and vertical spaces between them.

Now let us discuss another important attribute, **Align** for <IMG> tag, which aligns the image with respect to the base line of the text. The possible values for this attribute are the following:

Bottom : Aligns the bottom of the image with the baseline of the text and this is the default setting.

Middle : Aligns the middle of the image (vertically) with the baseline of the text.

Top : Aligns the image with the top of the text.

Let us see the effect of these values for the attribute Align. Example 4.17 and Figure 4.33 demonstrate this.

Example 4.17: To provide different alignments for an image

```
<HTML>
<HEAD>
    <TITLE> Alignment of Images </TITLE>
</HEAD>
<BODY Bgcolor= "#E0FFFF">
    <H2 Align= "center">Alignment of Images</H2>
    This Image is <I><U>aligned at the bottom </U></I>
```

```
    <IMG Src= "book3.jpg" Height= "40" Width= "50"
         Align= "Bottom"> <BR> <BR>
        This Image is <I><U>aligned at the Middle </U></I>
    <IMG Src= "book3.jpg" Height= "40" Width= "50"
         Align= "Middle"> <BR> <BR>
         This Image is <I><U>aligned at the Top </U></I>
    <IMG Src= "book3.jpg" Height= "40" Width= "50" Align="Top">
</BODY>
</HTML>
```

There are some more values for the Align attribute of <IMG> tag. They are left and right, which align the image the left and right sides of the browser window respectively.

Fig. 4.33: Different alignment of images with the text

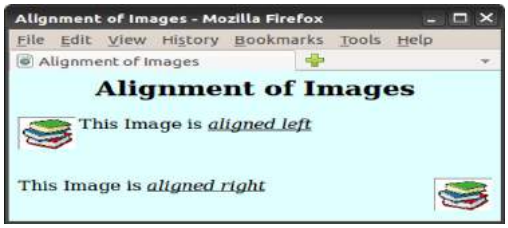


Fig. 4.34: Images aligned left and right of browser window

Let us do

Observe Figure 4.34, in which the left and right alignments of images in the browser window are shown. Write the HTML document for this web page.

Setting border around an image

Suppose we want to give a border to an image inserted in a web page. It is possible with the Border attribute of <IMG> tag. The thickness of the border can be set by giving appropriate value to this attribute. The HTML code in Example 4.18 and the corresponding web page shown in Figure 4.35 give the effect of border attribute.

Example 4.18: To give a border to an image

```
<HTML>
<HEAD>
    <TITLE> Inserting Images </TITLE>
</HEAD>
<BODY Bgcolor= "#E0FFFF">
    <H2 Align= "center">Inserting Border to Images</H2>
    Here is an image <B><I><U> with Border </U></I></B>
    attribute
```

```
    <IMG Src= "book3.jpg" Height= "50" Width= "70" Border="5">
    <BR>Here is an image<B><I><U> without Border</U></I></B>
    attribute
    <IMG Src= "book3.jpg" Height= "50" Width= "70">
</BODY>
</HTML>
```

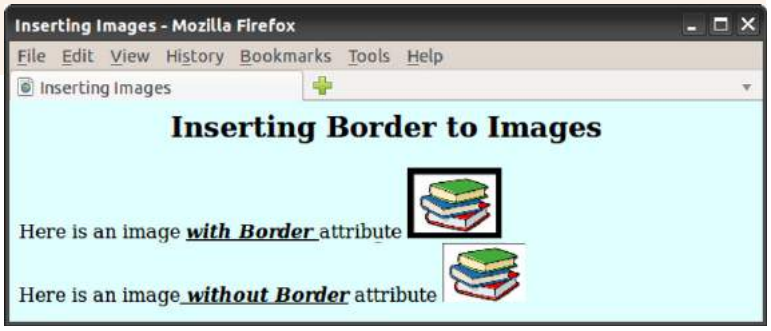


Fig. 4.35: Images with border and without border

We have learnt various attributes of `<IMG>` tag and their effects on the image. If the image file specified with the Src attribute is not found in the given path, how will the web page look? The specified place in the web page for the image will be left blank. Due to some other reason also, the web browser may not be able to display the image. In such a situation, we can provide an alternative text in the browser in the absence of image. HTML provides the attribute **`Alt`** to specify an alternate text for an image, if the browser cannot display the image. The code in Example 4.19 illustrates the effect of `Alt` attribute and Figure 4.36 shows the resultant web page.

Example 4.19: To illustrate the effect of `Alt` attribute of `<IMG>` tag

```
<HTML>
<HEAD> <TITLE> Inserting Images </TITLE> </HEAD >
<BODY Bgcolor= "#E0FFFF">
    <H2 Align= "center">Inserting Images</H2>
    If the browser cannot display the image, then the text
    entered in the <B>Alt</B> attribute will be displayed.<BR>
    <IMG Src= "book5.jpg" Height= "20%" Width= "20%"
         Alt= "Image of an opened book"> <BR>
</BODY>
</HTML>
```

Fig. 4.36 : Effect of Alt attribute in <IMG> tag

Know your progress

1. To insert images in an HTML document _____ tag is used.
2. _____ is the main attribute of <IMG> tag.
3. What is the use of Alt attribute of <IMG> tag?
4. Name the attributes that are used to display an image in a particular size.
5. Which are the attributes that provides horizontal and vertical spaces between two images?

Let us conclude

The security of communication over the Internet is a determining factor in the success of the Internet. The security of transactions over the Internet is implemented using HTTPS and digital certificates. Internet infrastructure consists of technologies like web server, software ports and DNS servers used to store and communicate data. Web servers consist of a single/several websites. Websites consists of web pages. We can design web pages either by writing HTML code or using web designing softwares. Webpages can be classified as static and dynamic. Dynamic web pages can be developed using scripts. We can perform client side validations using client side scripting languages like, JavaScript, VBScript, etc. Server side scripts like PHP, ASP, etc. are used to create pages dynamically in the web server. Cascading Style Sheet (CSS) can be used to provide a uniform style to the entire website. The basic concepts of HTML language and web page designing are discussed here. We are now familiar with different tags and their important attributes. We know that some of the tags are container tags, but some others are empty tags. We can make a web page with neatly formatted text using a variety of formatting tags. Different kinds of listing are discussed to make the text more presentable. The beauty of the web pages can be enhanced by including marquees, images, audio and video. We have also discussed the importance of hyper linking and have been familiarised with various kinds of hyperlinks. We should have a sound knowledge of the concepts discussed in this chapter and an excellent practical experience in creating HTML documents, so that we can easily internalise the concepts in the following chapters. Also, we will be able to design beautiful web sites and develop web applications ourselves.

Let us practice

1. Wrtie an HTML code for a web page of Kerala with the following details and features:
   - A heading followed by a paragraph of 5 sentences about Kerala using text formatting tags and attributes.
   - Background of the page needs an image of a scenary.
2. Wrtie an HTML code for a web page for your school with the following details and features:
   - A heading followed by a paragraph of 3 sentences about the district using text formatting tags and attributes.
   - Provide a colour to the background of the page.
   - Include an image of the school.
3. Wrtie an HTML code for a web page for your school with the following details and features:
   - A heading followed by a paragraph of 3 sentences about the district using text formatting tags and attributes.
   - Give the postal address of the school.
   - Include a marquee that "Admission for the new accademic year commences on 10th May"
4. Wrtie an HTML code for a web page to show the lyrics of our National Anthem with the following details and features:
   - A heading with a different font characteristics.
   - An image of our national flag

Let us assess

1. What is the role of routers in transporting data over the Internet?
2. The social media websites developed their own protocol for communicating over the Internet. How is this possible when Internet uses TCP/IP protocol?
3. The user name and password of an e-mail account has to be sent securely over the Internet.
   a. Name the technology used to send the data to the server.
   b. How does this technology support secure data communication?
4. What is the role of payment gateway in online purchases?
5. ABC Engineering College has about 1000 computers connected to the Internet, in its campus. What is the advantage of having a local DNS server in the college's intranet?

6. Write an example of a web server operating system and a web server package.
7. What is the use of software ports in a web server?
8. The port used for HTTP is _____.
9. Suppose you are browsing the website www.prdkerala.org. Explain how the DNS resolves the IP address.
10. What are scripts? Explain the different scripting languages.
11. Consider the home page of your school website and the web page that displays the results of class XI examinations.
    a. Compare the difference between the two web pages based on their creation.
    b. Write the technologies that can be used for developing these web pages.
12. a. The file name extension for a JavaScript file is _____.
    b. Write two popular uses of JavaScript in a web page.
13. What is Ajax? What is its use?
14. Your friend Ravi wishes to create a website that displays the marks of the students in your class in each examination.
    a. Suggest a technology to implement this.
    b. Justify your suggestion.
15. Consider that Manoj is developing a website using PHP that uses a database in MySQL. What are the requirements to implement this if he is using Linux web hosting?
16. "Almost all websites today use CSS for its development." What are the advantages of using CSS in web sites?
17. Who developed HTML?
18. In HTML, there are mainly two sections. Can you name them?
19. If you analyse web pages you can see different colours for links, visited links, background etc. Expalin how this can be done in HTML, with examples.
20. Compare container and empty tags in HTML with examples.
21. The default colour of the attribute `Alink` is _____.
22. The default color of the attribute `Vlink` is _____.
23. Classify the following HTML related words:
    `BR`, `IMG`, `ALIGN`, `FONT`, `FACE`
24. Name the tag which has `Noshade` attribute.
25. Write the main attribute of `<IMG>` tag to insert an image file in the webpage.
26. Mention the purpose of `Alt` atribute in `<IMG>` tag.
27. The default alignment of an image obtained by using `<IMG>` tag is _____.
28. List the main attributes of `<FONT>` tag.

5 Web Designing using HTML

Significant Learning Outcomes

After the completion of this chapter, the learner

- uses different list tags to present the content effectively in web pages.
- identifies the relevance of hyper linking and uses <A> tag for different types of linking.
- provides audio and video in web pages with the help of <EMBED> tag.
- produces inline sounds and videos in web pages.
- lists and explains the tag and attributes for creating a table.
- uses the tags associated with <TABLE> tag to design tables with different characteristics.
- constructs different tables using the tags and attributes.
- identifies the importance of frames in the web page.
- creates frames using appropriate tags to display different web pages in the same browser window.
- identifies the concept of Forms in the web page.
- explains various components in a Form and creates them using proper tags and attributes.
- designs web pages with tables, frames and forms.

In the previous chapter, we studied the basic tags of HTML. We have also learnt to create some simple web pages using those tags and their attributes. But we are familiar with websites that provide much more facilities and utilities. There are websites that contain different types of lists. Linking between web pages is the backbone of World Wide Web. The different types of linking are discussed in this chapter. You might have come across certain information in tabular form. Sometimes you see more web pages in the same browser window. We are also familiar with websites through which we submit the register number to obtain the mark list of examinations, apply for admission and scholarships, pay the bills of electricity and water consumption etc. How can we create tables in web pages to present information? Can we place more than one web page in a single browser window? If yes, how? How are web pages created to accept data from the user to provide information? HTML provides all these facilities for web developers. In this chapter, we discuss the HTML tags required to answer all these questions.

5.1 Lists in HTML

While presenting information, the facility of listing can make it more communicative. Lists are of different types. We are familiar with numbered lists and bulleted lists. HTML offers several mechanisms for specifying lists of information. All lists must contain one or more list elements. There are three kinds of lists in HTML - unordered lists, ordered lists and definition lists.

5.1.1 Unordered lists

Unordered lists or bulleted lists display a bullet or other graphic in front of each item in the list. We can create an unordered list with the tag pair **<UL>** and **</UL>**. Each item in the list is presented by using the tag pair **<LI>** and **</LI>**. Unordered lists are used when a set of items can be placed in any order.

The code in Example 5.1 presents some hardware components of a computer in bulleted list. The corresponding web page is shown in Figure5.1.

Example 5.1: To create an unordered list

```
<HTML>
<HEAD>
   <TITLE> Unordered Lists </TITLE>
</HEAD>
<BODY Bgcolor= "#DEB887">
   <CENTER> <H2> Unordered List </H2> </CENTER>
   While buying a computer, we have to consider many items.
   Here are some important items to consider.
   <UL>
     <LI> RAM </LI>
     <LI> Hard Disk </LI>
     <LI> Mother Board </LI>
     <LI> Processor </LI>
   </UL>
</BODY>
</HTML>
```

Fig.5.1: Web page containing unordered list

We can customise unordered lists by setting the **Type** attribute to three different values: `Disc` (default value), `Square` and `Circle`, which set the type of bullet that appears before each list item. The following code creates a list as shown in Figure 5.2.

```
<UL Type= "Square">
    <LI> RAM </LI>
    <LI> Hard Disk </LI>
    <LI> Mother Board </LI>
    <LI> Processor </LI>
</UL>
```

Fig.5.2: Unordered list with square bullet

5.1.2 Ordered lists

Ordered lists present the items in some numerical or alphabetical order. HTML provides the tag pair **<OL>** and **</OL>** to create an ordered list. The items in the ordered list are presented by <LI> tag in <OL> element. The ordered list is also called numbered list. Ordered lists or numbered lists are used to display a list of items that need to be placed in a specific order.

Example 5.2 is a code that presents an ordered list of items. The corresponding web page is shown in Figure 5.3.

Example 5.2: To create an ordered list

```
<HTML>
<HEAD>
    <TITLE> Ordered Lists </TITLE>
</HEAD>
    <BODY Bgcolor= "#DDA0DD">
    <H2 Align= "center"> Ordered List </H2>
    Consider the memory devices of a computer.
    Then according to the speed of data processing,
    we can arrange the memory devices as follows.
    <OL>
      <LI> Registers </LI>
      <LI> Cache </LI>
      <LI> RAM </LI>
      <LI> Hard Disk </LI>
    </OL>
</BODY>
</HTML>
```

Fig.5.3: Web page containing ordered list

We can see that the list in Figure 5.3 is numbered from one through four. There are other numbering styles for presenting the list

items. We can customise the numbering system used in ordered list by using the **Type** attribute, which can set with the values as detailed below:

1 Default numbering scheme (1, 2, 3, ...)

A Upper case letters (A, B, C, ...)

a Lower case letters (a, b, c, ...)

I Large roman numerals (I, II, III, ...)

i Small roman numerals (i, ii, iii, ...)

An ordered list, by default, starts with the first number in the series used in the list. That is, the starting number will be any one from 1, A, a, I and i. If we want to start with any other number in the series, then the **Start** attribute of <OL> tag enables us to change the beginning value. To start numbering a list at 5, for example, we may write: **<OL Start= "5">**. Thus, the numbering starts from 5 and then proceeds with 6, 7, 8, ... and so on.

The Start attribute sets the starting value of the item (it must be an integer) and the Type attribute sets the numbering style. For example, the following ordered list starts numbering from V and continues with VI, VII, ... and so on. The output of this code is shown in Figure 5.4.

```
<BODY Bgcolor ="#DDA0DD">
<H4 Align="center">Ordered List with Type attribute</H4>
<OL Type= "I" Start= "5">
  <LI> Registers </LI>
  <LI> Cache </LI>
  <LI> RAM </LI>
  <LI> Hard Disk </LI>
</OL>
</BODY>
```

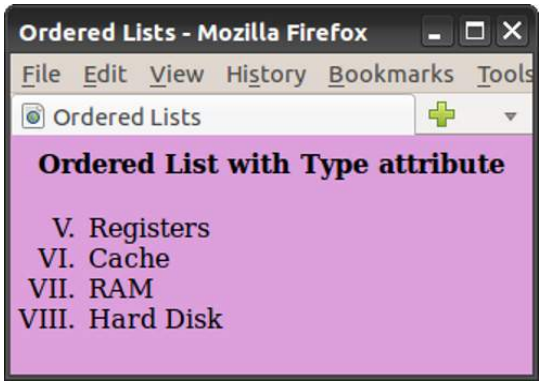


Fig.5.4: Ordered list with different numbering style and starting value

5.1.3 Definition lists

A definition list is a list of terms and the corresponding definitions. The definition text is typically indented with respect to the term. No bullet symbol or number is provided for the list items. The tag pair **<DL>** and **</DL>** enclose the definition lists. Each term in the list is created using the **<DT>** tag and the **<DD>** tag supplies the definition of the term.

The code in Example 5.3 creates a web page to present the definitions of some terms related to security aspects of Internet. Figure 5.5 shows the resultant web page.

Example 5.3: To create a definition list

```
<HTML>
<HEAD> <TITLE> Definition List </TITLE> </HEAD >
<BODY Bgcolor= "#FFE4C4" Leftmargin= "100" Rightmargin= "150">
    <H2 Align= "center"> Definition List </H2>
      Today cyber security has an immense role in the
      field of Internet. The following are some of
      the threats that affect a computer network.
    <DL>
      <DT>Spam :</DT>
      <DD> Spam is the unsolicited e-mail sent in the
           hope of increasing the sales of some product, or
           just for annoying people.</DD>
      <DT>Phishing :</DT>
      <DD> Phishing is an attempt to acquire information
           such as usernames, passwords and credit card details
           by posting as the original website, mostly that
           of banks and other financial institutions. </DD>
    </DL>
</BODY>
</HTML>
```

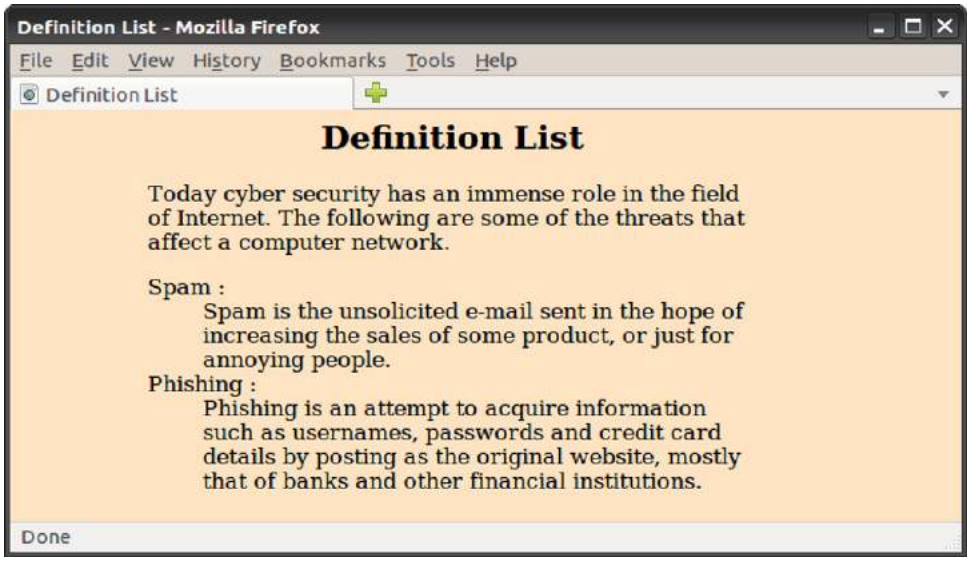


Fig. 5.5: Web page containing a definition list

In Figure 5.5, we can see that each annotation is indented under the corresponding term. Also note the left and right margins in the window.

5.1.4 Nested lists

A list of items can be given under each item of another list. It is known as nesting of lists. This is possible in many ways. For example, we can insert an unordered list into another unordered list, an unordered list into an ordered list, an ordered list inside an unordered list, etc. The code given in Example 5.4 demonstrates the concept of nested list and Figure 5.6 shows the resultant web page.

Example 5.4: To create a nested list

```
<HTML>
<HEAD>
    <TITLE> Nested Lists </TITLE>
</HEAD>
<BODY Bgcolor= "#E0FFFF">
    <H2 Align= "center"> Nested List </H2>
    Consider the devices of a computer.
    We can list some of them as follows.
    <OL>
        <LI> Input Devices </LI>
        <UL>
            <LI>Keyboard</LI>
            <LI>Mouse</LI>
            <LI>Scanner</LI>
            <LI>MICR</LI>
        </UL>
        <LI> Output Devices </LI>
        <UL Type= "Square">
            <LI>Printers</LI>
            <LI>Monitors</LI>
            <LI>Speakers</LI>
        </UL>
        <LI> Memory Devices </LI>
        <UL Type= "Circle">
            <LI>Hard Disc</LI>
            <LI>CD Rom</LI>
            <LI>Flash Drive</LI>
        </UL>
    </OL>
</BODY>
</HTML>
```

Fig.5.6: Classification of devices using nested list

In Example 5.4, three sets of unordered lists are nested into an ordered list.

Know your progress

1. What are the different types of lists in HTML?
2. Suppose your teacher asks you to display the list of students in your class using HTML document. Which type of list will you prefer? Why?
3. What are the common attributes of <UL> and <OL> tags?
4. What is the difference between <UL> tag and <OL> tag?
5. Name the tags used in the definition list.

5.2 Creating links

A hyperlink is an element, a text, or an image in a web page that we can click on, and move to another document or another section of the same document. Hyperlinks allow visitors to navigate between websites by clicking on words, phrases and images. HTML provides the ability to hyperlink text, image etc. to another document or section of a document. Hyperlink is often referred to as links. In HTML, the **<A>** tag provides the facility to give hyperlinks. This tag is called anchor tag and anything given between the tag pair <A> and </A> becomes part of the link and a user can click that part to reach the linked document. **Href** is the main attribute of <A> tag and it means hyper reference. The value of this attribute is the URL of the document (address of the web page/site) to which hyperlink is provided.

For example, consider the following code segment:

```
<A Href= "http://www.dhsekerala.gov.in">Higher Secondary</A>
```

This creates the target of the hyperlink to the website http:// www.dhsekerala.gov.in. At the time the user clicks the link, the browser opens the home page of this URL. The text inside the tag pair <A> and </A> will appear underlined and in different colour.

The HTML code in Example 5.5 and Figure 5.7 show how hyperlinks are created in web pages.

Example 5.5: To create a hyperlink in a web page

```
<HTML>
<HEAD>
    <TITLE> Anchor Tag </TITLE>
</HEAD>
<BODY Bgcolor= "#FFFFFF">
    <H2 Align= "center"> Hyperlinks </H2>
```

```
    <P>Now this will create a hyperlink to the website of
       Higher Secondary Department.<BR>
       Kindly click on the words
    <A Href= "http://www.dhsekerala.gov.in">Higher Secondary
       Education</A>.
</BODY>
</HTML>
```

As shown in Figure 5.7, the hyperlinked text is in underlined format (Higher Secondary Education) and in a different colour.

Hyperlinks are considered either "internal" or "external" depending on their target.

Fig.5.7: Web page containing hyperlink

5.2.1 Internal linking

A link to a particular section of the same document is known as internal linking. The attribute **Name** of <A> tag is required for this. We have to give a name to identify the section to be opened by the browser. This name is used with the Name attribute to establish the link.

For example, suppose we have three paragraphs on the subject "Environment Pollution". Let it be the Introduction section, Air pollution section, and Water pollution section. We refer to these sections by giving different values to the Name attribute of <A> tag.

```
<A Name= "Introduction"> INTRODUCTION </A>
<A Name= "Air"> Air Pollution </A>
```

Now, we can refer to these sections by giving the values of Href attribute as #Introduction, #Air (the # symbol is essential) from another section of the document as follows:

```
<A Href = "#Introduction"> Go to Introduction </A>
<A Href = "#Air"> Air pollution </A>
```

Let us create a web page incorporating the concepts of internal linking. Example 5.6 is an HTML code that has two internal links. Figure 5.8 shows the resultant page of this code.

Example 5.6: To create a web page containing internal links

```
<HTML>
<HEAD> <TITLE> Internal Linking </TITLE> </HEAD>
<BODY Bgcolor= "f8f8f8">
    <H2 Align= "center">ENVIRONMENTAL POLLUTION</H2>
    <A Name= "Introduction"><B>INTRODUCTION</B></A>
    <P><FONT Size= "15">E</FONT>nvironment pollution is a
    wide-reaching problem and it is likely to affect the
    health of human population.Here we discuss the environment
    pollution in the perspective of <A HREF= "#Air">air
    pollution </A>,
    <A Href= "#Water"> water pollution </A>and land/soil
    waste pollution. Studies find that these kinds of
    pollutions are not only seriously affecting humans
    but also animals and plants.
    </P>
    <A Name= "Air"><B> Air Pollution</B></A>
    <P>The air we breathe is an essential ingredient for our
    health and wellbeing. Unfortunately polluted air is common
    throughout the world, especially in developed countries.
    </P>
    <A Name= "Water"><B> Water pollution</B></A>
    <P>The water we drink is an essential ingredient for our
    health and wellbeing. Unfortunately polluted water and
    air are common throughout the world.Water pollution is
    caused by the discharge of industrial effluents, sewage
    water and agricultural or household waste.
    </P>
    <A Href= "#Introduction">Go to Introduction </A>
</BODY>
</HTML>
```

On clicking the hyperlink air pollution and water pollution on the web page shown in Figure 5.8, we get that particular section of the document as shown in Figure 5.9.

Fig. 5.8: Web page containing internal hyperlinks

Similarly, when we click on the link Go to Introduction at the bottom of the page shown in Figure 5.9, the introduction section will be displayed on the window.

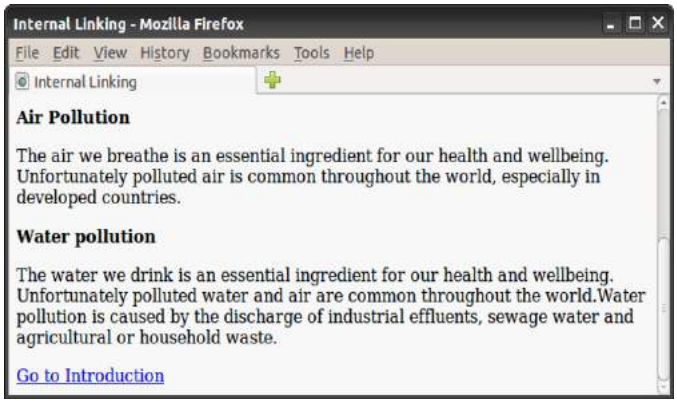


Fig. 5.9: Hyperlinked section of a page in the browser

5.2.2 External linking

The link from one web page to another web page is known as external linking. It is made possible by providing the URL of the external file in the Href attribute of **<A>** tag of the current page. The procedure for external linking is illustrated in the beginning of Section 5.2.

5.2.3 Concept of URL

URL stands for Uniform Resource Locator, and it means the web address. In fact, there are two types of URLs - relative URL and absolute URL.

The referencing `<A Href= "http://www.scertkerala.gov.in">` is an absolute URL, because it is referring to a specific URL. On the other hand, `<A Href = "image.html">` is relative reference. Here we give the file name, '*image.html*' as the URL, and it is a relative URL, relative to the current document. If the URL of the current web page document is *D:\HTML\hyperlink.html*, the web browser knows that the hyperlink `<A Href = "image.html">` points to the file *image.html* in the directory *D:\HTML* itself.

5.2.4 Creating graphical hyperlinks

In the previous sections, we gave hyperlinks to texts. We can make hyperlinks to images also using **<IMG>** tag inside the **<A>** tag. The HTML code given in Example 5.7 and Figure 5.10 show a web page containing graphical hyperlink.

Example 5.7: To create a web page containing graphical hyperlink

```
<HTML>
<HEAD> <TITLE> Graphical Hyperlink </TITLE> </HEAD>
<BODY Bgcolor = "#E0FFFF">
    <H2 Align= "center">Graphical Hyperlink</H2>
    Here is the image with <I>Graphical hyperlink </I>
    <A Href= "https://www.wikipedia.org">
```

```
    <IMG Src= "wiki.jpg" Alt= "Image of Wiki"
         Height= "30" Width= "40" Border= "1"> </A>.
         We can click over this image and the home page of
         linked site, wikipedia.org will open.
</BODY>
</HTML>
```

Suppose you move the mouse pointer upon the image of the wikipedia logo in the web page, the mouse pointer will change to the hand symbol. This indicates that, this image is a hyperlink. When we click on this image, then the corresponding web page www.wikipedia.org will be opened.

Fig. 5.10: Web page containing graphical hyperlink

5.2.5 Creating e-mail linking

We can create an e-mail hyperlink to a web page using the hyperlink protocol **mailto:**. Let us see the code in Example 5.8 and the web page shown in Figure 5.11 to understand the concept of e-mail hyper linking.

Example 5.8: To create a web page containing e-mail hyperlink

```
<HTML>
<HEAD> <TITLE> e-mail Linking </TITLE> </HEAD >
<BODY Bgcolor= "#E0FFFF">
    <H2 Align= "center">e-mail linking</H2>
    Now we can create an <B><I>e-mail hyperlink </I></B> to
    SCERT in the following way. Kindly click on the word
    <A Href= mailto: "scertkerala@gmail.com"> SCERT</A> Kerala.
</BODY>
</HTML>
```

As shown in Figure 5.11, the web contains a link SCERT. If we click on it, e-mail program will be opened with an empty message box addressed to the mail address scertkerala@gmail.com.

Fig. 5.11: Web page containing e-mail hyperlink

5.3 Inserting music and videos

Nowadays, web pages are enriched not only with text and images, but also with plenty of music, video and other multimedia resources. Let us have a basic idea about the inclusion of such resources in web pages. There are two ways of handling multimedia in a web browser. One is as inline and the other is as external data. The inline refers to files and data that are handled as part of the page. These files play the music or video when the page is visible in the browser window. We can also link a web page to external multimedia files with extensions .jpg, .gif, .avi, .png, .tiff, .mp3, .mp4, etc.

The easiest way to add music or video to the web page is with a special HTML tag, called **<EMBED>**. This tag includes the controls of the multimedia automatically in the browser window. In case, the browser does not support the `<EMBED>` tag, then we can use **<NOEMBED>** tag. The content within this tag pair will be displayed, if the `<EMBED>` tag is not supported by the browser.

The main attribute of the `<EMBED>` tag is **Src**, which specifies the URL of the music or video files to be included. The other attributes are **Height**, **Width**, **Align**, **Alt**, etc. The values of these attributes are familiar to us. One more important attribute of `<EMBED>` tag is **Hidden** which indicates whether the embedded component should be made visible or not. It is done by setting the value to `True` (by default) or `False`.

The code in Example 5.9 creates a web page that includes an audio link. Figure 5.12 shows the corresponding web page.

Example 5.9: To create a web page containing audio link

```
<HTML>
<HEAD>
      <TITLE> Embed Tag </TITLE>
</HEAD>
<BODY Bgcolor = "#DDFFFF">
    <H2 Align= "center"> Adding Music and Videos </H2>
    Here is a tag <B><I>EMBED </I></B> with the Multimedia
    features.<BR>
    <EMBED Src= "song1.mp3" Width= "300" Height= "60">
    </EMBED>
</BODY>
</HTML>
```

When the HTML document in Example 5.9 is opened, the webpage will play the music that we embedded along with the document. The audio controls like play/ pause, volume, etc. will also be displayed on the web page. Similarly we can add movies in a webpage by using the tag <EMBED> and its attribute **Src**.

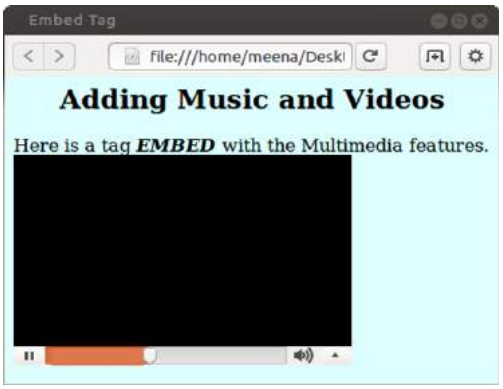


Fig. 5.12: Web page containing audio link

The code in Example 5.10 and the resultant web page shown in Figure 5.13 illustrate this linking.

Example 5.10: To create a web page containing video link

```
<HTML>
<HEAD>
    <TITLE> Embed Tag </TITLE>
</HEAD >
<BODY Bgcolor = "#DDFFFF">
    <H2 Align="center">Adding Music and Videos</H2>
    Here is a tag <B><I>EMBED </I></B>with the Multimedia
    features.
    <EMBED Src= "alan.mp4" Width= "300" Height= "150">
    </EMBED>
    <NOEMBED><IMG Src= "book2.jpg"
          Alt= "Alternative Media">
    </NOEMBED>
</BODY>
</HTML>
```

When we open this page, the video starts to play. Here again, we can control the different properties of the audio and video, such as volume, pause, full screen mode, etc. We can also include audios and videos from other web pages or sites by linking them to our pages.

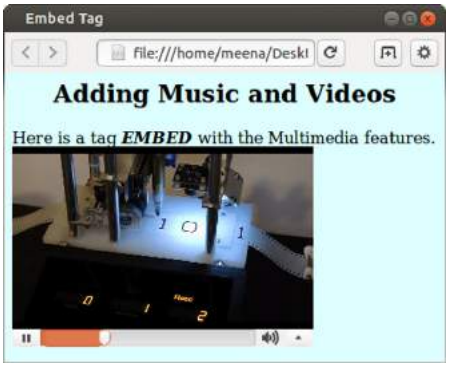


Fig. 5.13: Web page with video

As mentioned earlier, music can be played in the background while the page is viewed. This kind of audio inline is achieved by the tag **<BGSOUND>**. The code in Example 5.11 illustrate the use of this tag.

Example 5.11: To illustrate the use of <BGSOUND> tag

```
<HTML>
<HEAD> <TITLE> Background Music </TITLE> </HEAD >
    <BODY Bgcolor = "#DDFFFF">
    <H2 Align= "center">Adding Background Music </H2>
    Here is a tag <B><I> BGSOUND </I></B> which helps
    us to play background music in our web page.
    <BGSOUND Src= "Song2.mp3" Loop= "Infinite">
</BODY>
</HTML>
```

We can adjust the volume control by using the `Volume` attribute of `<BGSOUND>` to make the music softer or harder. Note that while creating HTML documents that include external files in `Src` and `Href` attributes, the correct path should be provided. If you try out the codes given in the examples, you have to make necessary modifications in these attributes. The attribute `Loop` determines the duration of play. The value `Infinite` causes the music play as long as the page is in view.

5.4 Creating tables in a web page

Sometimes, presenting a lot of information in a structured way is essential, especially in websites. Suppose we need to compare the data collected from a hospital for the period from 2012 to 2014, which states the number of newly diagnosed cancer patients among regular smokers, pan users and alcohol consumers. It can be presented in a tabular form as shown in Table 5.1.

NEWLY DIAGNOSED CANCER PATIENTS IN THE HOSPITAL FOR THE LAST 3 YEARS			
Year	**2012**	**2013**	**2014**
Smokers	129	140	143
Pan users	54	56	49
Alcohol users	74	68	77
Other cases	95	93	92

Table 5.1: Cancer patients diagnosed in a hospital

As we know, a table is a structured element that consists of rows and columns of cells. We can place any kind of content like text, image, and even other tables within these cells.In HTML, **<TABLE>** tag is used to create tables. This tag needs the support of some other tags like **<TH>**, **<TR>** and **<TD>** for constructing tables. Let us discuss these tags in the following sections.

5.4.1 <TABLE> tag

<TABLE> tag is a container tag. The whole content of the table should be enclosed within the tag pair <TABLE> and </TABLE>. This tag has many attributes to improve the general layout of the table. Some of the main attributes are listed below:

1. **Border**: This attribute specifies the thickness of the border line around the table. To draw the border of the table we have to specify a non-zero value to Border attribute in pixels. A border value of zero produces a table without border.
2. **Bordercolor**: It is used to assign colour to the table border.
3. **Align**: The position of the table inside the browser window is determined by Align attribute. The possible values are left (by default), right or center.
4. **Bgcolor**: We can change the background colour of the table using this attribute.
5. **Background:** It is used to assign a background image for a table. To set the background, specify the pathname of the image as value of this attribute. For example, <TABLE Background = "images/flower.gif"> sets the background of a table with the image in the file *flower.jpg* stored in the folder *images* in the current drive. When both Bgcolor and Background are specified, the Background will override Bgcolor.
6. **Cellspacing**: Table cells have space in between them. We can either increase or decrease the space between cells. The Cellspacing attribute determines the space to be left between cells. The value is given in number of pixels.
7. **Cellpadding**: There is a space between the content and cell border. We can increase or decrease the space between cell border and content. The Cellpadding attribute specifies the space in between the cell border and cell content. The value is given in number of pixels.
8. **Width and Height**: We can set a table width and height using Width and Height attributes. We can specify table width or height in terms of pixels or in terms of percentage of browser window.
9. **Frame**: This attribute with one of the values given in Table 5.2 indicates how table borders are displayed.

Value	Description
Void	Display no borders
Above	Display a border on the top of the table only
Below	Display a border on the bottom of the table only

Value	Description
Hsides	Display borders on the horizontal sides (top and bottom) only
lhs or rhs	Display the border only the left side or the right side
Vsides	Display borders on the vertical sides (right and left) only
box or border	Display borders on all sides of the table (It is the default value)

Table 5.2: Values of Frame attribute

10. **Rules**: We can use the Rules attribute to control what rules (borders between cells) are displayed in a table. Table 5.3 shows the values of Rules attribute.

Value	Description
none	Display no rules
cols	Display rules between columns only
rows	Display rules between rows only
groups	Display rules between row groups and column groups only
all	Rules will appear between all rows and columns

Table 5.3:Values of Rules attribute

Now let us discuss some other tags associated with <TABLE> tag.

5.4.2 <TR> tag

The rows in a table are created using <TR> tag. It is a container tag. The whole row is enclosed within the tag pair <TR> and </TR>. A <TR> tag always comes inside the <TABLE> tag. A row itself is a collection of cells. A cell is the smallest component of a table. There are two types of cells - heading cells and data cells. As seen in Table 5.1, the values given in red colour are of heading type. The values given in blue are just data cells.

5.4.3 <TH> tag

<TH> tag is used to define heading cells. It is also a container tag. The heading data should be enclosed between <TH> and </TH> tags. The heading cells are displayed in bold face and in centred form. <TH> tag always comes inside the <TR> tag.

5.4.4 <TD> tag

<TD> tag is similar to <TH> tag and is used to display data cells. It is also a container tag. The data is given in between <TD> and </TD> tags. Similar to <TH> tag,<TD> tag is also placed within the <TR> tag.

The code given in Example 5.12 creates a simple table and Figure 5.14 shows the resultant web page.

Example 5.12: To create a web page containing a simple table

```
<HTML>
<HEAD> <TITLE> Html Tables </TITLE>
</HEAD>
<BODY>
    <TABLE Border="1">
        <TR>
            <TH>Roll No</TH>
            <TH>Name</TH>
        </TR>
        <TR>
            <TD>1</TD>
            <TD>Aliya</TD>
        </TR>
        <TR>
            <TD>2</TD>
            <TD>Arun</TD>
        </TR>
    </TABLE>
</BODY>
</HTML>
```

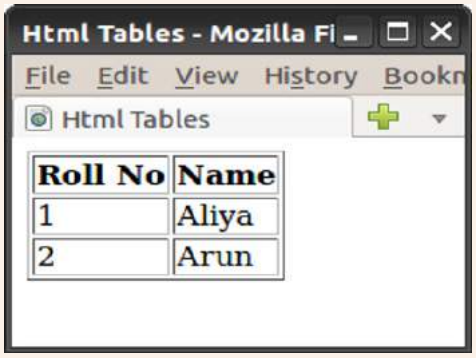


Fig. 5.14: Web page containing a simple table with two columns

Now, let us create the following table with some of the attributes discussed above.

The code given in Example 5.13 can create a web page as shown in Figure 5.15.

Year	2012 - 14
Smokers	412
Pan users	159
Alcohol users	219
Other cases	280

Example 5.13: To create a web page containing a table with border and colour

```
<HTML>
<HEAD> <TITLE> Hospital Table </TITLE> </HEAD>
<BODY>
    <TABLE Border= "3" Bordercolor= "RED" Bgcolor= "#4EB0AF"
           Align= "left" Cellspacing= "16" Cellpadding= "5"
           Width= "50%">
        <TR>
            <TH> Year </TH>
            <TH> 2012-14 </TH>
        </TR>
```

```
        <TR>
            <TH> Smokers </TH>
            <TD> 412 </TD>
        </TR>
        <TR>
            <TH> Pan users </TH>
            <TD> 159 </TD>
        </TR>
        <TR>
            <TH> Alcohol users </TH>
            <TD> 219 </TD>
        </TR>
        <TR>
            <TH> Other cases </TH>
            <TD> 280 </TD>
        </TR>
    <TABLE>
<BODY>
<HTML>
```

Fig. 5.15: Table using Cellspacing and Cellpadding

In Example 5.13, we used <TH> tags within all <TR> tag pairs, which make the first column of the table similar to header column as shown in Figure 5.15.

Attributes of <TR> tag

The characteristics of a row can be changed using the attributes of <TR> tag.

1. **Align**: This attribute specifies the horizontal alignment of the text in a cell in that particular row. This can take the values `left`, `right` or `center`. The default is `left` for data and `center` for headings (see Figure 5.15).
2. **Valign**: We can specify the vertical alignment of the content in a cell of any row using `Valign`. The possible values are `top`, `middle`, `bottom` or `baseline`. Baseline vertical alignment aligns the baseline of the text across the cells in the row.
3. **Bgcolor:** This attribute gives background colour to a particular row. Usually this helps in highlighting a row.

The following code segment is the modified part of the code given in Example 5.13. It modifies the third row of the table shown in Figure 5.15 with a background colour, and horizontal and vertical alignments.

```
<TR Bgcolor= "yellow" Align= "right" Valign= "middle">
    <TH> Pan users </TH>
    <TD> 159 </TD>
</TR>
```

The modified web page is shown in Figure 5.16.

Fig. 5.16: Table with modified row using Bgcolor, Align and Valign

Attributes of <TH> and <TD> tags

Most of the attributes of <TH> and <TD> tags are common, as both these tags are used for creating table cells. Let us discuss some of the main attributes.

1. **Align**: It specifies the horizontal alignment of the content within a cell. It can take the values `left`, `right` or `center`. The default value is `center` for <TH> and `left` for <TD>.
2. **Valign**: It specifies the vertical alignment of the content within a cell. It can take a value `top`, `bottom`, `middle` or `baseline`.
3. **Bgcolor**: We can set background colour for any cell using this attribute. The tags <TABLE>, <TR>, <TD>/<TH> all have `Bgcolor` attribute. If `Bgcolor` attribute is set for all these tags, the cell will take the colour assigned to this attribute in <TH>/<TD> tag.
4. **Colspan**: Usually a cell spans over a single column, but sometimes we may need columns occupying more than one cell. `Colspan` value defines the number of columns occupied by that particular cell. For example, `<TH Colspan="3">` makes the cell span over three columns.
5. **Rowspan:** Similar to `Colspan`, `Rowspan` attribute specifies the number of rows to be spanned by the cell. The default cell span is single row, but we can make it span multiple rows by setting `Rowspan` attribute a numeric value greater than one. For example, `<TD Rowspan="4">` makes the cell span over four rows.

Let us create a web page with a simple table to illustrate the effect of the attributes discussed above. The code given in Example 5.14 shows the use of these attributes and Figure 5.17 shows the corresponding web page.

Example 5.14: To create a table to demonstrate Colspan and Rowspan

```
<HTML>
<HEAD> <TITLE> Students Registration </TITLE>
</HEAD>
```

```
<BODY>
    <TABLE Border= "1" Cellspacing= "3" Cellpadding= "5">
        <TR>
            <TH Colspan= "3"> No. of Registered Students </TH>
        </TR>
        <TR>
            <TH Rowspan= "2"> Year </TH>
            <TD> 2014 </TD> <TD> 75 </TD>
        </TR>
        <TR>
            <TD> 2015 </TD> <TD> 88 </TD>
    </TR>
    </TABLE>
<BODY>
<HTML>
```

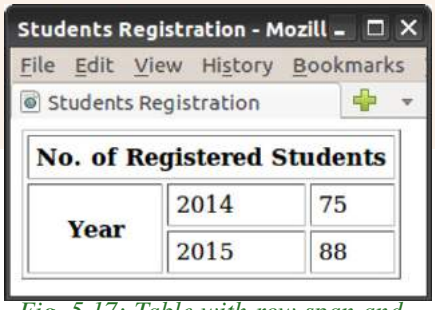


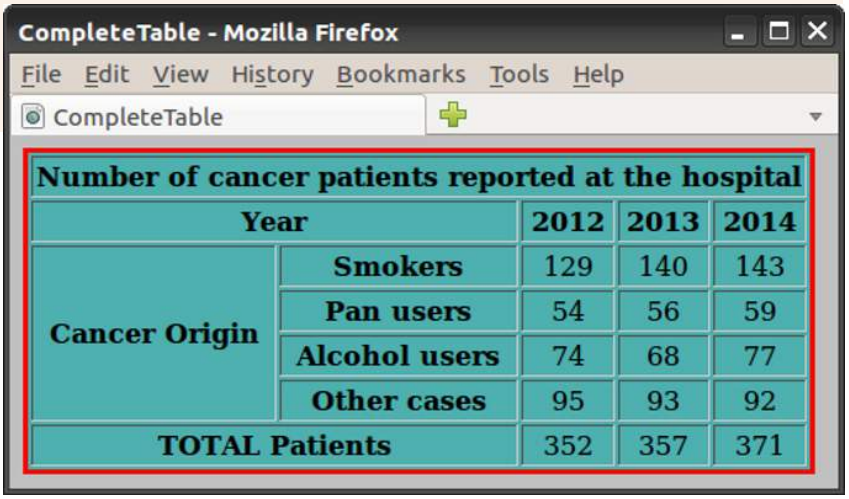
Fig. 5.17: Table with row span and column span

Now, let us create a web page to display the details of cancer patients as shown in Table 5.1. The code given in Example 5.15 can design a table as shown in Figure 5.18.

Example 5.15: To create a table to show the data of cancer patients

```
<HTML>
<HEAD> <TITLE> CompleteTable </TITLE> </HEAD>
<BODY Bgcolor= "silver">
    <TABLE Border= "3" Bordercolor= "red" Bgcolor= "#4EB0AF"
           Align= "left" Cellspacing= "2" Cellpadding= "2"
           Width= "50%">
        <TR>
            <TH Colspan= "5"> Number of cancer patients reported
                              at the hospital </TH>
        </TR>
        <TR Align= "center">
            <TH Colspan= "2"> Year </TH>
            <TH> 2012 </TH>
            <TH> 2013 </TH>
            <TH>  2014 </TH>
        </TR>
        <TR Align= "center">
            <TH Rowspan= "4"> Cancer Origin </TH>
            <TH> Smokers </TH>
            <TD> 129 </TD>
```

```
            <TD> 140 </TD>
            <TD> 143 </TD>
        </TR>
        <TR Align= "center">
            <TH> Pan users </TH>
            <TD> 54 </TD>
            <TD> 56 </TD>
            <TD> 59 </TD>
        </TR>
        <TR Align= "center">
            <TH> Alcohol users </TH>
            <TD> 74 </TD>
            <TD> 68 </TD>
            <TD> 77 </TD>
        </TR>
        <TR Align= "center">
            <TH> Other cases </TH>
            <TD> 95 </TD>
            <TD> 93 </TD>
            <TD> 92 </TD>
        </TR>
        <TR Align= "center">
            <TH Colspan= "2"> TOTAL Patients </TH>
            <TD> 352 </TD>
            <TD> 357 </TD>
            <TD> 371 </TD>
        </TR>
    </TABLE>
</BODY>
</HTML>
```



Number of cancer patients reported at the hospital				
Year		2012	2013	2014
Cancer Origin	Smokers	129	140	143
	Pan users	54	56	59
	Alcohol users	74	68	77
	Other cases	95	93	92
TOTAL Patients		352	357	371

Fig. 5.18: Web page containing a table of cancer patients Bgcolor, Rowspan, Colspan, Align and Valign

Let us do

Try to create a table containing the number of students studying different second languages available in your school. The table should provide details of each class of your higher secondary section.

