1. Introduction to C

Objectives of this lesson

1.1 **Introduction**:

C was specifically developed to write the operating system UNIX. UNIX was introduced by Bell laboratories in 1969. The original version of Unix was written in assembly language but the programs written in assembly language were less portable then the programs written in high level language. So the people at AT & T decided to re-write the operating system in high level language.

During the same period Kenneth Thompsom developed a language for systems programming which was named as "B" after its parent language BCPL. BCPL was developed by Martin Richards in 1967.

In 1972, "C" made its first appearance as an improved version of "B". It was developed by Dennis Ritchie. The name "C" derived from "B" as the letter C follows the letter B in the name BCPL. C is standardized by ANSI (American National Standards Institute) and ISO (International Standards Organization). These versions are known as ANSI C and ISO C. In the late seventies C began to replace the more familiar languages of that time like PL/I, ALGOL, etc. It wasn't made the 'official' Bell Labs language. Thus, without any advertisement C's reputation spread and its pool of users grew. Ritchie seems to have been rather surprised that so many programmers preferred C to older languages like FORTRAN or PL/I, or the newer ones like Pascal and APL. C seems so popular because it is reliable, simple and easy to use.

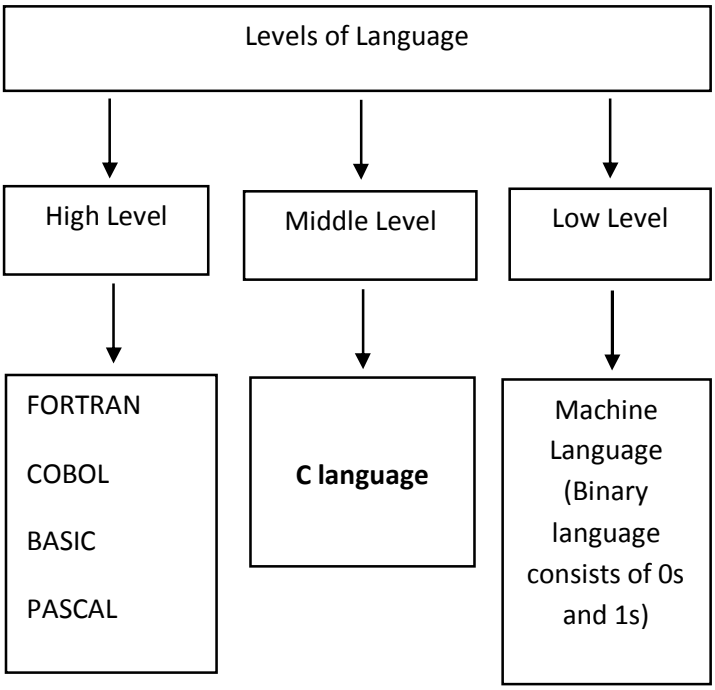


Fig: 1.1 Levels of language

1.2 Distinctive Features of C Language:

C is the most popular programming language, C has many advantages:

1. **General purpose programming language: I**t is general purpose programming language. It is usually called "system programming language" but equally suited to writing a variety of applications .C can be used to implement any kind of applications such as math's oriented, graphics, business oriented applications.
2. **Middle Level:** As a middle level language it bridges elements of high level language with the functionality of assembly language. C combines both the advantages of low level and high level languages. (arrays,pointers etc).
3. **Structured Programming:** C is very suited for structured programming. The programmers can easily divide a problem into a number of modules or functions.
4. **Simplicity:** C is simple to use because of its structured approach. It has a wide collection of inbuilt functions, keywords, operators and data types.
5. **Portability:** we can compile or execute C program in any operating system (unix,dos,windows). This refers to the ability of a program to run in different environments. With the availability of compilers for almost all operating systems and hardware platforms, it is easy to write code on one system which can be easily ported to another.
6. **Modularity:** modularity is one of the important characteristics of C. We can split the C program into no. of modules with the help of functions which help in increasing understanding of the programs, instead of repeating the same logic statements (sequentially). It allows reusability of modules.

7. **Efficient Compilation and Execution:** The process of compilation and execution of programs is quite fast in C Language as compared to other languages like BASIC (Beginner's All purpose Symbolic Instruction Code) and FORTRAN (FORmula TRANslator).
8. **Clarity:** The features like keywords, in-built functions and structures help to improve the clarity and hence understanding of the program.
9. **High Availability:** The software of C Language is readily available in market and can be easily installed on the computer.
10. **Easy Debugging:** The syntax errors can be easily detected by the C compiler. The error is displayed with line number of the code and the error message.
11. **Powerful programming language:** C is very efficient and powerful programming language, it is best used for data structures and designing system software.
12. **C is case sensitive language.** Case sensitive means upper-case letters and lower-case letters are different in C language.
13. **Memory Management:** Various memory management in-built functions are available in C language which helps to save memory and hence improve efficiency of the program. e.g. malloc(),alloc() and calloc().
14. **Rich set of Library Functions:** C has a rich set of library functions. These are the functions that provide readymade functionality to the users. It also supports graphic programming.