5.4.5 Table caption with <CAPTION> tag

We can provide a heading to a table using the **<CAPTION>** tag. It provides an easy method to add descriptive text to a table as its caption. Suppose we want to modify the table shown in Figure 5.19 with a caption. We can modify the code section before the first **<TR>** tag in Example 5.15 with following code segment:

```
<TABLE Border= "3" Bordercolor= "red" Bgcolor= "skyblue"
       Align= "left" Cellspacing= "2" Cellpadding= "2"
       Width= "50%">
<CAPTION> Number of new cancer patients reported at the
          hospital during 2012-14
</CAPTION>
```

The modified HTML code will produce a table as shown in Figure 5.19.

CompleteTable - Mozilla Firefox

File Edit View History Bookmarks Tools Help

CompleteTable

Number of new cancer patients reported at the hospital during 2012-14

Number of cancer patients reported at the hospital				
Year		2012	2013	2014
Cancer Origin	Smokers	129	140	143
	Pan users	54	56	59
	Alcohol users	74	68	77
	Other cases	95	93	92
TOTAL Patients		352	357	371

Fig. 5.19: Modified table with a caption

Know your progress

1. Name any two associated tags of <TABLE> tag.
2. Choose the odd one out:
   a. TABLE b. TR c. TH d. COLSPAN
3. Differentiate between <TD> and <TH>.
4. <TABLE> tag is an empty tag. State whether this statement is true or false.
5. Give any two attributes of <TR> tag.

5.5 Dividing the browser window

Sometimes we need to include more than one webpage inside a single browser window. The browser window can be divided into two or more panes to accommodate different pages simultaneously. HTML provides a facility called frameset to partition the browser window into different sections. Each section accommodates different HTML pages. Each individual section created by a frameset is called a frame. Figure 5.20 shows a simple frameset consisting of three frames. To create a frameset, we need **<FRAMESET>** and **<FRAME>** tags.

Fig. 5.20: A simple frameset with three frames

5.5.1 <FRAMESET> tag

<FRAMESET> is a container tag for partitioning the browser window into different frame sections. The frames are defined within the tag pair `<FRAMESET>` and `</FRAMESET>`. The main attributes are:

1. **Cols**: This attribute determines the number of vertical frames in the frameset page and its dimensions. The column width can be given either in percentage of total width or in number of pixels. For example, `<FRAMESET Cols = "30%,500,*">` will create three vertical frames. The first frame will occupy 30% of the total window width, the next will take 500 pixels space, and the asterisk(*) symbol given at the end denotes the remaining space which is given to the third frame.
2. **Rows**: Similar to `Cols`, `Rows` attribute defines the number and dimension of horizontal frames.
3. **Border:** We can specify the thickness of border for the frames by using `Border` attribute. Here the value is given in pixels.
4. **Bordercolor:** We can specify border colour by assigning a colour to this attribute.

5.5.2 <FRAME> tag

<FRAME> tag is an empty tag and it defines the frames inside the `<FRAMESET>`. For each division inside the `<FRAMESET>` tag, we should have a corresponding `<FRAME>` tag. `<FRAME>` tag always comes with **Src** attribute which specifies the HTML page to be loaded in the frame. The main attributes related to `<FRAME>` tag are:

1. **Src**: As we have already discussed, Src specifies the URL of the document to be loaded in the frame. For example, `<FRAME Src = "school.html">` loads the page *school.html* in a particular frame.
2. **Scrolling**: When the content inside a frame exceeds the frame size, a scrollbar appears depending upon the value of the `Scrolling` attribute. It can take values `Yes`, `No` or `Auto`. `Auto` is the default value and it displays scroll bars if the frame content exceeds the normal display area.
3. **Noresize**: It is used to prevent users from resizing the border of a specific frame by dragging on it. Just giving `Noresize` prohibits resizing the frame window. For example, `<FRAME Src= "school.html" Noresize>`.
4. **Marginwidth** and **Marginheight**: We can set horizontal and vertical margins to the frame by using `Marginwidth` and `Marginheight`, respectively. The values are given in pixels.
5. **Name**: It gives a name to a frame. This particular frame can be referenced using this name later in the code.

The following code can create the frameset shown in Figure 5.20. It is assumed that the three HTML pages *sampleframe1.html*, *sampleframe2.html* and *sampleframe3.html* are already created as in Figure 5.20.

```
<HTML>
<HEAD> <TITLE> A simple Frameset </TITLE> </HEAD>
    <FRAMESET Rows= "20%, 30%, 50%">
        <FRAME Src= "sampleframe1.html">
        <FRAME Src= "sampleframe2.html">
        <FRAME Src= "sampleframe3.html">
    </FRAMESET>
</HTML>
```

5.5.3 Targeting frames

In a frameset page, we can provide hyperlink in any frame and the linked page can be targeted to any other frame. When we click on the link in a frame, the corresponding page is opened within another frame in the frameset. For this, first we have to allot a name for the destination frame using its `Name` attribute. Then we can refer the named frame using `Target` attribute of the link to be displayed.

Let us do

Create three HTML files *bio.html*, *poem.html* and *fiction.html* to show a list of books belonging to biography, poetry and fiction, respectively.

Now let us create a web page with two frames - one for showing the links for these files and the other for opening the respective web pages. When a user clicks on any one of the three links available in one frame, the associated file will open in the other frame.

Once you have created the three files mentioned above, the two HTML codes given in Example 5.16 can satisfy our need.

Example 5.16: To illustrate the concept of targeting frames

Save the following code in a file ***main.html***

```
<HTML>
<HEAD> <TITLE> Left Frame </TITLE> </HEAD>
<BODY Bgcolor= "#00AFFF" Text= "#282D2F">
    <H2> Select the option </H2>
    <FONT Size= "5">
    <A Href= "bio.html" Target="right_frame">Biography</A><BR>
    <A Href= "poem.html" Target="right_frame">Poetry</A><BR>
    <A Href= "fiction.html" Target="right_frame">Fiction</A>
    </FONT>
</BODY>
</HTML>
```

Now create a file to store the following code:

```
<HTML>
<HEAD> <TITLE> Targeting Frames </TITLE> </HEAD>
<FRAMESET Cols= "200, *">
    <FRAME Src= "main.html" Name= "left_frame">
    <FRAME Name= "right_frame">
</FRAMESET>
</HTML>
```

When the above document is executed, a web page with two frames will be opened. The first column of the browser window will show the *main.html* and the second column will be blank. Note that **Src** attribute is not specified for that frame, but a name `right_frame` is assigned. If the first link (biography) is selected, the file *bio.html* will be opened in the second frame as shown in Figure 5.21. It is

Fig. 5.21 : Web page with target frame

assumed that the file *bio.html* contains the details as shown in the figure. Note the difference in colour in the selected link.

Note that, the HTML document main.htm contains three pairs of <A> tag to link the three HTML files. A tag named **Target** is used in <A> tags to specify the frame in which the linked file is to be opened. Here comes the relevance of names given to the frames.

<FRAMESET> tag is not supported in HTML 5 version.

There are many arguments against the use of frameset. Some of them are given below.

- The browser's back button will not work naturally inside a frameset.
- It is difficult to refer a specific document within a frameset.
- Any attempt to reload a specific frame may reload the entire frameset. As a result it will reset the frame contents to their default sources.
- Search engines can struggle when navigating through framed documents.
- It is difficult to print contents of a frameset.
- Bookmarking contents of a frameset is difficult.

5.5.4 Nesting of framesets

In Figures 5.20 and 5.21, we can see frames in the browser window. The first figure shows horizontal division of the browser window into three frames. But the second figure shows vertical division into two frames.

Suppose we want to divide the browser window as shown in Figure 5.22. It is possible with nested frameset. It is the process of inserting a frameset within another frameset.

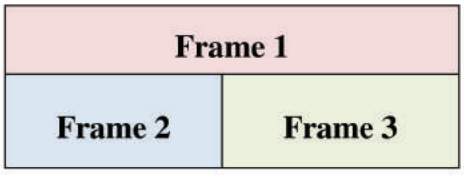


Fig. 5.22: Nesting of frameset

The following steps illustrate the nesting of frameset for the division shown in Figure 5.22.

1. Create the initial frameset, defining two rows as shown below:

```
<FRAMESET Rows= "85, *">
    <FRAME Src= "sampleframe1.html">
</FRAMESET>
```

This code horizontally divides the window into two frames and the first row is reserved for the file *sampleframe1.html*.

2. The second row is left free. Now we will divide it into two frames vertically. So, instead of a second <FRAME> tag, insert another opening <FRAMESET> tag as shown below:

```
<FRAMESET Rows= "85, *">
    <FRAME Src= "sampleframe1.html">
    <FRAMESET Cols= "220, *">
    </FRAMESET>
</FRAMESET>
```

This code will divide the second row into two columns. Now, we can add two <FRAME> tags within the inner frameset and complete the HTML code as given in Example 5.17. The resultant web page is shown in Figure 5.23. It is assumed that the three HTML pages are already created as in Figure 5.23.

Example 5.17: To implement the concept of nested frameset

```
<HTML>
    <HEAD>
        <TITLE> nesting frames </TITLE>
    </HEAD>
    <FRAMESET Rows= "85, *">
        <FRAME Src= "sampleframe1.html">
        <FRAMESET Cols= "200, *">
            <FRAME Src= "sampleframe2.html">
            <FRAME Src= "sampleframe3.html">
        </FRAMESET>
    </FRAMESET>
</HTML>
```

5.5.5 <NOFRAMES> tag

Earlier browsers did not support frames. In such a situation, the browser is expected to respond to the user in some way or the other. The tag pair **<NOFRAMES>** and **</NOFRAMES>** is used to display some text content in the window if the browser is unable to support frames. The following code illustrates the use of <NOFRAMES> tag.

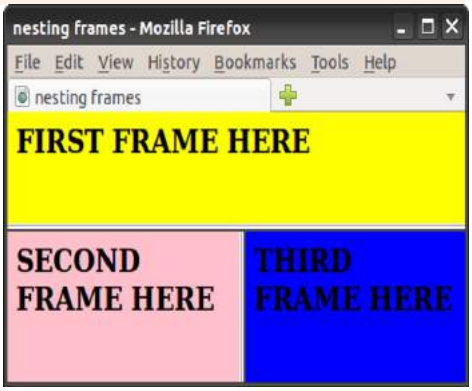


Fig. 5.23: Nested framesets

```
<HTML>
    <HEAD> <TITLE> A simple Frameset </TITLE>
    </HEAD>
```

```
    <FRAMESET Rows= "20%, 30%, 50%">
        <FRAME Src= "sampleframe1.html">
        <FRAME Src= "sampleframe2.html">
        <FRAME Src= "sampleframe3.html">
    </FRAMESET>
    <NOFRAMES>
        <P> Your browser doesnot support frames.<BR>
            Click <A Href="index.htm">here... </A></P>
    </NOFRAMES>
</HTML>
```

Here, if the browser does not support <FRAMESET> then a message "Click here…" will be displayed with an alternate link to the file *index.html*.

Know your progress

1. <FRAMESET Rows="100,*"> divides the frame into ____ sections.
2. List any three attributes of <FRAME> tag.
3. What is nested frameset?
4. What is the use of <NOFRAME> tag?
5. No <BODY> section is needed for frameset page. Say TRUE or FALSE.

5.6 Forms in web pages

HTML Forms are required when we have to collect some data from the webpage viewer for processing. For example, we use www.hscap.kerala.gov.in for the admission to Class XI in Kerala. We enter information such as name, SSLC Register Number, grades, choice of course and school, etc. in this web page. It is an implementation of HTML Form.

A Form will take input from the web page viewer and then it will post the data to a back-end application such as Common Gateway Interface (CGI), Active Server Pages (ASP), PHP, etc. for processing. We will study about these technologies in the next chapter. A form consists of two elements: a **<FORM>** container and any number of Form controls within that container. There are various Form controls like text fields, textarea fields, drop-down menus, radio buttons, checkboxes, etc.

5.6.1 <FORM> tag

A **<FORM>** tag provides a container for creating a Form. An HTML Form starts with <FORM> and ends with </FORM> tag. A web browser can only gather information through Forms. We must provide some back-end application to handle (save, modify, etc.) the data collected. We use Form handlers like CGI, JavaScript or PHP script for that purpose. All of the input elements associated with a single Form are processed by the same Form handler. A Form handler is a program on the web server that manages the data sent through the Form. The form handler is specified as the value for the Action attribute of <FORM> tag. The concept of server and other associated technologies will be discussed in the next chapter.

Most frequently used attributes of <FORM> are:

1. **Action**: It specifies the URL of the Form handler, which is ready to process the received data.
2. **Method**: It mentions the method used to upload data. The most frequently used Methods are Get and Post.
3. **Target**: It specifies the target window or frame where the result of the script will be displayed. It takes values like _blank, _self, _parent, etc. The main values of Target attribute are given in Table 5.4.

Value	Description
_blank	Opens the linked document in a new browser window
_self	Opens the linked document in the same frame as the link
_parent	Opens the linked document in the parent frameset
_top	Opens the linked document in the main browser window, replacing any frame present
name	Opens the linked document in the window with the specified name

Table 5.4: Values for Target attribute

A common use for JavaScript (or any client side scripting language) is to verify that users have filled in all the required fields in a Form and/or to check whether the input data is valid or not, before the browser actually submits the Form to the form handler on the web server.

Form controls

There are different types of Form controls that we can use to collect data using HTML Form. These controls include Text box, Password, Check Box, Radio Button,

Text Area, Select Box, Submit and Reset Button.We can create most of these controls in any HTML Form using **<INPUT>** tag.

5.6.2 <INPUT> tag

The visible part of a Form is a set of controls to accept the input from the viewers. Different types of input can be selected based on the nature of input. The **<INPUT>** tag is an empty tag that can be used to make different types of controls such as Text Box, Radio Button, Submit Button etc. The **Type** attribute determines the type of control created by this tag.

Attributes of <INPUT> tag

1. **Type**: This attribute determines the control type created by the <INPUT> tag. The main values of Type are given in Table 5.5.

Value	Description
text	creates a text box
password	same as text box. But here characters are represented by coded symbols such as asterisk.
checkbox	creates a checkbox where user can enter Yes or No values (check or uncheck).
radio	similar to checkbox but is used to select a single value from a group of values. When multiple radio buttons are assigned with the same value for Name attribute, users can select only one button at a time. When the user changes the selection, the one that is already selected becomes deselected.
reset	a special button used to clear all the entries made in the form and to bring it to the initial state.
submit	another special button used to submit all the entries made in the form to the server.
button	creates a standard graphical button on the form. We can call functions on clicking this button.

Table 5.5: Values of Type attribute

2. **Name**: It is used to give a name to the input control. When the Form is submitted, the data values are passed to the server along with the corresponding name of the control.
3. **Value**: It can be used to provide an initial (default) value inside the control.
4. **Size**: This attribute sets the width of the input text in terms of characters. It is applicable only to the input types text and password.

5. **Maxlength**: It limits the number of characters that the user can type into the field. It is also applicable only to the text and the password.

A password field helps in hiding the content visually. It doesn't encrypt or scramble the information in any way.

The HTML code given in Example 5.18 creates an HTML Form to input a name and password. Figure 5.24 shows the resultant web page.

Example 5.18: To create an HTML Form

```
<HTML>
<HEAD> <TITLE> Login </TITLE> </HEAD>
<BODY Bgcolor= "Pink">
    <FORM Action= "login.php" Method= "post">
        <P> Name:    
        <INPUT Type= "text" Name= "firstname" Size= "30"
               Maxlength= "25"> </P>
        <P>Password:
        <INPUT Type= "password" Name= "Psswd" Size= "30"
               Maxlength= "25"> </P> <BR>
                       
         
        <INPUT Type= "reset" Value= "Clear">
        <INPUT Type= "submit" Value= "Send">
    </FORM>
</BODY>
</HTML>
```

In Example 5.18, a file *login.php* is assigned to the `Action` attribute of the `<FORM>` tag. It means that the data entered through this Form will be captured and processed by the application stored in the file *login.php*. Also observe the use of HTML entity ` ` to provide non break spaces by referring to the code and Figure 5.24.

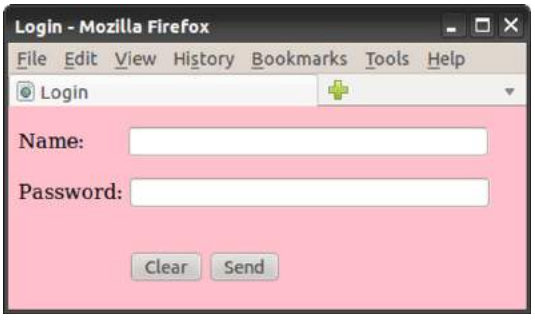


Fig. 5.24: HTML Form with text and password

Let us design an HTML Form to input data with the help of radio button (or option button) and check boxes. You might have heard that radio buttons are used to specify only one option from a set of alternatives, and that check boxes are used to mark one or more items in a given list.

The code given in Example 5.19 creates a Form to specify the gender and hobbies of a person with the help of radio buttons and check boxes. Figure 5.25 shows the corresponding web page.

Example 5.19: To create an HTML Form with radio buttons and check boxes

```
<HTML>
<HEAD> <TITLE>Checkbox Radio Button Control</TITLE> </HEAD>
<BODY Bgcolor= "#E9BEE5">
    <FORM> <BR>Sex:       
        <INPUT Type= "radio" Name= "sex" Value= "male"> Male
        <INPUT Type= "radio" Name= "sex" Value= "female"> Female
        <BR> <BR>Hobbies:
        <INPUT Type= "checkbox" Name= "Hobby" Value= "Games">
        Playing Games
        <INPUT Type= "checkbox" Name= "Hobby"
               Value="WatchingTV"> Watching TV
        <INPUT Type= "checkbox" Name= "Hobby" Value= "Reading">
        Reading
    </FORM>
</BODY>
</HTML>
```

Fig 5.25:Form with checkbox and radio button

5.6.3 <TEXTAREA> tag

We have seen single line text entry facility in a Form with the statement `<INPUT Type = "text">` (refer Example 5.18 and Figure 5.24). But in practice, there are occasions where we need to enter multiple lines of text in a Form. Postal address is an example. The container tag **<TEXTAREA>** can be used for this purpose. The area

between the tag pair gives space for multi line text depending on the values given to the attributes. The main attributes of <TEXTAREA> tag are:

1. **Name**: It is used to give a name to the control.
2. **Rows**: It specifies the number of rows in a text area control.
3. **Cols**: It indicates the number of columns in a text area, i.e., number of characters in a line.

Consider the following code segment and observe the usage of <TEXTAREA> tag:

```
<FORM Action= "guestbook.php" Method= "post">
    <TEXTAREA Rows= "10" Cols= "30" Name= "address">
        Enter your address.
    </TEXTAREA>
</FORM>
```

On completing the code and opening it, we can see a page as shown in Figure 5.26. Any text we include between tag pair <TEXTAREA> and </TEXTAREA> appears in a text box. When the user enters any data in the text box, it overrides default text.

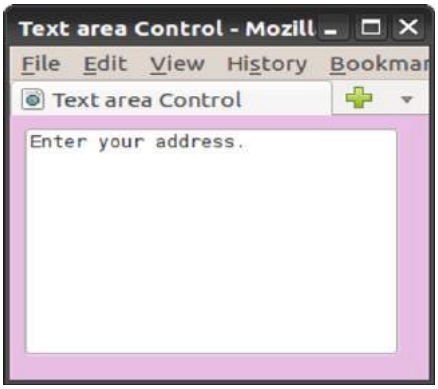


Fig. 5.26: HTML Form with text area

5.6.4 <SELECT> tag

A select box, also called dropdown box, provides a list of various options in the form of a dropdown list, from where a user can select one or more options. Select box is helpful when a number of options are to be displayed in a limited space. The options in the list are specified using **<OPTION>** tag, which will be contained in the **<SELECT>** tag pair.

The container tag pair <SELECT> and </SELECT> encloses a select box. The main attributes of <SELECT> tag are:

1. **Name**: It gives a name to the control, which is sent to the server to be recognized and to get the value.
2. **Size**: This can be used to present a scrolling list box. Its value will decide whether the select box should be a drop down list or a list box. If the value is 1, we get a dropdown list (or combo box).
3. **Multiple**: It allows users to select multiple items from the menu.

Now let us discuss the role of <OPTION> tag in select boxes. It is an empty tag placed inside the container tag <SELECT> and </SELECT>. It lists out the options provided in the select box. The main attributes of <OPTION> tag are:

1. **Selected**: This attribute is used to indicate default selection.
2. **Value**: It is used here to allow the submission of a value that differs from the content of the <OPTION> tag. If it is not present, the content is used as the value.

The HTML code given in Example 5.20 creates a web page that contains a drop down list. Note that the value of the Size attribute of <SELECT> tag is 1 and there are four options inside the select box.

Example 5.20: To create an HTML Form with a drop down list

```
<HTML>
<HEAD> <TITLE> Drop down list </TITLE> </HEAD>
<BODY Bgcolor= "#E9BEE5">
    <FORM Action= "guestbook.php" Method= "post">
        <P> Nationality:
        <SELECT Name= "Nationality" Size= "1">
            <OPTION Value= "Indian" selected> Indian
            <OPTION Value= "British"> British
            <OPTION Value= "German"> German
            <OPTION Value= "Srilankan"> Srilankan
        </SELECT>
    </FORM>
</BODY>
</HTML>
```

Figure 5.27 shows the dropdown list obtained on clicking the combo button.

Fig. 5.27: Select box in a Form

Let us do

Modify the code given in Example 5.20 with different values for the Size attribute and predict the changes in the select box.

5.6.5 Grouping Form data with <FIELDSET> tag

Sometimes grouping related controls in a Form become a necessity. The **<FIELDSET>** element helps in grouping related data in a Form. By using <FIELDSET> we can divide a Form into different subsections, each subsection containing related elements. To identify each <FIELDSET>, we can use the **<LEGEND>** tag. The <LEGEND> tag defines a caption for the <FIELDSET> element. It is a container tag.

Now let us create a Form with all the elements discussed so far, to submit the details such as name, age, sex, address, hobbies etc. of a student. The code given in Example 5.21 can be used for this so that an HTML Form as shown in Figure 5.28 will be obtained.

Example 5.21: To create an HTML Form to submit the details of a student

```
<HTML>
<HEAD> <TITLE> FormResume </TITLE> </HEAD>
<BODY Bgcolor= "#E9BEE5">
    <CENTER ><H3>Enter your details</H3></CENTER>
    <FORM Action= "guestbook.php" Method= "get">
        Name: 
        <INPUT Type= "text" Name= "first_name" Size= "20"
               Maxlength= "20" Value= "First Name Here"><BR><BR>
        Age:
        <INPUT Type="text" Name="age" Size="3" Maxlength="3"><BR>
        Sex:     
        <INPUT Type="radio" Name="sex" Value="male"> Male
        <INPUT Type="radio" Name="sex" Value="female"> Female
        <FIELDSET>
            <LEGEND>Nationality</LEGEND>
            <SELECT Name= "Nationality" Size= "4">
                <OPTION Value= "Indian" Selected> Indian
                <OPTION Value= "British"> British
                <OPTION Value= "German"> German
                <OPTION Value= "Srilankan"> Srilankan
            </SELECT><BR><BR>
            Nativity:
            <INPUT Type= "text" Name="State" Size="15"><BR><BR>
            District: 
            <INPUT Type= "text" Name= "District" Size= "15">
        </FIELDSET><BR>
        Hobbies:
        <INPUT Type= "checkbox" Name= "Hobby" Value= "games">
               Playing Games
        <INPUT Type= "checkbox" Name= "Hobby"
                Value= "WatchingTV"> Watching TV
        <INPUT Type= "checkbox" Name= "Hobby" Value= "Reading">
               Reading<BR><BR>
```

```
        <TEXTAREA Rows= "5" Cols= "25" Name= "address">Address
        </TEXTAREA><BR><BR>
        <INPUT Type= "submit" Value= "submit">
        <INPUT Type= "Reset" Value= "reset">
    </FORM>
</BODY>
</HTML>
```

Fig 5.28:An HTML Form to submit the details of a student

5.6.6 Form submission

When we click the submit button on the Form, it will send the collected input data to the server, a computer capable of processing the received data. In a web server, there are special server side programs for processing Form data coming from the browser of a client (a computer, a tab or a mobile phone through which we submit the data).

As mentioned earlier, the `Action` attribute defines the action to be performed when the Form is submitted. Normally it assigns the URL of the server side program to process the Form data. The common way to submit a Form to a server is by using a **submit** button. In Example 5.21, a server-side script *guestbook.php* is specified to handle the submitted form in the `<FORM>` tag.

The Method attribute of the <FORM> tag specifies the HTTP method (get or post) to be used when submitting the forms.

Know your progress

1. HTML provides _____ to input data through web pages.
2. Name the two tags used within <FORM> to enter text data.
3. How do radio button control and check box control differ?
4. Which tag is used to group data within the Form?
5. Name the tag used within <FORM> to input data.

5.7 Overview of HTML 5

HTML 5 is the major revision of the HTML standard after HTML 4.01. HTML5 was developed jointly by World Wide Web Consortium (W3C) and the Web Hypertext Application Technology Working Group (WHATWG). The new standard incorporates features like video playback, drag-and-drop etc.

The latest versions of all leading browsers including Google Chrome, Mozilla Firefox, Apple Safari, Opera and Internet Explorer support HTML5. Mobile web browsers that are pre-installed in iPhones, iPads, Android phones, etc. also support HTML5. HTML5 is designed, as much as possible, to be backward compatible with existing web browsers. The logo of HTML5 is given in Figure 5.29.

Fig. 5.29: HTML5 logo

Some new tags introduced in HTML5 are given below:

1. **<VIDEO> and <AUDIO>**: These tags offer the facility to easily embed media into HTML documents.
2. **<CANVAS>**: This tag gives us an easy and powerful way to draw graphics, build charts and graphs and customise graphics.
3. **<HEADER> and <FOOTER>**: The <HEADER> element specifies a header for a document or section. The <FOOTER> element specifies footer for a document or section. A footer typically contains the author of the document, copyright information, contact information, etc. We can have several <HEADER>, <FOOTER> elements in one document.
4. **<ARTICLE> and <SECTION>**: These tags are used to create articles and sections within a webpage. An article is an independent, stand-alone piece of discrete content like a blog post, or a news item. Article represents a full block of content. Section is used as a way of sectioning a page into different subject areas, or sectioning an article. <ARTICLE> and <SECTION> tags, if properly used will increase search engine visibility of the webpage.

5. **<OUTPUT>**: This tag represents the result of a calculation usually performed by a script.
6. **<DETAILS>**: Sometimes website needs to have an expanding/collapsing block of text. With <DETAILS> tag it is easier to make a simple block that expands and collapses the content when the header is clicked.
7. **<FIGURE>** and **<FIGCAPTION>**: <FIGURE> is a container for content like images, and <FIGCAPTION> element represents a caption or legend for a figure. It comes inside the <FIGURE> tag.
8. **<PROGRESS>** and **<METER>**: <PROGRESS> and <METER> are similar. The <PROGRESS> tag represents the progress of a task. It is useful for things like displaying the progress of a file upload. <METER> tag is used to display a scalar measurement within a known range, like showing hard drive usage.

Let us conclude

In this chapter, we have gone through some of the advanced features of HTML. Different kinds of lists are discussed to make the text more presentable. The beauty of the web pages can be enhanced by including marquees, images, audio and video. We have also discussed the importance of hyper linking and familiarised with various kinds of hyperlinks. Creating a table with the tags <TABLE>, <TR>, <TH>, and <TD>, and their attributes were discussed. The importance of Rowspan and Colspan attributes are identified. We can divide the browser window into different frames using framesets so that different pages can be opened and viewed at the same time. We have also seen that division of browser window can be done in different ways with the concept of nested framesets. We have discussed how Target attribute is used to create link between different frames of the browser window. We have also identified the utility of Form in the web page to submit data to the server using online facility. We have seen various elements such as text box, password, radio button, text area, select box, etc. that facilitate the input of data in different ways. We have just mentioned the concept of client side programs and server side programs. So, this chapter is a stepping stone to the next chapters in which we will discuss how the input data in the Form is verified and how the data is stored or processed in the server.

Let us practice

1. Wrtie an HTML code for a web page of a district in Kerala with the following details and features:
   - A heading followed by a paragraph of 3 sentences about the district using text formatting tags and attributes.
   - A list of the tourist places in the district.
2. Wrtie an HTML code for a web page for your school with the following details and features:
   - A heading followed by a paragraph of 3 sentences about the district using text formatting tags and attributes.
   - Include a list of five co-curricular activites like NCC, NSS, Clubs, etc.
3. Wrtie an HTML code for a web page to show the following details:

Components of a Computer

- **Hardware**
  1. I/O Devices
  2. RAM
  3. Hard Disk & DVD Drive
- **Software**
  1. Operating System
  2. Application Programs

4. Write an HTML code to present some details about Onam - the festival of Kerala, in a web page with the following features:
   - A heading with attractive font characteristics.
   - An image of "Vallam Kali" (boat race) in the background of the page.
   - Internal links to any two of the traditional events like "Pookkalam", "Thumpi Thullal", "Thiruvathira", "Onavillu", "Vallam kali", "Kummatti", "Pulikali", etc.
5. Write HTML codes to create two web pages to show some information about the higher secondary and high school sections of your school. Create another web page to divide the browser window horizontally into two. In the first frame, a brief introduction of the school and two links are to be provided - one for HSS and the other for HS. On clicking these links the respective web page is to be opened in the second frame.

6. Write an HTML code to show the following table in a web page and also provide an external link to the website of Kerala Police given below the table:

Road Accidents in Kerala during 2012 - 2014

Year	Total Number of		
	Cases	Persons Killed	Persons Injured
2012	36174	4286	41915
2013	35215	4258	40346
2014	36282	4049	41096

Data Source: www.keralapolice.org

7. Write an HTML code to display an application form as shown below:

APPLICATION FOR THE BEST STUDENT AWARD

Name: _____ Sex: Male ○ Female ○

Class & Division: Select ▼

Total Grade Point in Class XI: _____

Average Grade Point in Termly Exams in Class XII: _____

Cocurricular Activities:

NCC ☐ NSS ☐ Sports ☐ Arts ☐ Literary ☐

Other Achievements: _____

Let us assess

1. Find the errors in the following and correct it.

a.
```
<UL Type = "i" Start = 3>
```

b.
```
<IMG Src = "Myschool.jpg" Size  = "50" >
```

c.
```
<HTML>
      <HEAD><TITLE><HEAD></TITLE>
      <BODY> this is the body of the HTML
        document</BODY>
</HTML>
```

2. Rohit created a table in HTML but a border was not visible. What could be the reason?
3. _____ attribute is used with <A> tag to display the linked page in a specified frame.
4. Your computer teacher asked you to prepare a list of your best friends in a webpage. Which tag will you prefer? Write HTML code segment for this list.
5. Name the possible values of type attribute of <OL> tag.
6. Write the attributes of <UL> tag.
7. How do you create a list using upper case letters for numbering?
8. Suppose you want to create a list using lower case letters for numbering. How can this be made possible?
9. How can we create a list starting from 6 onwards?
10. Write the HTML code for creating the following webpage:

ABC Pvt. Ltd.
Kerala

1. Health Care
2. Baby Products
   - Toys
   - Dress
3. Ladies Wear
   - Kurthas
   - Jeans

11. Varun is creating a web page. He wants to create a link on the word 'sample' to a file named *sample.html* that is stored in a subdirectory *Exam* in *D drive*. Write the HTML command for this purpose.
12. Name the tag which has Noshade attribute.
13. Sunil developed a personal website, in which he has to create an e-mail link. Can you suggest the protocol used to achieve this link.
14. Shahir wants to connect his webpage to www.gmail.com. Write the tag and attribute required for this.
15. Mention the characteristics of two types of hyperlinks available in HTML.
16. Differentiate between Cellspacing and Cellpadding.
17. Differentiate between Text control and Textarea control used in Form.
18. Action and _____ are the main attributes of the <FORM> tag.

19. Say true or false:
    a. The default value of `Align` attribute of `<TABLE>` tag is `center`.
    b. `<FRAME>` is a container tag.
    c. Scrolling prevents users from resizing the border of a specific frame.
20. Which tag does allow partitioning of the browser window into different sections?
21. List down the main attributes of `<FRAME>` tag.
22. Create a frameset dividing the page into two sections vertically. Give names of your favorite football players in left frame. The profile of selected players should appear in the right Frame.
23. Predict the output of the following code:

```
<HTML>
<HEAD><TITLE>A simple table</TITLE></HEAD>
<BODY>
<TABLE border="1" Cellspacing= "1" Cellpadding= "10">
  <TR><TD> 1 </TD><TD> 2 </TD><TD> 3 </TD></TR>
  <TR><TD> 4 </TD><TD> 5 </TD><TD> 6 </TD></TR>
  <TR><TD> 7 </TD><TD> 8 </TD><TD> 9 </TD></TR>
</TABLE>
</BODY>
</HTML>
```

24. Write an HTML code to create the given table:

1	2	3
4	5	6
7		

25. Write an HTML code to accept your e-mail, address, phone number and password.
26. Write an HTML code to create a list of 3 bikes of your choice in a frame. Link each one to display the description with a picture in another frame.
27. Raju created a web page as follows. But he is unable to view any tabular format in the web page, when it is displayed in the browser. Find out the reason for it and correct it.

```
<HTML>
<HEAD><TITLE> My Page </TITLE></HEAD>
<BODY>
<TABLE><TR><TH>Roll No. </TH><TH> Name </TH></TR>
<TR><TD>1 </TD><TD> Huda </TD></TR>
<TR><TD>2 </TD><TD>Bincy</TD></TR>
</TABLE>
</BODY>
</HTML>
```

6 Client Side Scripting Using JavaScript

Significant Learning Outcomes

After the completion of this chapter, the learner

- distinguishes the use of client side and server side scripting language.
- explains the need of client side scripting language.
- identifies the importance of JavaScript as the client side scripting language.
- uses JavaScript functions in a web page.
- explains different data types in JavaScript.
- uses correct variables in JavaScript.
- uses appropriate control structures in program codes.
- uses appropriate built-in functions in JavaScript.
- explains the method to access Document Elements using javaScript.
- creates JavaScript functions that handle values in text boxes and combo boxes.

In the previous chapters we learned to create different types of web pages containing texts and graphics. Since this is the world of the Internet, most of us might have visited various websites in the Internet for different purposes. These web pages contain features that we have not learned. Creating such types of web pages requires the knowledge of scripting languages. Different scripting languages are used at the client side and server side. Even though JavaScript and VB Script are the two client side scripting languages, JavaScript is the most commonly used scripting language at the client side. The main reason for this is that the JavaScript is supported by all browsers, but VB Script is not. Since web pages are to be viewed by a large number of people over the Internet, we cannot expect that the user will use a specific browser itself. Hence, a web page should be made browser independent as much as possible. In this chapter, we will learn how JavaScript is used in a web page. Since all of us are familiar with C++, it is easy to understand JavaScript because the JavaScript follows the same syntax of C++.

JavaScript was developed by Brendan Eich for the Netscape Browser. The original name of JavaScript was Mocha. In 1995, when it was deployed in the Netscape browser version 2.0, the name was changed to JavaScript. In early days, only Netscape browser supported JavaScript. But due to the wide popularity of JavaScript, Internet Explorer provided support to JavaScript in 1996. Now, almost all browsers in the world support JavaScript.

Brendan Eich

6.1 Getting started with JavaScript

In Chapter 4 Web Technology, we discussed the use of client side and server side scripting languages. Client side scripting languages are used for validations of data at the client side itself. This reduces network traffic and workload on the server. Server side scripting languages are executed at the server and the web page produced is returned to the client browser. A server may store a large volume of data in the form of a database. So a server side scripting language may have to interact with this large volume of data for different purposes, but a client need not. So, the languages and commands used at the client side and server side are different.

Let us learn the basics of JavaScript as a client side scripting language in this chapter. Using JavaScript, we can embed program segments in different places in an HTML page. `<SCRIPT>` tag is used to include scripts in HTML page.

<SCRIPT> Tag

`<SCRIPT>` tag is used to include scripting code in an HTML page. The **Language** attribute of `<SCRIPT>` tag is used to specify the name of the scripting language used. Here we use JavaScript as the client side scripting language. Therefore, we will give `JavaScript` as the value for `Language` attribute.

The `<SCRIPT>` tag can be used in an HTML page as follows.

```
<SCRIPT Language= "JavaScript">
.....................
.....................
</SCRIPT>
```

The identifiers in JavaScript are case sensitive. It is common in JavaScript, to use camelCase names for identifiers. Examples are firstName, checkData, etc. CamelCase is a naming convention which is used when a single word is formed using multiple words. When the first letter of each word is capitalised, it is called UpperCamelCase and when the first letter of each word except the first word is capitalised, it is called lowerCamelCase. CamelCase improves readability of the word.

Now let us consider the following HTML code given in Example 6.1.

Example 6.1: To create a web page using JavaScript

```
<HTML>
<HEAD> <TITLE>Javascript - Welcome</TITLE> </HEAD>
<BODY>
    <SCRIPT Language= "JavaScript">
        document.write("Welcome to JavaScript.");
    </SCRIPT>
</BODY>
</HTML>
```

Fig. 6.1: A web page using JavaScript

The above code can be typed in any text editor. We use Geany editor that we used for creating HTML pages in the previous chapters. Save the file as 'Code 6.1.html'. Note that even if we use JavaScript in an HTML page, it is saved with **.html** extension. Then, open the file in any browser. We will get the web page as shown in Figure 6.1. Note that the statement `document.write` is written in lowercase letters. This is because JavaScript is a case sensitive scripting language. The keywords in JavaScript are all in the lowercase.

In the above HTML file, **`document.write()`** is a JavaScript command that includes a text in the body section of the HTML page. That is, the above HTML file has the same effect of the following HTML code.

Example 6.2: To create a web page using HTML

```
<HTML>
<HEAD> <TITLE>Javascript - Welcome</TITLE> </HEAD>
<BODY>
    Welcome to JavaScript.
</BODY>
</HTML>
```

Compare the above two examples (Example 6.1 and Example 6.2). In the second HTML code, the text `Welcome to JavaScript` is directly placed inside the body section, whereas in the first one, "`Welcome to JavaScript`" is included in the body section using the JavaScript method `document.write()`. Actually, `document` represents the body section of the web page. Therefore, `document.write()` is a JavaScript function that will include a text in the body

section of the web page. It is very important to note that like C++, every statement in JavaScript also ends in a semicolon.

`<SCRIPT Language= "JavaScript">` tells the browser that the code that follows is a JavaScript code. Now let us see how a browser handles the JavaScript code. The script code is interpreted at runtime by the JavaScript engine. Every browser has a JavaScript engine. JavaScript engine is a virtual machine for executing JavaScript code. When the browser sees a JavaScript code, it is passed to the script engine for processing. The script engine executes the code. If an HTML page does not contain any JavaScript code, the browser alone is able to render the HTML page. But if there is a JavaScript code, the browser takes the help of script engine also to render the HTML page. Hence, an HTML file without JavaScript is always rendered faster than that with JavaScript code.

Let us consider the code given in Example 6.3, that mixes JavaScript codes with HTML tags. The output of the HTML page is given in Figure 6.2.

Example 6.3: To create a web page containing heading tags

```
<HTML>
<HEAD> <TITLE>Javascript - Welcome</TITLE> </HEAD>
<BODY>
    <H1>
    <SCRIPT Language= "JavaScript">
        document.write("This is in H1 Head");
    </SCRIPT>
    </H1>
    <BR>
    <H2>
    <SCRIPT Language= "JavaScript">
        document.write("This is in H2 Head");
    </SCRIPT>
    </H2>
</BODY>
</HTML>
```

Fig. 6.2: Web page containing heading tags

In the above code, we used scripting codes in between the HTML tags more than once. Thus you can use the script codes any number of times in between the HTML tags. Wherever we use script codes, do remember to place them in between `<SCRIPT>` and `</SCRIPT>` tags.

All web browsers allow users to enable or disable the execution of JavaScript in a web page. By disabling the JavaScript, we are actually disabling the JavaScript engine in that browser. In Mozilla, the JavaScript can be enabled or disabled by choosing Tools -> Options -> Content -> Enabled JavaScript. In Google Chrome, it can be done by selecting the option 'Do not allow any site to run JavaScript' which is available in the Content Settings window. This window can be accessed from the menu Tools -> Settings -> Show Advanced Settings -> Content Settings.

If JavaScript is disabled in a web browser, the browser will not execute the script code at all. So, the browser will simply ignore the content written inside `<SCRIPT> .... </SCRIPT>` tags.

Performance of a browser mainly depends on the performance of its script engine. One can hardly see a dynamic web page that does not use any JavaScript. All broswers are competing with each other for the development of better, fast and powerful JavaScript engines.

Know your progress

1. Name an attribute of `<SCRIPT>` tag.
2. In JavaScript _____ function is used to print a text in the body section of an HTML page.
3. _____ is the value given to the language attribute of `<SCRIPT>` tag to specify that the code follows is JavaScript code.
4. What is the use of JavaScript engine?
5. State whether the following statements are true or false.
   a. Java Script is the only client side scripting language.
   b. `<SCRIPT>` tag is used to include client side scripting language in an HTML page.
   c. An HTML file can contain only one `<SCRIPT>` tag in it.
   d. We can mix JavaScript code with HTML code.
   e. The JavaScript code must be placed within `<SCRIPT>` and `</SCRIPT>` tags.
   f. Every JavaScript statement ends in a semicolon.

Let us do

Write an HTML code to create the following web page by using JavaScript inside the body section of the page. That is, the body section should be as shown below.

```
<BODY>
<SCRIPT Language= "JavaScript">
.................
.................
</SCRIPT>
</BODY>
```

6.2 Creating functions in JavaScript

We have already learned about functions in C++. In JavaScript also functions are defined and called almost in the same way as in C++. A function is a group of instructions with a name. JavaScript has a lot of built-in functions that can be used for different purposes. We will cover some of these functions later in this chapter. Besides, these built-in functions, we can define our own functions also. The biggest advantage of using a function is that if we need to execute a piece of program code more than once in a web page, the code has to be written only once within the function. We can call the function any number of times to execute it. Look at the following code:

```
function print()
{
    document.write("Welcome to JavaScript.");
}
```

Here, the keyword `function` is used to define a function. The word `print` that follows, is the name of the function. The function name can be any valid identifier. JavaScript uses the same rules for naming identifiers as in C++. Here the `print()` function contains only one statement. We can include any number of statements inside a function.

Defining a function does not execute the function automatically. It must be called for its execution. This means that even if we define a function in a web page and do not call the function, the function will not be executed at all. The function can be called by using the function name as follows:

```
print();
```

Note the semicolon after the function name. Now let us make use of the above `print()` function in an HTML page.

Example 6.4: To create a web page containing print() function

```
<HTML>
<HEAD> <TITLE>Javascript - Functions</TITLE>
    <SCRIPT Language= "JavaScript">
        function print()
        {
            document.write("Welcome to JavaScript.");
        }
    </SCRIPT>
</HEAD>
<BODY>
</BODY>
</HTML>
```

Note that nothing is written inside the body section of the HTML page. This code will not display '`Welcome to JavaScript`' in the browser window. This is because even though the function is defined, it is not at all called from any place in the page. Therefore, the function will not be executed at all. Hence nothing will be displayed on the screen. The body section of the HTML page in Example 6.4 should be modified as follows to get the output as shown in Figure 6.3.

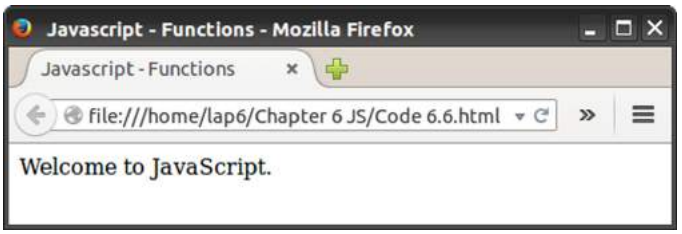


Fig. 6.3: Web page using functions

```
<BODY>
<SCRIPT Language= "JavaScript">
    print();
</SCRIPT>
</BODY>
```

Now let us see the syntax of a JavaScript function.

```
function function_name()
{
    statements;
}
```

Here the line `function function_name()` is called the **function header** and the code within the braces `{ and }` is called **function body**. The main difference from C++ is that, in JavaScript there is no return type, whereas in C++ the function has a return type. In JavaScript also, we can return some value from a function as in C++. Since this chapter is meant to give only a basic idea about the use of JavaScript, we do not discuss them for the time being. Another difference is that, C++ does not use the keyword `function` to define a function, but JavaScript does.

We might have noted that the function is defined within the head section of the HTML page. It is not necessary to define the function within the head section itself. We can define a function in the body section also as given below:

```
<BODY>
    <SCRIPT Language= "JavaScript">
        function print()
        {
            document.write("Welcome to JavaScript.");
        }
        print();
    </SCRIPT>
</BODY>
```

The above code also produces the same output as in Figure 6.3. Note that even if the function is defined within the body section, it must be called for its execution. For example, the following code does not display anything on the screen:

```
<BODY>
<SCRIPT Language= "JavaScript">
function print()
{
    document.write("Welcome to JavaScript.");
}
</SCRIPT>
</BODY>
```

Even though a function can be defined anywhere in an HTML page, it is always better to include the function definition within the head section. Now consider the following code in which the `print()` function is called twice. It will give the output as shown in Figure 6.4.

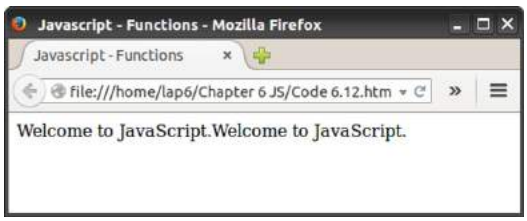


Fig. 6.4: Web page using two print functions without break

```
<BODY>
    <SCRIPT Language= "JavaScript">
        print();
        print();
    </SCRIPT>
</BODY>
```

When we call the function twice, it just places the content 'Welcome to JavaScript' twice in the body section, where the function is called. Hence, the body section in the above code has the same effect as the following code.

```
<BODY>
    Welcome to JavaScript.Welcome to JavaScript.
</BODY>
```

If the message should be displayed in the two different lines, we must use
 tag.

```
<BODY>
    Welcome to JavaScript.<BR>Welcome to JavaScript.
</BODY>
```

In order to create the same effect using JavaScript, the function can be modified as shown below.

```
<SCRIPT Language= "JavaScript">
    function print()
    {
        document.write("Welcome to JavaScript.<BR>");
    }
</SCRIPT>
```

This HTML code will give the output as shown in Figure 6.5.

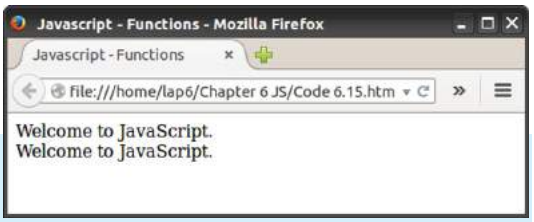


Fig. 6.5: Web page using two print functions

Know your progress

1. Say whether the following statements are true or false:
   a. A function is automatically executed when the browser opens the web page.
   b. A function is usually placed in the head section of a web page.
   d. A function can be called any number of times.
   e. Even though a function is defined within the body section, it will not be executed, if it is not called.
2. List the advantages of using functions in JavaScript.

Let us do

1. Consider the following two HTML codes (Code A and Code B) and try to predict the output.

Code A

```
<HTML>
<HEAD>
<SCRIPT Language= "JavaScript">
    function print()
    {
        document.write("Welcome to JavaScript");
    }
</SCRIPT>
</HEAD>
<BODY>
<SCRIPT Language= "JavaScript">
    print();
    document.write("<BR>");
    print();
</SCRIPT>
</BODY>
</HTML>
```

Code B

```
<HTML>
<HEAD>
<SCRIPT Language= "JavaScript">
function print()
{
      document.write("Welcome to JavaScript");
}
</SCRIPT>
</HEAD>
<BODY>
<SCRIPT Language= "JavaScript">
      print();
</SCRIPT>
<BR>
<SCRIPT Language= "JavaScript">
      document.write("<BR>");
      print();
</SCRIPT>
</BODY>
</HTML>
```

2. Given below is an HTML code to get the output as shown in the figure below. The code contains four functions namely `startGreen()`, `stopGreen()`, `startRed()` and `stopRed()`. These functions are called from different places from the body section of the HTML page. You can see that the body part in each function definition is blank except for the function `stopGreen()`. You have to complete the definition of all other functions so as to get the output as shown in figure.

```
<HTML>
<HEAD>
<SCRIPT Language= "JavaScript">
      function startGreen()
      {
            ..............
            ..............
      }
      function stopGreen()
      {
            document.write("</FONT>");
      }
      function startRed()
      {
            ..............
            ..............
      }
      function stopRed()
      {
            ..............
            ..............
      }
</SCRIPT>
</HEAD>
<BODY>
<SCRIPT Language= "JavaScript">
      startGreen();
</SCRIPT>
This is in Green colour with size 5
<SCRIPT Language= "JavaScript">
      stopGreen();
</SCRIPT>
```

```
<SCRIPT Language= "JavaScript">
      startRed();
</SCRIPT>
This is in Red colour with size 3
<SCRIPT Language= "JavaScript">
      stopRed();
</SCRIPT>
<SCRIPT Language= "JavaScript">
      startGreen();
</SCRIPT>
This is in Green colour with size 5
<SCRIPT Language= "JavaScript">
      stopGreen();
</SCRIPT>
</BODY>
</HTML>
```

6.3 Data types in JavaScript

All programing languages classify data items into different categories for the effective utilisation of memory. We have learned that the basic data types in C++ are `int`, `char`, `float`, `double` and `void`. Besides these data types, there are type modifiers also in C++. JavaScript reduces this complexity by limiting the number of basic data types into 3. The following are the three basic data types in JavaScript.

Number

All numbers fall into this category. All positive and negative numbers, all integer and float values (fractional numbers) are treated as the data type number. Thus, 27, -300, 1.89 and -0.0082 are examples of number type data in JavaScript.

String

Any combination of characters, numbers or any other symbols, enclosed within double quotes, are treated as a string in JavaScript. That is, "Kerala", "Welcome", "SCHOOL", "1234", "Mark20", "abc$" and "sanil@123" are examples of string type data.

Boolean

Only two values fall in boolean data type. They are the values `true` and `false`. Note that the values are not in double quotes. If they are put inside double quotes, they will be considered as of string type. Not only that, like C++, JavaScript is also case sensitive. So we cannot use TRUE and FALSE to represent boolean values.

6.4 Variables in JavaScript

As we know, variables are used for storing values. In JavaScript, variables can be declared by using the keyword **var** as given below:

```
var  x;
```

Here x is the name of the variable. Like C++, a variable can be given any name, as you like, with only one restriction, that it must be a valid identifier. We have used the keywords `int`, `float`, `char`, etc. to declare different types of variables in C++. But, in JavaScript, the same keyword, **var** is used to declare all type of variables.

In JavaScript, merely declaring a variable does not define the variable. The variable definition is complete only when it is assigned a value. JavaScript understands the type of variable only when a value is assigned to that variable. Consider the following declaration.

```
var x, y;
x = 25;
y = "INDIA";
```

In the above example, the variable x is of number type and y is of string type. Even though we say that x is of number type and y is of string type, note that we do not specify them in code segments. The following example explains how JavaScript defines variables.

Example 6.5: To illustrate the use of variables

```
<HTML>
<HEAD> <TITLE>Javascript - Variables</TITLE> </HEAD>
<BODY>
<SCRIPT Language= "JavaScript">
    var a, b, c, d, e, f;
    a = 25;
    b = 18.5;
    c = "INDIA";
    d = true;
    e = "true";
    document.write("a is of type : ");
    document.write(typeof(a));
    document.write("<BR>b is of type : ");
    document.write(typeof(b));
    document.write("<BR>c is of type : ");
    document.write(typeof(c));
```

```
    document.write("<BR>d is of type : ");
    document.write(typeof(d));
    document.write("<BR>e is of type : ");
    document.write(typeof(e));
    document.write("<BR>f is of type : ");
    document.write(typeof(f));
</SCRIPT>
</BODY>
</HTML>
```

Here, the code uses the function `typeof()`, which is not familiar to you. As the name indicates, this function is used to find the type of a variable. We will study this function in detail later. The output of the above example is given in Figure 6.6.

Fig. 6.6: Web page displaying the use of variables

Note that the variable `f` is declared, but it is not given a value. Therefore, the script engine is unable to understand its type and so it is declared as `undefined`. In JavasScript, `undefined` is a special datatype to represent variables that are not defined using `var`.

Any number of variables can be declared by using a single `var` keyword. The variables should be separated by comma (`,`). Let us consider a Javascript function that makes use of variables.

Example 6.6: To create a web page to find the sum of two numbers

```
<HTML>
<HEAD><TITLE>Javascript - Variables</TITLE>
<SCRIPT Language= "JavaScript">
function add()
{
    var m, n, sum;
    m = 20;
    n = 10;
    sum = m + n;
    document.write("Sum = ");
    document.write(sum);
}
```

```
</SCRIPT>
</HEAD>
<BODY>
<SCRIPT Language= "JavaScript">
    add();
</SCRIPT>
</BODY>
</HTML>
```

In the above example, the body section contains only one JavaScript statement - a call to the `add()` function. After executing the function, it will display the output as shown in Figure 6.7.

Fig. 6.7: Web page to find the sum of two numbers

6.5 Operators in JavaScript

Almost all operators in JavaScript are exactly similar to those in C++. Let us have a quick look at all of these operators.

6.5.1 Arithmetic operators

Table 6.1 shows the arithmetic operators used in JavaScript along with examples.

Operator	Description	Example	Value of y	Result (x)
+	Addition	x = y + 10	15	25
–	Subtraction	x = y – 10	15	5
*	Multiplication	x = y * 3	15	45
/	Division	x = y / 2	15	7.5
%	Modulus (division remainder)	x = y % 2	15	1
++	Increment	x = ++y x = y++	15 15	16 15
––	Decrement	x = --y x = y--	15 15	14 15

Table 6.1: Arithmetic operators

From the above table, we can see that all arithmetic operators work exactly as in C++.

6.5.2 Assignment operators

Table 6.2 displays the usage of various assignment operators in JavaScript.

Operator	Description	Example	Value of a	Value of b	Result (a)
=	Assignment	a = b	10	3	3
+=	Add and assignment	a+=b	10	3	13
-=	Minus and assignment	a-=b	10	3	7
*=	Multiply and assignment	a*=b	10	3	30
/=	Divide and assignment	a/=b	10	3	3.33
%=	Modulus and assignment	a%=b	10	3	1

Table 6.2: Assignment operators

It is very easy to understand the working of each operator from the above table. The 'Result' column gives the result after executing the statement in the 'Example' column with the given values of `a` and `b`.

6.5.3 Relational operators (Comparison operators)

Table 6.3 shows the various relational operators used in JavaScript, along with examples.

Operator	Description	Example	Value of a	Value of b	Result
==	Equal to	a==b	10	3	false
!=	Not equal to	a!=b	10	3	true
<	Less than	a<b	10	3	false
<=	Less than or equal to	a<=b	10	3	false
>	Greater than	a>b	10	3	true
>=	Greater than or equal to	a>=b	10	3	true

Table 6.3: Relational operators

From the table, it is clear that the result of a relational operation is either `true` or `false`. These operators compare the values on the two sides of the operator and give the result accordingly.

6.5.4 Logical operators

Table 6.4 shows the different logical operators in JavaScript along with examples.

Operator	Description	Example	Value of a	Value of b	Result
&&	AND	a && b	true	false	false
||	OR	a || b	true	false	true
!	NOT	!a	true		false

Table 6.4: Logical operators

From the above tables, it is clear that all these operators are exactly similar to C++. JavaScript provides a number of other operators also for different purposes. Since our discussion of JavaScript is limited only to this chapter, the discussion of those operators are beyond the scope of this book. Besides, the operators discussed above are enough to perform almost all the operations for a beginner. However, the following string operator will be of use to us in various situations. This operator is not available in C++.

6.5.5 String addition operator (+)

We have already seen that the operator + is used to add two numbers. The same operator + is used to add two strings also. Adding two strings means concatenating two strings.

Look at the following example code.

```
var x, y;
x = "A good beginning ";
y = "makes a good ending.";
z = x + y;
```

The + operator will add the two strings. Therefore the variable z will have the value `A good beginning makes a good ending`. The same + operator behaves differently based on the type of operands. If the operands are numbers, it will add the numbers. If the operands are strings it will concatenate the strings. Now, predict the value of z in the following code:

```
var  x, y;
x = "23";
y = 5;
z = x + y;
```

The answer is 235. If + operator sees any one operand as string, it will treat both the operands as string type and concatenate the strings to get the result 235. Suppose,

we want to add x and y in the form of numbers, we can rewrite the last statement as follows:

```
z = Number(x) + y;
```

After executing the above statement, the variable z will have the value 28. `Number()` is a function in JavaScript, that converts a string type data containing numbers to number type. We will make use of this function in some examples later in this chapter.

Know your progress

1. number, string and _____ are the basic data types in JavaScript.
2. `true` is a _____ type data.
3. `false` is a _____ type data.
4. The keyword used to declare a variable in JavaScript is _____.
5. The function used to know the type of data in JavaScript is _____.
6. What is the use of % operator?
7. List the logical operators.

6.6 Control structures in JavaScript

Control structures are used to change the sequential flow of execution in a program. All the control structures that we have learned in C++ can also be used in JavaScript without any difference. Let us discuss some of the frequently used control structures with an example each.

6.6.1 if

This is the most frequently used control structure in every programming language. It is used to execute a statement or a group of statements based on some condition. It can be used in two ways as given in Table 6.5.

Syntax of simple if	Syntax of if with else part
`if (test_expression)` `{` `  statements;` `}`	`if (test_expression)` `{` `  statements;` `}` `else` `{` `  statements;` `}`

Table 6.5: Syntax of if and if with else statement

The syntax in the left column of Table 6.5 is simple `if` statement and that in the right column is of `if – else` statement. Now, let us consider an example that makes use of the **`if`** statement.

Example 6.7: To create a web page that checks whether a student has passed or not

```
<HTML>
<HEAD> <TITLE>Javascript – if</TITLE> </HEAD>
<BODY>
<SCRIPT Language= "JavaScript">
    var score;
    score = 35;
    if (score < 30)
    {
        document.write("The student is failed.");
    }
    else
    {
        document.write("The student is passed.");
    }
</SCRIPT>
</BODY>
</HTML>
```

Fig. 6.8: Web page to illustrate `if` withe else statement

The above program code makes use of the `if` statement with `else` part. The output of the program is given in Figure 6.8. We can change the value of the score as a number below 30 and view the output to see the changes.

6.6.2 switch

`switch` is a multi-branching statement. Using this, different program codes can be selected for execution based on the value of an expression. Its syntax is:

```
switch (expression)
{
    case value1:
            statements;
            break;
    case value2:
            statements;
            break;
    ..................
```

```
        ..................
        default:
                    statements;
    }
```

The appropriate case is executed based on the value of the expression. Here the expression can be the name of a variable also. The following is a web page that prints the day corresponding to a given number.

Example 6.8: To create a web page to print the day of a week

```
<HTML>
<HEAD> <TITLE>Javascript - switch</TITLE> </HEAD>
<BODY>
<SCRIPT Language= "JavaScript">
    var d;
    d = 3;
    switch(d)
    {
        case 1:
            document.write("Sunday");
            break;
        case 2:
            document.write("Monday");
            break;
        case 3:
            document.write("Tuesday");
            break;
        case 4:
            document.write("Wednesday");
            break;
        case 5:
            document.write("Thursday");
            break;
        case 6:
            document.write("Friday");
            break;
        case 7:
            document.write("Saturday");
            break;
        default:
            document.write("Invalid Day");
    }
</SCRIPT>
</BODY>
</HTML>
```

The output of the above page is given in Figure 6.9.

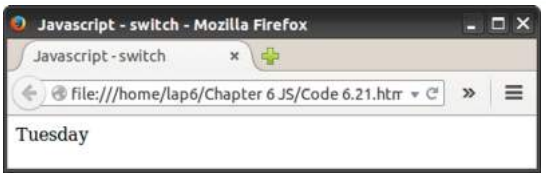


Fig. 6.9: Web page that illustrates the use of `switch` statement

6.6.3 `for` loop

`for` loop is used to execute a group of instructions repeatedly. `for` loop uses a loop variable to control the number of iterations. The syntax of `for` loop is:

```
for(initialisation; test_expression; update_statement)
{
    statements;
}
```

Here initialisation is used to initialise loop variables. `test_expression` checks the condition and update statement is used to increment or decrement loop variables. The following example explains the use of `for` loop.

Example 6.9: To create a web page to display the squares of first 10 numbers

```
<HTML>
<HEAD> <TITLE>Javascript - for</TITLE> </HEAD>
<BODY>
<SCRIPT Language= "JavaScript">
    var i, s;
    for (i=1; i<=10; i++)
    {
        s = i*i;
        document.write(s);
        document.write("<BR>");
    }
</SCRIPT>
</BODY>
</HTML>
```

Fig. 6.10 : Web page to illustrate the use of `for` loop

The output of the code is given in Figure 6.10.

Now let us modify the above code to get the output as 'Square of 1 is 1' and so on.

```
for (i=1; i<=10; i++)
{
    s = i*i;
    document.write("Square of " + i + " is " + s);
    document.write("<BR>");
}
```

The code uses the string addition operator + to generate the output.

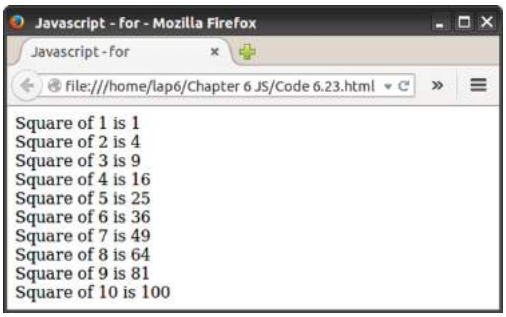


Fig. 6.11: Web page to display the square of numbers

6.6.4 `while` loop

`while` loop is a simple loop that repeatedly execute a group of statements based on a condition. The syntax is

```
while (test_expression)
{
    statements;
}
```

Here the `test_expression` is a condition. The statements inside the loop will be executed as long as the condition remains true. The following example displays all even numbers up to 10.

Example 6.10: To create a web page that displays even numbers upto 10

```
<HTML>
<HEAD> <TITLE>Javascript - while</TITLE> </HEAD>
<BODY>
<SCRIPT Language= "JavaScript">
    var i;
    i = 2;
    while (i<=10)
    {
        document.write(i);
        document.write("<BR>");
        i += 2;
    }
</SCRIPT>
</BODY>
</HTML>
```

Fig. 6.12: Web page that illustrates the use of while statement

The output of the above code is given in Figure 6.12.

In JavaScript, we can use `do while` loop also exactly as we do in C++. But since almost all the tasks can be done with `for` loops and `while` loops, we do not discuss the other loops here in this chapter.

Let us do

Write the output, if the statement `document.write("<BR>");` is omitted from the Example 6.10?

Know your progress

1. To select a group of statements in the program code for execution _____ or _____ control structures are used .
2. Give examples of looping statements in JavaScript.
3. _____ is a multi branching statement.
4. State whether the following statements are true or false
   a. `break` statement is used within the `switch` block.
   b. If we use `switch`, we will be using `break` within it, at least once.
   c. All the program codes written in `if-else` can be replaced by using `switch`.
   d. All the program codes written in `switch` can be replaced by using `if-else`.
5. What is the difference between `for` loop and `while` loop?

6.7 Built-in functions

JavaScript provides a large number of built-in functions. The functions are also called methods. Here we will discuss only the most commonly used functions.

a. `alert()` function

This function is used to display a message on the screen. For example, the statement `alert("Welcome to JavaScript");` will display the following message window on the browser window as shown in Figure 6.13. This function is used to display a message to the user at the time of data validation.

Fig. 6.13: `alert()` *window*

b. `isNaN()` function

This function is used to check whether a value is a number or not. In this function, `NaN` stands for Not a Number. The function returns `true` if the given value is not a number. For example, the following statements return the value `true`.

```
1. isNaN("welcome");
2. isNaN("A123");
3. isNaN("Score50");
4. isNaN("A");
```

The following statements return the value `false`.

```
1. isNaN("13");
2. isNaN(13);
3. isNaN("13.5");
4. isNaN("0.123");
```

The following statement gives the message as shown in Figure 6.14 on the browser window:

```
alert(isNaN("A"));
```

This function is very useful for data validation. For example, suppose the web page contains a text box to enter the age of a student. By mistake, the user may enter a character in the age box, instead of a number. The above function can check whether the entered value is a number or not. If it is not a number, a message will be displayed by using the `alert()` function.

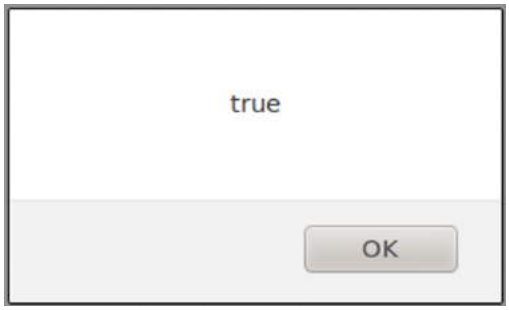


Fig. 6.14: Result of `isNaN()`

c. `toUpperCase()` function

This function returns the upper case form of the given string. Look at the following example code:

```
var x, y;
x = "JavaScript";
y = x.toUpperCase();
alert(y);
```

The output of the above code segment is shown in Figure 6.15. Here you can see how the `toUpperCase()` function is called. It is called along with the name of the string variable `x`. That is, `x.toUpperCase()` returns the upper case form of the string in the variable `x`. JavaScript is a case sensitive

Fig. 6.15 : Result of `toUpperCase()`

language. Hence you have to use the function exactly in the same case form as given in the code.

d. toLowerCase() function

It returns the lower case form of the given string.

Look at the following example code.

```
var x, y;
x = "JavaScript";
y = x.toLowerCase();
alert(y);
```

The output of the above code segment is shown in Figure 6.16. If all the characters in the string are already in the lower case, the `toLowerCase()` returns the same string.

Fig. 6.16: Result of `toLowerCase()`

e. charAt() function

It returns the character at a particular position. `charAt(0)` returns the first character in the string. `charAt(1)` returns the second character in the string and so on. Look at the following example code.

```
var x;
x = "JavaScript";
y = x.charAt(4);
alert(y);
```

The output is given in Figure 6.17. Since the fifth character in the variable `x` is `'S'`, the letter is displayed in the browser window.

Fig. 6.17: Result of `charAt()`

f. length property

Besides functions, a string variable also provides some properties which are of use to programmers. `length` property returns the length of the string. length means, the number of characters in the string. For example,

```
var x, n;
x = "JavaScript";
n = x.length;
alert(n);
```

Here, we can see how the `length` property is called. The property is called along with the variable as `x.length`. Figure 6.18 shows the output of the above code.

The difference between a function and a property is that the function has parentheses (and) with parameters after the function name, but the property does not.

Fig. 6.18: Result of `length` *property*

Know your progress

Write the value of the variable y in the following:

1.
```
x = "welcome";
y = x.length;
```
2.
```
x = "WELCOME";
y = x.toLowerCase();
```
3.
```
x = "Welcome";
y = x.toUpperCase();
```
4.
```
x = "welcome";
y = x.toLowerCase();
```
5.
```
x = "welcome";
y = isNaN(x);
```
6.
```
x = "welcome";
y = charAt(3);
```

6.8 Accessing values in a textbox using JavaScript

In Chapter 5, we learned how to place various controls like textbox, checkbox, radio button, submit button, etc. in a web page. Now we will discuss how to access these web page elements using JavaScript. Actually all the program codes that we have already discussed in this chapter did not accept any value from the user for processing. The necessary data for processing was given directly in the program code itself. After learning this section, we will be able to write a truely interactive web page. That is, the user can enter some values in a text box, and some processing can be done on this value and some result can be displayed in another text box. Go through the following HTML code carefully.

Example 6.11 : To create a web page that displays a web form

```
<HTML>
<HEAD> <TITLE>Javascript - Text box</TITLE> </HEAD>
<BODY>
    <FORM Name= "frmSquare">
```

```
        <CENTER>
            Enter a Number
            <INPUT Type= "text" Name= "txtNum">
            <BR><BR>
            Square is
            <INPUT Type= "text" Name= "txtSqr">
            <BR><BR>
            <INPUT Type= "button" Value= "Show">
        </CENTER>
    </FORM>
</BODY>
</HTML>
```

The output is given in Figure 6.19. Note that we have given the name `frmSquare` for the Form, `txtNum` and `txtSquare` for the two textboxes. Giving names to them is very much important to access them using JavaScript. If we do not give any name to a web page element, the JavaScript cannot access these elements. We can also note that we have not given a name to the submit button. This is because, this button need not be referred from the JavaScript.

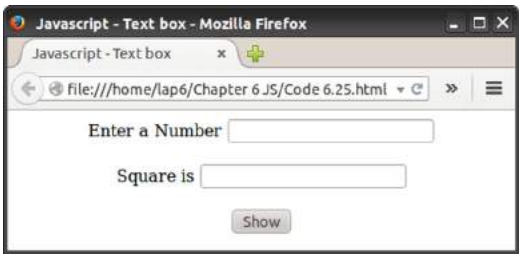


Fig. 6.19 : A form web page

Now let us make a slight modification in the above program code as given in Example 6.12.

Example 6.12: To create a web page that displays the square of a number

```
<HTML>
<HEAD> <TITLE>Javascript - Text box</TITLE>
    <SCRIPT Language= "JavaScript">
        function showSquare()
        {
            var num, ans;
            num = document.frmSquare.txtNum.value;
            ans = num * num;
            document.frmSquare.txtSqr.value = ans;
        }
    </SCRIPT>
</HEAD>
```

```
<BODY>
    <FORM Name= "frmSquare">
    <CENTER>
        Enter a Number
        <INPUT Type= "text" Name= "txtNum">
        <BR><BR>
        Square is
        <INPUT Type= "text" Name= "txtSqr">
        <BR><BR>
        <INPUT Type= "button" Value= "Show"
                              onClick= "showSquare()">
    </CENTER>
    </FORM>
</BODY>
</HTML>
```

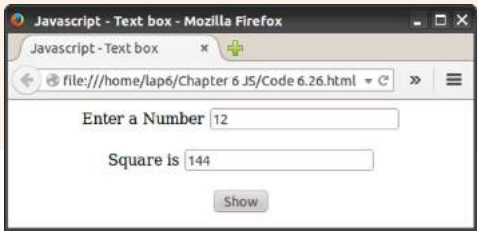


Fig. 6.20: Web page to find the square of a number

Go through the above code carefully. Note the additions made to the code in Example 6.11. A function with the name `showSquare()` is defined in the head section of the web page. This function is called using the following line of code:

```
<INPUT Type= "button" Value= "Show" onClick= "showSquare()">
```

`onclick= "showSquare()"` written within the button means that when the user clicks this button, the function with the name `showSquare()` is called.

Now go through the function definition. Look at the following line:

```
num = document.frmSquare.txtNum.value;
```

Here `document` refers the body section of the web page. `frmSquare` is the name of the Form we have given inside the body section. `txtNum` is the name of the text box within the `frmSquare` and `value` refers to the content inside that text box. That is `document.frmSquare.txtNum.value` means the `document`'s `frmSquare`'s `txtNum`'s `value`. Therefore the above line assigns the value of the first text box in a variable `num`.

Now, you may be able to understand the meaning of the following line.

```
document.frmSquare.txtSqr.value = ans;
```

The above line assigns the value of the variable `ans` in the second text box. Thus the above web page displays the square of the given number in the second text box when the button is clicked. User can type any number in the first text box and click the submit button to see its square. The screen shot of the web page while execution is given in Figure 6.20.

What will happen if the line

```
<INPUT Type= "button" Value= "Show" onClick= "showSquare()">
```

is replaced by

```
<INPUT Type= "button" Value= "Show"
                        onMouseEnter= "showSquare()">
```

The function will be called for execution when you move the mouse over the button. That means, you need not click the button for its execution. You may have noted that when you move the mouse over some buttons in a web page, the colour of the button may keep changing. This can be made possible by writing a JavaScript function to change the colour of the button and call that function on the event `onMouseEnter`. `onClick`, `onMouseEnter`, `onMouseLeave`, `onKeyDown`, `onKeyUp` are some of the commonly used events where we can call a function for its execution. Table 6.6 shows common JavaScript events and their descriptions.

Event	Description
`onClick`	Occurs when the user clicks on an object
`onMouseEnter`	Occurs when the mouse pointer is moved onto an object
`onMouseLeave`	Occurs when the mouse pointer is moved out of an object
`onKeyDown`	Occurs when the user is pressing a key on the keyboard
`onKeyUp`	Occurs when the user releases a key on the keyboard

Table 6.6: Common JavaScript events

Now let us dicuss another example (Example 6.13) that allows the user to enter two numbers in two different text boxes and display the sum of these numbers in a third text box, when a button is clicked. The output of this code is given in Figure 6.21.

Example 6.13: To create a web page that displays the sum of two numbers

```
<HTML>
<HEAD> <TITLE>Javascript - Sum</TITLE>
    <SCRIPT Language= "JavaScript">
        function showSum()
        {
            var num1, num2, ans;
            num1 = document.frmSum.txtNum1.value;
            num2 = document.frmSum.txtNum2.value;
            ans = num1 + num2;
            document.frmSum.txtSum.value = ans;
        }
    </SCRIPT>
```

```
</HEAD>
<BODY>
<FORM Name= "frmSum">
    <CENTER>
    Enter a Number 1
    <INPUT Type= "text" Name= "txtNum1">
    <BR><BR>
    Enter a Number 2
    <INPUT Type= "text" Name= "txtNum2">
    <BR><BR>
    The sum is
    <INPUT Type= "text" Name= "txtSum">
    <BR><BR>
    <INPUT Type= "button" Value= "Show" onClick= "showSum()">
    </CENTER>
</FORM>
</BODY>
</HTML>
```

This program displays 1020 as the result as shown in Figure 6.21. This is because the + operator is used to add strings also. By default, the content of the text box is always treated as of string type. Therefore, even though the content in the text box is a number, when we assign this to a variable, it will be treated as string type only. When we add the two strings "10"+"20" the answer is "1020". The function showSum() can be modified as follows to get the sum of two numbers.

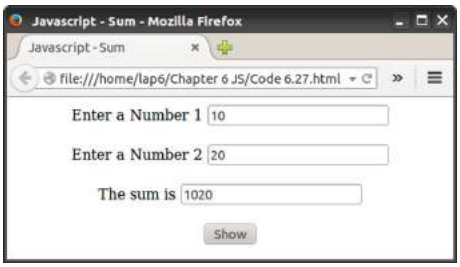


Fig. 6.21: Web page to illustrate the use of + operator

```
function showSum()
{
    var num1, num2, ans;
    num1 = Number(document.frmSum.txtNum1.value);
    num2 = Number(document.frmSum.txtNum2.value);
    ans = num1 + num2;
    document.frmSum.txtSum.value = ans;
}
```

We discussed the Number() function in the previous section of this chapter. The Number() function converts the data into number type and assigns that number into the variable num1. In the above function, num1 and num2 are treated as number

type data and hence the correct result is obtained after adding. The output of the web page is given in Figure 6.22 when the function in the Example 6.13 is replaced by the above function.

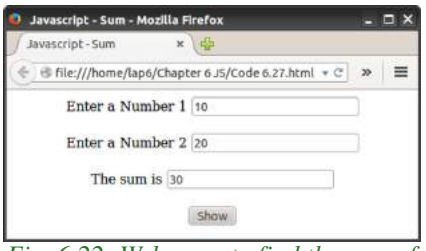


Fig. 6.22: Web page to find the sum of two numbers using + operator

Let us consider another web page which displays the sum of numbers upto a given limit. The user can enter the limit in a text box.

Example 6.14: To create a web page that displays sum of numbers upto a given limit

```
<HTML>
<HEAD> <TITLE>Javascript - Sum</TITLE>
    <SCRIPT Language= "JavaScript">
        function sumLimit()
        {
            var sum = 0, i, limit;
            limit = Number(document.frmSum.txtLimit.value);
            for(i = 1; i <= limit; i++)
                sum += i;
            document.frmSum.txtSum.value = sum;
        }
    </SCRIPT>
</HEAD>
<BODY>
    <FORM Name= "frmSum">
        <CENTER>
            Enter the limit
            <INPUT Type= "text" Name= "txtLimit">
            <BR><BR>
            Sum of Numbers
            <INPUT Type= "text" Name= "txtSum">
            <BR><BR>
            <INPUT Type= "button" Value= "Show"
                                  onClick= "sumLimit()">
        </CENTER>
    </FORM>
</BODY>
</HTML>
```

The output of the above code is given in Figure 6.23. In the above web page, on clicking the **show** button, the sum of numbers up to the given limit is shown in the second text box.

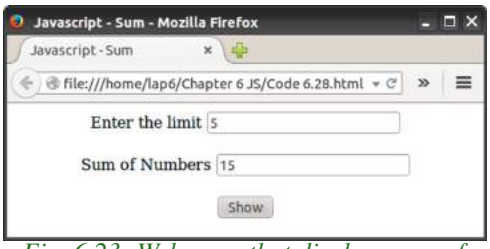


Fig. 6.23: Web page that displays sum of numbers upto a limit

If the **show** button in the Form is clicked without entering the limit, it does not display any message. That is, in this web page, if we click the **show** button without entering the limit, it will give 0 as the sum. If we do not enter any value in the text box, `document.frmSum.txtLimit.value` is empty. Hence `Number()` function will convert this empty value to the value 0. i.e., the limit variable in the function will have the value 0. Hence the loop will not be executed at all. Therefore, the variable `sum` will have the initial value 0 itself, which is displayed in the second text box.

Now, let us provide a check to the `sumLimit()` function as given below:

```
function sumLimit()
{
    var sum = 0, i, limit;
    if (document.frmSum.txtLimit.value == "")
    {
        alert("Please enter the limit!");
        return;
    }
    limit = Number(document.frmSum.txtLimit.value);
    for(i=1; i<=limit; i++)
        sum += i;
    document.frmSum.txtSum.value = sum;
}
```

JavaScript never gives an error. When the script engine is unable to execute an instruction, it will ignore that line as well as the rest of the lines in the function.If we click the **show** button without entering the limit, this code will show a message, reminding us to enter the limit. The `return` statement is used to exit from the function ignoring the rest of the lines. The `return` statement is similar to the `return` statement used in C++.

Let us now check whether a number or an alpbabet is entered in the text box for entering limit. For this, `isNaN()` function can be used. This helps to check whether the user has given the correct data for processing.

```
function sumLimit()
{
    var sum = 0, i, limit;
```

```
        if (document.frmSum.txtLimit.value == "")
        {
            alert("Please enter the limit!");
            return;
        }
        if (isNaN(document.frmSum.txtLimit.value))
        {
            alert("Please enter a number as the limit!");
            return;
        }
        limit = Number(document.frmSum.txtLimit.value);
        for(i = 1; i <= limit; i++)
            sum += i;
        document.frmSum.txtSum.value = sum;
    }
```

Java Script is mostly used for client side validation. When the user clicks the submit button after entering the data, JavaScript can be used to check whether the user has given all the necessary data, check whether the data is in the correct format or not, etc. If they are not in the correct format, a message can be shown reminding the user to enter the correct data.

The following example makes use of a drop-down list in JavaScript. This web page allows the user to select a state from a drop-down list. On clicking the show button, capital of the selected State is displayed in a text box. The output is given in Figure 6.24.

Example 6.15: To create a web page that displays the capital of a State

```
<HTML>
<HEAD> <TITLE>Javascript - switch</TITLE>
<SCRIPT Language= "JavaScript">
function capital()
{
    var n, answer;
    n = document.frmCapital.cboState.selectedIndex;
    switch (n)
    {
    case 0:
        answer = "Thiruvananthapuram";
        break;
    case 1:
        answer = "Bengaluru";
        break;
```

```
    case 2:
        answer = "Chennai";
        break;
    case 3:
        answer = "Mumbai";
        break;
    }
    document.frmCapital.txtCapital.value = answer;
}
</SCRIPT>
</HEAD>
<BODY>
    <FORM Name= "frmCapital">
    <CENTER> State
        <SELECT Size= 1 Name= "cboState">
            <OPTION>Kerala</OPTION>
            <OPTION>Karnataka</OPTION>
            <OPTION>Tamilnadu</OPTION>
            <OPTION>Maharashtra</OPTION>
        </SELECT>
        <BR><BR>
        Capital
        <INPUT Type= "text" Name= "txtCapital">
        <BR><BR>
        <INPUT Type= "button" Value= "Show" onClick= "capital()">
    </CENTER>
    </FORM>
</BODY>
</HTML>
```

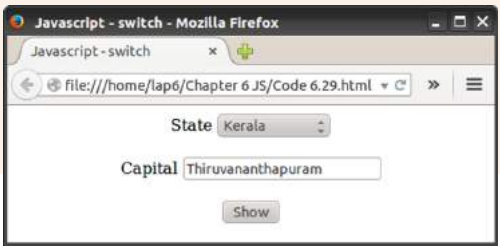


Fig. 6.24: Web page to find the capital of a state

Let us consider the following statement from the above program.

```
document.frmCapital.cboState.selectedIndex;
```

Here, '`cboState`' is the name of the drop-down list. '`selectedIndex`' gives the index of the selected item in the drop-down list. If the first item is selected, the index is 0, if the second item is selected, the index is 1, and so on. So, the above line assigns the index of the selected item in the variable `n`.

The following is a web page that allows the user to enter the name and age of a student. The name must contain at least 5 characters. The age should be a number in the range 15 to 20.

Example 6.16: To create a web page that validates name and age

```
<HTML>
<HEAD> <TITLE>Javascript - Validation</TITLE>
<SCRIPT Language= "JavaScript">
function checkData()
{
    var T_name, T_age, N_age;
    T_name = document.frmValid.txtName.value;
    if (T_name == "")
    {
        alert("Please enter name!");
        return;
    }
    if (T_name.length < 5)
    {
        alert("Name must contain at least 5 characters!");
        return;
    }
    T_age = document.frmValid.txtAge.value;
    if (T_age == "")
    {
        alert("Please enter age!");
        return;
    }
    if (isNaN(T_age))
    {
        alert("Please enter a number as the age!");
        return;
    }
    N_age = Number(T_age);
    if (N_age < 15 || N_age > 20)
    {
        alert("The age must be between 15 and 20!");
        return;
    }
}
</SCRIPT>
</HEAD>
```

```
<BODY>
    <FORM Name= "frmValid">
        <CENTER>Name
            <INPUT Type= "text" Name= "txtName">
            <BR><BR>
            Age
            <INPUT Type= "text" Name= "txtAge">
            <BR><BR>
            <INPUT Type= "button" Value= "Save"
                                  onClick= "checkData()">
        </CENTER>
    </FORM>
</BODY>
</HTML>
```

Fig. 6.25: Web page to validate name and age

The output of the above code is given in Figure 6.25. It checks all necessary validations for the data. At first, it tests whether there is an entry in the Name field or not. Then it tests whether the length of the name entry has minimum 5 characters or not. Then, it tests whether there is an entry in the age box. Then it checks whether the age entry is a number or not. Finally, it tests whether the age entry is in the range 15 to 20 or not.

6.9 Ways to add scripts to a web page

Scripts can be placed inside HTML code in different ways. In the previous examples we have placed the JavaScript code in the head section of the web page. Apart from head section, scripts can be placed inside the <BODY> tag or as an external file. Here we discuss the different ways of embedding scripts in web pages.

6.9.1 Inside <BODY>

Placing scripts inside the <BODY> tag has been discussed in the beginning of this chapter. Here, the scripts will be executed while the contents of the web page is being loaded. The web page starts displaying from the beginning of the document. When the browser sees a script code in between, it renders the script and then the rest of the web page is displayed in the browser window.

Let us discuss this method with an example. The following is a web page to get the result of a candidate. The user can enter a register number in the text box. On clicking the **Get Result** button, a JavaScript function should check whether there is any entry in the register number box. If there is, it must be a number and it should have seven digits. The output of the web page is given in Figure 6.26.

Example 6.17: To create a web page that accepts a register number after validation

```
<HTML>
<HEAD> <TITLE>Javascript - Validation</TITLE> </HEAD>
<BODY>
    <FORM Name= "frmValid">
        <SCRIPT Language= "JavaScript">
            function checkData()
            {
                var rno;
                rno = document.frmValid.txtRegno.value;
                if (rno == "")
                {
                    alert("Please enter Register No.");
                    return;
                }
                if (isNaN(rno))
                {
                    alert("Invalid Register No.");
                    return;
                }
                if (rno.length < 7)
                {
                    alert("The Register No. must have 7 digits");
                    return;
                }
            }
        </SCRIPT>
        <CENTER>
            <BR>Enter Register Number
            <INPUT Type= "text" Name= "txtRegno">
            <BR><BR>
            <INPUT Type= "button" Value= "Get Result"
                                    onClick= "checkData()">
        </CENTER>
    </FORM>
</BODY>
</HTML>
```

A script can also be at the bottom of <BODY> tag. If the scripts are loaded inside the <BODY> tag or in the <HEAD> tag, they will be loaded along with the HTML

code. This causes a visual delay in loading the web page. If the scripts are placed just before </BODY> tag, the contents of a web page like text, images, etc. will be displayed on the screen faster. However, the disadvantage of this method is that the scripts that are to be executed while loading the web page may not work.

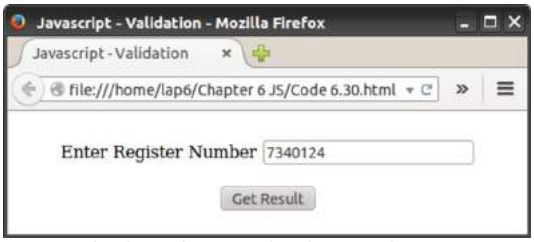


Fig. 6.26: Web page displaying the Form to accept register number

6.9.2 Inside <HEAD>

It is a usual practice to include scripts in the head section of the web page. We have been adding scripts in the head section in all our examples. The main reason for this is that the body section of most of the HTML pages contains a large volume of text that specifies the contents to be displayed on the web page. Mixing a function definition also with this will create a lot of confusion for the web designer for making any kind of modification in the page.More over, the head section of a web page is loaded before the body section. Therefore, if any function call is made in the body section, the function will be executed as the function definition is already loaded in the memory. In Example 6.17, the code within <SCRIPT> and </SCRIPT> has to be moved to the <HEAD> section of the HTML code.

6.9.3 External JavaScript file

We can place the scripts into an external file and then link to that file from within the HTML document. This file is saved with the extension '**.js**'. Placing JavaScripts in external files has some advantages. This is useful if the same script is used across multiple HTML pages or a whole website. It separates HTML and code which makes HTML and JavaScript easier to read and maintain. Storing JavaScript as separate files can speed up page loading. Note that in the above example (Example 6.17), the JavaScript code is stored as a separate file with the name "check.js". The contents of this file is as shown below:

```
function checkData()
{
    var rno;
    rno = document.frmValid.txtRegno.value;
    if (rno == "")
    {
        alert("Please enter Register No.");
        return;
    }
```

```
        if (isNaN(rno))
        {
            alert("Invalid Register No.");
            return;
        }
        if (rno.length < 7)
        {
            alert("The Register No. must have 7 digits");
            return;
        }
    }
```

Note that this file contains only JavaScript code and does not contain <SCRIPT> tag. <SCRIPT> tag is used inside the HTML file only. The file can be linked to HTML file using the <SCRIPT> tag. The Type attribute specifies that the linked file is a JavaScript file and the Src attribute specifies the location and file name of the external JavaScript file. The modified HTML code is given below.

```
<HTML>
<HEAD><TITLE>Javascript - Validation</TITLE>
    <SCRIPT Type= "text/JavaScript" Src= "checkdata.js">
    </SCRIPT>
</HEAD>
<BODY>
    <FORM Name= "frmValid">
        <CENTER>
            <BR>Enter Register Number
            <INPUT Type= "text" Name= "txtRegno">
            <BR><BR>
            <INPUT Type= "button" Value= "Get Result"
                                    onClick= "checkData()">
        </CENTER>
    </FORM>
</BODY>
</HTML>
```

If Src attribute is present, then contents of <SCRIPT> tag is ignored. That is, you cannot attach an external file and execute code in single <SCRIPT> tag. Two separate <SCRIPT> tags are needed for this. One with Src for external file and another with the code.

Know your progress

1. When will `onMouseEnter` event of JavaScript work?
2. The property of the drop-down list (`<SELECT>` element) used to get the index of the selected item is _____.
3. Compare `onKeyDown` and `onKeyUp` events of JavaScript.
4. What is the use of `Number()` function?
5. What is the advantage of placing JavaScript code just above the `</BODY>` tag?

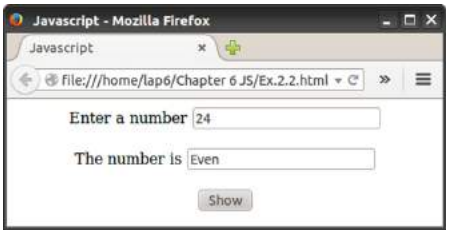
Let us conclude

This chapter introduces JavaScript as a client side scripting language that is used mainly for validations. The tag used in HTML to include JavaScript code and the popular functions used are explained here. The datatypes in JavaScript and the use of variables are also discussed in detail. The use of operators and control structures is similar to that of C++. The different built-in functions and the events in JavaScript are explained through examples. The different ways of including JavaScript code in HTML page are presented in detail.

Let us practice

1. Develop a web page to display the following screen. User can enter a number in the first text box. On clicking the show button, product of all numbers from 1 to the entered limit should be displayed in the second text box.

2. Develop a web page to display the following screen. User can enter a number in the first text box. On clicking the show button, Even or Odd should be displayed in the second text box depending on whether the number is even or odd.

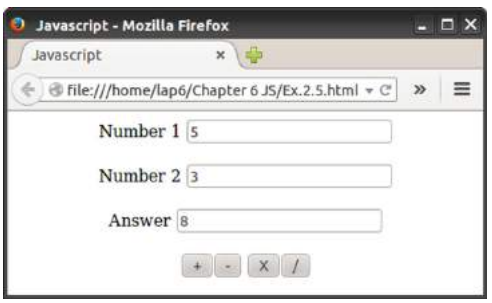


3. Develop a web page to display the following screen. The user can enter an age in the text box. If the user enters an alpbabet, instead of a number in the text

box, on clicking the show button, it should display a message "Invalid Age" to the user. Other wise it should display a message "Correct Data".

4. Develop a login page as shown in the figure. The page must contain one text box for entering the username and one password box for entering the password. The username must contain at least 4 characters and the password must contain at least 6 characters. The first two characters in the password must be numbers. On clicking the show button, if the valid data are given in boxes, a message "Correct Data" should be displayed. Otherwise, "Wrong Data" message should be displayed.

5. Develop a web page to implement a simple calculator. The page should have two text boxes to enter two numbers. It should also have 4 buttons to add, subtract, multiply and divide the two numbers. The answer should be displayed in a third text box on clicking the button. The web page should be as shown in the given figure.



Let us assess

1. Write the value of the variable z in each of the following:

a.
```
var x, y, z;
x = 5;
y = 3;
z = ++x - y--;
```

b.
```
var x, y, z;
x = "12";
y = 13;
z = x + y;
```

c.
```
var x, y, z;
x = 20;
y = 8;
x %= y;
z = x++;
```

d.
```
var x, y, z;
x = 1;
y = 4;
z = !(x < y);
```

e.
```
var x, y, z;
x = 5;
y = 6 ;
z = (x > y) || (y % 2 == 0);
```

2. Predict the output of the following

a.
```
<HTML>
<BODY>
<SCRIPT Language= "JavaScript">
var i;
for (i = 10; i >= 1; i--)
      document.write(i + "<BR>");
</SCRIPT>
</BODY>
</HTML>
```

b.
```
<HTML>
<BODY>
<SCRIPT Language= "JavaScript">
var i, s = 0;
for (i = 1; i <= 100; i += 2)
      s += i;
document.write("Sum = " + s);
</SCRIPT>
</BODY>
</HTML>
```

c.
```
<HTML>
<BODY>
<SCRIPT Language= "JavaScript">
var n, s = 0;
n = 0;
while (n <= 50)
{
      s = s + n;
      n = n + 5;
}
```

```
document.write("Sum = " + s);
</SCRIPT>
</BODY>
</HTML>
```

d.
```
<HTML>
<BODY>
<SCRIPT Language= "JavaScript">
var n, f = 1;
n = 5;
while ( n > 0)
{
      f = f * n;
      n--;
}
document.write("Product  = " + f);
</SCRIPT>
  </BODY>
  </HTML>
```

3. Following is an html code segment in a web page

```
<FORM Name= "frmStud">
<INPUT Type= "text" Name= "studentName">
</FORM>
```

Fill in the blanks to store the value of the text box to the variable n.

```
var n;
n = ..........................;
```

4. Suppose you have written a JavaScript function named checkData(). You want to execute the function when the mouse pointer is just moved over the button. How will you complete the following to do the same?

```
<INPUT Type= "button" .......... = "checkData()">
```

5. Explain <SCRIPT> tag and its attributes.
6. Write the syntax of a built-in function in JavaScript.
7. Classify the following values in JavaScript into suitable data types.
"Welcome", "123", "true", 67.4, .98, false, "hello"
8. What is meant by undefined data type in JavaScript mean?
9. Explain operators in JavaScript.

10. Write JavaScript functions to perform the following
    a. To check whether a variable N contains a number.
    b. To convert the string "scert" to all capitals.
    c. To convert the string "HTML" to all small letters.
    d. To display a message "Welcome to functions".
    e. To display the third character in the string "Computer".
11. Write JavaScript code to display the length of the string "Computer".
12. A web page contains a button. Write HTML code for the button which executes a function Message( ) on the occurance of the following events.
    a. When user clicks the mouse on the button.
    b. When user moves the mouse over the button.
13. What are the advantages of writing JavaScript code in the head section of an HTML page?
14. Design an HTML page that contains a text box to enter the marks in a given subject.
    a. Write HTML code for this web page.
    b. Provide validations for this text box in a separate JavaScript file and link it with the HTML file. The validations are (i) it should not be empty (ii) it should be a number (iii) it should be between 0 and 60.
    c. List the advantages of writing the script in a separate file.