1.3 Character set of C Language:

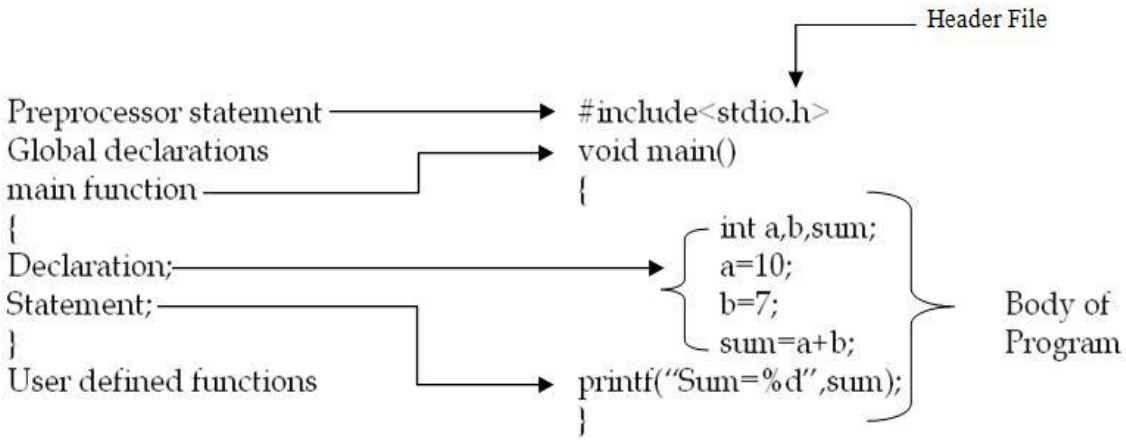
A character denotes any alphabet, digit or special symbol used to represent information. C compiler allows using many no. of characters:

1. Lower case letters : a b c … z
2. Upper case letters : A B C.…Z
3. Digits : 0 1 2 3…9
4. Special characters : + - * ( ) & % $ # { } [ ]' " : ; etc.
5. White space characters : blank, new line, tab space etc.

1.3.1 Escape Sequences: Escape sequences are used in input and output functions such as printf and scanf. This has '\' followed by a character. Following escape sequences are used in C language to specify actions such as carriage returns and tab movements:

1. ' \n ' new line
2. ' \t ' horizontal tab
3. ' \v ' vertical tab
4. ' \b ' back space
5. ' \? ' question mark
6. ' \" ' double quote

1.4 Structure of C Program:



1.4.1 Structure describes the way in which a program can be written, c structure includes:

1. **preprocessor statements:** these must be at the beginning of the program. It includes "include statements", user defined macros, to be processed before the actual program. These can be followed by #(hash) character.
2. **Header files:** Files that are placed at the header before main () of a C program are called header files normally having .h extension. It starts with # symbol. It is instructions to the compiler to include header files into the source program via # include statement. These statements direct the preprocessors to include header files and symbolic constants into a C program for e.g. # include<stdio.h>, # include<conio.h>, #include<math.h>, #include <string.h>
3. **Function prototypes:** function prototypes explain the nature of the function which includes return type, function name and list of the arguments type.
4. **Main function:** Every C program must have one necessary function, that is main(). Only statements in the main() function can be executed.
5. **Global declaration:** Variables or functions, whose existence is known in the main function and other user defined functions, are called global variables, and their declarations are called global declarations. It allows to declare variables as global whose scope is valid throughout the program.
6. **Braces:** Every C program uses a pair of curly braces ({,}). The left brace indicates the beginning and right brace indicates the end of main or a user defined function.
7. **Declarations:** It is a part of the C program where all the variables, arrays, functions etc., may be initialized with their basic data types.
8. **Statements:** These are instructions for specific operations. They may be I/O statements, arithmetic statements and other statements.

9. **Functions (sub program):** a reusable block of statements that gets executed upon calling, a program have any no. of function which contain local declarations and code.
10. **Local declarations:** every function at it's beginning may have local variables whose scope is valid with in the function only.
11. **Code:** code is some logical statements written according to the syntax of C to solve a problem.

1.4.2 Let us now try to understand the structure of C program with a simple program in C Language to add two numbers. But before we begin with our first C program we must follow some rules that are applicable to all C programs:

1. Each instruction in a C program is written as a separate statement. Therefore a complete C program would comprise of a series of statements.
2. The statements in a program must appear in the same order in which we wish them to be executed; unless of course the logic of the problem demands a deliberate 'jump' or transfer of control to a statement, which is out of sequence.
3. Blank spaces may be inserted between two words to improve the readability of the statement. However, no blank spaces are allowed within a variable, constant or keyword.
4. All statements are entered in small case letters.
5. C has no specific rules for the position at which a statement is to be written. That's why it is often called a free-form language.
6. Every C statement must end with symbol of semicolon (**;**). Thus **;** acts as a statement terminator.

For Example: Let us now write our first C program to add two numbers.

```
#include<stdio.h>          /*--- PRE-PROSSER STATEMENT BEGINS --*/
void main()                /*------ MAIN PROGRAM BEGINS ------------*/
{
        int a,b,sum;       /*----- DECLARATION STATEMENTS ------------*/
        a=10;              /*------ ASSIGNMENT STATEMENTS -----------*/
        b=7;
        sum=a+b;
        printf("Sum=%d",sum); /*------ PRINT STATEMENTS -----------*/
}
```

1.5 Function:

A function is a reusable block of statements that is executed to perform any task. Any C program is a collection of these functions. A function can be treated as sub program. It has following features:

1. Functions allows reusability (In a program, a function can be called any no. of times).
2. Functions reduce the program code.
3. Functions allow calls to other functions within one function.

1.5.1 Structure of a function:

```
Return-type function-name (arguments list….)
{
local declarations
……..
…..code
……..
}
```

1.5.2 Types of Functions: Broadly we can classify functions in two main categories:

1.5.2.1 Library functions

1.5.2.2 User defined functions.

1.5.2.1 **Library functions:** These are also called as built in or System defined functions. They can be viewed as small programs on their own. They are defined by C compiler and included in header files. For eg. : pow( ), sqrt( ), printf( ), scanf( ), and string functions etc.

1.5.2.2 User defined functions:

These are defined by the user. User can include any number of functions in the program, for which declaration and definition must be provided.

For example:

```
void sum( int,int);
        {
        printf (" sum=%d",x+y);
        }
```

1.5.3 Formatted I/O functions:

Formatted I/O function refers to the conversion of data to and from a stream of characters, for printing in plain text format. They allow us to supply the input in a fixed format and let us obtain the output in the specified format. In C, header file <stdio.h> contains basic I/O functions scanf( ) and printf( ).

Scanf(): reads from standard input.

Syntax:

Scanf(format_string, list_of_variable_addresses);

For example:

Scanf("%d",&a);

Printf(): writes to strandard output.

Syntax:

printf(format_string, list_of_ variables);

For example:

printf("%d",a);

1.6. Compilation and Execution of C Program:

To execute the program we need to type it and instruct the machine to execute it. To type C program we need a program called **Editor**. Once the program, which is also called the **source code** has been typed it needs to be converted to machine language (binary language of 0s and 1s), before the machine can execute it. To perform this conversion, which is also called **object code**, we need another program called **Compiler**. There are several IDEs available in the market. An Integrated Development Environment (IDE) consists of an Editor as well as the Compiler. For example, Turbo C, Turbo C++.

By using a Turbo C or Turbo C++ compiler following are the steps we need to follow to compile and execute our first C program…

Start the compiler at C> prompt. The compiler (TC.EXE is usually present in C:\TC\BIN directory).
Select New from the File menu.
Type the program.
Save the program using F2 under a proper name (say sum.c).
Use Alt + F9 and Ctrl + F9 to compile and execute the program.
Use Alt + F5 to view the output.

On compiling the program its machine language equivalent is stored as an EXE file (sum.EXE) on the disk. This file is called an executable file. If we copy this file to another machine we can execute it there without being required to recompile it. In fact the other machine need not even have a compiler to be able to execute the file.

- Note: Make sure that the default extension is 'C' rather than 'CPP'.

1.6.1 Different types of errors:

Error is a mistake or bug, which is commonly known as programming error in C. The process of removing or eliminating errors from a C program or making a program bug free is called **debugging**. These can be categorized into 3 types:

1.6.1.1 Compilation errors. (Syntax errors)
1.6.1.2 Linker errors.
1.6.1.3 Logical errors.

1.6.1.1 Compilation (Syntax) errors: These errors are raised when we compile the program. These can be located and corrected easily. Most common compilation or syntax errors are.

Undefined variable
Redeclaration of variable
Unterminated string character
Missing semicolon ;
Function call missing (,;")
Function should have prototype.