7 Web Hosting

Significant Learning Outcomes

After the completion of this chapter, the learner:

- describes the use of a web server and the concept of web hosting.
- classifies different types of hosting.
- explains the way to buy hosting space.
- registers a domain and hosts a website using FTP client software.
- explains the features of free hosting.
- identifies the use of Content Management Systems.
- describes the need for responsive web design.

We familiarised ourselves with creating web pages in the earlier chapters. A website consisting of several web pages are designed to give information about an organisation, product, service, etc. Suppose we have developed a website for our school using HTML. How can we make this website available on the Internet? These web pages are to be stored in the web servers connected to the Internet, to be made available to others. This chapter presents an overview of web hosting, its different types and general features. For accessing a website we need a domain name. How domain names are chosen and registered are also presented here. The various FTP client softwares available to transfer the files of the website (web pages, images, etc.) from our computer to the server are also discussed. After learning this chapter, one will be able to register a domain name and host a website.

7.1 Web hosting

To develop a website for our school we need to design web pages. In the previous chapters, we discussed how to design web pages. Any text editor or a web designing tool can be used to develop a home page, a page for the courses

in the school, facilities available, contact address, etc. and link them using a menu to form an attractive website.

After developing the school website in our computer, it has to be made available in the Internet. The designed website has to be uploaded to a web server to make it available to Internet users all over the world. For this, either storage space is rented on a web server to store the web pages that we have created or our own web server is set up. Setting up a web server is very expensive compared to hosting a website in a rented storage space.

Web hosting is the service of providing storage space in a web server to serve files for a website to be made available on the Internet. The companies that provide web hosting services are called web hosts. Web hosts own and manage web servers. These web servers offer uninterrupted Internet connectivity, software packages for offering additional services such as databases and support for programming languages such as PHP, Java, ASP.NET, etc.

7.1.1 Types of web hosting

Suppose the entire content of our school website including html files, images, etc., require 4 MB of hard disk space on the web server. Web hosts provide only standard packages of 10 MB, 20 MB, etc. of space on the web server. We may need to choose the web host considering the packages offered by them that suit our purpose. The number of visitors expected to visit our website is also a factor. If our website requires a database, contains scripts, etc. then the support of such features are also to be considered while choosing a web host.

The type of web hosting has to be decided based on requirements like the amount of space needed for hosting, the number of visitors expected to visit the website, the use of resources like databases, programming support, etc. Web hosts provide different types of hosting packages. They can be shared hosting, virtual hosting and dedicated hosting.

a. Shared hosting: Shared web hosting is the most common type of web hosting. It is referred to as shared because many different websites are stored on one single web server and they share resources like RAM and CPU. The features available on shared web servers are generally basic and are not flexible to suit a website that require specific features like high bandwidth, large storage space, etc. Shared hosting is most suitable for small websites that have less traffic. Shared servers are cheaper and easy to use, since they are configured with the most popular options. The updates and the security issues of the software installed in the web server are taken care of by the hosting company itself. A drawback is that since the bandwidth is shared by several websites, if any of these has a large volume of traffic, it will slow down all other websites hosted in the shared server.

b. Dedicated hosting: Dedicated web hosting is the hosting where the client leases the entire web server and all its resources. The web server is not shared with any other website. Websites of large organisations, government departments, etc. where there are large numbers of visitors, opt for dedicated web hosting. Here, the client has the freedom to choose the hardware and software for the server and has full control over the web server. Dedicated servers provide guaranteed performance, but they are very expensive. The advantage of dedicated servers is that such servers are usually hosted in data centers where the service provider facilitates Internet connectivity, round-the-clock power supply, etc. and the technical expertise for managing web servers. The cost of setting up and managing these facilities is thus reduced for the client. Since the bandwidth is not shared with other websites, it speeds up the access of the website. If the client is allowed to place their own purchased web server in the service providers facility, then it is called co-location.

c. Virtual Private Server: A Virtual Private Server (VPS) is a physical server that is virtually partitioned into several servers using the virtualization technology. Each VPS works similar to a dedicated server and has its own separate server operating system, web server software and packages like e-mail, databases, etc. installed in it. Unlike shared hosting, VPS provides dedicated amount of RAM for each virtual web server. Each of these VPS works as a fully independent web server, as if each were running on a separate physical server. The users of VPS are provided with the rights to install and configure any software on their VPS. They are also given the right to restart their VPS without affecting other virtual servers running on the same physical server.

VPS hosting provides dedicated bandwidth to each website on the server. This provides the advantages of a dedicated hosting, even though the actual server is shared. This type of hosting is suitable for websites that require more features than that provided by shared hosting, but does not require all the features of dedicated hosting. VPS provides almost the same services at a lesser cost than that of dedicated hosting. Some popular server virtualization softwares are VMware, Virtualbox, FreeVPS, User-mode Linux, Microsoft Hyper-V, etc.

Figure 7.1 gives a symbolic representation of the different types of web hosting packages.

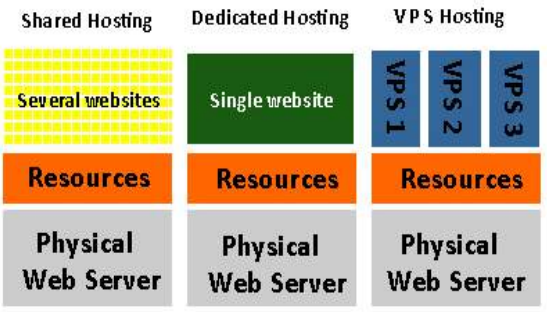


Fig. 7.1 : Types of web hosting

7.1.2 Buying hosting space

We have designed and stored the web pages of our school website in a folder in the computer. These files are to be copied to a web server to make it available on the Internet as shown in Figure 7.2. For this, the appropriate type of web hosting has to be chosen. It is preferred that the website is hosted using the services of a shared hosting service provider as it is cheaper and appropriate. Once the type of hosting server is decided, hosting space has to be purchased from a web hosting service provider.

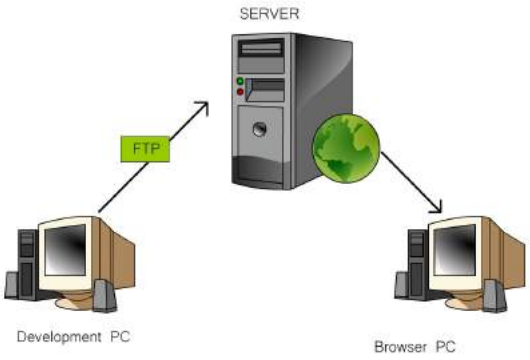


Fig. 7.2 : Making a website available on the Internet

While purchasing hosting space, several factors have to be taken into consideration. First, we have to decide on the amount of space required. We need to select a hosting space that is sufficient for storing the files of our website. If the web pages contain programming content, we need a supporting technology in the web server. The program used in the web page may require Windows hosting or Linux hosting. Here we need to choose between a Windows server or a Linux server as shown in Figure 7.3. If our website contains only HTML code, we can choose any server. Other features like database support, e-mail facility, etc. can also be considered while choosing the web host.

Fig. 7.3 : Hosting options

7.1.3 Domain name registration

We have now bought hosting space for our web pages and now we need an identification (URL) on the Internet. For this a suitable domain name has to be registered for our school. Domain names are used to identify a website in the Internet.

Most of the web hosting companies offer domain name registration services. After finalising a suitable domain name for our website, we have to check whether this domain name is available for registration or it is already in use by somebody else. The websites of the web hosting companies or web sites like www.whois.net provide an availability checker facility where we can check this. These websites check the database of ICANN that contains the list of all the registered domain names and gives a response as shown in Figure 7.4. If the domain name entered is available, we can proceed with the registration. The registration requires filling information for WHOIS database of domain names for ICANN. WHOIS information requires the name, address, telephone number and e-mail address of the registrant, as shown in Figure 7.5. This information can be made public or can be kept private according to the registrant's wish. After paying the annual registration fees online, the domain is purchased and is registered in our name. The shopping cart of the purchase is shown in Figure 7.6.

Fig. 7.4 : Domain name registration search result

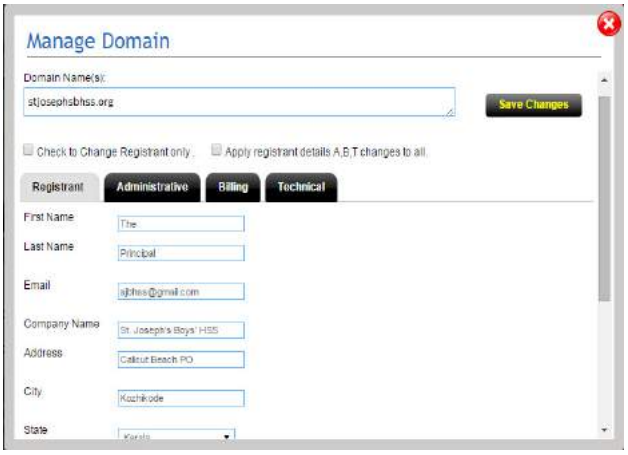


Fig. 7.5 : Providing WHOIS information

Thus, we have purchased a web server space for our school website and have registered a domain name for it. Now, when the user types our domain name, www.stjosephsbhss.org, in the browser window, it should display the webpage stored in the web server we have purchased. This will happen only if the DNS for server returns the IP address of our web server when the browser requests for it with www.stjosephsbhss.org.

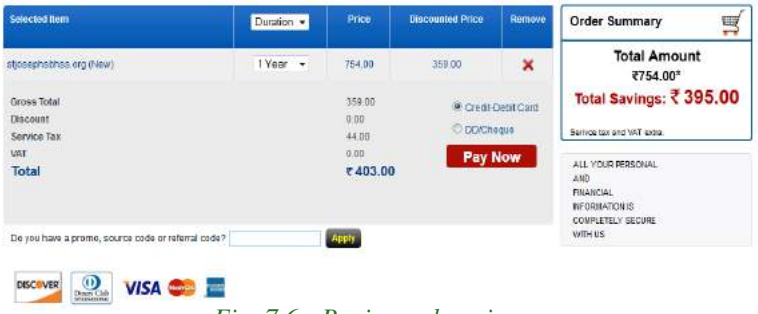


Fig. 7.6 : Buying a domain name

Therefore, our domain name has to be connected to the IP address of the web server where the web pages are stored. This is done using 'A record' (Address record) of the domain. An 'A record' is used to store the IP address of a web server connected to a domain name. The 'A record' can be modified by logging into the control panel of the domain. Here, you can set the 'A record' of the domain name to point to the IP address of web server as shown in Figure 7.7. After this, the DNS servers will be able to resolve our domain name to connect to our web server.

Fig. 7.7 : Changing 'A record' of the domain

A WHOIS search will provide information regarding a domain. It may include information like domain ownership, where and when registered, expiration date, etc. It is also used to determine whether a given domain name is available or not. A WHOIS lookup on www.kerala.gov.in returns the following.

```
Domain ID:D8944-AFIN
Domain Name:KERALA.GOV.IN
Created On:31-Dec-2003 05:00:00 UTC
Last Updated On:16-Jul-2014 11:37:59 UTC
Expiration Date:31-Dec-2016 05:00:00 UTC
Sponsoring Registrar:National Informatics Centre (R12-AFIN)
Status:OK
Registrant ID:R-R03120114034
Registrant Name:Government of Kerala
Registrant Organization:
Registrant Street1:Chief Minister's Office
Registrant Street2:
Registrant Street3:
Registrant City:Government Secretariate, Trivandrum, 69500
Registrant State/Province:Kerala
Registrant Postal Code:695001
Registrant Country:IN
```

Let us do

Prepare a comparative table of popular web hosts and their prices per year for their respective minimum hosting space. Provide pricing for both Windows and Linux web hosting.

Prepare a comparative table for the pricing for a year for .org and .com domain registrations in the popular service providers.

Search the WHOIS details of www.dhsekerala.gov.in and www.scert.kerala.gov.in in www.whois.net and prepare a chart of the details.

Know your progress

1. The companies that provide web hosting services are called _____.
2. List the factors that decide the type of web hosting.
3. VPS is
   a. Virtual Premium Service
   b. Virtual Private Service
   c. Virtual Premium Server
   e. Virtual Private Server
4. What is co-location?
5. What does WHOIS information contain?
6. Why is 'A record' important for a domain name?

7.1.4 FTP client software

After buying a space on the hosting server and a domain name, we need to transfer the files of the website from our computer to the web server. This requires an FTP client software.

We have already discussed FTP in Chapter 8, Computer Networks of Class XI. FTP is used to transfer files from one computer to another on the Internet. FTP client software establishes a connection with a remote server and is used to transfer files from our computer to the server computer. To connect to an FTP server, FTP client software requires a username and password and also the domain name. This is to be provided in the Site Manager dialog box as shown in Figure 7.8. FTP sends username and password to the server as plain text which is unsecure. Therefore nowadays, SSH FTP (SFTP) protocol which encrypts and sends usernames, passwords and data to the web server is used in the FTP software. SFTP uses Secure Shell (SSH) protocol which provides facilities for secure file transfer.

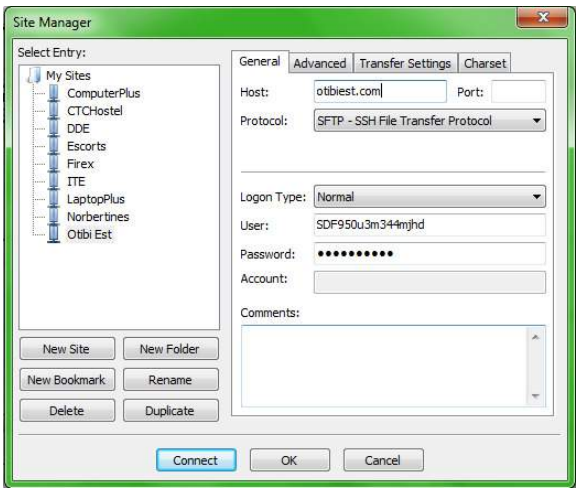


Fig. 7.8 : FTP Login

Once the FTP client is authenticated the IDE of FTP software appears as given in Figure 7.9. In this figure, the portion on the left side displays the folders and files in our computer and the right side displays the files in the web server computer. We

can use either the menu or 'drag and drop' files from the left side window to the right side window to upload the files to the web server. The files will then be transferred (copied) from our computer to the web server. The popular FTP client software are FileZilla, CuteFTP, SmartFTP, etc.

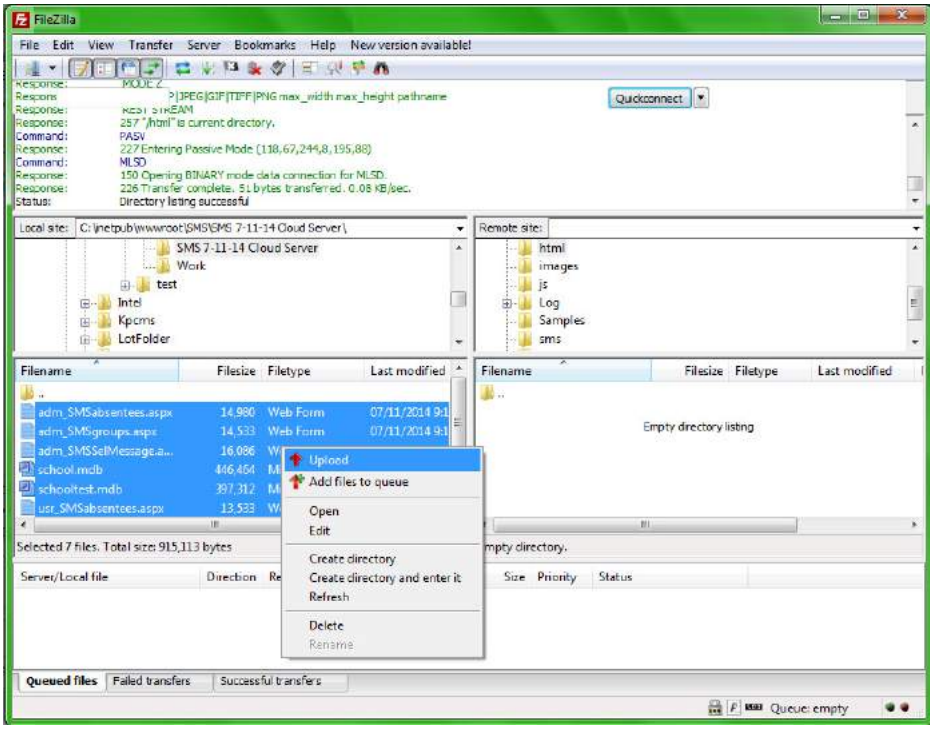


Fig. 7.9 : FTP client software IDE

Some web hosting companies provide their own control panel webpage through which users can upload the files. Such hosting companies do not allow third party FTP client software to upload files to their web servers. Figure 7.10 displays a control panel provided by a web hosting company to transfer files.

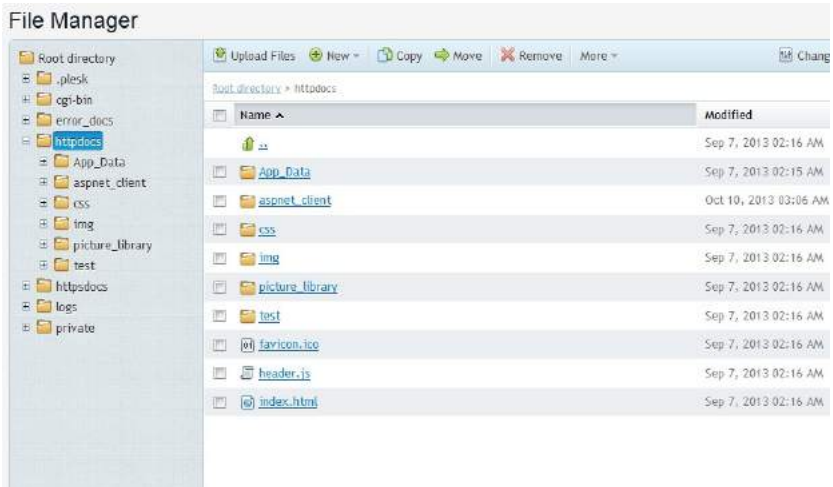


Fig. 7.10 : FTP control panel provided by a hosting company

7.2 Free hosting

Free hosting provides web hosting services free of charge. The service provider displays advertisements in the websites hosted to meet the expenses. Some free web hosting sites provide the facility to upload the files of our website in their server, but may place certain restrictions on the files. The size of the files that can be uploaded may be limited (supports upto 5 MB only), audio/video files (mp3, mp4, etc.) may not be permitted and so on. These websites usually provide control panels to upload files from our computer to the web server as shown in Figure 7.10. Some other websites only permit us to use the templates (pre-formatted designs) they provide for designing websites. They do not allow external files to be uploaded to their web server.

Free web hosting services usually provide either their own subdomain (oursite.example.com) or as a directory service (www.example.com/oursite) for accessing our websites. Some free web hosting companies provide domain name registration services also. Free web hosting is useful for sharing content on the web, among groups having similar interests like family unions, nonprofit organisations, etc. who are not able to spend money on web hosting. The availability of cheap web hosting services has reduced the need for free web hosting. Sites.google.com, yola.com, etc. are free web hosting services. A website of higher secondary school teachers that uses free hosting is given in Figure 7.11.

Fig. 7.11 : A free web site

7.3 Content Management System

Content Management System (CMS) refers to a web based software system which is capable of creating, administering and publishing websites. CMS provides an easy way to design and manage attractive websites.

Most CMS is available for free download at their websites. Copy the files of the CMS to the hosting space on our web server and configure the website as shown in Figure 7.12. Templates that provide a list of preset designs for websites are also

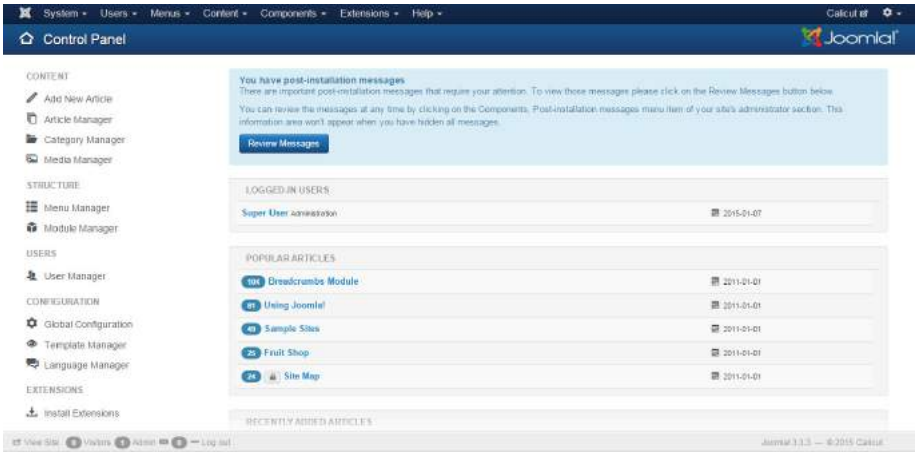


Fig. 7.12 : Configuring a website using Joomla!

available for download in some websites. Some sample templates available are displayed in Figure 7.13. The users need to choose a template from the given list, upload them to our website and add titles, images and other descriptions for the chosen design. This is as simple as designing a document in a word processor.

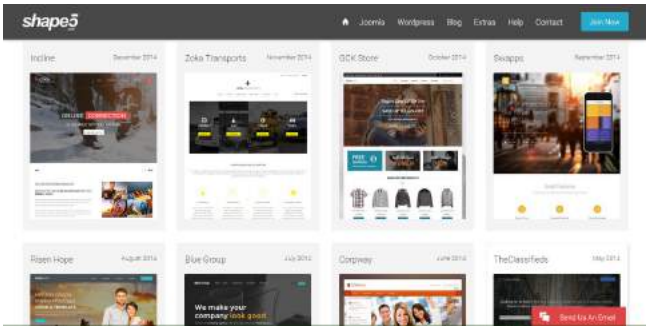


Fig. 7.13 : Templates available in Joomla!

Fig. 7.14 : A web site developed using Joomla CMS

CMS provides standard security features in its design, that help even people with less technical knowledge to design and develop secure websites. The CMS templates also reduce the need for repetitive coding and design for headings and menus that appear in all pages in a website, by internally providing features to add them in all pages. Templates are available for download for free or for a small price at

popular CMS websites. There are also third party vendors who customise the CMS for a fee. If you are hosting on a shared web server, verify whether the web server supports the CMS you have downloaded.

CMS is economical and now many organisations, bloggers, etc. use it for their websites. Some of the popular CMS software are WordPress, Drupal and Joomla! Figure 7.14 shows the website of Motor Vehicles department of Government of Kerala, developed using Joomla!

7.4 Responsive web design

Today we browse web pages using various devices like desktops, laptops, tablets and mobile phones. All these devices have different screen sizes. Traditional web pages are designed to be displayed on the screens of devices like desktops and laptops. These web pages are difficult to view when it is accessed using tablets and mobile phones. The user may have to use the scroll bar to move from one part of the web page to another. In earlier days, a separate website was created for the purpose of viewing in mobile devices. These websites contained web pages whose size matched the screen size of these devices. But maintaining two websites for a single organisation created issues. It would be better if the web page was able to adjust itself to the screen size of the device. This type of web page designing is called responsive web design. Figure 7.15 shows the appearance of a responsive website in different devices.

Fig. 7.15 : Responsive web design

Responsive web design is the custom of building a website suitable to work on every device and every screen size, no matter how large or small, mobile phone or desktop or television. The term 'responsive web designing' was coined by Ethan Marcotte, an independent designer and author, to describe a new way of designing for the ever-changing Web. Responsive web design can be implemented using flexible grid layout, flexible images and media queries. Flexible grid layouts set the size of the entire web page to fit the display size of the device. Flexible images and videos set the image/video dimensions to the percentage of display size of the device. Media queries provide the ability to specify different styles for individual devices. A horizontal menu in a web page for larger displays might have to be converted to a drop down menu for a mobile phone. These settings can be done using media queries inside the CSS file.

Screen sizes always vary - from a wearable device, mobile phones, tablets to laptops, desktops and televisions. Therefore, it is important that websites are designed to adapt to the screen size of the device.

Let us do

Design a website containing the name, age, total runs scored, no. of wickets taken, etc. of the members of the Indian cricket team and host the web site using any free hosting services.

Prepare a list of popular CMS providers and write their features.

In the web hosting space provided, upload the files of the school website you have prepared using any FTP software.

Let us conclude

After designing a website, it has to be hosted over the Internet using a suitable type of hosting. Websites of small organisations can be hosted using shared hosting, websites that have more traffic and need more security can opt for VPS hosting, whereas websites of large organisations or those with very high traffic require dedicated hosting. After buying a suitable web hosting space, FTP software can be used to connect to the web server and can securely transfer the files of the website to it. Domain name can be registered through a service provider and the 'A record' can be set to point the domain to the web server. There are free hosting websites that provide hosting free of cost. Content Management Systems (CMS) offer tools and standard security features in its design that helps even people with less technical knowledge to design and develop secure websites. Today, since we browse websites using devices like mobile phones, tablets, laptops, etc. that have different screen sizes, it is important to design websites that display themselves according to the size of the screen they are viewed from.

Let us assess

1. What do you mean by web hosting? Explain the different types of web hosting.
2. A supermarket in a city wishes to take its business online. It plans to accept orders for its products through a website and receive payments online.
   a. Which type of hosting is suitable for this website?
   b. Explain the reason for your choice.
3. Emil wishes to purchase the web hosting space required to host a website for his medical shop. List the features to be taken into consideration while buying hosting space on a web server.
4. How can we connect a website hosted in a webserver to a domain name?
5. What is the advantage of using SFTP protocol in FTP software?
6. Raju wishes to host a website for his family. What are the advantages that free web hosting companies provide?
7. What is CMS? What are the features of CMS? Give Examples.
8. Explain the need for applying responsive web design while developing websites today.
9. How is responsive web design implemented?

8 Database Management System

Significant Learning Outcomes

After the completion of this chapter, the learner

- recognises the need for files.
- identifies the major limitations of the conventional file management system.
- lists and explains the different advantages of the Database Management System.
- lists the various components of DBMS and explains their purpose.
- recognises the types of users and their roles in the DBMS environment.
- explains the levels of data abstraction and data independence in DBMS.
- explains relational data model by citing examples.
- uses different terminologies in RDBMS, appropriately.
- applies and evaluates various operations in Relational algebra.

This is an era of information. The survival of organisations in this competitive world largely depends on the need for information obtained with high accuracy and speed. We know that information is obtained through the processing of data. For this, a huge amount of data is to be collected, stored and processed to generate information. Every organisation that we can think of like schools, banks, business organisations, etc. are in need of information. Can you imagine how this huge quantity of data was handled traditionally? In earlier days, this was handled manually with data recorded in books termed as 'book keeping'. It is obvious that large storage space is a hazard and processing of data is laborious. With the advent of computers the data could be stored effectively, but the possibility of duplication, inconsistency, invalidity, etc. remained. This chapter provides an effective mechanism to overcome these limitations. The concept of Database Management System (DBMS) is introduced in this chapter as an effective record keeping system (earlier called book keeping). Various operations are also discussed to retrieve the required and relevant information from the database.

8.1 Concept of database

Consider the Single Window System (SWS) of the Department of Higher Secondary Education, maintaining a large collection (say 5,00,000 applications or 19 GB) of data concerning students, courses, schools and grades each year for Class XI admission process. This data is accessed simultaneously by several schools and students. Questions about the allotment of students and schools must be answered quickly, changes made to the data by different schools must be applied consistently, and access to certain parts of the data (e.g., Grades or WGPA) must be restricted.

We can manage the data by storing it in the conventional file management system. But, this approach has many drawbacks:

- We must keep more copies of the same data for different applications. This storage leads to duplication of data.
- There is no mechanism to protect the data from inconsistent changes made by different users accessing the database simultaneously.
- If the data are not properly organised, retrieval of information will be difficult, and time consuming, and there may be chances of inaccuracy in the information..
- There is no way to ensure that data is restored to a consistent state if the system crashes while changes are being made.
- Operating systems provide only a password mechanism for security. This is not sufficiently flexible to enforce security policies in data.
- There is no standardisation on data.

8.1.1 Need of database

The drawbacks mentioned above can be overcome by using database. In situations, where huge amount of data is to be mantained and manipulated, conventional file system will not be sufficient. In such a situation we use database. ***Database*** is an organized collection of inter-related data stored together with minimum redundancy, in a manner that makes them accessible for multiple applications. Now we can try to manage the data in SWS by storing it in a database. The software Database Management System (DBMS) is essentially a set of programs which facilitates storage, retrieval and management of database. The primary goal of DBMS is to provide an environment that is both convenient and efficient to use in retrieving and storing database.

8.1.2 Advantages of DBMS

Database systems are designed to manage large amounts of data. DBMS involves both the database structures definition for the storage of data and the provision of

mechanisms for the manipulation of data. In addition, the database system must ensure the safety of the data stored, against unauthorized access or system failure. If data is to be shared among several users, the system must avoid possible anomalous results. The database management system has a number of advantages over the traditional file management system. They are listed below.

- **Controlling data redundancy**: In file management systems, data may be placed in many files. The storing of the same data in multiple locations (may be in the same file or different files) or duplication of data is known as data redundancy. Redundancy leads to higher cost in storage and data access. Database systems do not maintain redundant data, instead all the data is kept at one place in a centralized manner. All the applications or users that require data refer to the centrally maintained database. Sometimes there can be technical or business reasons for maintaining several copies of the same data. However, redundancy should be carefully controlled in any case.
- **Data consistency**: Data redundancy may lead to data inconsistency; that is, the various copies of the same data show different values in different files. Assume that your class teacher and Principal maintain separate copies of the address list of all students admitted in your class. During periodic address change a few students report to the Principal and a few students report to the changes to the class teacher. After a certain period of time both the address lists become irrelevant and inconsistent since total corrections are not updated in both. By controlling data redundancy, data consistency is obtained. If a data item appears only once, any update to its value has to be performed only once and the updated value is immediately available to all users.
- **Efficient data access**: A DBMS utilises a variety of techniques to store and retrieve data efficiently.
- **Data integrity**: Data integrity refers to the overall completeness, accuracy and consistency of data in the database. This can be indicated by an absence of any alteration in data between two updates of a data record. Data integrity is imposed within a database at its design stage through the use of standard rules and procedures. Data integrity can be maintained through the use of error checking and validation routines.
- **Data security**: The information inside a database is valuable to any company or organization. Therefore it must be kept secure and private. Data security refers to the protection of data against accidental or intentional disclosure or unauthorized destruction or modification by unauthorized persons. The various programs and users may share data in common. But access to specific information can be limited to selected users by setting the access rights. Through

the use of passwords, the information in a database is made available only to authorised persons.

- **Sharing of data**: The data stored in the database can be shared among several users or programs even simultaneously and each may use it for different purposes.
- **Enforcement of standards**: With central control of the database, a Database Administrator (DBA) defines and enforces the necessary standards. Standards can be defined for data formats to facilitate exchange of data between systems. Applicable standards might include naming conventions, display formats, report structures, terminology, documentation standards, update procedures, access rules and so on. This facilitates communication and cooperation among various departments, projects and users within the organization.
- **Crash recovery**: When a system crashes, all or a portion of the data can become unusable. DBMS provides some mechanism to recover data from the crashes. Thus the DBMS protects data from the effects of system failures.

Know your progress

1. Storing of the same data in different places is called _____.
2. Address of a particular student is stored in two ways in a school record; this situation is known as _____.
3. Unauthorised accessing of data can be prevented from using _____.
4. Sharing of data can reduce data redundancy. State whether true or false?
5. Data redundancy will reduce data consistency. State whether true or false?

8.2 Components of the DBMS environment

DBMS have several components, each performing very significant tasks in its environment. The components are

- Hardware
- Software
- Data
- Users
- Procedures

Hardware: The hardware is the actual computer system used for storage and retrieval of the database. This includes computers (PCs, workstations, servers and supercomputers), storage devices (hard disks, magnetic tapes), network devices (hubs, switches, routers) and other supporting devices for keeping and retrieval of data.

Software: The software part consists of the actual DBMS, application programs and utilities. DBMS acts as a bridge between the user and the database. In other words, DBMS is the software that interacts with the users, application programs, and databases. All requests from users for access to the database are handled by the DBMS. Database management system software consists of several software components that handle various tasks such as data definition, data manipulation, data security, data integrity, data recovery and performance optimization. One general function provided by the DBMS is thus the shielding of database from complex hardware-level detail. The DBMS controls the access and helps to maintain the consistency of the data.

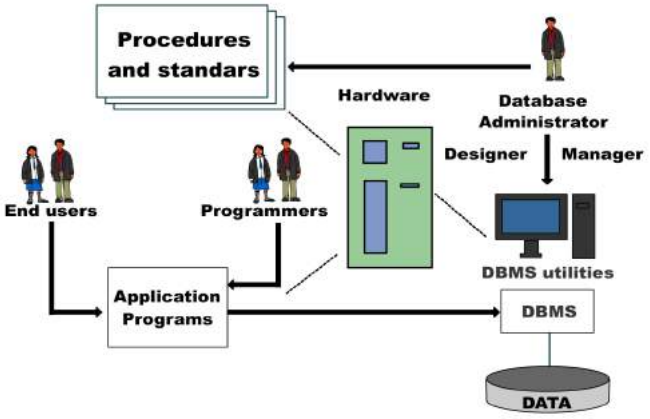


Fig. 8.1: Database system environment

Application programs are most commonly used to access data found within the database to generate reports, tabulations, and other information to facilitate decision making. Utilities are the software tools used to help manage the database system. For example, all major DBMS provide graphical user interfaces (GUIs) to help create database structures, control database access, and monitor database operations.

Data: It is the most important component of DBMS environment from the end users point of view. The database contains operational data and the meta-data (data about data). The database should contain all the data needed by the organization. The major feature of databases is that the actual data and the programs that uses the data are separated from each other. For effective storage and retrieval of information, data is organized as fields, records and files.

Assume a box containing a collection of cards which stores the Admission Number, Name, Batch, Result, Marks of students in a class. Each card will have the same format but the data written on them is different as in Figure 8.2.

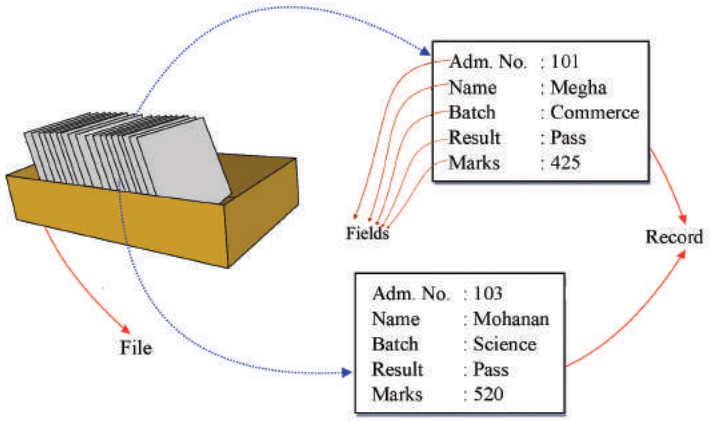


Fig. 8.2: Concept of data organisation

Fields: A field is the smallest unit of stored data. Each field consists of data of a specific type. In Figure 8.2 Adm. No., Name, Batch, Result, and Marks are the fields.

Record: A record is a collection of related fields. In the Figure 8.2, each card in the box contains the related fields of a record. For example, the fields 103, Mohanan, Science, Pass, and 520 constitute a record.

File: A file is a collection of all occurrences of same type of records. The box shown in Figure 8.2 may be considered as a file.

Users: There are a number of users who can access data on demand using application programs. The users of a database system can be classified depending on the mode of their interactions with DBMS. The different categories of users are Database Administrator (DBA), Application Programmers, Sophisticated users and Naive Users.

Procedures: Procedures refer to the instructions and rules that govern the design and use of the database. The users of the system and the person that manages the database require documented procedures on how to use or run the system. These may consist of instruction on how to:

i. log onto the DBMS.
ii. use a particular DBMS facility or application program.
iii. start and stop the DBMS.
iv. make backup copies of the database or handle hardware or software failures
v. reorganise the database across multiple disks, improve performance, or archive data to secondary storage.

Know your progress

1. Data about data is called _____.
2. List the name of major components of a database system?
3. Categorize the following components of DBMS environment. Hard disk, Switch, DBA, Payroll system, End user, railway reservation system.

8.3 Data abstraction and data independence

For the system to be usable, it must retrieve data efficiently. The need for efficiency has led designers to use complex data structures to represent data in the database. Since many database system users are not computer trained, developers hide the complexity from users through several levels of abstraction. The data in a DBMS is described at three levels of abstraction, as illustrated in Figure 8.3. The database description consists of a structure at each of these three levels of abstraction: the physical level, conceptual level and view level.

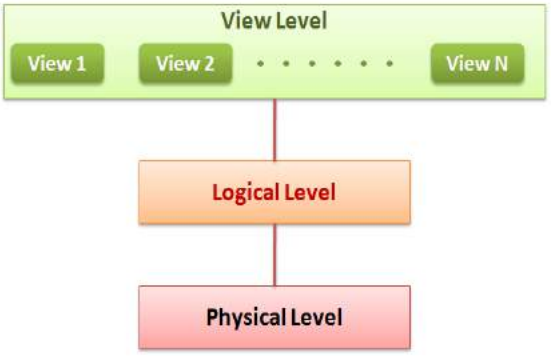


Fig. 8.3: Levels of abstraction

a. Physical level

The lowest level of abstraction describes how data is actually stored on secondary storage devices such as disks and tapes. The physical level describes complex low-level data structures in detail. We must decide what file organisations are to be used to store the relations and create auxiliary data structures, called indexes, to speed up data retrieval operations.

A sample physical schema for the SWS database follows:

- Store all relations as unsorted files of records. (A file in a DBMS is either a collection of records or a collection of data)
- Create indexes on the first column of the files Student, School, and Course.

b. Logical level

The next-higher level of abstraction describes what data is stored in the database, and what relationships exist among those data. The logical level thus describes the entire database in terms of a small number of relatively simple structures. Although

implementation of the simple structures at the logical level may involve complex physical-level structures, the user of the logical level does not need to be aware of this complexity. Database administrators, who must decide what information to keep in the database, use the logical level of abstraction. Logical level is also referred as conceptual level.

c. View level

View level is the highest level of database abstraction and is the closest to the users. It is concerned with the way in which individual users view the data. It describes only a part of the entire database. Most of the users of the database are not concerned with all the information that is contained in the database. Instead they need only a part of the database that is relevant to them. This simplifies their interaction with the system. The system may provide many views for the same database. Figure 8.4 shows the three levels of data abstraction for a STUDENT file with fields AdmNo, Name, Batch, Result, Marks.

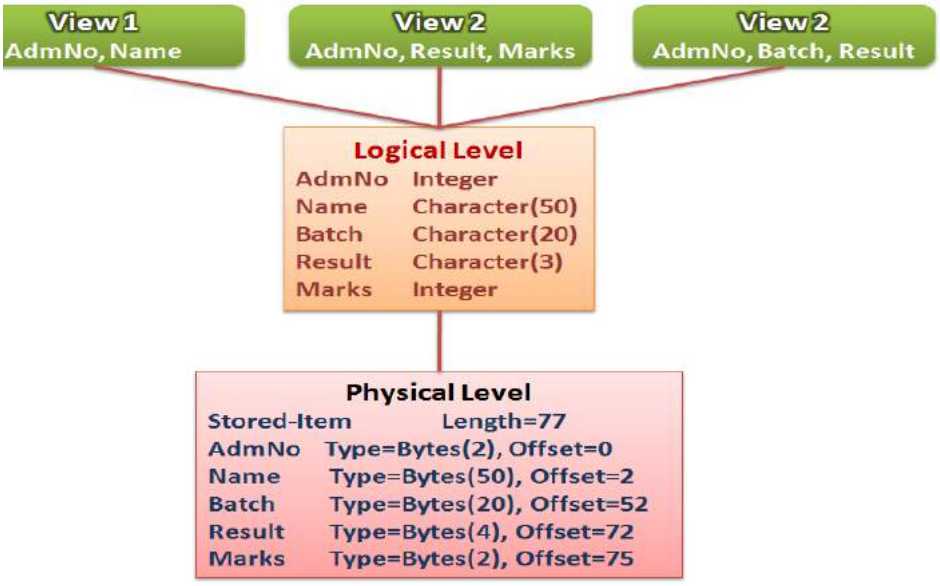


Fig. 8.4: Example for levels of abstraction

8.3.1 Data independence

Since a database may be viewed through three levels of abstraction, any change in the database structure at a particular level may affect the schema of other levels. The frequent changes made on database should not lead to the redesigning and re-implementation of the database. The ability to modify the schema definition (data structure definition) in one level without affecting the schema definition at the next

higher level is called data independence. There are two levels of data independence, physical data independence and logical data independence.

a. Physical data independence

Physical data independence refers to the ability to modify the schema followed at the physical level without affecting the schema followed at the conceptual level. That is, the application programs remain the same even though the schema at physical level gets modified.

b. Logical data independence

Logical data independence refers to the ability to modify a conceptual schema without causing any changes in the schema followed at view (external) level. The logical data independence ensures that the application programs remain the same.

It is more difficult to achieve logical data independence than physical data independence because the application programs are heavily dependent on the logical structure of the database.

8.4 Users of database

Depending on the degrees of expertise or the mode of the interactions with DBMS the users of a database system can be classified into the following groups:

- Database Administrator (DBA)
- Application Programmers
- Sophisticated Users
- Naive Users

8.4.1 Database Administrator

The person who is responsible for the control of the centralized and shared database is the Database Administrator (DBA). The DBA is responsible for many critical tasks such as,

Design of the conceptual and physical schemas: The DBA is responsible for interacting with the users of the system to understand what data is to be stored in the DBMS and how it is likely to be used. Based on this knowledge, the DBA must design the conceptual schema and the physical schema.

Security and authorization: The DBA is responsible for ensuring authorized access of data. For example, in a school, teachers allow students to find out course details, results of the student and the details as to who teaches a particular subject. At the same time students shall not be permitted to see teachers' salaries or grades of

other students. The DBA can enforce this policy by giving permission to students to read only the course view.

Data availability and recovery from failures: The DBA must take steps to restore the data to a consistent state when the system fails to complete a transaction or in case of a system crash. The DBMS provides software support for these functions, but the DBA is responsible for implementing procedures to back up the data periodically and maintain logs (special files for storing all activities in a database such as insertion, deletion, updation etc.) of system activity (to facilitate recovery from a crash).

8.4.2 Application programmers

Application programmers are computer professionals who interact with the DBMS through application programs. Application programs are programs written in any host language (for example Visual Basic, C, C ++, Java, etc.) and interact with the DBMS through Data Manipulation Language (DML). Application programs should ideally access data through the external schema.

8.4.3 Sophisticated users

Sophisticated users include engineers, scientists, business analysts, and others who are thoroughly familiar with the facilities of the DBMS. They interact with the systems through their own queries (a request to a database) to meet their complex requirements.

8.4.4 Naive users

Naive users interact with the system by invoking one of the application programs that were written previously. They are not concerned with or even aware of the details of the DBMS. Naive users deal only with the higher level of abstraction. People accessing data over the web, clerical staff in an office, billing clerk in a supermarket or hotels, bank clerk, etc. are examples of some naive users.

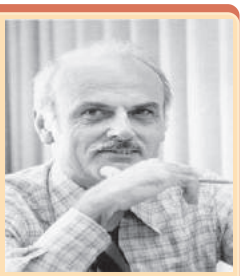
Know your progress

1. The person who interacts with the database through query language is called _____
2. The billing clerk in a Supermarket is a _____ user.
3. Who provides data security in a database?
4. Who changes the structure of a database?
5. _____ interacts with the database through the prewritten application program.

8.5 Relational data model

The relational data model represents database as a collection of tables called relations, each of which is assigned a unique name. In relational model, both data and the relationships among them are represented in tabular form. This representation enables even beginners to understand the concepts of a database easily.

Edgar Frank Codd (19 August 1923 - 18 April 2003) was an English computer scientist who invented the relational model for database management. He was born on the Isle of Portland in England. He served as a pilot in the Royal Air Force during the Second World War. In 1948, he joined IBM. He received the Turing Award in 1981. He died of heart failure at his home in Williams Island, Florida, at the age of 79 on 18 April 2003.



Today, a vast majority of database products are based on the relational model and they are known as Relational DataBase Management System (RDBMS). The major advantages of the relational model over the other data models are its simple data representation and the ease with which even complex queries can be expressed. The popular RDBMS are Oracle, Microsoft SQL Server, MySQL, DB2, Informix and Ingress. Most commercial relational database systems offer a query language that includes Structured Query Language (SQL), Query-by-Example (QBE) or Datalog. We shall study the widely used query language SQL in the Chapter 9.

8.6 Terminologies in RDBMS

Before discussing the operations on relational databases, let us be familiar with some terminologies associated with RDBMS.

a. Entity

An entity is a person or a thing in the real world that is distinguishable from others. For example, each student is an entity, and each school can be considered as another entity.

b. Relation

Relation is a collection of data elements organized in terms of rows and columns. A relation is also called Table. A sample relation named STUDENT is shown in Table 8.1.

STUDENT relation

AdmNo	Roll	Name	Batch	Marks	Result
101	24	Sachin	Science	480	EHS
102	14	Rahul	Commerce	410	EHS
103	4	Fathima	Humanities	200	NHS
104	12	Mahesh	Commerce	180	NHS
105	24	Nelson	Humanities	385	EHS
106	8	Joseph	Commerce	350	EHS
107	24	Shaji	Humanities	205	NHS
108	2	Bincy	Science	300	EHS

Table. 8.1: A sample relation

c. Tuple

The rows (records) of a relation are generally referred to as tuples. A row consists of a complete set of values used to represent a particular entity. In Table 8.1, each row in the STUDENT relation represents the result of a particular student.

d. Attribute

The columns of a relation are called attributes. AdmNo, Roll, Name, Batch, Marks and Result are attributes of the STUDENT relation. The values of each attribute are taken from the range of possible values called domain.

e. Degree

The number of attributes in a relation determines the degree of a relation. The relation STUDENT has six columns or attributes and therefore the degree of the STUDENT relation is 6.

f. Cardinality

The number of rows or tuples in a relation is called cardinality of the relation. The relation STUDENT has eight tuples and hence the cardinality of the STUDENT relation is 8.

g. Domain

A domain is a pool of values from which actual values appearing in a given column are drawn. For example, the domain of the column Batch in the relation STUDENT shown in Table 8.1 is the set of values {Science, Humanities, Commerce}. That is,

any one of the values from this set only can appear in the column Batch. Similarly, the set {EHS, NHS} is the domain of the column Results.

h. Schema

The description or structure of a database is called the database schema, which is specified during database design. In the relational model, the schema for a relation specifies its name, the name of each column, and the type of each column. As an example, student information in a School database may be stored in a relation with the following structure:

STUDENT (Admno : integer,
Roll : integer,
Name : character(50),
Batch : character(20),
Marks : decimal,
Result : character(4))

i. Instance

An instance of a relation is a set of tuples in which each tuple has the same number of fields as the relational schema. The preceding schema says that each row in the STUDENT relation has six columns, with column names and types as indicated. An example for instance of the STUDENT relation is shown in Table 8.1.