1.6.1.2 Linker errors: The program must be linked to the 'C' library. If it fails in such case these errors are raised. Most common errors are:

Unable to link cos.obj
Undefined symbol

1.6.1.3 Logical (Run-time) errors: These errors are raised due to logical inefficiency. We can know them when program gets executed. It is very difficult to locate them, most common logical errors are:

Divide by zero
Floating point error
Null value
Garbage result in printing.

1.7 Starting with Programming:

The basic steps for creating and successfully executing of a program with Turbo C are:

1. First, download the Turbo C tool kit from CD or Internet. Then, Go to the C drive (by default) and open the TC folder.

2. Now, Double click on TC executive file (TC.exe). It's icon appears as shown below (**Fig: 1.2**):

Fig: 1.2

3. This will open a window. Now open the file menu by pressing Alt + F and select new to create a new program as shown below (**Fig: 1.3**).

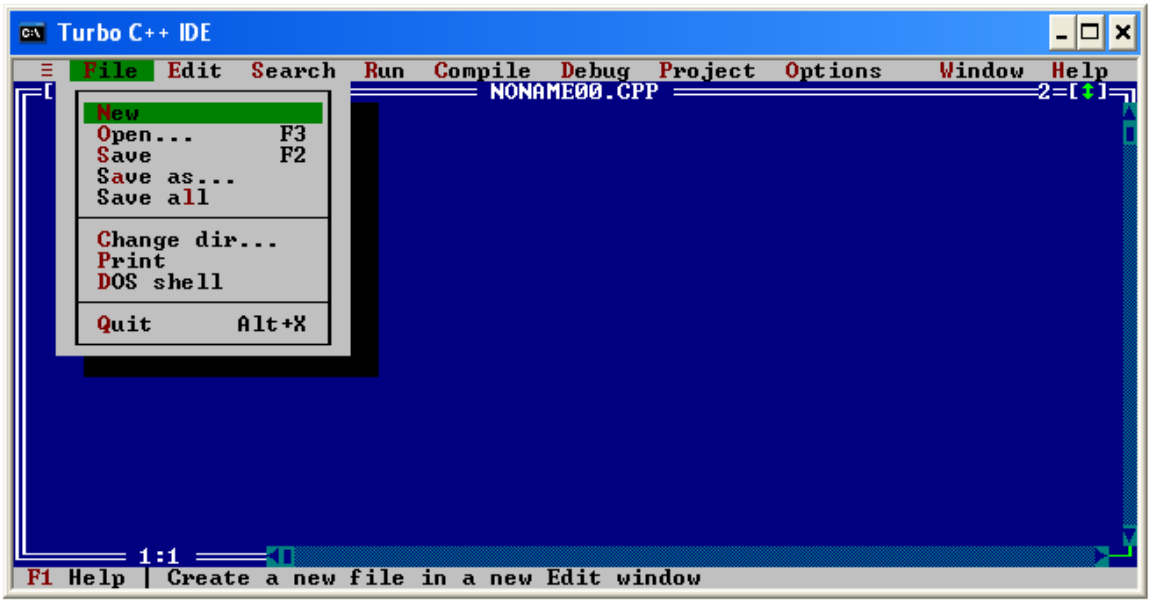


Fig: 1.3

4. Now type the program as shown in the below figure (**Fig: 1.4**).

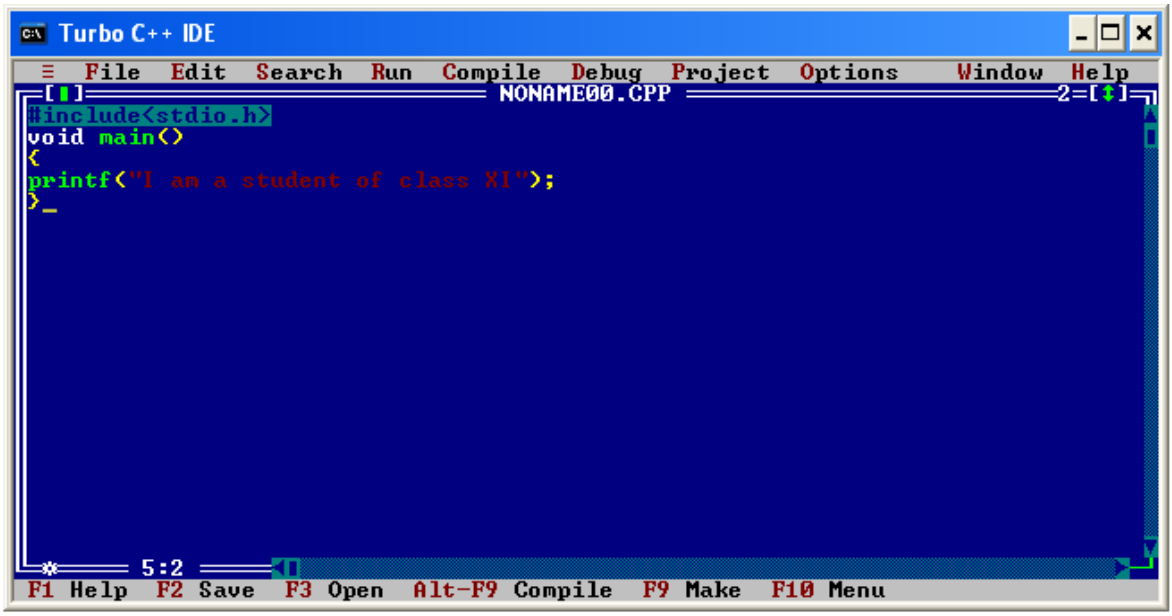


Fig: 1.4

5. After the completion of typing program, you have to save the program by pressing Alt + S then select **save** options from the menu as shown below (**Fig: 1.5**).

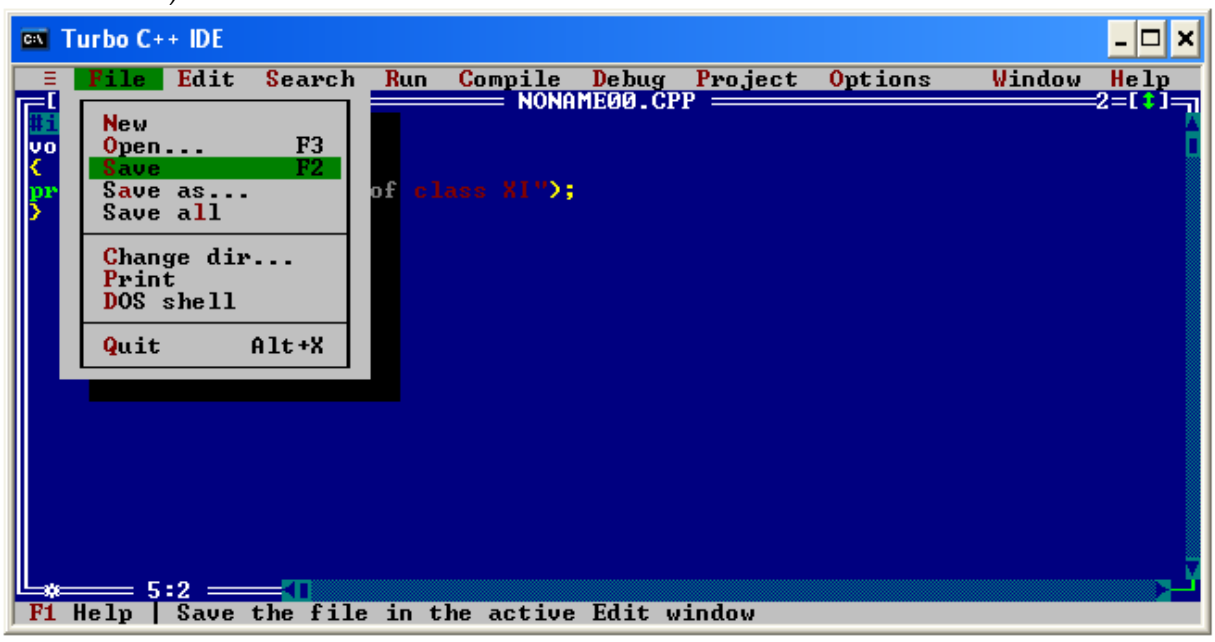


Fig: 1.5

6. Give the file name with extension ".C" in Save File As Dialog Box**.** Then, Click the OK button or press Enter Key from keyboard (**Fig: 1.6**).

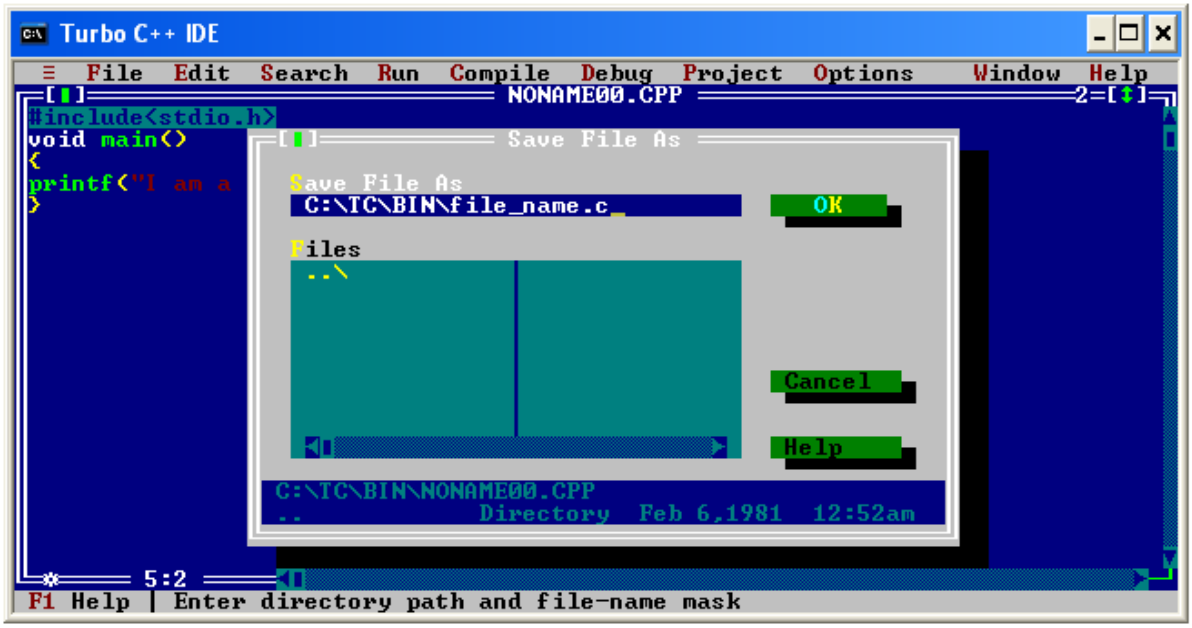


Fig: 1.6

7. Press Alt + F9 keys simultaneously for Compiling the program, if the compiler finds some errors in the program, then remove these errors & compile it again. If there is no error, then it will show zero errors shown below (**Fig: 1.7**), that means that this program is ready for execution.

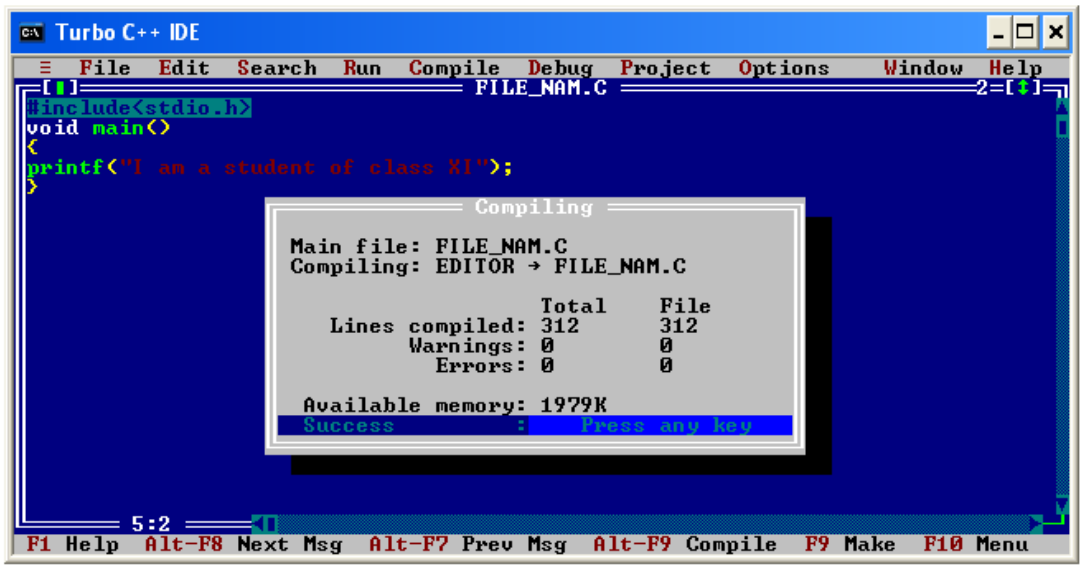


Fig: 1.7

8. Press Ctrl + F9 keys simultaneously for executing the program. Now, this program is executed. To see the output press Alt + F5, This will show you output on the screen as shown in the figure below (**Fig: 1.8**):