Know your progress

1. Organization of data in terms of rows and columns is called _____.
2. _____ in a table gives the complete data of a particular entity.
3. Number of rows in a relation is called _____.
4. Number of _____ in a relation is called degree of the relation.
5. In the relational model, data is organised as _____.

8.6.1 Keys

A relation is defined as a set of tuples. All the tuples in a relation must be distinct. That is, no two tuples can have the same combination of values for all their attributes. Therefore there should be a way to identify a tuple in a relation. The concept of a key allows us to make such distinctions. A key is an attribute or a collection of attributes in a relation that uniquely distinguishes each tuples from other tuples in a

given relation. If a key consists of more than one attribute then it is called a composite key. In the extreme, the entire tuple is the key since each tuple in the relation is guaranteed to be unique. However, we are interested in smaller keys, if they exist, for a number of practical reasons.

a. Candidate key

A candidate key is the minimal set of attributes that uniquely identifies a row in a relation. In the STUDENT relation of Table 8.1, AdmNo can uniquely identify a row. Therefore it can be considered as a candidate key. There may be more than one candidate key in a relation. Also, a candidate key need not be just one single attribute. It can be a composite key. For example, a combination of Roll, Batch and Marks can also be used to identify a particular student. Therefore, Roll + Batch + Year can be considered as another candidate key of STUDENT relation.

b. Primary key

A primary key is one of the candidate keys chosen to be the unique identifier for that table by the database designer. A primary key is a set of one or more attributes that can uniquely identify tuples within the relation. As it uniquely identifies each entity, it cannot contain null value and duplicate value.

Candidate keys are considered as candidates for primary key position. From the candidate keys the one with the least number of attributes may be selected as primary key. In our example (STUDENT relation), the attribute AdmNo can be used as the primary key. That is, no two students in the STUDENT relation can have the same AdmNo. In Table 8.1, we can see unique values in the column Name, but in real case scenario, more than one student can have the same name.

c. Alternate key

A candidate key that is not the primary key is called an alternate key. In the case of two or more candidate keys, only one of them can serve as the primary key. The rest of them are alternate keys. In our example the combination of Roll + Batch + Year is the alternate key since AdmNo is taken as the primary key.

d. Foreign key

A key in a table can be called foreign key if it is a primary key in another table. Since a foreign key can be used to link two or more tables it is also called a reference key. Suppose we have used Batch code instead of Batch name as shown in Table 8.2 and we have a relation BATCH as shown in Table 8.3. It is clear that BatchCode is the primary key in relation Batch, but it is used in STUDENT table as a non-key attribute. So, BatchCode is referred to as a foreign key with respect to STUDENT relation.

STUDENT relation

AdmNo	Roll	Name	BatchCode	Marks	Result
101	24	Sachin	S2	480	EHS
102	14	Rahul	C2	410	EHS
103	4	Fathima	H2	200	NHS
104	12	Mahesh	C2	180	NHS
105	24	Nelson	H2	385	EHS
106	8	Joseph	C2	350	EHS
107	24	Shaji	H2	205	NHS
108	2	Bincy	S2	300	EHS

Table 8.2: Modified STUDENT relation

BATCH relation

BatchCode	BatchName	Strength
S1	Science	150
S2	Science	150
C1	Commerce	100
C2	Commerce	100
H1	Humanities	100
H2	Humanities	100

Table 8.3: Instance of Batch relation

Know your progress

1. The minimal set of attributes that uniquely identifies a row in a relation is _____
2. In a table Employee Emp_code, Pan_no are the candidate keys. If Emp_code is taken as the primary key then Pan_no is _____ key.
3. How many primary keys are possible in a relation?
4. If a key consists of more than one attributes then it is called _____

8.7 Relational algebra

We have discussed the features of relational model, which provides facilities for craeting a database. Once the database is designed and data is stored, the required information is to be retrieved. A variety of operations are provided by RDBMS. The collection of operations that is used to manipulate the entire relations of a database is known as relational algebra. These operations are performed with the

help of a special language associated with the relational model, called query language. We will learn this language in the next chapter to perform these operations. The operations involved in relational algebra take one or two relations as input and produces a new relation as the result. The fundamental operations in relational algebra are SELECT, PROJECT, UNION, INTERSECTION, SET DIFFERENCE, CARTESIAN PRODUCT, etc. The SELECT and PROJECT operations are unary operations because they operate on one relation. The remaining operations are binary operations as they operate on pairs of relations.

8.7.1 SELECT operation

SELECT operation is used to select rows from a relation that satisfies a given predicate. The predicate is a user defined condition to select rows of user's choice. This operation is denoted using lower case letter sigma (σ). The general format of select is as follows:

$$\sigma_{\text{condition}}(\text{Relation})$$

The result of SELECT operation is another relation containing all the rows satisfying the given predicate (or conditions). The relational algebra uses various comparison operators < (less than), <= (less than or equal to), > (greater than), >= (greater than or equal to), = (equal to) and < > (not equal to) to set up simple conditions, and logical operators $\vee$ (OR), $\wedge$ (AND) and ! (NOT) to construct composite conditions.

To illustrate the SELECT operation, consider the relation STUDENT given in Table 8.1. The following examples show how SELECT operations are expressed in relational algebra and what output they produce.

Example 8.1: To select all the students who are eligible for higher studies.

$$\sigma_{\text{Result="EHS"}}(\text{STUDENT})$$

The output of this operation is a relation as shown in Table 8.4.

AdmNo	Roll	Name	Batch	Marks	Result
101	24	Sachin	Science	480	EHS
102	14	Rahul	Commerce	410	EHS
105	24	Nelson	Humanities	385	EHS
106	8	Joseph	Commerce	350	EHS
108	2	Bincy	Science	300	EHS

Table 8.4: Output of Example 8.1

Example 8.2: To select all the students in the Commerce batch who are failed.

$$\sigma_{\text{Result="NHS"} \wedge \text{Batch="Commerce"}} (\text{STUDENT})$$

The result of this operation is a relation as shown in Table 8.5

AdmNo	Roll	Name	Batch	Marks	Result
104	12	Mahesh	Commerce	180	NHS

Table 8.5: Output of Example 8.2

Example 8.3: To select all the students in the batch Science or Commerce.

$$\sigma_{\text{Batch="Science"} \vee \text{Batch="Commerce"}} (\text{STUDENT})$$

Table 8.6 shows the output this operation.

AdmNo	Roll	Name	Batch	Marks	Result
101	24	Sachin	Science	480	EHS
102	14	Rahul	Commerce	410	EHS
104	12	Mahesh	Commerce	180	NHS
106	8	Joseph	Commerce	350	EHS
108	2	Bincy	Science	300	EHS

Table 8.6: Output of Example 8.3

8.7.2 PROJECT operation

The PROJECT operation selects certain attributes from the table and forms a new relation. If the user is interested in selecting the values of a few attributes, rather than all the attributes of the relation, then use PROJECT operation. It is denoted by lower case letter π. The general format of project operation is as follows:

$$\pi_{A1, A2, \ldots, An} (\text{Relation})$$

Here A1, A2, ..., An refer to the various attributes that would make up the relation specified.

Example 8.4: Select Name, Result and Marks attributes in STUDENT relation.

$$\pi_{\text{Name, Marks, Result}} (\text{STUDENT})$$

The output of this operation is given in Table 8.7.

It is possible to combine the SELECT and PROJECT operations into a single statement. The illustration of this is shown in Examples 8.5 and 8.6.

Name	Marks	Result
Sachin	480	EHS
Rahul	410	EHS
Fathima	200	NHS
Mahesh	180	NHS
Nelson	385	EHS
Joseph	350	EHS
Shaji	205	NHS
Bincy	300	EHS

Table 8.7: Output relation of Example 8.4

Example 8.5: To select admission number and name of students who are Eligible for Higher Studies.

$$\pi_{\text{AdmNo, Name}}(\sigma_{\text{result="EHS"}}(\text{STUDENT}))$$

The resultant relation of this operation is given in Table 8.8. Compare it with Table 8.4 for the verfication of correctness of the result.

AdmNo	Name
101	Sachin
102	Rahul
105	Nelson
106	Joseph
108	Bincy

Table 8.8: Output relation of Example 8.5

Example 8.6: To select name and marks of those students in the Humanities batch who are Not eligible for Higher Studies.

$$\pi_{\text{Name, Marks}}(\sigma_{\text{result="NHS"} \wedge \text{Batch="Humanities"}}(\text{STUDENT}))$$

The result of this nested operation is shown in Table 8.9.

Name	Marks
Fathima	200
Shaji	205

Table 8.9: Output relation of Example 8.6

8.7.3 UNION operation

UNION operation is a binary operation and it returns a relation containing all tuples appearing in either or both of the two specified relations. It is denoted by $\cup$. The two relations must be union-compatible, and the schema of the result is defined to be identical to the schema of the first relation. If two relations are union-compatible, then they have the same number of attributes, and corresponding attributes, taken in order from left to right, have the same domain. Note that attribute names are not used in defining union-compatibility.

Consider two relations ARTS and SPORTS given in Tables 8.10 and 8.11 containing the details of students who participate in the arts festival and the sports meet of a school, respectively. Both the relations ARTS and SPORTS consist of AdmNo, Name and BatchCode as fields. It is clear that that these two relations are union

ARTS relation

AdmNo	Name	BatchCode
101	Sachin	S2
103	Fathima	H2
106	Joseph	C2
110	Nikitha	S1
132	Vivek	C1
154	Nevin	C1

Table 8.10: Instance of ARTS relation

SPORTS relation

AdmNo	Name	BatchCode
102	Rahul	C2
103	Fathima	H2
105	Nelson	H2
106	Joseph	C2
108	Bincy	S2
132	Vivek	C1
164	Rachana	S1

Table 8.11: Instance of SPORTS relation

compatible. That is, the two relations have the same number of attributes and the type of the corresponding attributes are also the same.

Relation ARTS ⋃ SPORTS returns the details of the students participated in arts or sports or both. That is, the expression ARTS ⋃ SPORTS returns a table as shown in Table 8.12. This table consists of the records belonging to the tables ARTS or SPORTS or both, eliminating the duplication. In Table 8.12, we can see that records of students with admission numbers 103, 106, and 132 appear only once in the relation.

AdmNo	Name	BatchCode
101	Sachin	S2
103	Fathima	H2
106	Joseph	C2
110	Nikitha	S1
132	Vivek	C1
154	Nevin	C1
102	Rahul	C2
105	Nelson	H2
108	Bincy	S2
164	Rachana	S1

Table. 8.12: Relation of ARTS U SPORTS

8.7.4 INTERSECTION operation

INTERSECTION operation is also a binary operation and it returns a relation containing the tuples appearing in both of the two specified relations. It is denoted by ⋂. The two relations must be union-compatible, and the schema of the result is defined to be identical to the schema of the first relation.

If we apply INTERSECT operation on the relations in Tables 8.10 and 8.11, the expression ARTS ⋂ SPORTS returns the details of the students participated in both arts and sports. That is ARTS ⋂ SPORTS returns a table consisting of rows common to ARTS and SPORTS as shown in Table 8.13.

AdmNo	Name	BatchCode
103	Fathima	H2
106	Joseph	C2
132	Vivek	C1

Table. 8.13: Relation of ARTS SPORTS

8.7.5 SET DIFFERENCE operation

SET DIFFERENCE operation is also a binary operation and it returns a relation containing the tuples appearing in the first relation but not in the second relation. It is denoted by – (minus). The two relations must be union-compatible, and the schema of the result is defined to be identical to the schema of the first relation.

The result of SET DIFFERENCE operation ARTS – SPORTS on Tables 8.10 and 8.11 returns the details of the students participated in arts but not in sports. That is, the resultant table will contain the rows appearing in relation ARTS but not in relation SPORTS as given in Table 8.14.

Relation SPORTS – ARTS returns the details of the students participated in sports but not in arts as shown in Table 8.15.

AdmNo	Name	BatchCode
101	Sachin	S2
110	Nikitha	S1
154	Nevin	C1

Table. 8.14: Relation of ARTS - SPORTS

Union and Intersection operations are commutative, that is the order of relation is not important. For example, the result of ARTS ∪ SPORTS and SPORTS ∪ ARTS are same. Also the result of ARTS ∩ SPORTS and SPORTS ∩ ARTS are the same. But Set Difference operation is not commutative, that is the order of relation is important. For example, the result of ARTS – SPORTS and SPORTS – ARTS are not same (refer to Tables 8.14 and 8.15).

AdmNo	Name	BatchCode
101	Rahul	C2
105	Nelson	H2
108	Bincy	S2
164	Rachana	S1

Table. 8.15: Relation of SPORTS - ARTS

8.7.6 CARTESIAN PRODUCT operation

CARTESIAN PRODUCT returns a relation consisting of all possible combinations of tuples from two relations. It is a binary operation on relations, which has a degree (number of attributes) equal to the sum of the degrees of the two relations operated upon. The cardinality (number of tuples) of the new relation is the product of the number of tuples of the two relations operated upon. CARTESIAN PRODUCT is denoted by × (cross). It is also called CROSS PRODUCT. All the tuples of the first relation are concatenated with tuples of the second relation to form tuples of the new relation.

Let us consider a relation TEACHER as shown in Table 8.16, which contains the details of teachers in the school. We can use this relation to perform cartesian production operation with STUDENT relation in Table 8.2. The output of the operation STUDENT × TEACHER is shown in Table 8.17. This table shows that each record of STUDENT relation is concatenated with the rows in TEACHER relation.

TEACHER relation

TeacherId	Name	Dept
1001	Viswesaran	English
1002	Meenakshi	Computer

Table 8.16: Instance of TEACHER relation

Adm No	Roll	Name	Batch Code	Marks	Result	TeacherId	Name	Dept
101	24	Sachin	S2	480	EHS	1001	Viswesaran	English
101	24	Sachin	S2	480	EHS	1002	Meenakshi	Computer
102	14	Rahul	C2	410	EHS	1001	Viswesaran	English
102	14	Rahul	C2	410	EHS	1002	Meenakshi	Computer
103	4	Fathima	H2	200	NHS	1001	Viswesaran	English
103	4	Fathima	H2	200	NHS	1002	Meenakshi	Computer
104	12	Mahesh	C2	180	NHS	1001	Viswesaran	English
104	12	Mahesh	C2	180	NHS	1002	Meenakshi	Computer
105	24	Nelson	H2	385	EHS	1001	Viswesaran	English
105	24	Nelson	H2	385	EHS	1002	Meenakshi	Computer
106	8	Joseph	C2	350	EHS	1001	Viswesaran	English
106	8	Joseph	C2	350	EHS	1002	Meenakshi	Computer
107	24	Shaji	H2	205	NHS	1001	Viswesaran	English
107	24	Shaji	H2	205	NHS	1002	Meenakshi	Computer
108	2	Bincy	S2	300	EHS	1001	Viswesaran	English
108	2	Bincy	S2	300	EHS	1002	Meenakshi	Computer

Table 8.17: Result of STUDENT X TEACHER operation

A Database model defines the logical design of data. The model describes the relationships between different parts of the data. The different models used in database design are Hierarchical Model, Network Model, Relational Model and Object-Oriented Model. Hierarchical structures were widely used in the early mainframe database management systems, such as the Information Management System (IMS) by IBM. Popular DBMS product in Network model were Cincom Systems' Total and Cullinet's IDMS.

Let us conclude

We have discussed the basic concepts of DBMS and its components. The advantages of database over traditional file system have been detailed. A brief idea about various terminologies associated with database is presented in the context of relational data model. Once the data is organised systematically in database, the operations provided by relational algebra to generate required information are

experienced with the help of sample relations. A good understanding about the concepts introduced by this chapter is essential to learn the next chapter effectively. In that chapter we will discuss how database is created and information is retrieved using a query language.

Let us assess

1. Who is responsible for managing and controlling the activities associated with the database?
   a. Database administrator b. Programmer
   c. Naive user d. End user
2. In the relational model, cardinality is the
   a. number of tuples b. number of attributes
   c. number of tables d. number of constraints
3. Cartesian product in relational algebra is
   a. a Unary operator b. a Binary operator
   c. a Ternary operator d. not defined
4. Abstraction of the database can be viewed as
   a. two levels b. four levels
   c. three levels d. one level
5. In a relational model, relations are termed as
   a. tuples b. attributes
   c. tables d. rows
6. In the abstraction of a database system the external level is the
   a. physical level b. logical level
   c. conceptual level d. view level
7. Related fields in a database are grouped to form a
   a. data file b. data record
   c. menu d.bank
8. A relational database developer refers to a record as
   a. criteria b. relation
   c. tuple d. attribute
9. An advantage of the database management approach is
   a. data is dependent on programs
   b. data redundancy increases
   c. data is integrated and can be accessed by multiple programs
   d. none of the above

10. Data independence means
    a. data is defined separately and not included in programs
    b. programs are not dependent on the physical attributes of data
    c. programs are not dependent on the logical attributes of data
    d. both (b) and (c)
11. Key to represent relationship between tables is called
    a. primary key
    b. candidate Key
    c. foreign Key
    d. alternate Key
12. Which of the following operations is used if we are interested only in certain columns of a table?
    a. PROJECTION
    b. SELECTION
    c. UNION
    d. SELECT
13. Which of the following operations need the participating relations to be union compatible?
    a. UNION
    b. INTERSECTION
    c. SET DIFFERENCE
    d. All of the above
14. Which database level is closest to the users?
    a. External
    b. Internal
    c. Physical
    d. Conceptual
15. The result of the UNION operation between R1 and R2 is a relation that includes
    a. all the tuples of R1
    b. all the tuples of R2
    c. all the tuples of R1 and R2
    d. all the tuples of R1 and R2 which have common columns
16. A file manipulation command that extracts some of the records from a file is called
    a. SELECT
    b. PROJECT
    c. JOIN
    d. PRODUCT
17. An instance of relational schema R (A, B, C) has distinct values of A including NULL values. Which one of the following is true?
    a. A is a candidate key
    b. A is not a candidate key
    c. A is a primary Key
    d. Both (a) and (c)

18. How many distinct tuples are there in a relation instance with cardinality 22?
    a. 22 b. 11
    c. 1 d. None
19. A set of possible data values is called
    a. Attribute b. Degree
    c. Tuple d. Domain
20. Why should you choose a database system instead of simply storing data in conventional files?
21. Explain the different levels of data abstraction in DBMS?
22. How are schema layers related to the concepts of logical and physical data independence?
23. Consider the instance of the EMPLOYEE relation shown in the following table. Identify the attributes, degree, cardinality and domain of Name and Emp_code.

Emp_Code	Name	Department	Designation	Salary
1000	Sudheesh	Purchase	Manager	25000
1001	Dhanya	Sales	Manager	25000
1002	Fathima	Marketing	Clerk	12000
1003	Shajan	Sales	Clerk	13000

24. Identify primary key, candidate keys and alternate keys in the instance of EMPLOYEE relation in Question 23.
25. Consider the instance of the STUDENT relation shown in the following table. Assume Reg_no as the primary key.
    a. Identify the candidate keys and alternate keys in the STUDENT relation.
    b. How are the primary key and the candidate key related?

Reg_no	Name	Batch	Result	Marks
101	Sachin	Science	Pass	480
103	Fathima	Humanities	Fail	200
106	Joseph	Commerce	Pass	350
108	Bincy	Science	Pass	300

26. What is a database? Describe the advantages and disadvantages of using DBMS.
27. What is data independence? Explain the difference between physical and logical data independence.
28. Enforcement of standard is an essential feature of DBMS. How are these standards applicable in a database?
29. Cardinality of a table T1 is 10 and of table T2 is 8 and the two relations are union compatible. If the cardinality of result T1 $\cup$ T2 is 13, then what is the cardinality of T1 $\cap$ T2? Justify your answer.
30. Cardinality of a table T1 is 10 and of table T2 is 8 and the two relations are union compatible
    a. What will be the maximum possible cardinality of T1 $\cup$ T2?
    b. What will be the minimum possible cardinality of T1 $\cap$ T2?
31. Consider the relations, City (city_name, state) and Hotel (name, address, city_name). Answer the following queries in relational algebra
    a. Find the names and address of hotels in Kochi.
    b. List the details of cities in Kerala state.
    c. List the names of the hotels in Thrissur.
    d. Find the names of different hotels.
    e. Find the names of hotels in Kozhikode or Munnar .
32. Using the instance of the EMPLOYEE relation shown in question 23, write the result of the following relational algebra expressions.
    a. $\sigma_{\text{Department="Sales"}}$ (EMPLOYEE).
    b. $\sigma_{\text{salary>20000} \wedge \text{Department="Sales"}}$ (EMPLOYEE).
    c. $\sigma_{\text{salary>20000} \vee \text{Department="Sales"}}$ (EMPLOYEE).
    d. $\pi_{\text{name, salary}}$ (EMPLOYEE).
    e. $\pi_{\text{name, salary}}(\sigma_{\text{Designation="Manager"}}(\text{EMPLOYEE}))$.
    f. $\pi_{\text{name, Department}}(\sigma_{\text{Designation="Clerk" ^ salary > 20000}}(\text{EMPLOYEE}))$.
33. Consider the instance of the BORROWER and DEPOSITOR relations shown in following figure which stores the details of customers in a Bank. Answer the following queries in relational algebra.
    a. Display the details of the customers who are either a depositor or a borrower.
    b. Display the name of customers who are both a depositor and a borrower.

c. Display the details of the customers who are depositors but not borrowers.

d. Display the name and amount of customer who is a borrower but not depositor.

BORROWER

Acc_No	Name	Amount
AC123	Albin	50000
AC103	Rasheeda	25000
AC106	Vishnu	25000
AC108	Aiswarya	30000

DEPOSITOR

Acc_No	Name	Amount
AC123	Albin	500
AC105	Shabana	25000
AC116	Vishnu	125000
AC108	Aiswarya	3000

34. Consider the instance of the CUSTOMER and BRANCH relations shown in the following table. Write the Cartesian Product of the two relations.

CUSTOMER

Acc_No	Name	Branch_ID	Amount
AC123	Albin	B1001	50000
AC103	Rasheeda	B1001	25000
AC106	Vishnu	B1001	25000
AC108	Aiswarya	B1077	30000

BRANCH

Branch_ID	Name
B1001	Kochi
B1002	Guruvayur
B1077	Idukki

9 Structured Query Language

Significant Learning Outcomes

After the completion of this chapter, the learner

- recognises the importance and features of Structured Query Language.
- explains the components of SQL.
- distinguishes the features of DDL, DML and DCL commands.
- identifies the characteristics of MySQL.
- lists different data types and their features.
- explains the effect of different constraints in SQL.
- performs operations using DDL commands like CREATE, ALTER, DROP.
- uses DML commands like SELECT, INSERT, UPDATE, DELETE for data manipulation.
- identifies various clauses associated with SQL commands and their purpose.
- uses operators for setting different conditions.
- lists different aggregate functions and explains their usage.
- constructs nested queries for information retrieval.

In the last chapter, we discussed the Relational Database Management System (RDBMS). We know that relational database is a set of related data stored in tables called relations. We also have a basic idea about relational algebra, which deals with various operations performed on relations. Now, we need more clarity on these operations which include creating a table, inserting data into a table, manipulating the data stored in a table and deleting data from a table, modifying the structure of a table, removing a table, etc. on a relational database. This chapter introduces a language called Structured Query Language (SQL) for these operations. Most of the relational database management systems like MySQL, Oracle, Sybase, Informix, Postgres, SQL Server and MS Access use SQL as standard database language. We use one of the most popular open source RDBMS, like MySQL, to implement Structured Query Language.

9.1 Structured Query Language

Structured Query Language (SQL) is a language designed for managing data in relational database management system (RDBMS). SQL provides an easy and efficient

way to interact with relational databases. There are numerous versions of SQL. The original version was developed in the 1970's by Donald D. Chamberlin and Raymond F. Boyce at IBM's San Jose Research Laboratory (now the Almanden Research Centre). This language was originally called Structured English Query Language (Sequel) and later its name was changed to SQL. In 1986, American National Standard Institute (ANSI) published an SQL standard.

As we know a relational database system is a structured collection of tables (relations) and the data is stored in these tables. Tables are uniquely identified by their names and are comprised of columns and rows. A column (field) in a table represents a particular type of information. In a table, each row represents a collection of related data. We know that rows in a table are known as tuples (or records) and columns are known as attributes.

Let us do

Examine the following table, named "Student" (refer to Table 9.1) and answer the questions given below for recollecting the basic terminologies related to database.

Adm_no	Name	Gender	Date_Birth	Income	Course
1001	Alok	M	2/10/1998	24000	Science
1002	Nike	M	26/11/1998	35000	Science
1003	Bharath	M	1/1/1999	45000	Commerce
1004	Virat	M	5/12/1998	22000	Science
1005	Meera	F	15/8/1998		Science
1006	Divakar	M	21/2/1998		Humanities

Table 9.1: Student table

i. The cardinality of the table is _____.

ii. The degree of the table is _____.

iii. List out the different tuples in the table.

iv. List out the different attributes in the table.

v. What are the values in the domain of the attribute 'Course'?

SQL is a powerful tool for implementing RDBMS. It provides facilities to create a table, insert data into a table, retrieve information from a table, modify data in the table, delete the existing data from a table, modify the structure of a table, remove a table from a database, etc.

9.1.1 Features of SQL

Structured Query Language is an ANSI/ISO standard language for writing database queries. A query is a request to a database. It can perform all the relational operations mentioned earlier. SQL is effective in framing queries because of the following features:

- SQL is a relational database language, not a programming language like C, C++.
- It is simple, flexible and powerful.
- It provides commands to create and modify tables, insert data into tables, manipulate data in the tables etc.
- It gives guidelines to major popular RDBMS like Oracle, SQL Server, MySQL, MS Access, Sybase, Informix and Postgres to perform database operations.
- SQL is a non-procedural language since it describes what data to retrieve, delete, or insert, rather than how to perform the operation.
- As part of ensuring data security, SQL provides facility to add or remove different types of access permissions to users on databases or tables.
- It provides the concept of views (This concept will be discussed later in this chapter).

9.1.2 Components of SQL

SQL has three components, namely Data Definition Language (DDL), Data Manipulation language (DML) and Data Control Language (DCL). Let us discuss these components and their roles in developing RDBMS.

Data Definition Language

Consider Table 9.1 (Student table). How can we create such a table? Is it possible to add a new column to this table? How can we remove a table from the database? The Data Definition Language (DDL) will give solutions to all these questions.

DDL is a component of SQL that provides commands to deal with the schema (structure) definition of the RDBMS. The ***DDL commands*** are used to create, modify and remove the database objects such as tables, views and keys. The common DDL commands are `CREATE`, `ALTER`, and `DROP`.

Data Manipulation Language

In Table 9.1, we can see several tuples (or rows or records). How are these tuples inserted into the table? Suppose the monthly family income of a particular student is to be modified. Is it possible? How can we delete the record of a student from

the table? The Data Manipulation Language (DML) provides commands for these types of manipulations.

DML is a component of SQL that enhances efficient user interaction with the database system by providing a set of commands. ***DML*** permits users to insert data into tables, retrieve existing data, delete data from tables and modify the stored data. The common DML commands are `SELECT`, `INSERT`, `UPDATE` and `DELETE`.

Data Control Language

Data Control Language (DCL) is used to control access to the database, which is very essential to a database system with respect to security concerns. ***DCL*** includes commands that control a database, including administering privileges and committing data. The commands `GRANT` and `REVOKE` are used as a part of DCL.

`GRANT` : Allows access privileges to the users to the database.

`REVOKE` : Withdraws user's access privileges given by using `GRANT` command.

Know your progress

1. SQL stands for _____.
2. Which are the three components of SQL?
3. SQL can be used to:
   a. create database structures only. b. query database data only.
   c. modify database data only. d. All of these can be done by SQL.
4. SQL is:
   a. a programming language. b. an operating system.
   c. a data sublanguage. d. a DBMS.
5. Which of the following is not an RDBMS package?
   a. ORACLE b. SQL SERVER c. MySQL d. HTML

9.2 Working on MySQL

The American National Standards Institute (ANSI) in 1986, and the International Organization for Standardization (ISO) in 1987, standardised SQL. Since 1986, the SQL standard has been evolving to include a larger set of features. The standard has been revised several times and several versions exist. SQL:2011 is the seventh revision of the ISO and ANSI standard for the SQL database query language. It was formally adopted in December 2011. Despite the existence of such standards, the different database software packages provide their own versions of the standard

ANSI SQL. Therefore, most SQL codes are not completely portable among different database software without adjustments. In this chapter, SQL will be discussed using open source database software MySQL.

MySQL is a free, fast, easy-to-use RDBMS, used for many applications. It is developed, marketed, and supported by MySQL AB, which is a Swedish company. MySQL is becoming very popular for many reasons:

- MySQL is released under an open-source license. So it is customizable.
- It provides high security to the database.
- It is portable as it works on many operating systems and with many languages including PHP, PERL, C, C++, JAVA, etc.
- MySQL works rapidly and effectively even with large volume of data.
- It is highly compatible with PHP, one of the popular languages for web development.

MySQL was developed by Michael "Monty" Widenius and David Axmark in 1995. It was originally owned by a Swedish company called MySQL AB, and was bought over by Sun Microsystems in 2008. Sun Microsystems was acquired by Oracle in 2010.

MySQL is often deployed in a Linux-Apache-MySQL-PHP (LAMP), Windows-Apache-MySQL-PHP (WAMP), or Mac-Apache-MySQL-PHP (MAMP) environment. All components in LAMP are free and open-source, inclusive of the Operating System. The official site for MySQL is www.mysql.com. The reference for MySQL is the "MySQL Reference Manual", available at http://dev.mysql.com/doc

9.2.1 Opening MySQL

We can work on MySQL by giving commands at the `mysql>` prompt. In Ubuntu Linux, we have to open the Terminal window using the following command sequence to get this prompt:

Applications → Accessories → Terminal

In the Terminal window we give the following command to start MySQL:

```
mysql -u root -p
```

MySQL in Windows OS can be opened by proceeding as follows:
Start → Programs → MySQL → MySQL Server (version number) → MySQL command line client

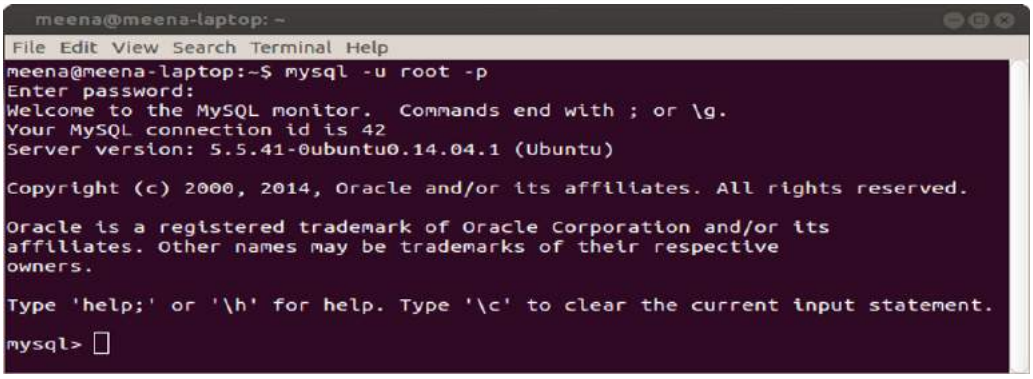


Fig 9.1: MySQL prompt at Ubuntu Linux Terminal window

When we open MySQL, it may ask for the password for verification. (Here we should use the same password that we entered during the installation process). After the password verification, we will get the prompt of MySQL as shown in Figure 9.1.

SQL is not case sensitive. That is, commands can be given in the upper or the lower case or even in a mix. But hereafter, we will use some styles to distinguish SQL commands and keywords from other texts. Commands and keywords will be specified in the upper case letters, whereas user-defined words such as table name, column name etc. will be in the lower case. Commands and outputs (or responses) can be stored in a text file after using the command **tee**. For example, `tee E:\outputs.txt` will create a file outputs.txt in E: drive to store whatever appears in the screen after the execution of this command. In this chapter the outputs stored in this file will be presented as figures.

The prompt gives us the message that MySQL is ready to accept any query from the user. Now we can input our queries at this prompt.

To exit from MySQL, give the command **QUIT** or **EXIT** at the prompt as:

```
mysql> EXIT;
```

9.2.2 Creating a database

We need to create a database before we work on the data. The database is the container in which we store the tables. To create a database in MySQL, we use the **CREATE DATABASE** command. The syntax is as follows:

```
CREATE DATABASE <database_name>;
```

While creating a database, the following points are to be remembered:

- The `<database_name>` in the syntax indicates the name of the database that we want to create. It is recommended that the database name should be as meaningful and descriptive as possible.

- The <database_name> should be unique. We cannot have two databases with the same name in a MySQL database server.

Let us start our database operations with the creation of a new database called "***school***". Figure 9.2 shows the screen shot after the execution of the command. We can see the MySQL command prompt and the command in the first line. The second line is the message returned by MySQL as a response to the command being executed. From the message, it is clear that a new database with the name *school* has been created successfully. (*Note that here onwards we will avoid such screen shots as figures, instead the command required for the specified operation will be presented in a separate font.*)

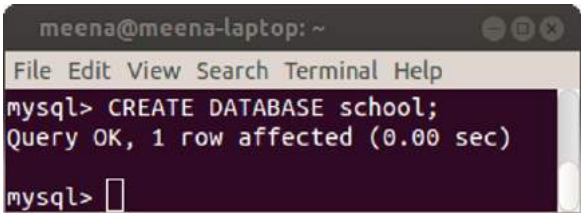


Fig. 9.2: MySQL window after the execution of a command

9.2.3 Opening database

To perform operations on a database, we have to open it explicitly. When we open a database, it becomes the active database in the MySQL server. MySQL gives a command **USE** to open a database. The syntax is:

```
USE <database_name>;
```

Let us open the data base school using the command as follows:

```
USE school;
```

The response of this command after the execution is given below:

```
Database changed
```

Now the database named *school* is the active database in our system. That means, the different DDL, DML and DCL commands we execute hereafter will be related to the database *school*. We can check the existence of a database. The SHOW DATABASE command is used to check whether a database exists or not. It will list the entire databases in our system. The syntax is:

```
SHOW DATABASES;
```

The output of this command is shown in Figure 9.3.

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| school             |
| test               |
+--------------------+
4 rows in set (0.00 sec)
```

Fig. 9.3: Output of SHOW DATABSES *command*

9.2.4 Data types in SQL

Data type defines the type of value that may be entered in the column of a table. Data types ensure

the correctness of the data, if we use them in a meaningful way. Care should be taken to assign correct data types for columns during the designing of the database. For example, if the numeric value 2 is designated as a text data type, such as a string, then it cannot be used in a mathematical operation; whereas the same number stored in an integer column can be used mathematically. So, let us understand the concept of SQL data types like, the type of data they represent, range of values supported by each of them, etc. Data types differ in different versions of SQL. Here, we look at the different data types available in MySQL.

MySQL data types are classified into three. They are numeric data type, string (text) data type, and date and time data type. All numerical values like 7, 100.234, -456, 0, etc. can be represented by any of the numeric data types. The data "Aleena" (name of a student), "Kerala" (name of a state), 'F' (specification of a gender), etc. are string type by nature. Data like '01-01-2020', '23:34:3' can be represented by Date and Time data types.

a. Numeric Data types

Numeric data type values can be used like any normal number. They can be added, subtracted, multiplied and divided. The most commonly used numeric data types in MySQL are **INT** or **INTEGER** and **DEC** or **DECIMAL**.

(i) INT or INTEGER

As we know, integers are whole numbers without a fractional part. They can be positive, zero or negative. An integer value can be represented in MySQL by `INT` or `INTEGER` data type. The data items like 69, 0, -112 belong to `INT` data type.

(ii) DEC or DECIMAL

Numbers with fractional parts can be represented by `DEC` or `DECIMAL` data type. The standard form of this type is `DECIMAL(size,D)` or `DEC(size,D)`. The parameter `size` indicates the total number of digits the value contains including decimal part. The parameter **D** represents the number of digits after the decimal point. For example, the type specification `DEC(5,2)` or `DECIMAL(5,2)` denotes that 5 is the precision and 2 is the scale. The column with this specification is able to store any value having a maximum of five digits, out of which two are after the decimal point. That is, the range of values will be from -999.99 to 999.99.

Table 9.2 shows an overview of numeric data types in MySQL. Remember that the values are version dependent.

Data types	Usage	Signed	Unsigned	Storage in Bytes
TINY INT	Very small integer values	-128 to 127	0 to 255	1
SMALL INT	A small integer	-32768 to 32767	0 to 65535	2
MEDIUM INT	A medium-sized integer value	-8388608 to 8388607	0 to 16777215	3
INT	Normal sized integer value	2147483648 to 2147483647	0 to 4294967295	4
BIG INT	Large integer value	Value up to 19 digits	Value up to 264	8
FLOAT(M,D)	Floating point numbers	Decimal precision can go to 24 places		4
DOUBLE(M,D)	A double precision floating-point number	Decimal precision can go to 53 places		8
DECIMAL(M,D)	Store exact precision values	A Decimal type can store a Maximum of 65 Digits, with 30 digits after decimal point.		8

Table 9.2: Numeric data types of MySQL and their characteristics

b. String (Text) data types

String is a group of characters. The most commonly used string data types in MySQL are **CHARACTER** or **CHAR** and **VARCHAR**.

(i) CHAR or CHARACTER

Character includes letters, digits, special symbols etc. The CHAR is a fixed length character data type. The syntax of this data type is CHAR(x), where x is the maximum number of characters that constitutes the data. The value of x can be between 0 and 255. CHAR is mainly used when the data in a column are of the same fixed length and small in size. For example, if we want to store data like 'M' for male and 'F' for female, in the column *Gender* of a table, it is better to declare that column as of type CHAR. It always uses the specified amount of space even though the data

need not require that much. If the number of characters in the data is less than the declared size of the column, the remaining character positions in the string will be filled with white spaces (spacebar character). But when we retrieve this value from the table, all trailing spaces are removed. Note that if the size of a column of type CHAR is 1, it is not necessary to mention the size, because the default size of CHAR type is 1.

(ii) VARCHAR(size)

VARCHAR represents variable length strings. It is similar to CHAR, but the space allocated for the data depends only on the actual size of the string, not on the declared size of the column. For example, if we want to store data in the column *Name* of a table, it is better to declare that column as of type VARCHAR, because the data in the column may contain different number of characters. The length of the string can vary from 0 to 65535 characters (MySQL version dependent). The VARCHAR type saves memory space since VARCHAR type did not append spaces with the values when they are stored. The data like name of people, addresses etc. are examples of this data type.

C. Date and Time data types

MySQL has data types for storing dates and times. The data type used to store date type value is **DATE** and to store time value is **TIME**.

(i) DATE

The DATE data type is used to store dates. MySQL represents date values in YYYY-MM-DD format. The supported range is from 1000-01-01 to 9999-12-31. Dates are displayed in MySQL in one format, but we can use various date formats in our SQL statements. The YYYY-MM-DD is the standard format. But we can use any punctuation character between the date parts. For example, '2011-01-24', '2011/01/25', '20110126' are valid date combinations in MySQL. We can insert date type values into a column of DATE data type in any of the above formats. Although MySQL tries to interpret values in several formats, date parts must always be given in year-month-day order (for example, '98-09-04').

(ii) TIME

The TIME data type is used to specify a column to store time values in MySQL. It shows values in the standard HH:MM:SS format. The TIME data type can be used to store a specific point in time (like 10 hours 05 minutes 25 seconds) as well as an interval of time between two points in time (like the time between now and the weekend) that may sometimes be larger than 23 hours. When manually entering a time into MySQL it is highly recommended that you use the exact format HH:MM:SS.

Let us do

Now look at Table 9.3 and fill the **Data type** column with suitable MySQL data type for the given data.

Value	Data type
325.678	
'A'	
'Computer'	
'2016-01-01'	
450	
22:32:45	
456787	

Table 9.3: Data type of values

Know your progress

1. SQL stands for _____.
2. _____ command is used to make a database active.
3. How can we see the names of databases in the system?
4. What is the difference between `CHAR` and `VARCHAR` data types?
5. Which is the format for storing date type data in MySQL?
6. Can we store the number 234 in a column declared with `CHAR(5)` data type?

9.3 SQL commands

SQL provides commands to perform different operations on database. As we mentioned earlier, the commands are classified as DDL commands, DML commands and DCL commands. Here, we will discuss the most commonly used DDL commands and DML commands. DDL commands are used to perform operations associated with the structure of database. The operations include creation of tables, modification in the structure of tables and removal of tables. DML commands are associated with the operations on the content of tables. These operations include insertion of records, retrieval of records, modification or updation of records and deletion of records.

These commands will be introduced in such a way that we can create a table for organizing data of a particular entity, retrieve required information and remove those we do not want to keep further.

9.4 Creating tables

Tables are the central and the most important objects in any relational database. The primary purpose of any database is to hold data that are stored in tables. Now

let us again consider the table *student* given in Table 9.1. How can we create a table with a set of columns? The DDL command **CREATE TABLE** is used to define a table by specifying the name of the table and giving the column definitions consisting of name of the column, data type and size, and constraints if any, etc. Remember that each table must have at least one column. The syntax of CREATE TABLE command is:

```
CREATE TABLE <table_name>
(<column_name> <data_type> [<constraint>]
[, <column_name> <data_type> [<constraint>,]
  ..............................
  ..............................    );
```

Here, the <table_name> represents the name of the table that we want to create; <column_name> represents the name of a column in the table; <data_type> represents the type of data in a column of the table; and <constraint> specifies the rules that we can set on the values of a column. All the columns are defined within a pair of parentheses, and are separated by commas. We can define all the columns even in a single line.

9.4.1 Rules for naming tables and columns

While naming the tables and columns, certain points are to be remembered and they are listed below.

- The name may contain letters (A - Z, a - z), digits (0 - 9), under score (_) and dollar ($) symbol.
- The name must contain at least one character. (Names with only digits are invalid).
- The name must not contain white spaces, special symbols.
- The name must not be an SQL keyword.
- The name should not duplicate with the names of other tables in the same data base and with other columns in the same table.

In some MySQL versions, the table name can be a quoted identifier. The identifier quote character is the backtick ("`"). If we use table name as quoted, then we can include any special symbols in the name of the table.

Now, let us create a table *student* to store the details of a group of higher secondary students in a school. The fields of the table and their descriptions are given in Table 9.4.

Sl. No.	Attributes	Description
1	Admission number	Integer value
2	Name	String of 20 characters long
3	Gender	A single character
4	Date of birth	Date type
5	Course	String of 15 characters long
6	Family income	Integer value

Table 9.4: Attributes of student table

The SQL statement or query to create a table in MySQL to incorporate the details shown in Table 9.4 is given in Query 9.1.

Query 9.1

```
CREATE TABLE student
(adm_no INT,
name VARCHAR(20),
gender CHAR,
dob DATE,
course VARCHAR(15),
f_income INT);
```

Here, the `CREATE TABLE` statement creates a table named student with five columns `adm_no`, `name`, `gender`, `dob`, `course` and `f_income`. The columns `adm_no` can contain integer values, `name` can contain strings up to a maximum of 20 characters, `gender` can store a single character indicating whether a student is a boy or a girl, `dob` can store date of birth, `course` can contain the group in which he/she is studying and `f_income` can hold the monthly income of the family. The columns `name` and `course` will spare only the actual space required by the respective string data.

After creating a table, when we insert records in that table, some kind of restriction may need to be enforced in certain columns. Restrictions may be in the form of denying empty values in some columns and refusing duplicate values in some other columns. But it should be addressed during the table creation itself. MySQL provides some keywords called constraints, for this purpose.

9.4.2 Constraints

Constraints are the rules enforced on data that are entered into the column of a table. When we create a table, we can apply constraints on the values that can be entered into its fields. If this is specified in the column definition, SQL will not

accept any values that violate the criteria concerned. This ensures the accuracy and reliability of the data in the database. The constraints ensure database integrity and hence they are often called data base integrity constraints. Constraints could be column level or table level.

a. Column Constraints

Column constraints are applied only to individual columns. They are written immediately after the data type of the column. The following are column constraints:

i. NOT NULL

This constraint specifies that a column can never have NULL values. NULL is a keyword in SQL that represents an empty value. It is important to remember that NULL does not equate to a blank or a zero; it is something else entirely. Though a blank is equal to another blank and a zero is equal to another zero, a NULL is never equal to anything, not even another NULL. Two NULL values cannot be added, subtracted or compared.

ii. AUTO_INCREMENT

MySQL uses the AUTO_INCREMENT keyword to perform an auto-increment feature. If no value is specified for the column with AUTO_INCREMENT constraint, then MySQL will assign serial numbers automatically and insert the newly assigned value in the corresponding column of the new record. By default, the starting value for AUTO_INCREMENT is 1, and it will be incremented by 1 for each new record. This special behavior also occurs if we explicitly assign the value NULL to the column. The AUTO_INCREMENT feature makes it easy to assign a unique ID to each new row, because MySQL generates the values for us. The auto increment column must be defined as the primary key of the table. Only one AUTO_INCREMENT column per table is allowed.

iii. UNIQUE

It ensures that no two rows have the same value in the column specified with this constraint.

iv. PRIMARY KEY

This constraint declares a column as the primary key of the table. This constraint is similar to UNIQUE constraint except that it can be applied only to one column or a combination of columns. The primary keys cannot contain NULL values. In other words, it can be considered as a combination of UNIQUE and NOT NULL constraints. A PRIMARY KEY constraint is used to enforce a rule that a column should contain only unique, non-NULL data.

v. DEFAULT

Using this constraint, a default value can be set for a column, in case the user does not provide a value for that column of a record.

Let us apply some of these constraints in the student table and modify Query 9.1 as given in Query 9.2.

Query 9.2

```
CREATE TABLE student
(adm_no INT PRIMARY KEY AUTO_INCREMENT,
name VARCHAR(20) NOT NULL,
gender CHAR DEFAULT 'M',
dob DATE,
course VARCHAR(15)
f_income INT);
```

In Query 9.2, the constraints `PRIMARY KEY` and `AUTO_INCREMENT` are applied to the column `adm_no`. So, this column will not allow duplicate values during data entry. If we donot specify a value for this column, MySQL will generate a data automatically. The constraint `NOT NULL` applied to the column `name` does not allow to leave the column with a `null` value. That is, data is a must in this column. Similarly, if we do not give a value to the column `gender`, `'M'` will be stored as the default value.

b. Table constraints

Table constraints are similar to column constraints; the main difference is that table constraints can be used not only on individual columns, but also on a group of columns. When a constraint is to be applied on a group of columns of a table, it is called table constraint. The table constraint appears at the end of the table definition. For example, Query 9.3 creates a table named ***stock***. The constraint `UNIQUE` is applied to the combination of `icode` and `iname`.

Query 9.3

```
CREATE TABLE stock
(icode CHAR(2) PRIMARY KEY AUTO_INCREMENT,
iname VARCHAR(30) NOT NULL,
dt_purchase DATE,
rate DECIMAL(10,2),
qty INT,
UNIQUE (icode, iname));
```

In Query 9.3, the constraint `UNIQUE` is applied to the combination of values of columns `icode` and `iname`. It enforces a situation that no two rows can have the same values for the columns `icode` and `iname` when taken together.

9.4.3 Viewing the structure of a table

We have created two tables, *student* and *stock*. How do we know the structure of the table after its creation? The **DESCRIBE** command is used to display the structure definitions of a table. The syntax is:

```
DESCRIBE <table_name>;
```

OR

```
DESC <table_name>;
```

The structure of the table student can be viewed using the command:

```
DESC student;
```

Figure 9.4 shows the output of this command.

```
+----------+-------------+------+-----+---------+----------------+
| Field    | Type        | Null | Key | Default | Extra          |
+----------+-------------+------+-----+---------+----------------+
| adm_no   | int(11)     | NO   | PRI | NULL    | auto_increment |
| name     | varchar(20) | NO   |     | NULL    |                |
| gender   | char(1)     | YES  |     | M       |                |
| dob      | date        | YES  |     | NULL    |                |
| course   | varchar(15) | YES  |     | NULL    |                |
| f_income | int(11)     | YES  |     | NULL    |                |
+----------+-------------+------+-----+---------+----------------+
6 rows in set (0.02 sec)
```

Fig. 9.4: Structure of student table

Note that in Query 9.2, we did not mention the size for the columns `adm_no` and `f_income` while creating the table. But MySQL takes the default size 11 as the size of these two columns. The column `adm_no` is declared with `PRIMARY KEY` and `AUTO_INCREMENT`. So this column will not allow duplicate values and null values. If we do not specify a value for this column, a new value will be generated by adding 1 to the value of the respective column of the previous record. In the absence of a value for this column for the first record, MySQL gives 1 as the value. We can also see that a default value 'M' is set for the column gender. Figure 9.4 illustrates these aspects. There is a command `SHOW TABLES`, which shows the tables created in the current database as given in Figure 9.5.

```
+------------------+
| Tables_in_school |
+------------------+
| student          |
+------------------+
1 row in set (0.00 sec)
```

Fig. 9.5: Output of `SHOW TABLES`

Let us do

Write the structure of the table ***stock***, referring to Figure 9.4 and Query 9.3. Verify your answer in the lab with the command, `DESC stock;`

Know your progress

1. Which of the following commands is used to display the structure of a table?

   a. `LIST` b. `SHOW` c. `DESCRIBE` d. `STRUCT`

2. Write the syntax of `CREATE TABLE` command.
3. Name the different column constraints.
4. What is the difference between primary key constraint and unique constraint?
5. What are the features of `AUTO_INCREMENT` constraint?
6. Write down the rules for naming a table.
7. How many columns in a table can be specified as primary key of the table?

9.5 Inserting data into tables

We have created a database and its tables. Now we need to put some records into these tables. We have the details of six students in Table 9.1. Let us discuss how these records can be inserted into the table.

The DML command **`INSERT INTO`** is used to insert tuples into tables. The syntax is:

```
INSERT INTO <table_name> [<column1>,<column2>,...,<columnN>]
VALUES(<value1>,<value2>,...,<valueN>);
```

Here `<table_name>` is the name of the table into which the tuples are to be inserted; `<column1>`, `<column2>`, …., `<columnN>` indicate the name of columns in the table into which values are to be inserted; `<value1>`,`<value2>`, …., `<valueN>` are the values that are inserted into the columns specified in the `<column_list>`.

For example, let us insert a new record into the table *student* with data *1001, 'Alok', 'M', 1998/10/2, 'Science', 24000* into the columns `adm_no`, `name`, `gender`, `dob`, `course` and `f_income`, respectively. Query 9.4 makes it possible.

Query 9.4

```
INSERT INTO student
VALUES (1001,'Alok','M','1998/10/2', 'Science',
        24000);
```

The response of the system will be:

```
Query OK, 1 row affected (0.05 sec).
```

The INSERT statement adds a new row to the table *student* giving a value for every column in the table. While inserting a row, if we provide values for all the columns of the table, we need not specify the name of column(s) in the query. But we need to make sure that the order of the values is in accordance with the order of the columns in the table. Now let us insert another row with some modifications as shown in Query 9.5.

Query 9.5

```
INSERT INTO student (name, dob, course, f_income)
VALUES  ('Nike','1998/11/26','Science',35000);
```

The response for this statement will be:

```
Query OK, 1 row affected (0.01 sec)
```

In Query 9.5, admission number and gender are not provided. Being a record after the one with admission number 1001, the admission number of this student will be 1002. The gender will be set with the default value 'M'.

While inserting data into tables, the following points are to be taken care of:

- While adding a new row, we should ensure the data type of the value and the column matches.
- We follow the integrity constraints, if any, defined for the table.
- CHAR or VARCHAR type data should be enclosed in single quotes or double quotes.
- Column values for DATE type columns are to be provided within single quotes. The string will internally be converted into DATE data type.
- Null values are specified as NULL (or null) without quotes.
- If no data is available for all columns, then the column list must be included, following the table name.

MySQL allows inserting several rows into a table with a single INSERT command by specifying multiple value lists. The general format is as follows:

```
INSERT INTO <table-name> VALUES(...), (...), ... ;
```

Let us insert two more records given in Table 9.1 using Query 9.6.

Query 9.6

```
INSERT INTO student (name, dob, course, f_income)
VALUES('Bharath','1999/01/01','Commerce',45000),
      ('Virat','1998/12/05','Science',22000);
```

The response of the system will be:

```
Query OK, 2 rows affected (0.02 sec)
Records: 2 Duplicates: 0  Warnings: 0
```

We can observe that the two records are given within separate pairs of parentheses. The response indicates that the two records are inserted successfully.

Suppose we do not have the monthly income data of a student. How can we insert the record? Query 9.7 illustrates the solution.

Query 9.7

```
INSERT INTO student(name, dob, gender, course)
VALUES('Meera','1998/08/15','F','Science');
```

As a response to this query, the value for `adm_no` will be generated by the system, but the value for `f_income` column will be kept as `NULL`. The order of columns is also changed in this query. The absence of values in a row can be managed in another way also as shown in Query 9.8.

Query 9.8

```
INSERT INTO student(name, dob, gender, course,
                                        f_income)
VALUES('Divakar','1998/02/21','Science',NULL);
```

Note that, in the `VALUES` clause, `NULL` is given for `f_income`.

Let us do

Let us insert some more records in table *student*. Write queries to store the details of students shown in Table 9.5.

adm_no	name	gender	dob	course	f_income
1025	Kaushi	M	1998/10/2	Commerce	17000
1026	Niveditha	F	1999/03/04	Humanities	52000
1027	Sreekumar	M	1998/06/06	Science	15000
1057	Chaithanya	F	1999/06/03	Science	

Table 9.5: More records for Student table

Know your progress

1. Which of the following is used to add a row into a table?
   a. ADD b. CREATE c. INSERT d. MAKE
2. Which statement is used to insert new data into a table?
   a. ADD RECORD b. INSERT RECORD
   c. INSERT INTO d. INSERT ROW
3. Write the essential keywords used along with `INSERT` command.

Let us do

Write SQL statements to insert some records in the table stock created using Query 9.3. While giving values to the columns, utilise the facility of AUTO_INCREMENT and UNIQUE constraints.

9.6 Retrieving information from tables

We have created a database school and a table student, and then inserted ten records into it. Now let us learn how information is retrieved from the data stored in tables. It is a kind of data manipulation operation and SQL provides the command **SELECT** for this purpose. It is used to retrieve information from specified columns in a table. The SELECT command has several forms of its own. The simplest form of SELECT command is:

```
SELECT <column_name>[,<column_name>,<column_name>, ...]
FROM <table_name>;
```

Here <column_name> indicates the column from which data is retrieved and <table_name> denotes the name of the table from which the information is retrieved. The name of the table is given with the keyword **FROM**, which is an essential clause with SELECT command. The SELECT command will display the data in columns, in the order in which they appear along with the SELECT command.

Now let us illustrate the execution of SELECT command through various queries. On executing Query 9.9 we get the name and course of students in the table *student* as shown in Figure 9.6.

Query 9.9

```
SELECT name, course
FROM student;
```

```
+------------+------------+
| name       | course     |
+------------+------------+
| Alok       | Science    |
| Nike       | Science    |
| Bharath    | Commerce   |
| Virat      | Science    |
| Meera      | Science    |
| Divakar    | Science    |
| Kaushi     | Commerce   |
| Niveditha  | Humanities |
| Sreekumar  | Science    |
| Chaithanya | Science    |
+------------+------------+
10 rows in set (0.00 sec)
```

Fig. 9.6: Output of Query 9.9

If we want to display the entire column values of a table, we need not give a complete list of columns of the relation. Instead, an asterisk (*) symbol can be used to substitute the complete list of columns as shown in Query 9.10. The output is shown in Figure 9.7.

Query 9.10

```
SELECT * FROM student;
```

```
+--------+------------+--------+------------+------------+----------+
| adm_no | name       | gender | dob        | course     | f_income |
+--------+------------+--------+------------+------------+----------+
|   1001 | Alok       | M      | 1998-10-02 | Science    |    24000 |
|   1002 | Nike       | M      | 1998-11-26 | Science    |    35000 |
|   1003 | Bharath    | M      | 1999-01-01 | Commerce   |    45000 |
|   1004 | Virat      | M      | 1998-12-05 | Science    |    22000 |
|   1005 | Meera      | F      | 1998-08-15 | Science    |     NULL |
|   1006 | Divakar    | M      | 1998-02-21 | Science    |     NULL |
|   1025 | Kaushi     | M      | 1998-10-02 | Commerce   |    17000 |
|   1026 | Niveditha  | F      | 1999-03-04 | Humanities |    52000 |
|   1027 | Sreekumar  | M      | 1998-06-06 | Science    |    15000 |
|   1057 | Chaithanya | F      | 1999-06-03 | Science    |     NULL |
+--------+------------+--------+------------+------------+----------+
10 rows in set (0.00 sec)
```

Fig. 9.7: Entire content of student table

We can see NULL values in column f_income of some rows in Figure 9.7. It is due to the missing of values in the insertion of those rows (refer Queries 9.7, 9.8 and Table 9.1, and 9.5).

9.6.1 Eliminating duplicate values in columns using DISTINCT

Suppose we want to know the names of different courses in the table *student*. If we construct a query **SELECT course FROM student;** it will display the column course with all the ten values as shown in Figure 9.7. Data will be selected from all rows of the relation to display, even if the data appearing in the result get duplicated. This duplication can be eliminated using the keyword **DISTINCT** as given in Query 9.11. Observe the output given in Figure 9.8.

Query 9.11

```
SELECT DISTINCT course
FROM student;
```

In the output, there are no duplicate values. If the column used with DISTINCT keyword contains more than one NULL value, only one will be shown in the result.

If we give the keyword **ALL** in the place of DISTINCT, then the result will contain all the duplicate values in the column. That is, the output will be the same as that in the case when we neither specify DISTINCT nor ALL.

```
+------------+
| course     |
+------------+
| Science    |
| Commerce   |
| Humanities |
+------------+
3 rows in set (0.25 sec)
```

Fig. 9.8: Use of DISTINCT

9.6.2 Selecting specific rows using WHERE clause

In certain situations, we need to display only a subset of a table. For example, we may require the details of female students only, or the details of students whose family monthly income is below Rs. 25000/-. In these cases, there is a selection in

the retrieval of records. Obviously the selection is based on some condition(s). SQL enables us to impose some selection criteria for the retrieval of records with **WHERE** clause of SELECT command. The syntax of SELECT command with WHERE clause is:

```
SELECT <column_name>[,<column_name>,<column_name>, ...]
FROM <table_name>
WHERE <condition>;
```

When a WHERE clause is present, the SELECT command goes through the entire table one row at a time and examines whether the row satisfies the condition(s) or not. If a row satisfies the condition, that row is displayed in the output. The conditions are expressed with the help of relational operators and logical operators. MySQL provides a variety of such operators and they are listed in Table 9.6.

	Operator	Meaning/Result
Relational Operators	=	Equal to
	<> or !=	Not equal to
	>	Greater than
	<	Less than
	>=	Greater than or equal to
	<=	Less than or equal to
Logical operators	NOT	True when condition is false
	AND	True if both the conditions are true
	OR	True if either of the conditions is true

Table 9.6: Operators used for setting conditions

Let us write an SQL statement to display the details of female students in the table. Query 9.12 is the required statement which contains a WHERE clause and = operator for setting the condition and Figure 9.9 shows the output.

Query 9.12

```
SELECT * FROM student
WHERE gender='F';
```

```
+--------+------------+--------+------------+------------+----------+
| adm_no | name       | gender | dob        | course     | f_income |
+--------+------------+--------+------------+------------+----------+
|   1005 | Meera      | F      | 1998-08-15 | Science    |     NULL |
|   1026 | Niveditha  | F      | 1999-03-04 | Humanities |    52000 |
|   1057 | Chaithanya | F      | 1999-06-03 | Science    |     NULL |
+--------+------------+--------+------------+------------+----------+
3 rows in set (0.06 sec)
```

Fig. 9.9: List of female students - the output of Query 9.12

Suppose we want to see the name, course and monthly family income of only the Science group students whose income is below Rs. 25000. Here we have to constitute two conditions - one for checking the income and the other for checking the course. When these two conditions are matched while going through the records in the table, the values of their `name`, `course` and `f_income` columns will be retrieved for display. Query 9.13 is the required query and Figure 9.10 shows the output. Note that **AND** operator is used to combine the two conditions.

Query 9.13

```
SELECT name, course, f_income FROM student
WHERE course='Science' AND f_income<25000;
```

Write a query to display the name, course and monthly income of students studying in courses other than *Science* group (i.e., the students of *Commerce* and *Humanities* groups).

```
+-----------+---------+----------+
| name      | course  | f_income |
+-----------+---------+----------+
| Alok      | Science |    24000 |
| Virat     | Science |    22000 |
| Sreekumar | Science |    15000 |
+-----------+---------+----------+
3 rows in set (0.00 sec)
```

Fig. 9.10: Use of AND operator

While writing the above query, you might have used **OR** operator for setting the condition. The same information can be obtained by the use of **NOT** operator. Query 9.14 illustrates this and Figure 9.11 shows the corresponding output.

Query 9.14

```
SELECT name, course, f_income FROM student
WHERE NOT course='Science';
```

Identify some requirements with respect to the table ***stock*** which can be attained by setting different conditions, and construct queries using appropriate operators.

```
+-----------+------------+----------+
| name      | course     | f_income |
+-----------+------------+----------+
| Bharath   | Commerce   |    45000 |
| Kaushi    | Commerce   |    17000 |
| Niveditha | Humanities |    52000 |
+-----------+------------+----------+
3 rows in set (0.00 sec)
```

Fig. 9.11: Use of NOT operator

SQL has a set of special operators for setting conditions. These include **BETWEEN...AND**, **IN**, **LIKE** and **IS**. Let us discuss how these operators help us to constitute conditions.

a. Condition based on a range of values

A range of values can be given as condition. The SQL operator **BETWEEN...AND** is used to specify the range. Suppose we need a list of students whose monthly income falls in the range of Rs. 25000/- to Rs. 45000/-. We know that this can be obtained by the statement given in Query 9.15. Figure 9.12 shows the output.

Query 9.15

```
SELECT name, f_income FROM student
WHERE f_income>=25000 AND f_income<=45000;
```

The same output can be obtained by using the statement given in Query 9.16. In this statement the operator BETWEEN...AND is used to construct the condition. The output includes both the lower and upper values given in the range.

Query 9.16

```
SELECT name, f_income FROM student
WHERE f_income BETWEEN 25000 AND 45000;
```

From Query 9.16 and Figure 9.12, we can conclude that the BETWEEN...AND operator allows specifying a range of values which belong to either numeric or date and time type data.

```
+---------+----------+
| name    | f_income |
+---------+----------+
| Nike    |    35000 |
| Bharath |    45000 |
+---------+----------+
2 rows in set (0.00 sec)
```

Fig. 9.12: Use of BETWEEN...AND operator

b. Conditions based on a list of values

While setting conditions for the retrieval of records, a list of values can be provided. The values may be of any data type, but it should match with those of the column used in the condition. In such a case, the operator **IN** is used with the list. Suppose we want to retrieve the details of students in *Commerce* and *Humanities* groups. The statement in Query 9.17 can generate the details as shown in Figure 9.13.

Query 9.17

```
SELECT * FROM student
WHERE course='Commerce' OR course='Humanities';
```

```
+--------+----------+--------+------------+------------+----------+
| adm_no | name     | gender | dob        | course     | f_income |
+--------+----------+--------+------------+------------+----------+
|   1003 | Bharath  | M      | 1999-01-01 | Commerce   |    45000 |
|   1025 | Kaushi   | M      | 1998-10-02 | Commerce   |    17000 |
|   1026 | Niveditha| F      | 1999-03-04 | Humanities |    52000 |
+--------+----------+--------+------------+------------+----------+
3 rows in set (0.00 sec)
```

Fig. 9.13: Output of Query 9.17

The same output can be obtained by using IN operator in the condition as shown in Query 9.18.

Query 9.18

```
SELECT * FROM student
WHERE course IN('Commerce', 'Humanities');
```

As we see in Query 9.18, the IN operator checks whether the value in the specified column (here it is course) of a record matches any of the values in the given list. When matches are found while going through the records, they will be displayed. Since there are only three courses in the table, the result shown in Figure 9.13 can also be obtained using Query 9.19.

Query 9.19

```
SELECT * FROM student
WHERE course NOT IN('Science');
```

c. Conditions based on pattern matching

There may be the need of retrieving data based on some pattern matching. That is, string data having some similarities in characters may be the criterion for the retrieval of records. SQL provides a pattern matching operator **LIKE** for this purpose. Patterns are specified using two special wildcard characters % and _ (underscore), where % (percentage) matches a substring of characters and _ (underscore) matches a single character. Patterns are case sensitive; i.e.; uppercase characters do not match lower case characters. Consider the following examples:

- "Ab%" matches any string beginning with "Ab".
- "%cat%" matches any string containing "cat" as substring. For example, "education", "indication", "catering" etc.
- "_ _ _ _" matches any string of exactly four characters without any space in between them.
- "_ _ _ %" matches any string of at least 3 characters.

Query 9.20 illustrates the use of LIKE operator. It gives a list of names in the table that end with the substring 'ar'. Figure 9.14 shows the output of this query.

Query 9.20

```
SELECT name FROM student
WHERE name LIKE '%ar';
```

```
+-----------+
| name      |
+-----------+
| Divakar   |
| Sreekumar |
+-----------+
2 rows in set (0.00 sec)
```

Fig. 9.14: Use of pattern matching

The following query will display only 'Divakar' since there are two _ (underscore) characters for pattern matching.

```
SELECT name FROM student
WHERE name LIKE 'Div__ar';
```

d. Conditions based on NULL value search

We have seen that there may be NULL values in some fields of a record. We can retrieve such records with the help of **IS** operator. The condition will be true when the specified column contains a NULL value in the records. In the *student* table, the f_income column contains NULL value in three records (refer to Figure 9.7). Query 9.21 retrieves these records and Figure 9.15 shows the output.

```
+------------+---------+
| name       | course  |
+------------+---------+
| Meera      | Science |
| Divakar    | Science |
| Chaithanya | Science |
+------------+---------+
3 rows in set (0.00 sec)
```

Fig. 9.15: NULL *value checking*

Query 9.21

```
SELECT name, course FROM student
WHERE f_income IS NULL;
```

On the other hand, if we want to retrieve the records containing non-null values in the f_income column, the following statement can fulfill the task:

```
SELECT name, course FROM student
WHERE f_income IS NOT NULL;
```

Let us do

Identify some requirements with respect to the table ***stock*** which need the use of the special SQL operators we discussed, and construct queries to solve the problems.

Know your progress

1. Name the keyword used with SELECT command to avoid duplication in the values of a column.
2. Which is the essential clause for SELECT query?
3. Which of the following operators is used to check for NULL value in a column?

   a. IN b. LIKE c. IS d. NOT
4. The operator used for checking pattern matching is _____.
5. What is wrong with the following statement?

   ```
   SELECT * FROM emp WHERE grade = NULL;
   ```
6. The command that extracts records from a table is _____.

9.6.3 Sorting results using ORDER BY clause

As we can see in the results of the SELECT queries discussed so far, the records are always obtained in the order in which they appear in the table. Is it possible to display them in a specific order - ascending or descending, of some column values? Yes. The result of a query can be sorted in the ascending or descending order by making use of **ORDER BY** clause. The order is to be specified by using the keyword **ASC** (for ascending) or **DESC** (for descending) along with the column name that is used with ORDER BY clause. By default, the display will be in the ascending order. Remember that only the results that appear on the screen are sorted. The order of records in the table will be kept unaltered.

Query 9.22 will display the details of students in the alphabetical order of their names. Figure 9.16 shows the output.

Query 9.22

```
SELECT * FROM student ORDER BY name;
```

adm_no	name	gender	dob	course	f_income
1001	Alok	M	1998-10-02	Science	24000
1003	Bharath	M	1999-01-01	Commerce	45000
1057	Chaithanya	F	1999-06-03	Science	NULL
1006	Divakar	M	1998-02-21	Science	NULL
1025	Kaushi	M	1998-10-02	Commerce	17000
1005	Meera	F	1998-08-15	Science	NULL
1002	Nike	M	1998-11-26	Science	35000
1026	Niveditha	F	1999-03-04	Humanities	52000
1027	Sreekumar	M	1998-06-06	Science	15000
1004	Virat	M	1998-12-05	Science	22000

10 rows in set (0.00 sec)

Fig. 9.16: Details of students according to the alphabetical order of their names

Suppose we want to see the details according to their monthly family income. If we have to obtain the result from the highest income to the lowest, Query 9.23 can serve the purpose. Figure 9.17 shows the output.

Query 9.23

```
SELECT * FROM student
ORDER BY f_income DESC;
```

adm_no	name	gender	dob	course	f_income
1026	Niveditha	F	1999-03-04	Humanities	52000
1003	Bharath	M	1999-01-01	Commerce	45000
1002	Nike	M	1998-11-26	Science	35000
1001	Alok	M	1998-10-02	Science	24000
1004	Virat	M	1998-12-05	Science	22000
1025	Kaushi	M	1998-10-02	Commerce	17000
1027	Sreekumar	M	1998-06-06	Science	15000
1006	Divakar	M	1998-02-21	Science	NULL
1005	Meera	F	1998-08-15	Science	NULL
1057	Chaithanya	F	1999-06-03	Science	NULL

10 rows in set (0.00 sec)

Fig. 9.17: Age-based listing of students (younger to older)

As shown in Figure 9.17, if the column used with `ORDER BY` clause contains `NULL` values, they will be listed last.

Multiple sorting can be performed after selection with the help of `ORDER BY` clause. For example, if we need a course-based listing of students in the alphabetical order of their names, a statement as given in Query 9.24 can be used. Figure 9.18 shows its output.

Query 9.24

```
SELECT name, course FROM student
ORDER BY course, name;
```

Look at the output shown in Figure 9.18. First the values in the column `course` is arranged alphabetically and then the names are arranged under each course.

```
+------------+------------+
| name       | course     |
+------------+------------+
| Bharath    | Commerce   |
| Kaushi     | Commerce   |
| Niveditha  | Humanities |
| Alok       | Science    |
| Chaithanya | Science    |
| Divakar    | Science    |
| Meera      | Science    |
| Nike       | Science    |
| Sreekumar  | Science    |
| Virat      | Science    |
+------------+------------+
10 rows in set (0.00 sec)
```

Fig. 9.18: Multiple sorting

Earlier we have used the `WHERE` clause for condition based retrieval of records. The retrieved records can be displayed in a particular order using `ORDER BY` clause. Suppose we want to display the name and family income of *Science* group students according to the descending order of income. The two clauses can be combined to generate this output as shown in Query 9.25. The output is shown in Figure 9.19.

Query 9.25

```
SELECT name, course, f_income FROM student
WHERE course='Science'
ORDER BY f_income DESC;
```

As shown in Figure 9.19, the records with `NULL` values in the sorted column will appear at last as they appear in the table. Another important aspect is that, if an `ORDER BY` clause is used in a `SELECT` statement, it should appear only after the `WHERE` clause, if any. Because, the records should be selected first, then only they are given for reordering.

```
+------------+---------+----------+
| name       | course  | f_income |
+------------+---------+----------+
| Nike       | Science |    35000 |
| Alok       | Science |    24000 |
| Virat      | Science |    22000 |
| Sreekumar  | Science |    15000 |
| Meera      | Science |     NULL |
| Divakar    | Science |     NULL |
| Chaithanya | Science |     NULL |
+------------+---------+----------+
7 rows in set (0.00 sec)
```

Fig. 9.19: Condition based selection and sorting

Know your progress

1. What is the meaning of `ORDER BY` clause ?
2. Which keyword is used for sorting the data in descending order in MySQL?

   a. `DESC` b. `ASC` c. `SORT` d. `MODIFY`
3. In which order are the records displayed in the absence of `ORDER BY` clause.
4. In SQL, what is the default sort order of the `ORDER BY` clause?

Let us do

Identify some requirements with respect to the table ***stock*** which need the sorting of records based on different conditions, and construct queries to solve the problems.

9.6.4 Aggregate functions

MySQL provides a number of built-in functions that can be applied to all rows in a table or to a subset of the table specified by `WHERE` clause. These functions are called aggregate functions because they operate on aggregate of tuples (records). The result of an aggregate function is a single value.

Commonly used aggregate functions are given in Table 9.7.

Function	Return value
`SUM()`	Total of the values in the column specified as argument.
`AVG()`	Average of the values in the column specified as argument.
`MIN()`	Smallest value in the column specified as argument.
`MAX()`	Largest of the values in the column specified as argument.
`COUNT()`	Number of non `NULL` values in the column specified as argument.

Table 9.7: Some built-in functions of MySQL

Let us discuss these functions to generate useful information from the table. Query 9.26 gives the highest, lowest and average family monthly incomes of the students. Figure 9.20 shows the result.

Query 9.26

```
SELECT MAX(f_income), MIN(f_income), AVG(f_income)
FROM student;
```

```
+---------------+---------------+---------------+
| MAX(f_income) | MIN(f_income) | AVG(f_income) |
+---------------+---------------+---------------+
|         52000 |         15000 |    30000.0000 |
+---------------+---------------+---------------+
1 row in set (0.03 sec)
```

Fig. 9.20: Use of aggregate functions

While calculating these values, the `NULL` values in the argument column are not considered.

These functions can be applied to a subset of the table formed by some selection crierion as shown in Query 9.27. It gives the number of students in the *Science* group. Figure 9.21 shows the output.

Query 9.27

```
SELECT COUNT(*), COUNT(f_income)
FROM student
WHERE course='Science';
```

If we refer to Figure 9.19, we can see that the correct answer is 7. Then, why is there a mismatch between the two column values in Figure 9.21? The first column is **COUNT(*)** and it gives the number of records having *Science* as the value in column course. Note that the * (asterisk) symbol stands for the collection of all the columns in the table. So, if there is at least one field in a record, that record will be taken into consideration for COUNT(*). But COUNT(f_income) in Query 9.27 counts only the non-NULL values in column f_income of the records which contains *Science* in column course. This is why the second column in Figure 9.21 contains only 4. Refer to Figure 9.17 and identify the rows that conatins NULL in column f_income.

```
+----------+-----------------+
| COUNT(*) | COUNT(f_income) |
+----------+-----------------+
|        7 |               4 |
+----------+-----------------+
1 row in set (0.03 sec)
```

Fig. 9.21: Use of COUNT() function

9.6.5 Grouping of records using GROUP BY clause

Sometimes, we need to divide the tuples in a table into different groups based on the values in a column. The rows of a table can be grouped together based on a common value using the **GROUP BY** clause. The attribute (column) specified in the **GROUP BY** clause is used to form groups. Tuples with the same value in the attribute specified in the GROUP BY clause are placed together in one group. Thus different groups are formed based on distinct values in the column. So this process can be considered as categorisation of records.

Using Query 9.27, number of students in *Science* group is obtained. Suppose we want to know the number of students in each group along with average family monthly income. Query 9.28 can be used. The output is shown in Figure 9.22.

Query 9.28

```
SELECT course, COUNT(*),AVG(f_income)
FROM student
GROUP BY course;
```

Here different groups are formed under different values ('*Commerce*', '*Humanities*' and '*Science*') in column course and the functions COUNT(*) and AVG(f_income) are applied to each group.

```
+------------+----------+---------------+
| course     | COUNT(*) | AVG(f_income) |
+------------+----------+---------------+
| Commerce   |        2 |    31000.0000 |
| Humanities |        1 |    52000.0000 |
| Science    |        7 |    24000.0000 |
+------------+----------+---------------+
3 rows in set (0.00 sec)
```

Fig. 9.22: Grouping of rows using GROUP BY clause

9.6.6 Applying conditions to form groups using HAVING clause

We have already learnt about the WHERE clause, which places conditions on individual rows. We can apply conditions to form groups with the help of **HAVING** clause.

This clause is used along with GROUP BY clause. The condition in HAVING clause is applied to form groups of records, not individual rows.

Query 9.29 applies a condition for forming groups. Here the groups will be formed only if the condition provided with HAVING clause is satisfied.

Query 9.29

```
SELECT course, COUNT(*) FROM student
GROUP BY course
HAVING COUNT(*) > 3;
```

```
+---------+----------+
| course  | COUNT(*) |
+---------+----------+
| Science |        7 |
+---------+----------+
1 row in set (0.00 sec)
```

Fig. 9.23: Condition based grouping

If we refer to Figures 9.22 and 9.23, we can find that only *Science* group has more than 3 students. Groups are not formed based on values '*Commerce*' and '*Humanities*', since the number of records with each of these values are 2 and 1, respectively. That is why details of these groups are not shown by Query 9.29.

Know your progress

1. List the aggregate functions of SQL.
2. How do COUNT(*) and COUNT(column_name) differ?
3. What is the difference between WHERE clause and HAVING clause?
4. The usage SUM(*) or MAX(*) is invalid. Why?
5. What will be the result of the following query?

```
SELECT COUNT(DISTINCT course) FROM student;
```

Let us do

Identify some queries with respect to the table ***stock*** which requires the utilisation of aggregate functions and ORDER BY clause. Write SQL statements to answer these queries.

9.7 Modifying data in tables

In certain situations, we need to change the values in the columns of a table. For example, following a revision in salary or wages, the family monthly income of a student may be changed. Similarly, the missing values (NULL) in monthly income of some students may need to be replaced by valid data at a later stage. This kind of changes can be done using the DML command **UPDATE**. It changes the values in one or more columns of specified rows. These changes may be affected in all or only in selected rows of the table. The new data for the column within these rows is given using the keyword **SET**, which is the essential clause of UPDATE command. The new data can be a constant, an expression or data from other tables.

The syntax of UPDATE command is:

```
UPDATE <table_name>
SET <column_name> = <value> [,<column_name> = <value>,...]
[WHERE <condition>];
```

Suppose the family monthly income of the student *Kaushi* is to be changed to Rs. 27000/-. Query 9.30 can perform this modification.

Query 9.30

```
UPDATE student
SET f_income=27000
WHERE name='Kaushi';
```

After the execution of this query, the response of MySQL will be as follows:

```
Query OK, 1 row affected (0.08 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Let us see the change in the record using the query,

```
SELECT * FROM student WHERE name='Kaushi';
```

The output of this query is shown in Figure 9.24. Compare this figure with Figure 9.17 and see the change.

```
+--------+--------+--------+------------+----------+----------+
| adm_no | name   | gender | dob        | course   | f_income |
+--------+--------+--------+------------+----------+----------+
|   1025 | Kaushi | M      | 1998-10-02 | Commerce |    27000 |
+--------+--------+--------+------------+----------+----------+
1 row in set (0.00 sec)
```

Fig. 9.24: Details of Kaushi with modified data in f_income column

We can use expressions to set the column values in records. Query 9.31 illustrates this concept.

Query 9.31

```
UPDATE student
SET f_income=f_income+1000
WHERE f_income<25000;
```

The output of this query will be as follows:

```
Query OK, 3 rows affected (0.06 sec)
Rows matched: 3  Changed: 3  Warnings: 0
```

The output reveals that three records satisfy the given condition and hence the values in f_income column of these records are incremented by 1000 (refer to Figures 9.17, 9.25).

The columns with NULL values can also be modified with UPDATE command. Suppose we want to store Rs. 20000/- as the monthly income in the respective field of the

records in which the column contains NULL value at present. Query 9.32 makes it possible.

Query 9.32

```
UPDATE student
SET f_income=20000
WHERE f_income IS NULL;
```

The changes made by Queries 9.31 and 9.32 can be observed in Figure 9.25, which is obtained by the query: `SELECT * FROM student;`

```
+--------+------------+--------+------------+------------+----------+
| adm_no | name       | gender | dob        | course     | f_income |
+--------+------------+--------+------------+------------+----------+
|   1001 | Alok       | M      | 1998-10-02 | Science    |    25000 |
|   1002 | Nike       | M      | 1998-11-26 | Science    |    35000 |
|   1003 | Bharath    | M      | 1999-01-01 | Commerce   |    45000 |
|   1004 | Virat      | M      | 1998-12-05 | Science    |    23000 |
|   1005 | Meera      | F      | 1998-08-15 | Science    |    20000 |
|   1006 | Divakar    | M      | 1998-02-21 | Science    |    20000 |
|   1025 | Kaushi     | M      | 1998-10-02 | Commerce   |    27000 |
|   1026 | Niveditha  | F      | 1999-03-04 | Humanities |    52000 |
|   1027 | Sreekumar  | M      | 1998-06-06 | Science    |    16000 |
|   1057 | Chaithanya | F      | 1999-06-03 | Science    |    20000 |
+--------+------------+--------+------------+------------+----------+
10 rows in set (0.00 sec)
```

Fig. 9.25: Modifications in column f_income

Let us do

With respect to the table ***stock***, write SQL statements to make some modifications in some column values.

9.8 Changing the structure of a table

In some situations, we may need to modify the structure of tables. It is an operation related to the schema and hence SQL provides a DDL command **`ALTER TABLE`** to modify the structure of a table. The alteration will be in the form of adding or dropping column(s), changing the data type and size of existing column(s), renaming a table, etc. Let us see how these changes are incorporated.

9.8.1 Adding a new column

One or more columns can be added at any position in an existing table. The syntax for adding a new column to a table is:

```
ALTER TABLE <table_name>
ADD <column_name> <data_type>[<size>] [<constraint>]
[FIRST | AFTER <column_name>];
```

Here the <table_name> indicates the name of the table which is to be altered; **ADD** is the keyword to add the new column; <column_name> <data_type> [<size>] indicates the description of the new column. **FIRST | AFTER** indicates the position of the newly added column.

If we want to add a column at the first position, use the clause FIRST. If we want to add a column at a specified position, use the clause AFTER <column_name>. Note that if we do not specify the position of the new column, then by default, it will be added as the last column of the table. Remember that the newly added column contains only NULL values.

Let us add two columns **gr_mks** and **reg_no** in the table ***student*** to hold the grace marks awarded to the students and their register number for the Higher Secondary examinations, respectively. Query 9.33 can make these changes in the structure of the table.

Query 9.33

```
ALTER TABLE student
ADD gr_mks INTEGER AFTER dob,
ADD reg_no INTEGER;
```

The response of MySQL for this query will be as follows:

```
Query OK, 10 rows affected (0.25 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

9.8.2 Changing the definition of a column

We can modify the characteristics of a column like data type, size and/or constraints by using the clause **MODIFY** with ALTER TABLE command. The syntax to modify the definition of a column is:

```
ALTER TABLE <table_name>
MODIFY <column_name> <data_type>[<size>] [<constraint>];
```

Let us modify the newly added column reg_no by providing a constraint **UNIQUE** to ensure that no two students have the same register number. Query 9.34 makes this change.

Query 9.34

```
ALTER TABLE student
MODIFY reg_no INTEGER UNIQUE;
```

The response of MySQL will be the same as that we obtained for Query 9.30. We can observe the changes made by the Queries 9.33 and 9.34 using the command: DESC student;. The output is shown in Figure 9.26.

```
+----------+-------------+------+-----+---------+----------------+
| Field    | Type        | Null | Key | Default | Extra          |
+----------+-------------+------+-----+---------+----------------+
| adm_no   | int(11)     | NO   | PRI | NULL    | auto_increment |
| name     | varchar(20) | NO   |     | NULL    |                |
| gender   | char(1)     | YES  |     | M       |                |
| dob      | date        | YES  |     | NULL    |                |
| gr_mks   | int(11)     | YES  |     | NULL    |                |
| course   | varchar(15) | YES  |     | NULL    |                |
| f_income | int(11)     | YES  |     | NULL    |                |
| reg_no   | int(11)     | YES  | UNI | NULL    |                |
+----------+-------------+------+-----+---------+----------------+
8 rows in set (0.00 sec)
```

Fig. 9.26: Structure of table student after alteration

We can see from Figure 9.26 that column `gr_mks` is added after `dob` and `reg_no` as the last column. Also note that column `Key` for `reg_no` contains **`UNIQUE`** constraint. Now, if we see the contents in the new columns using the command `SELECT * FROM student;`, we can see `NULL` values as shown in Figure 9.27.

```
+--------+------------+--------+------------+--------+------------+----------+--------+
| adm_no | name       | gender | dob        | gr_mks | course     | f_income | reg_no |
+--------+------------+--------+------------+--------+------------+----------+--------+
|   1001 | Alok       | M      | 1998-10-02 |   NULL | Science    |    25000 |   NULL |
|   1002 | Nike       | M      | 1998-11-26 |   NULL | Science    |    35000 |   NULL |
|   1003 | Bharath    | M      | 1999-01-01 |   NULL | Commerce   |    45000 |   NULL |
|   1004 | Virat      | M      | 1998-12-05 |   NULL | Science    |    23000 |   NULL |
|   1005 | Meera      | F      | 1998-08-15 |   NULL | Science    |    20000 |   NULL |
|   1006 | Divakar    | M      | 1998-02-21 |   NULL | Science    |    20000 |   NULL |
|   1025 | Kaushi     | M      | 1998-10-02 |   NULL | Commerce   |    27000 |   NULL |
|   1026 | Niveditha  | F      | 1999-03-04 |   NULL | Humanities |    52000 |   NULL |
|   1027 | Sreekumar  | M      | 1998-06-06 |   NULL | Science    |    16000 |   NULL |
|   1057 | Chaithanya | F      | 1999-06-03 |   NULL | Science    |    20000 |   NULL |
+--------+------------+--------+------------+--------+------------+----------+--------+
10 rows in set (0.00 sec)
```

Fig. 9.27: Contents in the new columns after modifying the structure of table student

Let us do

Write SQL statements to fill in data in the new columns with proper values and display the changes in the records.

Make some structural modifications in the table ***stock*** and store values according to the changes.

9.8.3 Removing column from a table

If we want to remove an existing column from a table, we can use **`DROP`** clause along with `ALTER TABLE` command. The syntax for removing a column from a table is:

```
ALTER TABLE <table_name>
DROP <column_name>;
```

For example, if we want to remove the column `gr_mks` from the table *student*, Query 9.35 is sufficient.

Query 9.35

```
ALTER TABLE student
DROP gr_mks;
```

The response of MySQL will be the same as we received for the earlier `ALTER TABLE` statements. The change can be observed using the command: `DESC student;`

9.8.4 Renaming a table

We can rename a table in the database by using the clause **`RENAME TO`** along with `ALTER TABLE` command. We can rename a table even though it contains tuples. The syntax for renaming a table is:

```
ALTER TABLE <table_name>
RENAME TO <new_table_name>;
```

For example, if we want to rename the table *student* as ***student2015***, Query 9.36 can help us.

Query 9.36

```
ALTER TABLE student
RENAME TO student2015;
```

The response of this query will be as follows:

```
Query OK, 0 rows affected (0.06 sec)
```

```
+------------------+
| Tables_in_school |
+------------------+
| student2015      |
+------------------+
1 row in set (0.00 sec)
```

Fig. 9.28: Renaming of table

The change can be viewed using the query: `SHOW TABLES;` and the output will be as shown in Figure 9.28.

9.9 Deleting rows from a table

Sometimes we need to remove one or more records from the table. The DML command **`DELETE`** is used to remove individual or a set of rows from a table. The rows which are to be deleted are selected by using the `WHERE` clause. If the `WHERE` clause is not used, all the rows in the table will be deleted. The `DELETE` command removes only records and not the individual field values. The syntax of the `DELETE` command is:

```
DELETE FROM <table_name> [WHERE <condition>];
```

For example, if the record of *Sreekumar* with admission number *1027* is to be deleted from the table, Query 9.37 can be used.

Query 9.37

```
DELETE FROM student2015 WHERE adm_no=1027;
```

The output of this query will be: **`Query OK, 1 row affected (0.08 sec)`**

Note that the name of the table is specified as *student2015*, because we have renamed the table *student* as *student2015*. The change in the table can be observed in Figure 9.29, which is obtained by the query: `SELECT * FROM student2015;`

```
+--------+------------+--------+------------+------------+----------+--------+
| adm_no | name       | gender | dob        | course     | f_income | reg_no |
+--------+------------+--------+------------+------------+----------+--------+
|   1001 | Alok       | M      | 1998-10-02 | Science    |    25000 |   NULL |
|   1002 | Nike       | M      | 1998-11-26 | Science    |    35000 |   NULL |
|   1003 | Bharath    | M      | 1999-01-01 | Commerce   |    45000 |   NULL |
|   1004 | Virat      | M      | 1998-12-05 | Science    |    23000 |   NULL |
|   1005 | Meera      | F      | 1998-08-15 | Science    |    20000 |   NULL |
|   1006 | Divakar    | M      | 1998-02-21 | Science    |    20000 |   NULL |
|   1025 | Kaushi     | M      | 1998-10-02 | Commerce   |    27000 |   NULL |
|   1026 | Niveditha  | F      | 1999-03-04 | Humanities |    52000 |   NULL |
|   1057 | Chaithanya | F      | 1999-06-03 | Science    |    20000 |   NULL |
+--------+------------+--------+------------+------------+----------+--------+
9 rows in set (0.02 sec)
```

Fig. 9.29: Contents in the renamed table after deleting a record

Figure 9.29 shows only seven columns. This is because of the removal of column `gr_mks` as a result of Query 9.35.

The deletion of values in a column is not possible. The effect may be brought out by filling the column with `NULL` values using `UPDATE` command as follows:

```
UPDATE <table-name>
SET <column-name> = NULL
[WHERE <condition>];
```

9.10 Removing table from a database

If we do not need a table, it can be removed from the database using **`DROP TABLE`** command. This DDL command removes a table from the database permanently even though the table contains tuples. We have to be very careful while deleting any existing table because once data is lost, it cannot be recovered later. The syntax to remove a table is:

```
DROP TABLE <table_name>;
```

For example, if we want to remove the table *student2015* from the database *school*, Query 9.38 can be used.

Query 9.38

```
DROP TABLE student2015;
```

Know your progress

1. The command used to edit the structure of a table is _____.
2. Restructuring of a column affects the values in it. State whether true or false.
3. How will you remove a column from a table?
4. Name the command used to remove a row from a table.
5. What happens when we use `DELETE FROM` command without a `WHERE` clause?

9.11 Nested queries

We are all familiar with the concept of nesting as we discussed it in nested if, nested loop, etc. Now, we will discuss nested queries. We know that nested means one inside another. Here the result of one query is dynamically substituted in the condition of another. A MySQL inner query is also called sub query, while the query that contains the sub query is called an outer query. SQL first evaluates the inner query (sub query) within the `WHERE` clause and the result of inner query is then substituted in the condition of the outer query. While using relational operators, ensure that the sub query (inner query) returns a single row output.

Suppose we want to display the name and course of the students who have the highest family monthly income. Query 9.39 can perform this task and Figure 9.30 shows the output.

Query 9.39

```
SELECT name, course FROM student2015
WHERE f_income=(SELECT MAX(f_income)
                FROM student2015);
```

In this example, the sub query (here query inside the pair of parentheses) is evaluated first and the highest value in `f_income` column (here 52000) is returned. This value is then compared with the value in `f_income` attribute of the records in the table and when a match is found, it is retrieved by the outer query.

```
+-----------+------------+
| name      | course     |
+-----------+------------+
| Niveditha | Humanities |
+-----------+------------+
1 row in set (0.03 sec)
```

Fig. 9.30: Result of nested query

9.12 Concept of views

MySQL supports the concept of views, which is a feature of RDBMS. A ***view*** is a virtual table that does not really exist in the database, but is derived from one or more tables. The table(s) from which the tuples are collected to constitute a view is called base table(s). These tuples are not physically stored anywhere, rather the definition of the view is stored in the database. Views are just like windows through which we can view the desired information that is actually stored in a base table. The contents of a view are taken from other tables depending upon a query condition. A view can be created with the DDL command `CREATE VIEW`. The syntax is:

```
CREATE VIEW <view_name>
AS SELECT <column_name1> [,<column_name2],...]
FROM <table_name>
[WHERE <condition>];
```

Let us create a view that contains the details of only those students who were born before January 1, 1999. Query 9.40 creates this view named ***student1998***.

Query 9.40

```
CREATE VIEW student1998
AS SELECT * FROM student2015
WHERE dob < '1999-1-1';
```

The output of this query will be: **`Query OK, 0 rows affected (0.31 sec)`**

Figure 9.31 shows the structure of the view and Figure 9.32 shows the contents of the view. These can be obtained by the commands `DESC student1998;` and `SELECT * FROM student1998;` respectively.