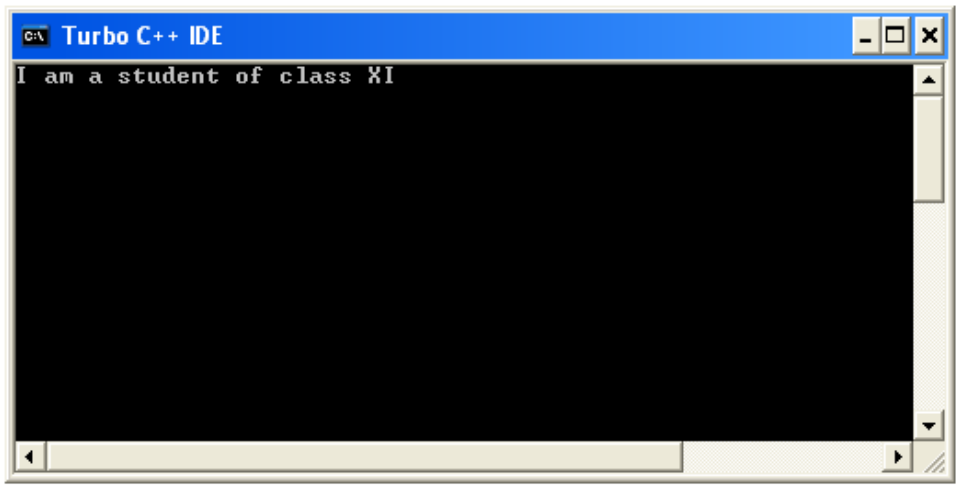


Fig: 1.8

Exercise

Fill in the Blanks:

1. In 1970's, C was developed by _____.
2. C is a _____ level language as it combines both the advantages of low level and high level language.
3. The syntax errors can easily be detected at the time of _____.
4. The header files starts with _____ symbol and have _____ extension.
5. A function is a _____ block of statements.

True/ False

1. Every C statement must be terminated with (.) full-stop.
2. To compile a C program ctrl+f9 keys are used.
3. The extension .CPP can be used to save C program.
4. Syntax errors raised at the time of compilation.
5. To view the output of a C program alt+f5 keys are used.

Short answer type questions:

1. What is a pre-processor statement?
2. Define debugging.

3. What is editor?
4. Tell about the character set of C language.
5. What are escape sequences?

Long Answer type questions:

1. Describe the features of C languge.
2. Define the structure of a C program.
3. Define function and its types.
4. Write the steps for compilation and execution of C program.
5. What is an error? Explain its classification?

****

2. Constant, Variables and Data Types

Objectives of this lesson

2.1 Introduction

Like any other human Language, C language consists of alphabets, numbers and special symbols. When these alphabets, numbers and special symbols are combined together they form constants, variables and keywords, and are known as 'C Tokens'.

The basic and the smallest units of C program are called tokens. There are six types of tokens in C.

C tokens include:

1. Identifiers
2. Keywords
3. Constants
4. Variables
5. Operators
6. Special Symbols.

Every word in a C program is either a keyword or an identifier. Let us now study in detail about them.

2.2 Identifiers:

An identifier is a name given to the program elements such as variables, arrays and functions. It is a sequence of alphabets, digits and underscore. It can't include white-spaces. The first character must be an alphabet. Identifiers are the names of variables, functions, labels and various others user defined items. For eg. Pi, area, add_cd, sum() etc.

There are some rules that are followed for the formulation of identifiers names are given below:

1. The first character must be an alphabet or underscore.
2. All succeeding characters must be either letters or digits.
3. Uppercase and lowercase identifiers are different in C.
4. No special Character or punctuation symbols are allowed except the underscore "_".
5. No two successive underscores are allowed.
6. Keywords should not be used as identifiers.

2.3 Keywords:

All keywords (all reserved words) are basically the sequence of characters that have one or more fixed meaning and this meaning, in any circumstances cannot be changed. The following identifiers are reserved as keywords and are defined by C compiler. Each word has different meaning and purpose in C language. These cannot be used as names of variables, functions or labels. All C keywords must be written in lowercase letters; because, in C both uppercase and lowercase letters are significant. They are 32 in number.

auto	double	int
struct	break	else
long	switch	case
enum	register	typedef
char	extern	return
union	const	float
short	unsigned	continue
for	signed	void
default	goto	sizeof
volatile	do	if
static	while	

Fig: 2.1

2.4 Constants:

In C, any quantity which never changes is known any constant. This quantity can be stored at any location in the memory of the computer. In C programming we can define variables with constant values.