```
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| adm_no   | int(11)     | NO   |     | 0       |       |
| name     | varchar(20) | NO   |     | NULL    |       |
| gender   | char(1)     | YES  |     | M       |       |
| dob      | date        | YES  |     | NULL    |       |
| course   | varchar(15) | YES  |     | NULL    |       |
| f_income | int(11)     | YES  |     | NULL    |       |
| reg_no   | int(11)     | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
7 rows in set (0.01 sec)
```

Fig. 9.31: Structure of the view student1998

```
+--------+---------+--------+------------+----------+----------+--------+
| adm_no | name    | gender | dob        | course   | f_income | reg_no |
+--------+---------+--------+------------+----------+----------+--------+
|   1001 | Alok    | M      | 1998-10-02 | Science  |    25000 |   NULL |
|   1002 | Nike    | M      | 1998-11-26 | Science  |    35000 |   NULL |
|   1004 | Virat   | M      | 1998-12-05 | Science  |    23000 |   NULL |
|   1005 | Meera   | F      | 1998-08-15 | Science  |    20000 |   NULL |
|   1006 | Divakar | M      | 1998-02-21 | Science  |    20000 |   NULL |
|   1025 | Kaushi  | M      | 1998-10-02 | Commerce |    27000 |   NULL |
+--------+---------+--------+------------+----------+----------+--------+
6 rows in set (0.00 sec)
```

Fig. 9.32: Contents in the view student1998

Note that in Figure 9.31, the column `Key` does not contain the constraints of `adm_no` and `reg_no`, and Figure 9.32 shows the details of students born in the year 1998 only.

We can use all the DML commands with the view in quite a similar fashion as in the case of tables. Figure 9.32 is a result of such a query. Remember that we are accessing the base table though the associated views. So, take care when operations like `update` and `delete` are performed on views as they actually reflect in the base tables. Query 9.41 illustrates this concept.

Query 9.41

```
UPDATE student1998
SET reg_no=2201020
WHERE adm_no=1001;
```

Let us check the data in the table '*student2015*' after executing this query. Figure 9.33 shows the output of the query: `SELECT * FROM student2015;`

```
+--------+------------+--------+------------+------------+----------+---------+
| adm_no | name       | gender | dob        | course     | f_income | reg_no  |
+--------+------------+--------+------------+------------+----------+---------+
|   1001 | Alok       | M      | 1998-10-02 | Science    |    25000 | 2201020 |
|   1002 | Nike       | M      | 1998-11-26 | Science    |    35000 |    NULL |
|   1003 | Bharath    | M      | 1999-01-01 | Commerce   |    45000 |    NULL |
|   1004 | Virat      | M      | 1998-12-05 | Science    |    23000 |    NULL |
|   1005 | Meera      | F      | 1998-08-15 | Science    |    20000 |    NULL |
|   1006 | Divakar    | M      | 1998-02-21 | Science    |    20000 |    NULL |
|   1025 | Kaushi     | M      | 1998-10-02 | Commerce   |    27000 |    NULL |
|   1026 | Niveditha  | F      | 1999-03-04 | Humanities |    52000 |    NULL |
|   1057 | Chaithanya | F      | 1999-06-03 | Science    |    20000 |    NULL |
+--------+------------+--------+------------+------------+----------+---------+
9 rows in set (0.00 sec)
```

Fig. 9.33: Effect of updation in the base table through the view

The advantages of views lie in the fact that without sparing extra storage space, we can use the same table as different tables (but virtual). Another advantage is that the views implement sharing along with privacy. It also helps to reduce the complexity of conditions with `WHERE` clause while retrieving, updating or deleting records from tables.

If a view is no longer needed, it can be removed from the database with **`DROP VIEW`** command. But it will not affect the base table(s). The syntax is:

```
DROP VIEW <view_name>;
```

For example, if we want to delete the view *student1998* from the database, Query 9.42 can be used.

Query 9.42

```
DROP VIEW student1998;
```

Know your progress

1. What is mean by nested query?
2. What is view in SQL?
3. Creation of a view needs a `SELECT` command. State whether true or false.
4. Removal of a view deletes a table from the database. State whether true or false.
5. Can we update a table using its view?

Let us do

We have discussed all the DDL and DML commands of SQL, which are implemented through MySQL, with examples and their outputs. Let us summarise them in Table 9.8. The first two rows are filled in and the remaining rows are left to you.

SQL	Command	Essential Keyword	Purpose of the command
DDL	CREATE TABLE ALTER TABLE DROP TABLE CREATE VIEW DROP VIEW INSERT	ADD/MODIFY	To create tables To change the structure of tables
DML	SELECT UPDATE DELETE		

Table 9.8: Summary of SQL command

Let us conclude

Structured Query Language is used to operate on relational database. MySQL is a popular RDBMS package by which we can implement SQL commands. We have used various DDL commands to perform operations related with the structure of database. Constraints have been presented to ensure data validity and integrity in database. We have also discussed different DML commands to perform operations associated with the data contained in tables of a database. These operations include insertion, retrieval, modification and deletion of data in tables. We have also familiarized ourselves with nested queries and the concept of views. A good understanding of this chapter is essential for your higher studies in the field of computers.

Let us practice

1. The structure of a table is given to store the details of marks scored by students in an examination.

Data	Type	Description
Register number	Numeric	A unique and essential data to identify a student
Name	String	A maximum of 30 characters
Course	String	It can be Science, Commerce or Humanities
Marks of six subjects	Numeric each	Six separate columns are required

Write SQL statements for the creation of the table and the following requirements:

a. Insert data into the fields (at least 10 records).
b. Display the details of all students.
c. List the details of Science group students.
d. Count the number of students in each course.
e. Add a new column named Total to store the total marks.
f. Fill the column Total with the sum of the six marks of each student.
g. Display the highest total in each group.
h. Find the highest, lowest and average score in Subject 6 in Commerce group.
i. Display the names in the alphabetical order in each course.
j. Display the name of the student with the highest total.

2. The structure of a table is given to store the details of items in a computer shop.

Data	Type	Description
Item number	Numeric	A unique and essential data to identify an item
Item name	String	A maximum of 30 characters
Date of purchase	Date	Duplication is allowed
Unit price	Fractional Number	Price of a single item
Quantity	Numeric	Number of items
Manufacturer	String	Name of supplier (can duplicate)

Write SQL statements for the creation of the table and the following requirements:

a. Insert data into the fields (at least 10 records).
b. Display the details of all items in the table.
c. Display the names of items and total price of each.
d. List the items manufactured by a company *(specify the name)* avialable in the table.
e. Find the number of items from each manufacturer.
f. Display the details of items with the highest price.

g. List the names of items whose price is more than the average price of all the items.

h. Display the names of items purchased after 1-1-2015.

i. Get the details of items manufactured by two or three companies (*specify the names*) avialable in the table .

j. Display the details of items from a company *(specify the name)* with a stock of more than 20 pieces.

3. The structure of a table is given to store the details of higher secondary school teachers.

Data	Type	Description
Teacher ID	Numeric	A unique and essential data to identify a teacher
Name	String	A maximum of 30 characters
Gender	Character	Male or Female
Date of joining	Date	Duplication is allowed
Department	String	Science, Commerce, Humanities or Language
Basic pay	Numeric	Basic salary of a teacher

Write SQL statements for the creation of the table and the following requirements:

a. Insert data into the fields (at least 10 records).

b. Display the details of all female teachers in the table.

c. List the details of male teachers in the Science department.

d. Display the names and basic pay of teachers in the Language department whose basic pay is Rs. 21000/- or more.

e. Display the names and 71% of basic pay of the teachers.

f. Find the number of teachers in each department.

g. Display the details of teachers whose basic pay is less than the average basic pay.

h. List the male teachers who joined before 1-1-2010.

i. Increment the basic pay of all teachers by Rs. 1000/-.

j. Delete the details of teachers from the Language department.

4. The structure of a table is given to store the details of customers in a bank.

Data	Type	Description
Account number	Numeric	A unique and essential data to identify a customer
Name	String	A maximum of 30 characters
Gender	Character	Male or Female
Date of joining	Date	Duplication is allowed
Type of account	String	SB or Current
Balance amount	Numeric	Can be a fractional number

Write SQL statements for the creation of the table and the following requirements:

a. Insert data into the fields (at least 10 records).

b. Display the details of customers having SB account.

c. Display the names of customers with a balance amount greater than Rs. 5000/-.

d. Display the details of female customers with a balance amount greater than Rs. 10000/-.

e. Count the number of male and female customers.

f. Display the names of customers with the highest balance amount.

g. Display the names of customers whose names end with 'kumar'.

h. Update the balance amount of a particular customer with a deposit amount of Rs. 2000/-.

i. Display the details of customers with a tax deduction of 2% of the balance amount for those who have Rs.2,00,000/- in their account.

j. Delete the details of customers with current account.

Let us assess

1. The command to remove rows from a table 'CUSTOMER' is:

   a. `REMOVE FROM CUSTOMER` b. `DROP TABLE CUSTOMER`

   c. `DELETE FROM CUSTOMER` d. `UPDATE CUSTOMER`

2. If values for some columns are unknown, how is a row inserted?
3. Distinguish between `CHAR` and `VARCHAR` data types of SQL.
4. What is the difference between `PRIMARY KEY` and `UNIQUE` constraints?
5. What do you mean by `NULL` value in SQL?

6. Which of the following is the correct order of clauses for the SELECT statements?

   a. SELECT, FROM, WHERE, ORDER BY

   b. SELECT, FROM, ORDER BY, WHERE

   c. SELECT, WHERE, FROM, ORDER BY

   d. SELECT, WHERE, ORDER BY, FROM

7. The SQL operator _____ is used with pattern matching.

8. Read the following strings:

   (i) 'Sree Kumar' (ii) 'Kumaran' (iii) 'Kumar Shanu' (iv) 'Sreekumar'

   Choose the correct option that matches with the pattern '%Kumar', when used with LIKE operator in a SELECT statement.

   a. Strings (i) and (ii) only

   b. Strings (i), (iii) and (iv) only

   c. Strings (i) and (iii) only

   d. All the strings

9. List any five built-in functions of SQL and the value returned by each.

10. Distinguish between WHERE clause and HAVING clause.

11. Write any four DML commands in SQL.

12. Write the essential clause required for each of the following SQL command.

    a. INSERT INTO b. SELECT c. UPDATE

13. Consider the given table Customer and write the output of the following SQL queries:

AccNo	Name	Branch	Amount
1001	Kumar	Calicut	10000
1002	Salim	Trivandrum	20000
1003	Fida	Kottayam	18000
1004	John	Kannur	30000
1005	Raju	Thrissur	5000

```
a. SELECT * FROM customer WHERE Amount>25000;

b. SELECT Name FROM customer
   WHERE Branch IN ('Calicut, 'Kannur');

c. SELECT COUNT(*) FROM customer WHERE Amount < 20000;

d. SELECT Name FROM customer WHERE Name LIKE "%m%";

e. SELECT * FROM customer ORDER BY Amount DESC;
```

14. Distinguish between `COUNT(*)` and `COUNT(column-name)`.
15. Consider the given table ITEMS.

Item Code	Name	Category	Unit Price	Sales Price
0001	Pencil	Stationery	5.00	8.00
0002	Pen	Stationery	8.00	10.00
0003	NoteBook	Stationery	10.00	20.00
0004	Chappal	Footwear	50.00	70.00
0005	Apple	Fruits	60.00	90.00
0006	Orange	Fruits	40.00	60.00
0007	Pen	Stationery	10.00	12.00

a. Suggest a suitable primary key for the above table. Give justification.

b. Write SQL statements for the following:

   i. To list all stationery items.

   ii. To list itemcode, name and profit of all items.

   iii. To count the number of items in each category.

   iv. To list all stationery items in the descending order of their unit price.

   v. To find the item with the highest selling price.

   vi. To create a view that contains the details of all stationery items.

16. What are the different modifications that can be made on the structure of a table? Which is the SQL command required for this? Specify the clauses needed for each type of modification.
17. A table is created in SQL with 10 records. Which SQL command is used to change the values in a column of specified rows? Write the format.
18. Name the keyword used with `SELECT` command to avoid duplication of values in a column.
19. Distinguish between `DISTINCT` and `UNIQUE` in SQL.
20. Pick the odd one out and give reason:

    a. `CREATE` b. `SELECT` c. `UPDATE` d. `INSERT`

10 Enterprise Resourse Planning

Significant Learning Outcomes

After the completion of this chapter, the learner

- identifies the need of Enterprise Resource Planning (ERP) in an enterprise.
- lists different functional units of an ERP system.
- explains the importance of Business Process Re-engineering (BPR) in the implementation of ERP.
- recognises different phases in implementation of an ERP system.
- lists some important ERP packages in the market.
- explains the benefits and risks of ERP implementation in an enterprise.
- becomes familiar with some related technologies of ERP system.

In today's competitive world, one has to manage the future of an enterprise more cleverly. Managing the future means managing the information. A large enterprise may generate huge amount of data such as financial data, customer details, purchase details, employee data etc. Only the organization that makes the best possible use of this information can succeed. In this age of information explosion, it is very difficult to manage this huge information by people alone. Information technology and its related technologies can be used for planning and organizing resources and information of an enterprise. Hence most of the organizations are moving to Enterprise Resource Planning (ERP) packages as a solution to their information management problem.

10.1 Overview of an enterprise

An enterprise is a group of people and other resources working together for a common goal. An enterprise may consist of different sections or departments such as manufacturing or production, planning, sales, purchase, finance, distribution etc. Each department will have their own duties and responsibilities and they are working to achieve the objective which

is set for the enterprise. There are different types of resources in an enterprise like men, material, money and machine. Information system can be designed for various departments of an enterprise so that accurate and timely data can be provided to the concerned persons. Figure 10.1 is a schematic diagram of an enterprise and it shows that an enterprise contains resources and people working for a common goal or objective.

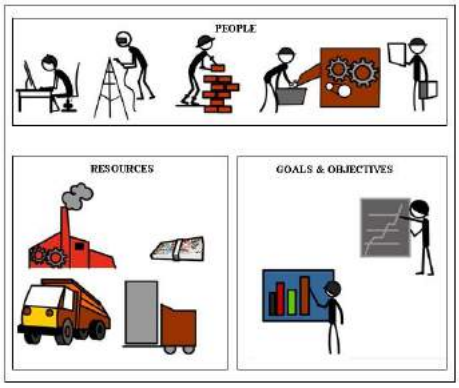


Fig 10.1: An Enterprise

10.2 Concept of Enterprise Resource Planning

In some enterprises, different departments function independently. So the information that is produced by each department may be available only to the top management of the department and it is not available to the other departments. Here, there is no communication between different sections of an enterprise. An organization or an enterprise where there is no communication between departments can be depicted as shown in Figure 10.2.

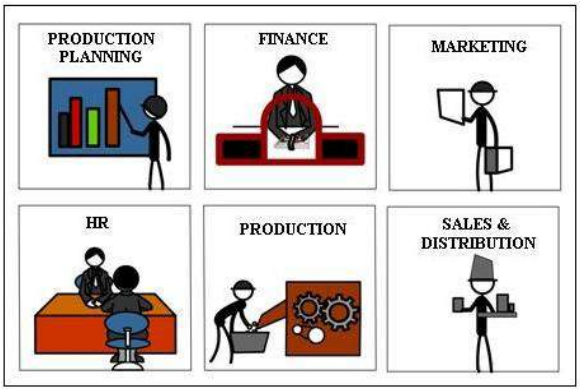


Fig. 10.2: An enterprise with no or little communication between departments

For better benefit and efficiency, each department must know what the other departments are doing. An enterprise can be considered as a system and all its departments as its sub systems. Information about the entire enterprise can be stored in a centralized database and it is made available to all departments. Such an enterprise can be depicted as shown in Figure 10.3.

ERP combines all the business requirements of an enterprise or company together into a single,

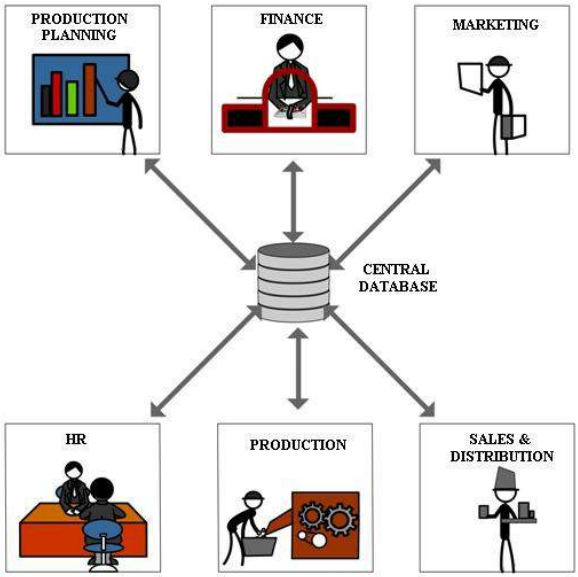


Fig. 10.3: An enterprise with Central Database

integrated software that runs off a single database so that the various departments of an enterprise can share information and communicate with each other more easily. Conceptually, ERP replaces the old standalone computer systems in each area of an enterprise such as finance, human resource, manufacturing, sales, etc. with a single software program that facilitates various functional modules. Thus, employees in any department get the required information related to the activities of the respective department. In addition to this, the information will be available across the departments. For example, someone in the finance department can use ERP to see if any sale order has been shipped from the warehouse so that he can now confidently plan for working capital management for the next period. It organises and integrates operation processes and information flows to make optimum use of resources such as men, material, money and machine.

ERP system is a fully integrated business management system covering functional areas of an enterprise like Finance, Human Resources, Production, Sales , Logistics etc. The key element of ERP is the use of a single database to store data for various system modules. ERP systems utilise components of both computer software and hardware.

10.3 Functional units of ERP

The resources available in an enterprise must be utilized effectively. So it is the responsibility of the management to plan the resources. The ERP system helps the management in making the planning process more productive and efficient.

The entire ERP package contains many modules or sub units. The number and functioning of modules vary with the nature of enterprise and the type of ERP package. Some commonly used modules, available in almost all packages are briefly explained below.

Financial module

This module is the core of many ERP software packages. It can collect financial data from various functional departments and generate valuable financial reports. Financial reports include balance sheets, general ledger, trial balance, financial statements, etc. This module also includes financial accounting, investment management, enterprise controlling and treasury.

Manufacturing module

Manufacturing module contains necessary business rules to manage the entire production process. This module of ERP enables an enterprise to combine technology and business processes to get integrated solutions. It provides information for the entire operation to be carried out for the production. It also

provides freedom to change manufacturing and planning methods as and when required.

Production planning module

This module is used for optimising the utilisation of available resources and helps the organisation to plan their production. Optimisation is the process of making the best effective use of resources. This module identifies the materials required, allocates optimal resources using historical production and performs sales forecasting with the sales data.

HR module

HR stands for human resource. HR module of ERP focuses on the management of human resources and human capital. HR module maintains an updated and complete employee database including personal information, salary details, attendance, performance, promotion, etc. of all employees in an enterprise.

Inventory control module

This module covers processes of maintaining the appropriate level of stock in the warehouse. It is responsible for identifying the inventory requirements, setting target, monitoring item usages, reporting inventory status, etc. Integration of inventory control with sales, purchase and finance helps an enterprise to generate intelligent reports.

Purchasing module

Purchase module is used for making the required raw materials available in the right time and at the right price. Generating purchase order for the supplier, evaluating the supplier, and billing are made available in this module. Purchase module is closely connected with the inventory, finance and production planning module.

Marketing module

Marketing module is used for monitoring and tracking customer orders, increasing customer satisfaction and for eliminating credit risks. It helps marketers to analyze, plan, execute and measure all marketing activities.

Sales and distribution module

Revenue from sale is the life-blood of any enterprise. Sales module deals with important parts of a sales cycle. Sales cycle includes inquiries, order placement, order scheduling, dispatching and invoicing. This module is closely integrated with the e-commerce website of the organization.

Quality management module

This module is used for managing the quality of the product. The quality management module fulfills the following functions-Quality planning, Quality inspection and Quality control. Based on the size and nature of enterprise, other than above modules, there may be different types of modules in the functional unit of an enterprise. An ERP system integrates all the above components into a single software system. Figure 10.4 shows the various functional units of ERP.

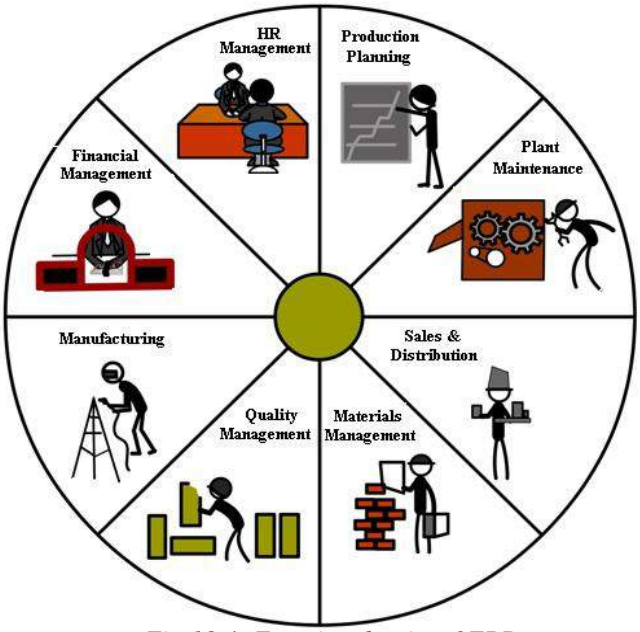


Fig 10.4: Functional units of ERP

10.4 Business Process Re-engineering

Business Process Re-engineering (BPR) is the analysis and redesign of work flow within an enterprise. It involves changes in structure and process of a business environment. Due to rapid change in business field, especially in information technology, the process of a business needs to be redesigned for better service and benefit. Information technology plays a major role in business process re-engineering as it provides office automation, online business, flexible manufacturing, high speed delivery and paperless transactions.

Re-engineering of a business will result in efficient time management, reduced cost and effective utilization of resources. In the business world, competition is based on price, quality, selection and service. Nowadays, with the advent of internet and e-commerce, distance and time is not a barrier for a business. These changes impose the need for organizational transformation, where the entire processes and organizational structure are changed.

In Business Process Re-engineering, re engineering and process may be defined separately. Re-engineering is the rethinking and redesigning of a business process to achieve improvements in the performance of an enterprise. Business process is a structured set of activities designed to produce a specified output for an enterprise. Each process consists of related steps or activities that use people, information and other resources.

A business process consists of three elements.

- **Inputs**: Data such as application forms, customer enquiries and materials for processing.
- **Processing**: Set of activities or stages of operations to get an output.
- **Outcome**: It is the output of processing steps.

Here, processing is the time consuming and problematic element of business process. Basically, a business process is the way in which we perform our work and, business process re-engineering is the process of changing this way in a better style to accomplish the goals of an enterprise. So BPR mainly focuses on processing parts, which is to be re-engineered to reduce limitations and to increase performance.

The fundamental rethinking and redesign of the business process to achieve improvements in performance such as cost, quality, service and speed of an enterprise is called business process re-engineering. Figure 10.5 illustrates the phases in the life cycle of BPR which includes identification of business process, analysis of current business process, designing of a revised process and implementation of revised process.

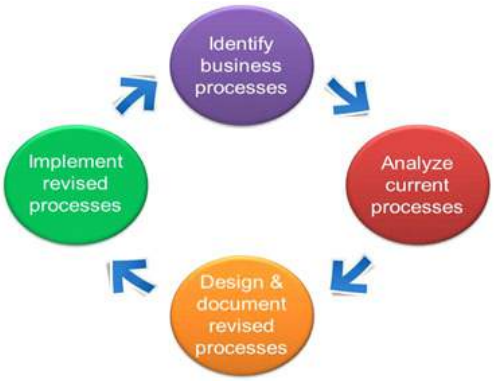


Fig 10.5: Business Process Re-engineering (BPR)

Connection between ERP and BPR

Before implementing ERP in a business, the need of such a new system must be ensured. Business process re-engineering will help an enterprise to determine the changes in the structure or process of a business for better aspects. So ERP and BPR are related and they go hand in hand, and in most of the cases business process re engineering is performed before enterprise resource planning. Conducting business process re engineering before implementing enterprise resource planning will help an enterprise to avoid unnecessary modules from the software. BPR first ensures that business processes are optimized before software is configured and also ensures that software functionality will closely match the actual process steps.

All business process re- engineering does not necessary lead to the implementation of a new ERP system. Some BPR may find that there is no need of BPR in the enterprise because of cost, effectiveness and other challenges.

In some other cases BPR and ERP are used together to achieve better result for the enterprise and to improve existing ERP.

Know your progress

1. Define the term enterprise.
2. _____ is a fully integrated business management system.
3. Analysis and redesign of work flow within an enterprise is called _____
4. List any four commonly available modules in ERP.
5. In ERP, information about all the departments of an enterprise is stored in _____?

10.5 Implementation of ERP

Nowadays almost all major organisations are moving to enterprise resource planning package as a solution to their information management problem. If an ERP package is correctly selected and implemented in an enterprise, then the productivity and profit of the organisation will increase dramatically. The implementation should be well planned and executed perfectly. So, the ERP implementation has to go through different phases. But, there are no clear lines separating these phases and in many cases, one phase will start before the previous one is completed.

The different phases of ERP implementation are given below.

- Pre evaluation screening
- Package selection
- Project planning
- Gap analysis
- Business Process Reengineering
- Installation and configuration
- Implementation team training
- Testing
- Going live
- End user training
- Post implementation

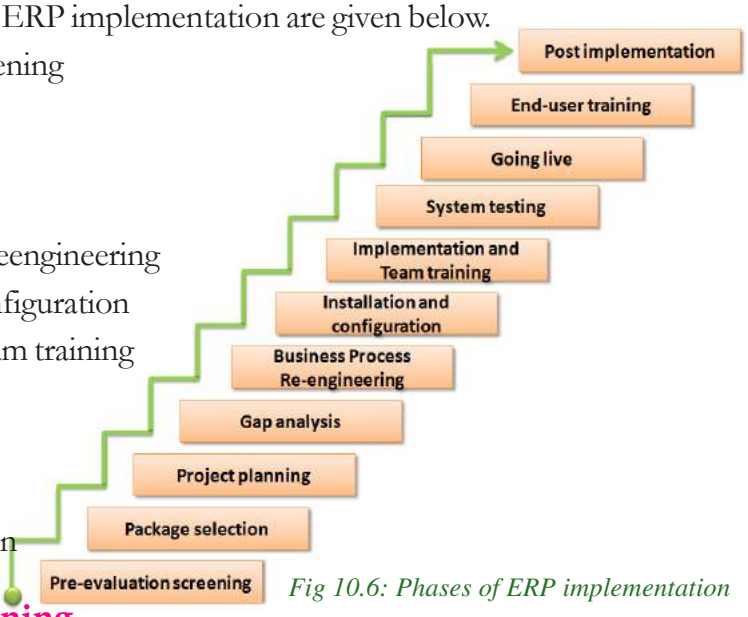


Fig 10.6: Phases of ERP implementation

Pre evaluation screening

There may be a number of packages available in the market for implementing ERP in an enterprise. Instead of evaluating all the packages, for selecting an appropriate package, we have to limit the number of packages to be evaluated. So pre evaluation screening is the first phase of implementation of a ERP package.

Package selection

In the previous phase, we selected a few packages for evaluating their performance and efficiency. The package that we select will decide the success or failure of the project. Since an ERP system needs huge investment, once a package is selected and purchased, it is not an easy task to switch to another one. The most important factor kept in mind while evaluating the different screened packages is that, it should be flexible enough to meet the enterprise requirements.

Project planning

In this phase, the implementation process is planned and designed. The time schedule and deadlines for different activities are determined. Roles and responsibilities of the concerned staff are identified and assigned to each person. This phase decides when to begin the project, how to do it and when to complete it.

Gap analysis

There is no ERP package available in the market which fulfils all the requirements of an enterprise perfectly. Although ERP vendors may claim that their software will solve all problems, there will still be some gaps. Even the best ERP package will be able to meet only 80 percentage of the needs of the enterprise. So, the gap should be analysed and considered in the following phases of ERP implementation.

Business Process Reengineering (BPR)

The fundamental rethinking and redesign of the business process to achieve improvements in performance such as cost, quality, service and speed of an enterprise is called business process re engineering. The importance of BPR in the implementation of an ERP system has been already discussed (refer to Section 10.4)

Installation and configuration

This is the main functional phase of an ERP implementation. Before installing a new ERP package, the whole process of the enterprise should be analyzed in detail. Instead of replacing the old system with the new ERP system, it will be better and effective to develop an appropriate prototype. This prototype is a miniature of the actual ERP that is going to be implemented. In future, we have to conduct continuous testing of the prototype to find out the weakness, and steps can be taken to avoid these mistakes when the real ERP system gets introduced. The task of configuration of ERP system should be assigned to experienced staff.

Implementation team training

This is the phase where the company trains its employees to implement and later run the system. The ERP vendors and the hired consultants will leave the organisation after the implementation is over. So the company must select employees with the

right attitude, willingness to change and learn new things, and who are not afraid of technology.

System testing

The software is tested to ensure that it performs properly from both the technical and functional areas of an enterprise. The validity of output can be determined with the help of sample data. If any mistakes are found out at this stage, it should be corrected before the operation of ERP system.

Going live

This is the phase where ERP is made available to the entire organization. After this phase, the system is ready for use. After configuration and testing, removing errors, and checking the correctness of reports, the system becomes 'live' to perform the enterprise operations. For the smooth functioning, the ERP vendors will provide support and service to the enterprise.

End-user training

The actual users of the ERP system need training on how to use the system. This phase may be started before the system goes live. The employees who are going to use the new system are identified. Their skills are noted and they are divided into groups, based on the skill. Each group is then given training on the new system. Since the success of ERP system is in the hands of end-users, this training is very important.

Post implementation

After installing and operating a new ERP system, there may be the need of updating and evaluating the system. So, after implementation, it is to be checked whether the objective set for the ERP system is met. In this phase, errors may be corrected and necessary steps may be taken to improve the processing efficiency.

10.6 ERP solution providers/ERP packages

Selection of ERP package is very crucial in the implementation of an ERP system. If an ERP package is chosen correctly, implemented judiciously and used efficiently, the productivity of the enterprise will be increased. ERP package vendors are investing huge amount of time, money and effort in the research and development of packaged solutions. There are so many ERP vendors in the world. Some of the popular ERP packages are Oracle, SAP, Odoo, Microsoft Dynamics, and Tally.

Oracle

Oracle corporation is headquartered in Redwood shores , California, USA. Oracle was originally known for its database system rather than its ERP system. The ERP

package from Oracle provides strong finance and accounting module. It also provides good customer and supplier interaction, effective production analysis, efficient human resource management and better pricing module. Oracle also developed Customer Relationship Management (CRM) and Supply Chain Management (SCM) software.

JD Edwards was a popular ERP package provider which was purchased by PeopleSoft company in 2003. Later PeopleSoft company was purchased by Oracle Corporation in 2005.

SAP

SAP stands for Systems, Applications and Products for data processing. It is a German multinational software corporation headquartered in Walldorf and founded in 1972. The company started by developing software for integrated business solutions. In the beginning, the software was developed aiming at large multinational companies. After gaining good acceptance from them, the company started developing packages for small scale enterprises also. SAP also developed Customer Relationship Management (CRM), Supply Chain Management (SCM), and Product Life cycle Management (PLM) software.

Odoo

Odoo is an open source ERP. In open source ERP the source code or program can be modified as necessary, based on the requirement of organization. Odoo was formerly known as OpenERP until May 2014.

Microsoft Dynamics

Microsoft is an American multinational corporation headquartered in Redmond, Washington. Microsoft Dynamics is part of Microsoft business solutions. It provides a group of enterprise resource planning products primarily aimed at midsized enterprises. This package can be installed and used easily and it provides good user interface. It also provides customer relationship management (CRM) software.

Tally ERP

Tally solutions Pvt Ltd is a Bangalore based Software Company in India. Tally ERP is a business accounting software for accounting, inventory and payroll.

In the near future, new ERP vendor may introduce new ERP packages and existing ERPs may get more facilities and capabilities. Selecting an ERP solution is a serious exercise and has to be executed with great care.

Other than the above packages there are so many popular packages in the market. Some of them aEpicore, Infor, Sage , Reach , Openbravo, Apache OFBiz, Compiere, WebERP, ERP5 etc.

10.7 Benefits and risks of ERP

Installing an ERP package in an enterprise has many direct and indirect advantages. More over, it has some challenges also. Let us discuss the benefits and limitations of an ERP system.

10.7.1 Benefits of ERP system

There are so many advantages on implementing an ERP system in an enterprise. Some of the major benefits are briefly explained:

1. Improved resource utilization

An enterprise can plan and manage its resources effectively by installing ERP software. So the wastage or loss of all types of resources can be reduced, and improved resource utilization can be ensured.

2. Better customer satisfaction

Customer satisfaction means meeting maximum customers' requirements for a product or service. Using an ERP system, a customer will get more attention and service of an enterprise without spending more money and time. With the introduction of web based ERP, a customer can place the order, track the status of the order and make the payment from his/her home.

3. Provides accurate information

In today's competitive world, an enterprise has to plan and manage the future cleverly. So, an enterprise needs high quality, relevant and accurate information. An ERP system will be able to provide such information for the better future planning of the enterprise.

4. Decision making capability

Accurate and relevant information given to decision makers will help them to take better decisions for running a system more smoothly. Better decision from an enterprise will help them to go a step ahead of its competitors.

5. Increased flexibility

An ERP system improves the flexibility of manufacturing operations and the entire organization. A flexible organization can adapt to the changes in the environment rapidly. ERP system helps organizations to remain flexible by making enterprise information available without any departmental barriers.

6. Information integrity

The most important advantage of ERP is in its promotion of integration of various departments and hence we will get an integrated form of information about the enterprise. The entire information about an enterprise is stored in a centralized data base, so that complete visibility into all the important processes across various departments of an organisation can be achieved.

Moreover, an ERP system provides high security, business intelligence, unified reporting system, improved supplier performance, use of latest technologies, high speed delivery of product and better analysis and planning capabilities.

10.7.2 Risks of ERP implementation

Some of the problems and limitations of using an ERP package in an enterprise are the following:

1. High cost

The cost of ERP software configuration and implementation is very high. The high price of the package, associated license fees and other charges are the main problems of ERP installation. There may be additional indirect costs due to ERP implementation like new IT infrastructure, upgrading the network facilities etc. For small scale enterprises purchasing and installing such an expensive package may not be preferable.

2. Time consuming

The ERP implementation process has to go through different phases and it is highly time consuming. It may take months or even one or two years to get completed and fully functional.

3. Requirement of additional trained staff

To run an ERP system, trained and experienced employees are to be appointed in the enterprise. The correct selection of an ERP package alone cannot guarantee the success of an enterprise. In addition, the contribution of skilled and trained persons in using ERP system is very important.

4. Operational and maintenance issues

Implementation of an ERP needs major changes in the current process of an enterprise. Sometimes, it will be difficult to adjust with these changes by employees and management of an enterprise, as it is human nature to resist changes. Most enterprises are implementing ERP in the assumption that someday the project will end. But actually the implementation is a life cycle and the ERP system cannot end with the implementation and it is a continuous process. So, maintenance of an ERP system is very difficult.

10.8 ERP and related technologies

An ERP system integrates separate business functions - material management, product planning, sales, distribution, financial and others - into single applications. If some other technologies which are going to be discussed in this section are used along with stand alone ERP package, the performance of the enterprise will be increased significantly. Let us discuss some of the related technologies used along with ERP packages.

Product Life Cycle Management (PLM)

Product life cycle is used for determining the lifespan of a product. Product life cycle management is the process of managing the entire life cycle of a product.

Figure 10.7 shows the general schematic diagram of four stage product life cycle which consists of development and introduction of a new product, then its growth in the market, its maturity and at last its decline if it cannot compete with similar products of other companies.

Product Life cycle Management is performed for increasing the quality of a product, for increasing marketing opportunities and for ensuring usage of latest technology in production for introducing a new model. To create a new product or to modify an existing product, the company must understand its customer, market and competitors. The information gathered from product life cycle will help an enterprise to understand the state/status of a product in the existing market.

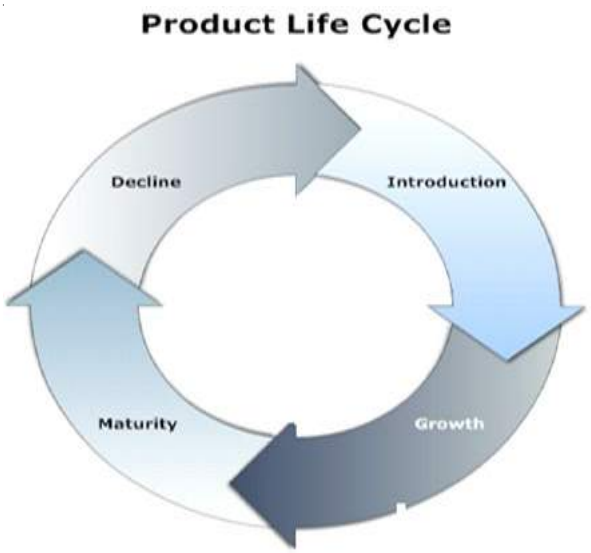


Fig 10.7: General schematic diagram of four stage product life cycle

Customer Relationship Management (CRM)

Customer is an individual or group who receives or consumes goods and services. Customers are the most important part of any enterprise. The success of any enterprise depends on good relationship with its customers. CRM is a broadly used term that covers policies used by an enterprise to manage their relationships with customers. The objectives of a CRM strategy must consider a company's specific situation and its customers' needs and expectations. Customer Relationship Management is a comprehensive approach for creating, maintaining and expanding customer relationships. It is not only the responsibility of customer service group or IT team. It touches all major part of an enterprise. Figure 10.8 shows the processes involved in CRM.

It includes the capture, storage and analysis of customer information. The data gathered as a part of CRM must consider customer privacy and data security. Customers want the assurance that their data is not shared with third parties without their consent and not accessed illegally by third parties. Customers also want their data used by companies to provide a benefit to them.

Fig 10.8: Processes in CRM

The technology requirement of customer relationship management consists of a database to store entire information about the customer and a software for interacting, analyzing and supporting customers.

Management Information System (MIS)

We can see that there are three components in MIS - Management, Information and System. Management is the ultimate user or the decision maker, information is the processed data and system is the integration and holistic view of the enterprise. In the technical perspective, an information system collects, stores, and distributes information from an organization's environment and internal operations to support organizational functions. It is also used for decision making, communication, coordination, control, analysis, and visualization of an enterprise. Information systems transform raw data into useful information through three basic activities: input, processing, and output.

An enterprise may contain different categories of employees like clerks, assistants, officers, executives, managers etc. All of them are the users of MIS. A management information system will collect relevant data from inside and outside an enterprise. This data is processed and stored in a centralized database and is made available to

its users whenever it is needed. MIS has the capability to generate reports as and when the user demands it. It is mainly used to create reports on the basis of which decisions are made. Figure 10.9 shows the role of MIS in an organisation.

So Management Information System can be defined as an integrated system of man and machine for providing the information to support the operations, the management and the decision making function in an organisation.

Fig 10.9: Role of MIS in an organisation

Supply Chain Management (SCM)

The supply chain consists of all the activities associated with moving goods from the supplier to the customer. It begins with collecting raw materials and ends with receiving the goods by the consumer, as shown in Figure 10.10. It is very important for companies to move product to their customers quickly. Faster product delivery or availability will increase the sale and satisfaction of customers. So it is very important to manage the activities in supply chain. Software packages are available in the market for managing the same.

Supply Chain Management (SCM) is a systematic approach for managing supply chain. SCM activities include inventory management, transportation management, warehouse management, distribution strategy planning, customer service performance monitoring, computer simulation etc.

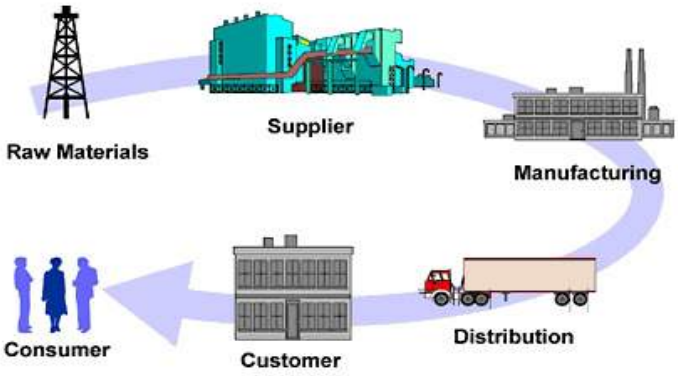


Fig 10.10: Activities involved in SCM

Decision Support System (DSS)

Decision Support Systems are interactive, computer-based systems that aid users in judgment and choice activities. It is a computer program application that analyses business data and presents it so that users can make business decisions more easily. DSS focuses on providing help in analysing situations rather than providing right information in the form of various types of reports. DSS is only a support in nature,

but human decision makers still retain their supremacy. DSS needs an effective database management system to provide the support in decision making. Figure 10.11 shows the function of DSS.

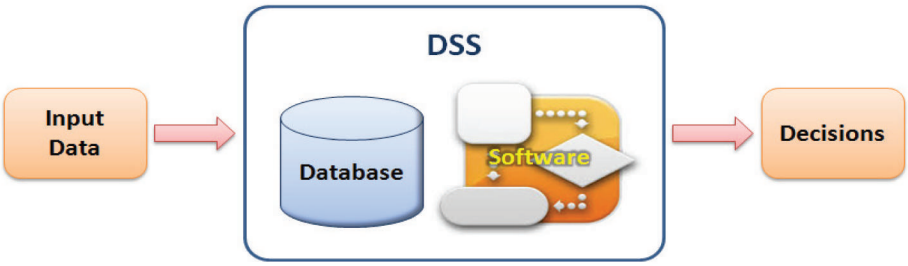


Fig 10.11: Function of DSS

Let us conclude

This chapter has given us an awareness about the growing trends in the business filed which make business productive, economic and profitable. A basic information about Enterprise Resource Planning (ERP) has been conveyed with its major functional units. The importance of business process re-engineering in the implementation of ERP has been cited. The different phases in the implementation of ERP are mentioned precisely. The most commonly used ERP packages available in the market are presented. The benefits and limitations of ERP implementation are also listed. Short descriptions of some importnat technologies related to ERP have been given at the end.

Let us assess

1. DSS stands for _____.
   a. Decision Signal system
   b. Design Support System
   c. Decision Support Scheme
   d. Decision Support System
2. Pick the odd one out (SAP, Oracle, C++, Odoo) and justify.
3. SAP stands for _____.
4. Define enterprise resource planning.
5. Give the significance of HR module in an ERP package.
6. Give the relation between business process re engineering (BPR) and enterprise resource planning (ERP).
7. Write a short note about any two ERP solution providers.
8. List the advantages of ERP implementation in an enterprise.
9. Write a short note about any one of the technologies related with ERP.
10. Briefly explain different phases of ERP implementation.

11 Trends and Issues in ICT

Significant Learning Outcomes

After the completion of this chapter, the learner

- identifies the various mobile computing terminologies.
- recognises the features of mobile operating systems.
- discovers the features of Android operating system.
- lists and explains various Intellectual Property Rights.
- explains cyberspace.
- distinguishes different types of cyber-crimes.
- details cyber law and ethics.
- identifies the importance of IT Act.
- recognises infomania.

The use of Internet has increased rapidly in the last few decades. The modern society cannot think of a day without the Internet and the services provided by it. The developments in computer technology have led to the development of powerful but cheaper mobile devices. People have started accessing these services on the Internet using their mobile devices like mobile phones, tablets, etc.

Information and Communication Technology (ICT) is the term often used as an extended synonym for Information Technology (IT). ICT is more specific in integrating telecommunication and computers. In this chapter, we discuss the different technologies and services in mobile communication. The various mobile operating systems available are discussed here with special focus on Android operating system. The technologies like bigdata, Radio Frequency Identification (RFID) that improve the way of doing business is also discussed here.

Similar to rights on property like land, house, etc., intellectual properties like music, films, software, designs, etc. have also ownership rights. These rights are called Intellectual Property Rights (IPR) and the issues related to

them are discussed in this chapter. Every technology has a dark side. Internet also has its share of issues and threats. Some people use this medium for performing illegal activities. We will discuss about such actions and how we can safeguard ourselves in the Internet.

11.1 Mobile computing

The advances in computer technology have led to the development of more computing power in small devices. Light weight computing devices with low power consumption are now cheap and common. Devices like laptops, tablets, smart phones, etc. have drastically changed the lifestyle and work culture of people. Today people are able to connect to others, send and receive files or other information anywhere anytime. This has increased computing demands on a daily basis.

Mobile computing is a technology that has computing capability and can transmit/ receive data while in motion. Mobile computing requires portable computing devices like laptops, tablets, smart phones, etc., wireless communication networks and connectivity to the Internet. The demand for mobile computing started the growth and development of mobile communication technology.

11.2 Mobile communication

The term 'mobile' has completely revolutionised communication by opening up innovative ways for exchanging information. Today, mobile communication has become the backbone of society. Mobile system technologies have improved standards of living. Mobile communication networks do not depend on any physical connection to communicate between two devices.

11.2.1 Generations in mobile communication

The mobile phone was introduced in the year 1946. In the initial stage, growth in mobile communication was very slow. With the increase in the number of users, accommodating them within the limited available frequency spectrum became a major problem. To solve this problem, the concept of cellular communication was evolved.The cellular concept was developed in the 1960's at Bell Laboratories. However, in the late 1990's, when the Government of India offered spectrum licenses for mobile communication, cellular technology became a common standard in our country. Let us discuss different generations in mobile communication .

a. First Generation networks

First Generation (1G) networks refer to the first generation of wireless telephone technology (mobile telecommunications) developed around 1980. 1G mobile phones were based on the analog system and provided basic voice facility only.

b. Second Generation networks

Second Generation (2G) networks follow the digital system for communication. This improved the audio quality in transmission. In 2G networks, phone conversations are digitally encrypted.These networks provided far greater mobile phone coverage. 2G networks also introduced data services for mobile. Picture messages and Multimedia Messaging Service (MMS) were introduced. The two popular standards introduced by 2G systems are Global System for Mobiles (GSM) and Code Division Multiple Access (CDMA). A detailed discussion on the GSM and CDMA standards are given below:

i. Global System for Mobiles

Global System for Mobiles (GSM) is a globally accepted standard for digital cellular communication. It is a digital, circuit-switched network for voice telephony. The frequency band for GSM is 900 MHz to 1800 MHz. GSM follows a uniform international standard that made it possible to use a mobile device around the world. The network is identified using the Subscriber Identity Module (SIM). Users can select a handset of their choice. GSM is the most successful family of cellular standards. Let us now discuss the different technologies that are used to enhance data communication features to GSM.

General Packet Radio Service and
Enhanced Data rates for GSM Evolution

2G network was expanded later to include improved data communication features using General Packet Radio Services (GPRS) and Enhanced Data rates for GSM Evolution(EDGE).

GPRS is a packet oriented mobile data service on GSM. When compared to conventional GSM, users of GPRS benefited from shorter access time and higher data rates. GPRS allows billing based on volume of data transferred. Although GPRS is a 'data only' technology, it helps to improve GSM's voice quality.

EDGE is a digital mobile phone technology that allows improved data transmission rates for GSM. EDGE is a superset to GPRS and can function on any network with GPRS deployed on it. It provides nearly three times faster speeds than the GPRS system. Both the phone and the network must support EDGE, otherwise the phone will revert automatically to GPRS.

ii. Code Division Multiple Access

In Code Division Multiple Access (CDMA) system, several transmitters can send information simultaneously over a single communication channel. CDMA provides wider coverage than GSM and provides better reception even in low signal strength conditions. The voice quality in CDMA is better than GSM. It has a signal with wider bandwidth and increased resistance to interference. CDMA technology provides better security to the mobile network when compared to GSM.

c. Third Generation networks

The Third Generation (3G) wireless network technology provides high data transfer rates for handheld devices. The high data transfer rates allow 3G networks to offer multimedia services combining voice and data. 3G is also referred to as wireless broadband as it has the facility to send and receive large amounts of data using a mobile phone. The access part in 3G networks uses WCDMA (Wideband Code Division Multiple Access). It requires upgrading the base stations (mobile towers) and mobile phones. Besides, the base stations need to be close to each other.

d. Fourth Generation networks

A Fourth Geneartion (4G) system, also called Long Term Evolution (L.T.E.), provides mobile ultra-broadband Internet access to mobile devices. 4G networks offer very high speeds and provides excellent performance for bandwidth intensive applications such as high quality streaming video. One of the key requirements for 4G is a wireless IP-based access system. The access part in 4G networks uses Orthogonal Frequency Division Multiple Access (OFDMA). 4G provides good quality images and videos than TV.

e. Fifth Generation networks

Fifth Geneartion (5G) is the next step in the evolution of mobile communication. It will offer faster, more number of connections, more energy-efficient and cost-effective data communication than its predecessors. 5G will be a key component that will help to realise the vision of unlimited access to information and sharing of data anywhere and anytime. 5G will also provide wireless connectivity for a wide range of new applications in wearable devices, smart homes, traffic safety/control, industry applications, etc. By 2020, it is thought that 5,000 crore to 1,00,000 crore devices will be connected to the Internet. 5G will be a wireless access solution which is aimed at addressing the demands and requirements of mobile communication beyond 2020.

11.2.2 Mobile communication services

Mobile communication industry uses a number of acronyms for the various technologies that are being developed. Here, we will discuss a few popular mobile communication technologies like SMS, MMS, GPS and smart cards.

a. Short Message Service

Short Message Service (SMS) is a text messaging service in mobile communication systems that allows exchanging short text messages. SMS is the most widely used data application by mobile phone users. The GSM standard allows to send a message containing upto 160 characters. When a message is sent, it reaches a Short Message Service Center (SMSC), which provides a 'store and forward' mechanism. SMSC attempts to send messages to the recipients. If a recipient is not reachable, the SMSC waits and then retries later. Some SMSC's also provide a 'forward and forget' option where transmission is tried only once and if it fails, the message is not sent again. SMS messages are exchanged using the protocol called Signalling System No. 7 (SS7).

The First SMS

The first SMS message was sent on 3rdDecember 1992 from a personal computer to a cellular phone on the Vodafone GSM network in the UK. The content of the message was 'Merry Christmas'.

b. Multimedia Messaging Service

Multimedia Messaging Service (MMS) is a standard way to send and receive messages that consists of multimedia content using mobile phones. It is an extension of the capability of SMS to send text messages. Unlike SMS, MMS does not specify a maximum size for a multimedia message. MMS supports contents such as text, graphics, music, video clips and more. An MMS server is responsible for storing and handling incoming and outgoing messages. Associated with the MMS server is the MMS proxy relay, which is responsible for transferring messages between different messaging systems.

c. Global Positioning System

The Global Positioning System (GPS) is a satellite based navigation system that is used to locate a geographical position anywhere on earth, using its longitude and latitude. GPS is designed and operated by the U.S. Department of Defence and it consists of satellites, control and monitoring stations, and receivers.

The basis of the GPS is a group of satellites that are continuously orbiting the earth. These satellites transmit radio signals that contain their exact location, time,

and other information. The radio signals from the satellites, which are monitored and corrected by control stations, are picked up by the GPS receiver. GPS receivers take the information transmitted from the satellites to calculate a user's exact location on earth. A GPS receiver needs only three satellites to plot a 2D position, which will not be very accurate. Ideally, four or more satellites are needed to plot a 3D position, which is much more accurate.

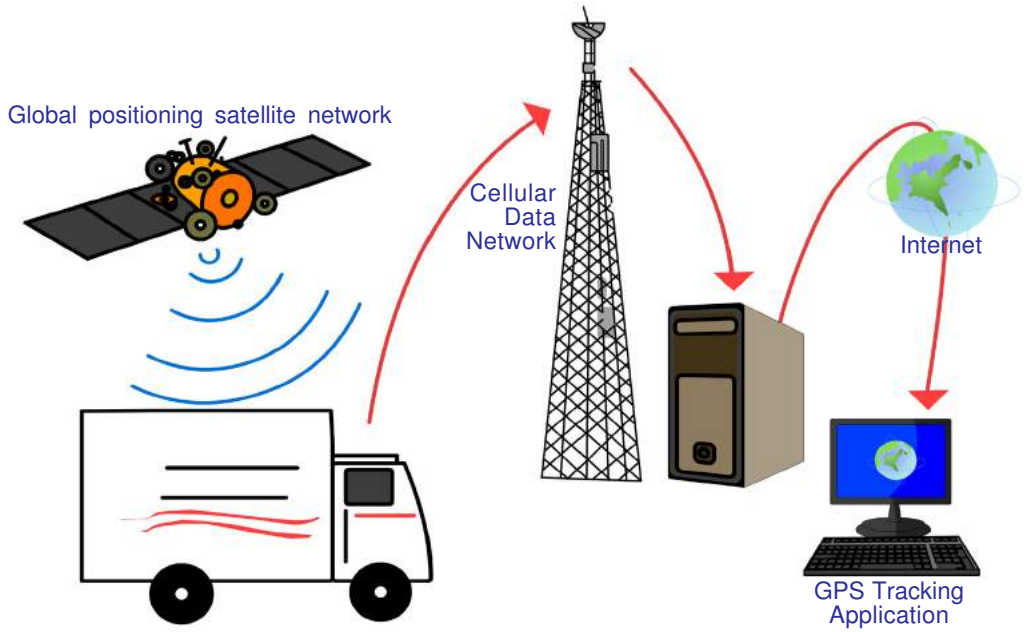


Fig. 11.1: GPS fleet tracking

GPS is used for vehicle fleet tracking by transporting companies to track the movement of their trucks. Figure 11.1 depicts the working of a truck tracking application using GPS. Vehicle navigation systems will direct the driver to his or her destination through the best route. In commercial aviation GPS is used for aircraft navigation. GPS is also used in oil exploration, farming, atmospheric studies etc. GPS receivers are now integrated in many mobile phones for implementing various tracking applications.

d. Smart cards

Let us recollect about smart cards and smart card readers that we discussed in Chapter 2 of Class XI. A smart card is a plastic card embedded with a computer chip / memory that stores and transacts data. The advantages of using smart cards is that it is secure (data is protected), intelligent (it can store and process data) and that it is convenient (it is easy

Fig. 11.2: A model smart card

to carry). That is why business firms and other organisations use smart cards for authentication and storing data.A model of smart card issued by the Government of India for RSBY scheme is shown in Figure 11.2.

In mobile communication, the smart card technology is used in Subscriber Identity Modules (SIM) for GSM phone systems (refer to Figure 11.3). The smart card is inserted or integrated into the mobile handset. The card stores personal subscriber information and preferences. SIM cards help to identify a subscriber, roam across networks and provide security to value added services like Internet browsing, mobile commerce, mobile banking, etc. Smart cards also work as credit cards, ATM cards, fuel cards, authorisation cards for television receiver, high-security identification cards, etc.

Fig. 11.3 : GSM SIM card

11.3 Mobile operating system

A mobile operating system is the operating system used in a mobile device (smart phone, tablet, etc.), similar to an operating system used in a desktop or laptop computer.

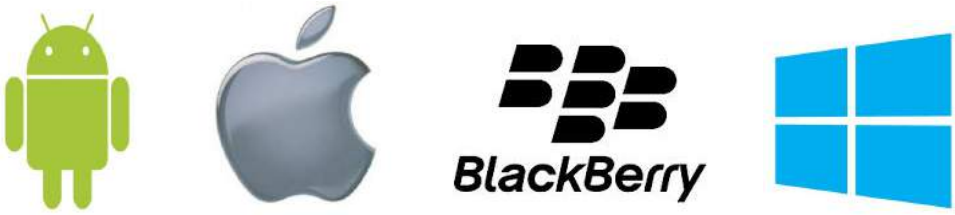


Fig. 11.4: Icons of popular mobile operating systems

A mobile OS manages the hardware, multimedia functions, Internet connectivity, etc. in a mobile device. It is the software platform on which other programs, called application programs, run. Popular mobile operating systems are Android from Google, iOS from Apple, BlackBerry OS from BlackBerry and Windows Phone from Microsoft.

Android operating system

Android is a Linux-based operating system designed mainly for touch screen mobile devices such as smart phones and tablet computers. It was originally developed by Android Inc. that was founded in Palo Alto, California in 2003 by Andy Rubin and his friends. In 2005, Google acquired Android Inc. making it a wholly owned subsidiary of Google. At Google, the team led by Rubin developed a mobile device platform powered by the Linux kernel. In 2007, the Open Handset Alliance, a consortium of several companies which include Google, HTC, Intel, Motorola, etc. was established with a goal to develop open standards for mobile devices.

Android was released along with the formation of the Open Handset Alliance (OHA).

The user interface of Android is based on touch inputs like swiping, tapping, pinching and reverse pinching to manipulate on-screen objects. Android allows users to customise their home screens with shortcuts to applications and widgets. Since its launch in 2007, the Android operating system's market share has been growing at a fast pace to make it the most widely used mobile operating system today. Major Android versions have been developed under a codename and released according to alphabetical order. Table 11.1 shows the Android version names.

The Android OS consists of a kernel based on Linux kernel. Android uses Linux kernel as it has a powerful memory and process management system, permissions-based security structure and open source nature. An Application Framework for developers to build applications using the Android Software Development Kit is available in Android. Android Software Development Kit can be used to develop applications like Google Maps, Facebook, etc. that run on Android.

Version	Code name
4.4	KitKat
4.1	Jelly Bean
4.0.3	Ice Cream Sandwich
3.1	Honeycomb
2.3	Gingerbread
2.2	Froyo
2.0	Eclair
1.6	Donut
1.5	Cupcake

Table 11.1: Android version names

The Android code is released under the Apache License. Apache licensing allows the software to be freely modified and distributed by device manufacturers and developers. Android has a large community of developers writing applications called 'apps' that enhance the functionality of devices. Apps are written using a customised version of the Java programming language.

The acceptance of Android is due to factors like its open source nature. This makes it easy for developers to make programs/applications for Android OS called Android apps. Most of these apps are available for free download from the Android Play Store. It enhances the popularity of this OS.

With remarkable advances in mobile hardware and software, these trimmed-down operating systems may be heading in a direction that could replace a desktop OS. It

is also likely that mobile OS could be further developed to be included in electronic devices like televisions, washing machines, watches, etc.

Let us do

Prepare a chart displaying the different mobile operating systems available in the market and their features.

Know your progress

1. SIM is
   a. Subscriber Identity Module
   b. Subscriber Identity Mobile
   c. Subscription Identification Module
   d. Subscription Identification Mobile
2. What is a GPS?
3. The protocol used to send SMS messages is _____.
4. Name the technology used to send multimedia content using mobile phones.
5. What are the functions of a mobile operating system?

11.4 ICT in business

In Chapter 11, IT Applications of Class XI, we discussed the advantages of using ICT in business. Here, we will discuss some tools which help to improve the way of doing business. ICT has become a vital part of the business of many companies. The use of ICT to improve the quality of the product or service given to customers is growing at a fast pace. The way people shop and the way the product reaches the customer have undergone remarkable changes. The Internet has changed the shopping habits of people. The major change in business is through the introduction of ICT in business. The use of ICT in all areas of business is shifting business from traditional shops to online retailers.

Marketing and sales have undergone a strategic shift. Customer preferences have an important role in today's business. Along with mass marketing, marketing specific to a customer has also gained prominence. Once shopping turned online, retailers could track not only what customers bought, but also what else they looked for; how they navigated through the site; how much they were influenced by promotions, social media reviews, and page layouts; and similarities across individuals and groups. This is achieved with the help of ICT.

Once a customer purchases a product, the product has to be billed and transported to his location. This has to be done with minimum cost and time. The use of ICT in transporting and tracking the product through its transit improved efficiency and thereby winning the confidence of the customers. We will now discuss some of the major developments in business through the use of ICT.

11.4.1 Social networks and big data analytics

Today before buying a product either online or from a nearby shop, people tend to search websites for their price, user reviews and for comparing similar products. They also search different social network groups and receive opinions from friends on a particular product. Therefore business firms have started watching the conversations and opinions posted in social media. Business firms are using the power of social media to gain a better understanding of their markets. They use this data from the social media and also from browsing patterns of users for their planning. The volume of such data is very large. This data that comes from posts to social media sites, digital pictures and videos, purchase transaction records, cell phone data, etc. which are related to a particular business can be considered as big data in business.

Analysis of large data is difficult. The data sources can be either structured from databases or unstructured from audio, documents, spreadsheets, social media posts, etc. Different software tools can be used to analyse big data, store it and take decisions in the business. Big data analytics is the process of examining large data sets containing a variety of data types to uncover hidden patterns, market trends, customer preferences and other useful business information. The analytical findings/reports can lead to refining marketing campaigns, new revenue opportunities, better customer service, improved operational efficiency, competitive advantages over rival organisations and other business benefits. A schematic diagram of big data analytics is given in Figure 11.5.

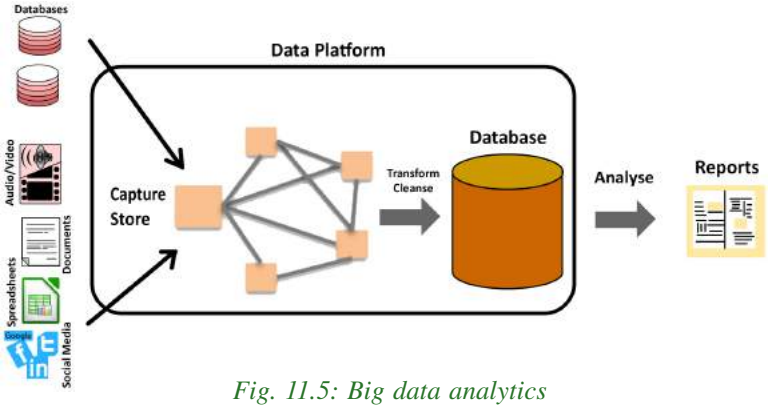


Fig. 11.5: Big data analytics

Big data analytics provide an opportunity for business firms to answer questions that were previously considered beyond their reach. This opened the doors to a new world of possibilities to interact effectively with the customer, with the knowledge of his preferences.

11.4.2 Business logistics

Business logistics is the management of the flow of goods/resources in a business between the point of origin and the point of consumption in order to meet the requirements of customers or corporations. The resources managed in logistics can include items such as food, products, animals, etc.

The objective of business logistics is ensuring the availability of the right product, in the right quantity and condition, at the right place and time for the right customer, at the right cost. Business logistics includes

i. purchase of materials from a company's suppliers
ii. transportation of those materials to the company's production facilities
iii. movement of finished goods through warehouses and transportation channels to customers.

It also involves the integration of information flow, inventory, packaging and security to products. The complexity of logistics can be reduced by the use of hardware and software.

Radio Frequency Identification (RFID) technology can be used to identify, track, sort or detect a wide variety of objects in logistics. RFID hardware consists of a RFID tag and a reader. The RFID tag consists of a combination of transmitter and responder. This tag contains a microchip for storing data and a sending and receiving antenna for data exchange with the RFID reader. These thin tags are inserted or pasted on product containers or products. Each RFID tag contains identifiable information about a product. In RFID, communication takes place between a reader and an RFID tag. The advantage of RFID tag is that the tag need not be in a straight line with the reader. Also, the reader can read the contents of a tag from a distance of several meters. An RFID tag is given in Figure 11.6.

Fig. 11.6: RFID tag

Tags can either be active, i.e., powered by battery, or passive, i.e., powered by the reader. Apart from logistics, RFID is gaining popularity in large supermarkets, payments in highway toll booths, etc. as an alternative to bar codes. RFID tags are also fixed on animals like tigers, lions, etc. in the forests, which makes their census easier.

RFID gives transportation and logistics operations increased visibility into product movement and business processes. It increases efficiency by providing real-time data that gives up-to-date information about the products. RFID based business

logistic softwares help to lower the operating costs, increase productivity in the distribution centers, maximize on-time deliveries and improve customer service and satisfaction.

11.5 Information security

Today, almost all activities that we perform in real life like communication, buying goods, banking, etc. can be achieved through the Internet. While executing all these activities, information is exchanged between computers. The security of information passed over Internet is a primary concern. In this section, we will discuss in detail about the cyber world and the various issues like copyright and trademark violations, cyber crimes, etc. that we come across while surfing the Internet.

11.5.1 Intellectual Property Right

Many people are engaged in creative work like music, literary work, artistic work, discoveries, inventions, designs and software development. These works are the creation of the mind. They demand a lots of hard work and time. The outcome of such work is called intellectual property. It is unjust to use the idea of a person without his permission. The person involved in developing such properties must get the benefits of it. So, it should be protected from unauthorised access.

The importance of intellectual property was first recognised in the Paris Convention for the Protection of Industrial Property in 1883 and the Berne Convention for the Protection of Literary and Artistic Work in 1886. Both treaties are now administered by the World Intellectual Property Organization (WIPO). WIPO was established in 1960 as an agency of United Nations (UN). It is dedicated to ensure that the rights of creators or owners of intellectual property are protected worldwide and the inventors and authors are recognised and rewarded by their creation. The logo of WIPO and its Indian counterparts is given in Figure 11.7.

Intellectual Property Rights (IPR) are similar to any other property right like right over land, house, etc. IPR refers to the exclusive right given to a person over the creation of his/her mind, for a period of time. IPR enables people to earn recognition and financial benefit from what they invent or create. IPR owners can disclose their creations in exchange for money. The company which receives the rights, markets and sells this innovation to the society. In this way the IPR owner, the company and the society benefit from the creation. IPR is encouraged by UN and almost all countries worldwide with the intention of encouraging innovations.

Fig. 11.7: Logo of WIPO

Each country has its own IPR registration system which is applicable to that country. WIPO serves as an international registration system for trademark, industrial design, etc. which is applicable to its member countries.

Intellectual property is divided into two categories - industrial property and copyright.

A. Industrial property

Fig. 11.8: Logo of IP India

Industrial property right applies to industry, commerce and agricultural products. It protects patents to inventions, trademarks, industrial designs and geographical indications. In India, industrial property rights can be registered with Controller General of Patents Designs and Trademarks under Ministry of Commerce and Industry. Logo of Intellectual Property India is given in Figure 11.8.

Patents: A patent is the exclusive rights granted for an invention. An invention means a new product or process (procedure) involving an inventive step, and capable of industrial application. It is the legal right obtained by an inventor for exclusive control over the production and sale of his/her mechanical or scientific invention for a limited period of time. To be patentable, an invention must:

- relate to a process or product
- be new
- involve an inventive step
- be capable of industrial use
- not be developed with the intention to harm others

Patent protection means that the invention cannot be commercially made, used, distributed or sold without the patent owner's consent. A patent provides the right to the patent owner to decide how the invention can be used by others. The owner can sell the right to the invention to someone else, who will then become the new owner of the patent. The term of every patent in India is 20 years from the date of filing of patent application. Once a patent expires, the protection ends and an invention can be used by the public freely. The details of the first patent filed for a zipper is given in Figure 11.9.

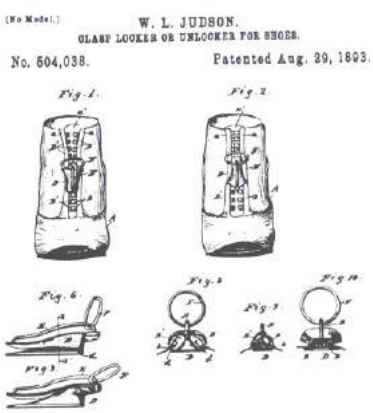

(No Model.)
W. L. JUDSON.
CLASP LOCKER OR UNLOCKER FOR SHOES.
No. 504,038. Patented Aug. 29, 1893.

Fig. 11.9: The first patent for zipper

Trademark: A trademark is a distinctive sign that identifies certain goods or services produced or

provided by an individual or a company. A trademark can be a name, logo, symbol, etc. that can be used to recognise a product or service. It provides protection to the owner of the trademark by ensuring the exclusive right to use it to identify goods or services. It helps consumers to identify and purchase a product or service. A trademark must be registered. The initial term of registration is for 10 years. Thereafter it can be renewed. In order to determine whether any person or company is using a particular trademark, a trademark search can be conducted through the trademark registry maintained by Controller General of Patents Designs and Trademarks (http://ipindiaonline.gov.in). The effect of a trademark registration is limited to that country. Some popular trademarks in India are given in Figure 11.10.

Fig. 11.10: Popular Indian trademarks

Industrial designs: An industrial design refers to the ornamental or aesthetic aspects of an article. A design may consist of three-dimensional features such as the shape, surface of an article or two-dimensional features, such as patterns, lines or colour. An industrial design right protects the visual design of objects that are not purely functional. Industrial designs are applied to a wide variety of industrial products and handicrafts like medical instruments, watches, jewellery, vehicles, textile designs, etc. The registered designs of Coca-Cola bottle and iPhone are given in Figure 11.11.

Fig. 11.11: Popular industrial designs

Geographical indications: Geographical indications are signs used on goods that have a specific geographical origin and possess qualities or a reputation that are due to that place of origin. Agricultural products typically have qualities that derive from their place of production and are influenced by factors like climate and soil. Place of origin may be a village or town, a region or a country. Some of the popular products with geographical indications related to Kerala are Aranmula Kannadi and Palakkadan Matta Rice (Figure 11.12).

Fig. 11.12: Geographical indications related to Kerala

B. Copyright

A copyright is a legal right given to the creators for an original work, usually for a limited period of time. Copyright applies to a wide range of creative, intellectual or artistic forms of works which include books, music, painting, sculpture, films, advertisement and computer software. This covers rights for reproduction, communication to the public, adaptation and translation of the work.

In India, the Copyright Act, 1957 came into effect from January 1958. This Act has been amended five times. Some of the important amendments to the Copyright Act in 2012 are extension of copyright protection in the digital environment, liabilities of Internet Service Providers, ensuring right to receive royalties for music composers and exception of copyrights for physically disabled to access any work.

Under Indian Copyright Act, a work is automatically protected by copyright when it is created. The general rule is that the copyright lasts for 60 years after the death of the last surviving author. Registration of copyright gives a legal status to a creative work . This makes it an intellectual property, giving exclusive legal right over the creation. It should be noted that it is not necessary to register to get the copyright. Copyright registrations in India are handled by the Copyright Office under the Ministry of Human Resource Development. The logo of Indian Copyright Office is given in Figure 11.13.

Fig. 11.13: Logo of Copyright Office, India

Table 11.2 shows the different symbols used to specify the registration status of intellectual property.

INTELLECTUAL PROPERTY	
Registered trademark	®
Unregistered trademark	™
Copyright	©
Sound-recording copyright	!

Table 11.2: Symbols used for intellectual property

The copyright holders of a work can authorise or prohibit

- its reproduction in all forms, including print form and sound recording;
- its public performance and communication to the public;
- its broadcasting;
- its translation into other languages;
- its adaptation, such as from a novel to a screenplay for a film.

Creators often sell the rights of their works to individuals or companies for financial benefit.

Computer software (source code, databases, websites, etc.) can be copyrighted as a literary work. Even though computer software is protected under copyright, nowadays they are increasingly getting patented. This is because software development is considered as an industry and patents provide better protection when compared to copyright. It should be noted that the criteria required for obtaining patent protection is more stringent.

Table 11.3 displays the rights and the related intellectual property. The differences in the registration of intellectual property rights are shown in Table 11.4.

Intellect	Property	Right
Idea	Invention/innovation	Patent
Idea	Quality + identity	Trademark
Idea	Appearance	Design
Goods	Geographical origin	Geographical Indication
Idea	Expression	Copyright

Table 11.3: Rights related to intellectual property

	Patent	Trademark	Copyright
Refers to	Product, Process	Name, Logo, Symbols	Creative, intellectual or artistic forms of work
Registration	Required	Required	Automatic, can be registered
Duration	20 years	10 years	Until 60 years after the death of the last surviving author
Renewable	No	Yes	

Table 11.4: Differences in the registration of rights for intellectual property

11.5.2 Infringement

Unauthorised use of intellectual property rights such as patents, copyrights and trademarks are called intellectual property infringement. It may be a violation of civil law or criminal law, depending on the type of intellectual property, jurisdiction (countries) and the nature of the action.

Patent infringement is caused by using or selling a patented invention without permission from the patent holder. Many countries allow using patented inventions for research purpose. The legal dispute between Apple and Samsung over their mobile phone technologies is an example of patent infringement.

Trademark infringement occurs when one party uses a trademark that is identical to a trademark owned by another party, where both the parties use it for similar

products or services. It is better to register the trademarks to get the legal advantages. An example of trademark infringement is the legal suit where a small company manufactured toffees under the trademark 'HORLIKS'. This violated the trademark rights enjoyed by 'HORLICKS'.

Copyright infringement is reproducing, distributing, displaying or adaptating a work without permission from the copyright holder. It is often called piracy. Software piracy is the illegal copying, distribution, or use of software. Music piracy is the copying and distributing of a piece of music without the consent of the composer, artist or the copyright holding music company. Enforcement of copyright is generally the responsibility of the copyright holder.

Know your progress

1. Creation of the mind is known as _____.
2. Expand WIPO.
3. Intellectual property rights are categorised into _____ and _____.
4. Patents are exclusive rights given to _____.
5. _____ is a sign used to recognise a product or service.
6. What is industrial design?
7. What is the importance of geographical indicators?

11.5.3 Cyber space

Cyber space is a virtual environment created by computer systems connected to the Internet. It is the term used to refer to a simulated world, where we can do many things at one place. Let us now discuss various instances where cyberspace has influenced our lives.

Formerly, communication was mainly dependent on postal service. Nowadays e-mail has gained wide acceptance and legal validity for communication. In the Department of Higher Secondary Education (DHSE), all communications to higher secondary schools related to admissions, examinations, National Service Scheme, administration, etc. are done through *website/ e-mail*. Today, Internet has become a popular medium for communication among common people.

In December 2012, a girl student was subjected to inhuman physical torture in a moving bus in a metro city in India. After a few days the girl succumbed to her injuries. This incident invoked protests among people from various walks of life. Social media like facebook, twitter, etc. became a platform for the protest. These

people organised a movement in support of the girl through messages over the social media. Thousands of people gathered in the streets of Delhi with lighted candles as a protest. Taking this incident into account, this incident the Loksabha passed 'The Criminal Law (Amendment) Act, 2013', which strictly deals with sexual assault cases. Such movements organised over *social media,* show its influence on society.

People purchase mobile phones, shoes, apparels, etc. by visiting showrooms. Today, many people prefer to purchase these goods online from e-commerce websites. The *e-commerce* sites provide catalogs which display different search options like price range, brands, etc. about the product. This make shopping easier, from the comfort of home and saves time too. e-Commerce requires payments through credit cards, debit cards, Internet banking, etc. Nowadays more and more people are attracted to online buying and selling. Figure 11.14 shows e-Commerce websites.

Fig. 11.14 : Sample e-commerce websites

Almost all banks offer *Internet banking* facilities to its customers. Through this facility we can transfer funds, pay telephone bills, electricity bills, payments for online purchases, train tickets, cinema tickets, etc. This saves a lot of time and effort when compared to performing these activities by actually visiting the respective offices. Hence banking is also a popular service over Internet. An Internet banking website is displayed in Figure 11.15.

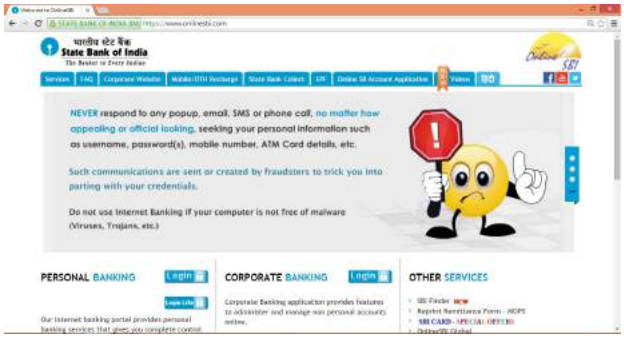


Fig. 11.15 : An Internet banking website

With the passage of time, more services are going online. Many people spend a considerable amount of time surfing the Internet to avail of various services in the web. Internet is often referred to as cyberspace. Figure 11.1**6** shows a symbolic representation of cyberspace.

Cyberspace is an unreal world in which communication over computer networks occurs. It is an information superhighway where individuals gather information, interact, exchange ideas, provide social support, conduct business, play games, engage in discussions and so on.

Fig. 11.16 : Symbolic representation of cyberspace

Cyberspace is a space where social interactions dominate. Some people consider cyberspace as an uncontrolled and unregulated electronic space where anyone is free to do anything as they wish and express themselves freely. Such acts of people affect badly or influence negatively many others. Hence, when an individual works on the Internet, they have to follow some rules and ethics which are beneficial to all users.

This unregulated space also provides room for criminals. Since activities like communication, financial transactions, etc. in the cyberspace increase day by day, it has opened a new medium for criminal activities. Today cyberspace security has become a serious concern.

'Cyberspace' is a term coined by William Gibson, a Canadian science fiction writer, in his short story 'Burning Chrome' (1982). He used it to represent a world where events or transactions that are not in real world occur. He believed that cyberspace is a consensual hallucination. According to Gibson, consensual hallucination is the nature of existence within cyberspace.

11.5.4 Cyber Crimes

We know that the Internet has opened new opportunities in entertainment, business, communication education, sports, etc. At the same time, it is a reality that some people use the Internet for committing illegal activities.

The increasing use of the Internet through smart phones and tablets for online banking and other financial transactions has increased the risk. Rising Internet penetration and online banking provides opportunities for cyber criminals to target online financial transactions, using malicious software (malware) or through illegal hacking. The statistics of cyber crime committed in the recent years according to National Crime Records Bureau is given in the graph shown in Figure 11.17. This graph shows the significant rise of cyber crimes in India through the years.

Cyber Crime is defined as a criminal activity in which computers or computer networks are used as a tool, target or a place of criminal activity. The victims of cyber crime lose money, reputation and face mental trauma.

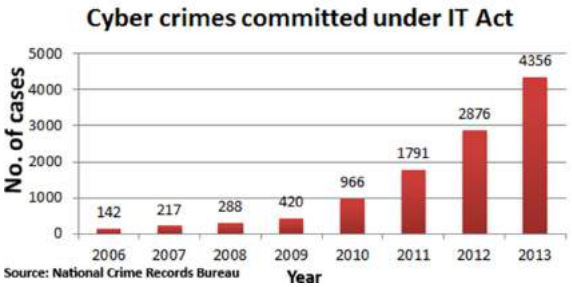


Fig. 11.17 : Cyber crime statistics in India

An important aspect of cybercrime is its nonlocal character. A crime can occur in jurisdictions separated by vast distances. That is an attacker may operate from a country and attack a destination in some other country. So the investigating team and the judiciary of different countries must keep hand in hand. Due to the anonymous nature of the Internet, it is possible for people to engage in a variety of criminal activities. People commit cyber crimes knowingly or unknowingly.

Cyber crimes include phishing, hacking, denial of service attacks, etc. which we learned in Chapter 9 of Class XI. Computer crime mainly consists of unauthorised access to computer systems, credit card frauds, illegal downloading, child pornography, cyber terrorism, creation and/or distribution of viruses, spam and so on.

Cyber crimes can be basically divided into 3 major categories: They are cyber crimes against individuals, against property and against government.

A. Cyber crimes against individuals

The popularity of broadband Internet increased the dependence of Internet for daily activities. This increased the risk of online crimes. An act of a person in cyberspace that causes physical or mental trouble to an individual is referred as cyber crime. Harassment, assuming someone's identity, impersonation and violation of privacy are some examples of cyber crimes.

i. Identity theft: Identity theft occurs when someone uses another person's identifying information, like their name, credit card number, etc. without their permission to commit fraud or other crimes. It is a form of stealing a person's identity, by which someone pretends to be someone else, to gain access to resources like bank account, social media accounts, etc. This is done with the intention of transferring money from the victim's bank account, payments for purchases, defaming the person through social media, etc. This is done with the intention of obtaining credit from financial institutions or gaining benefits in that persons name.

Before transferring money from bank accounts, the thief may change the mailing address of the account so that the victim will not be able to realize that money is withdrawn. The stolen personal information may be used to open new accounts, open new credit card accounts, to apply for a mobile phone connection and more.

ii. Harassment: Posting humiliating comments focusing on gender, race, religion, nationality at specific individuals in chat rooms, social media, e-mail, etc. is harassment. The use of vulgar or indecent language, threatening any illegal or immoral act through a computer or a computer network is considered as harassment.

The use of the Internet, to harass someone is called cyber stalking. We might have read about statements from celebrities that the facebook accounts in their names are fake. This is because such accounts may contain pictures or posts with defaming content. These profiles might be created using the photographs and personal information of the celebrity by people with criminal intentions. There are people who defame others by sending humiliating e-mails, facebook posts, etc. All these amount to harassment. Common characteristics of cyber stalking include threats, false accusations, monitoring, identity theft and data destruction or manipulation. Cyber stalking also includes sexual or other exploitation of minors. It can destroy friendships, careers, self-image and confidence.

iii. Impersonation and cheating: Impersonation is an act of pretending to be another person for the purpose of harming the victim. There are people who utilise the anonymity of the Internet to commit impersonation online.

Sometimes we may receive e-mails seeking help for transferring large amounts of money from a distant country to India. Usually, the sender states that this money is in the form of an asset (land, gold, etc.) that is to be sold. To dispose the asset and complete the legal formalities, the sender requires some amount of money. The mail requests us to share a portion of the expenses and offers us up to 50% of the value of the asset. After receiving the cheque or money order, the sender tells that due to some complications more money is required. In this way the victim may lose large amounts. We often receive similar mails with a different story. Mailing scams like this are examples of Internet fraud/cheating.

Similar crimes occur in online auction also. In some cases, individuals announce products for sale on Internet auction sites. They demand money before the delivery of the item and never carryout their order.

iv. Violation of privacy: Violation of privacy is the intrusion into the personal life of another, without a valid reason. This gives the person whose privacy has been invaded, the right to file a lawsuit for damages against the person/organisation that intruded. It consists of distributing private information like personal data, photography, workplace monitoring videos, etc.

You might have heard of issues regarding the use of hidden cameras, mobile cameras, etc. to capture images of women in public places. Photography of any person (men or women) without his/her permission amounts to violation of privacy. Posting images of others in social media, transferring them to others through e-mail/copying, etc. without their permission are considered as violation of privacy.

v. Dissemination of obscene material: The Internet has provided a medium for the facilitation of crimes like pornography. The distribution and posting of obscene material is one of the important cyber crimes today. Pornography on Internet may take various forms. It may include hosting website containing prohibited materials, use of computers for producing obscene material, downloading obscene materials through the Internet, etc. These obscene content may misguide adolescents.

Most of the cyber crimes occur without the knowledge of the victim, whereas in some others, the victim also takes part in crime to become rich easily. It should be noted that various types of preventive mechanisms have been installed to avoid such crimes. e-mail service providers use spam filter that filters unwanted mails. Different types of authentication mechanisms that exist for online financial transactions prevent fraud to a certain extent. Above all, an individual has to be careful while using the Internet.

B. Cyber crimes against property

Cyber crimes against all forms of property like credit cards, intellectual property, etc. are termed as cyber crime against property. These crimes include hacking, piracy, computer vandalism (destruction of others property), unauthorised intrusion through cyberspace, unauthorised possession of information stored in computers, etc. Some classifications of cyber crimes against property are given below.

i. Credit card fraud: Credit card fraud involves an unauthorised usage of another person's credit card information for the purpose of payments for purchases or transferring funds from it. There are instances where the web servers of large organisations are hacked and credit card information of a large number of people is stolen by Internet thieves. They use this information to make payments or they sell this information to other fraudsters over Internet for a small price.

ii. Intellectual property theft: The infringement of IPR's come under this category. Violation of copyright, patent, trademark, etc. are intrusions against property.

Recently, an Indian IT company developed a software for correcting errors in program code. One of the employees of this company copied this software in a CD and tried to sell it to a competitor for a big price. This led to huge financial and property loss to the company. This is considered as theft of intellectual property. Software piracy is also a crime under cyber law.

Nowadays intellectual property theft has become common. Information about any topic is now freely available on the Internet. Copying of an other person's language, thoughts, ideas or expressions and presenting them as one's own original work is called plagiarism. There are stringent copyright laws available in India which protect the intellectual property rights. Plagiarism can be easily detected through various tools available online.

iii. Internet time theft: Today almost all modems/routers have wireless Internet facility. They provide sharing of Internet at homes, schools, business establishments, etc. If this is not properly secured using passwords, other people may use our Internet. The usage of the Internet time by unauthorised persons, without the permission of the person who pays for it is called Internet time theft. This leads to loss of Internet time and money. Above this, others may use our Internet account for committing crimes, for which we may also be held responsible.

The different types of attacks like viruses, worms, man in the middle attack, etc., discussed in Chapter 9 of Class XI also fall under cyber crimes against property.

C. Cyber crimes against government

Increasing popularity of e-governance has made governments vulnerable to cyber attacks. The various governmental computer networks and websites are vulnerable to risks and threats of cyber crimes. The different categories of cyber attacks against government are cyber terrorism, website defacement and e-governance denial attack.

i. Cyber terrorism: Cyber terrorism is a cyber attack against sensitive computer networks like nuclear power plants, air traffic controls, gas line controls, telecom, etc. These types of attacks against governments are increasing globally. Cyber terrorism focuses on the use of the Internet by anti nationals to affect a nation's economic and technological infrastructure.

In 2010, a computer virus called tuxnet was used to carry out an invisible attack on Iran's secret nuclear programme. This virus was aimed at disabling Iran's Uranium enrichment programme. This virus infected a large number of nuclear controls and gave false instructions leading to nuclear malfunctions and break down.

Cyber terrorism may prove to be very costly to a country. Therefore governments provide very powerful security mechanisms for their servers.

ii. Website defacement: This is a common cyber attack against a government. Defacement of websites includes hacking of government websites and posting derogatory comments about a government in those websites.

iii. Attacks against e-governance websites: These types of attacks deny a particular online government service. This is done using a Distributed Denial of Service (DDoS) attack which we discussed in Chapter 9 of Class XI. In another case the website may be hacked and controlled by the hackers. They gain access to website administration through content management system and destroy the data. This causes huge loss to the government.

Cyber terrorism has increased in the modern world because of the anonymous nature of cyberspace. The availability of a variety of targets that affect a large number of people, and the fact that attacks can be conducted from a distance have increased terrorism through cyberspace.

11.5.5 Cyber ethics

The available statistics shows that the volume of cyber crimes is increasing rapidly all over the world. Internet crime has many faces and is committed in diverse ways. Awareness is the first step in protecting ourselves, our family and our business. We should ensure that our actions in the cyberspace do not harm others. We should also remember that our actions over the Internet are being monitored by several others.

Suggestions given below can be considered as ethical practices over the Internet.

- Use anti-virus, firewall, and spam blocking software for your PC.
- Ensure security of websites (https and padlock) while conducting online cash transactions.
- Do not respond or act on e-mails sent from unknown sources.

- Use unique and complex passwords for accounts and change your passwords on a regular basis. (Should have a minimum of 8 characters, contain alphabets, numbers and special characters)
- Do not select the check boxes or click OK button before reading the contents of any agreement/message.
- Avoid the use of unauthorised software.
- Do not hide your identity and fool others.
- Do not use bad or rude language in social media and e-mails.
- Remove the check mark against 'Remember me' before logging into your account using computers other than your personal ones.

Let us do

- Conduct an awareness program about various cyber crimes.
- Prepare a chart listing how we can protect ourselves in cyber space.

Know your progress

1. What do you mean by cyberspace?
2. Criminal activity using computer, mobile phone and Internet is termed as _____.
3. What is cyber crime against a person?
4. Cyber terrorism is a type of cyber crime against _____.
5. Stealing one's information such as username and password is _____.

11.5.6 Cyber laws

The term cyber law in general refers to the legal and regulatory aspects of the Internet. Cyber law can be defined as a law governing the use of computers and Internet.

We have already discussed that there are different kinds of cyber crimes. Criminal activities such as theft, fraud, forgery and defamation are traditional in nature and are subject to the Indian Penal Code. The abuse of computers (cyber crime) has become wide spread in India which is addressed by the Information Technology Act, 2000 and IT Act Amendment Bill 2008.

Cyber law is important because it touches almost all aspects of transactions and activities using Internet. Whether we realise it or not, every action and every reaction in cyberspace has some legal perspectives.

11.5.7 Information Technology Act 2000 (Amended in 2008)

The Information Technology Act, 2000 is India's legislation regulating the use of computer, servers, computer networks, data and information in electronic format. The legislation has touched various aspects related to authentication, digital signature, cyber crime and liability of network service provider. The acts aim to provide legal recognition for transactions carried out by means of electronic data interchange and other means of electronic communication. IT Act allows using alternatives to paper based communication and facilitates electronic filing of documents with government agencies. It gives legal acceptance for electronic communication. It also addresses offenses and disputes in the cyberspace and provides justice to victims of cyber crimes.

IT Act aims to provide legal infrastructure for e-commerce in India. This Act was developed to promote IT industry, regulate e-commerce, facilitate e-governance and prevent cyber crimes. It promotes security practices within India that would serve the country in a global context. Under the Act certain violations are treated as serious crimes and offenders are liable to penal actions. Therefore, it is important to understand the various perspectives of the IT Act, 2000 and what it offers.

In May 2000, the Indian Parliament passed the Information Technology Bill and it is known as the Information Technology Act, 2000. The Act was amended by the Information Technology Amendment Bill 2008, which was passed in December 2008. This amendment accommodates further development of IT and related security concerns, since IT Act 2000 was passed. Compensation has to be given to affected persons if damage is done to the computer system or computer network by the introduction of virus, denial of service, etc. Sections 65-74 of the Act specifically deals with cyber crimes.

SOME COMMON OFFENCES AND THEIR PUNISHMENTS NOTED IN IT ACT 2000 AND IT AMENDMENT 2008

Section	Offences	Punishments
65	Tampering with computer source code	Imprisonment up to three years, or with fine which may extend up to two lakh rupees, or with both.
66	Computer related offences	Imprisonment for a term which may extend to two three years or with fine which may extend to five lakh rupees or with both.
66B	Punishment for dishonestly receiving stolen computer resource or communication device.	Imprisonment for a term which may extend to three years or with fine which may extend to rupees one lakh or with both.

Section	Offences	Punishments
66C	Punishment for identity theft.	Imprisonment for a term which may extend to three years and shall also be liable to fine which may extend to rupees one lakh.
66D	Punishment for cheating by personation by using computer resource.	Imprisonment for a term which may extend to three years and shall also be liable to fine which may extend to one lakh rupees.
66E	Punishment for violation of privacy.	Imprisonment which may extend to three years or with fine not exceeding two lakh rupees, or with both.
66F	Punishment for cyber terrorism.	Imprisonment which may extend to imprisonment for life.
67	Punishment for publishing or transmitting obscene material in electronic form.	Imprisonment for a term which may extend to two to three years and with fine which may extend to five lakh rupees. In the event of a second or subsequent conviction with imprisonment for a term which may extend to five years and also with fine which may extend to ten lakh rupees.
67A	Punishment for publishing or transmitting of material containing sexually explicit act, etc. in electronic form.	Imprisonment for a term which may extend to five years and with fine which mayextend to ten lakh rupees.
67B	Punishment for publishing or transmitting of material depicting children in sexually explicit act, etc. in electronic form.	Imprisonment for a term which may extend to five years and with a fine which may extend to ten lakh rupees. In the event of second or subsequent conviction with imprisonment for a term which may extend to seven years and also with fine which may extend to ten lakh rupees.
Punishment for sending offensive messages through online communication service like e-mail, social media, SMS, etc. will be according to the sections in Indian Penal Code (IPC).		

11.5.8 Cyber Forensics

The increase in computer crimes affects our daily lives and national security. Internet and digital technologies bring us a lot of convenience. But at the same time they also provide criminals more chance to commit crimes. Traditional law enforcement tools and methodologies do not successfully address the detection, investigation and prosecution of cyber crime.

Forensics is the process of using scientific knowledge for identifying, collecting, preserving, analyzing and presenting evidence to the courts. Cyber forensics can be defined as the discipline that combines elements of law and computer science to collect and analyze data from computer systems, networks, communication systems and storage devices in a way that is admissible as evidence in a court of law. The goal of computer forensics is to analyse data in a way that preserves the integrity of the evidence collected so that it can be used effectively in a legal case.

11.5.9 Infomania

Information is considered as the key to success. It has to be collected, managed and processed well. But what happens if a person gets overloaded with information? Infomania is the state of getting exhausted with excess information. It occurs due to accumulation of information from many sources like Internet, e-mail and cell phones, but cannot be processed. While collecting information, its quality and relevance are also to be considered. Infomania is the excessive enthusiasm for acquiring knowledge. This may result in neglecting the more important things like duties, family, etc. We may see people browsing for information during dinner. It is now treated as a psychological problem. Constantly checking e-mails, social networking sites, online news, etc. are the symptoms of infomania. Many people do this to keep themselves up to date with the fear of being out of the group.

Studies prove that people addicted to infomania lose concentration and sleep. Excessive use of technology reduces intelligence. Some people give high priority to respond to incoming messages through mail and social media. These people may skip or interrupt their important family or professional engagements to answer an e-mail or a social media post.

To get away from such situations, users must learn to log off social media, online news and discussion groups while they are at work or spending time with family.

Know your progress

1. The year in which IT Act came into existence in India is _____.
2. What is the purpose of IT Act?
3. What do you mean by cyber forensics?
4. _____ is the excessive enthusiasm for acquiring knowledge.

Let us conclude

The convergence of a mobile phone and a computer has shifted the focus from desktops to mobile computing devices. These devices are always connected to the Internet and therefore mobile communication technology has gained importance. The mobile communication technology has evolved through generations from 1G to 4G, the prime focus being speed. This also has led to the development of features like SMS, MMS, GPS, etc. Android is one of the popular mobile operating systems, developed by Google based on Linux kernel. The advances in the development of this OS are in such a way that it will soon replace desktop operating systems and could be used in devices like television, washing machines, etc. A person can do a variety of jobs through cyberspace. Due to its anonymous nature, cyber space has become a venue for criminals. Crimes can be against a person, property or government. The best way to be safe is to be aware of the traps in the Internet. The rights of the creator or owner of music, software, artistic works, inventions, etc. are protected by Intellectual Property Rights. There are cyber laws which provide the framework for a secure environment in cyberspace. India has a robust IT Act. It addresses the different crimes committed using computers, Internet and mobile phones. Cyber forensics has emerged as a part of legal proceedings of almost all civil and criminal cases. Any crime committed using IT infrastructure leaves a digital evidence that makes it easily detectable. While using Internet, ensure that our actions do not harm others. The term Infomania is used for addressing the problems created by information overloading.

Let us assess

1. What is mobile computing?
2. Explain generations in mobile communication?
3. Compare GSM and CDMA standards.
4. Write short notes on SMS.

5. Differentialte GPRS and EDGE?
6. _____ is a standard way to send and receive message with multimedia content using mobile phone.
7. Expand GPS?
8. What is a smart card? How is it useful?
9. Explain the features of Android OS.
10. What is cyberspace?
11. Why is cyberspace called a virtual world?
12. Phishing is an example of _____.
13. Explain different categories of cyber crimes in detail.
14. "Awareness is the best way to protect ourselves in cyberspace". Comment.
15. How do trademark and industrial design differ?
16. What is copyright? How does it differ from patent?
17. Explain the exclusive right given to the owner by IPR?
18. What is piracy?
19. What do you meant by infringement?
20. Why is a cyber law important?
21. "Non-local nature of cyber crime creates problems to the investigators". Explain.
22. "Infomania has became a psychological problem". Write your opinion.
23. What do you mean by big data in business? Explain big data analytics.
24. What do you mean by business logistics?
25. How does RFID improve the way business is done?

References

- Stroustrup, B. (2013). *The C++ Programming Language*. New Delhi : Addison-Wesley Professional
- Lafore, R. (2009). *Object-Oriented Programming in C++*. Chennai : Sams Publishing
- Balagurusamy, E. (2008). *Object Oriented Programming with C++*. New Delhi: Tata McGraw-Hill Education
- Powel, T. A. (2010). *The Complete Reference : HTML & XHTML*. New Delhi: OSborne/Tata MC Graw-Hill
- Lloyd, I. (2008). *The Ultimate HTML Reference*. Melbourne : Sitepoint
- Holzner, S. (2000). *HTML Black Book*. New Delhi : DreamTech Press
- Frain, B. (2012). *Responsive Web Design with HTML5 and CSS3*. Mumbai : Packt Publishing
- Powel, T. A. & Schenider, F. (2008). *The Complete Reference JavaScript*. New Delhi : Tata McGraw-Hill
- Zakas, N. C. (2012). *Professional JavaScript for Web Developers*. Birmingham : Wrox
- The Information Technology (Amendment) Act, 2008 : The Gazette of India, No.13, Feb 5, 2009
- National Crime Records Bureau. www.ncrb.gov.in, Accessed on 31st March 2015
- Leon, A. (2010). Enterprise Resource Planning. New Delhi : Tata McGraw - Hill