For example:

```
int a=5;
char c='s';
char school[ ]="GHS";
```

2.4.1 Types of Constants: C constants can be divided in two categories:

1. Primary constants
    a. Numeric constant
    b. Character constant

2. Secondary constants
    a. Array
    b. Structure
    c. Union etc.

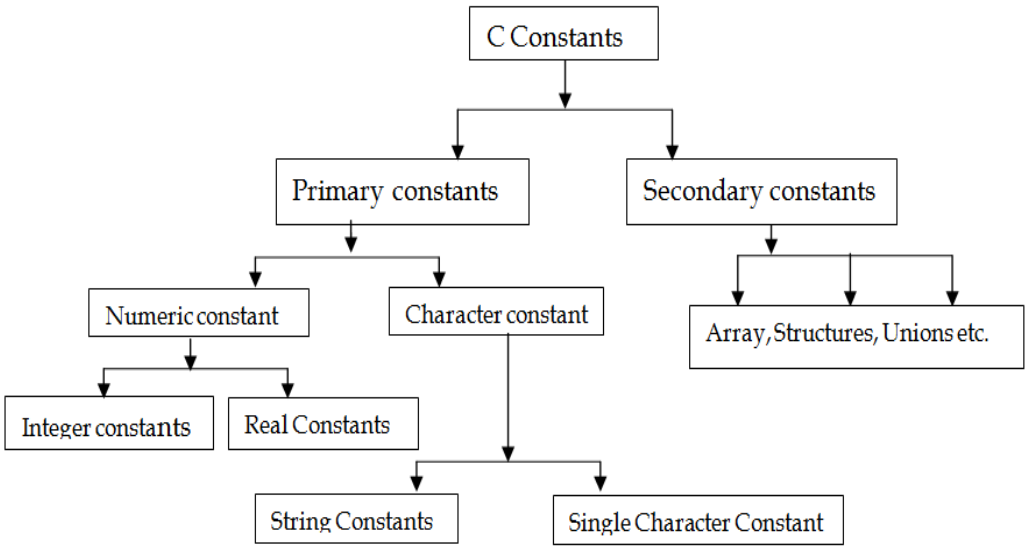


Fig: 2.2

- **Numeric Constants** :

Numeric Constants consist of numbers. They are of two types: Integer constant and Real constants.

a. Integer constants:

An integer constant must have at least one digit. It must not have a decimal point. It can be either positive or negative. If no sign precedes an integer constant it is assumed to be positive. No commas or blanks are allowed within an integer constant. The range for integer constants is -32768 to 32767. For eg.: 46, +72, -1000, 755 etc.

b. Real constants:

Real constants are mostly called Floating Point constants. The real constants could be written in two forms—

i. Fractional form
ii. Exponential form.

i. Fractional form:

In fractional form, a real constant must have at least one digit. It must have a decimal point. It could be either positive or negative. Default sign is positive. No commas or blanks are allowed within a real constant. For eg.: 446.45, -100. 75 etc.

ii. Exponential form.

In Exponential form, a real constant is represented in two parts. The part appearing before 'e' is called mantissa, whereas the part following 'e' is called exponent.

The mantissa part and the exponential part should be separated by a letter e. The mantissa part may have a positive or negative sign. Default sign of mantissa part is positive. The exponent must have at least one digit, which must be a positive or negative integer. Default sign is positive. Range of real constants expressed in exponential form is-3.4e38 to 3.4e38. For eg.: +3.2-5, 4.1e8, -0.2e+3 etc.

- **Character Constants:**

a. Single Character Constants :

A character constant is a single alphabet, a single digit or a single special symbol enclosed within single inverted commas. Both the inverted commas should point to the left. For example, 'A' is a valid character constant whereas 'A' is not.
The maximum length of a character constant can be 1 character. For eg.: 'A', '5', '=' etc.

b. String Constants:

A string constant, or string literal, is a sequence of zero or more characters surrounded by double quotes. For example:

```
"I am a string"
```

or

```
""              /* the empty string */
```

The quotes are not part of the string, but serve only to delimit it. String constants can be concatenated at compile time. For example:

"hello, " "world" is equivalent to "hello, world"

This is useful for splitting up long strings across several source lines. Technically, a string constant is an array of characters. The internal representation of a string has a null character '\0' at the end, so the physical storage required is one more than the number of characters written between the quotes. A sequence of characters in double quotes, like "hello, world\n", is called a character string or string constant.

2.5 Variables:

An entity that may vary or change on executing the program is called a variable. Variable names are names given to locations in memory. These locations can contain integer, real or character constants. Let us try to understand this by an example of a box, which may contain a toy or a cake or anything else but named as gift. Just in the same manner, a variable named as 'a' may contain any value integer, float, character or string.

Variables are tokens defined by user and used in programming. It must start with a letter or underscore (_), it can't include spaces, and it can contain digits.

2.5.1 Naming conventions for variables:

a. A variable name is any combination of 1 to 31 alphabets, digits or underscores. But we should not create unnecessarily long variable names.
b. The first character in the variable name must be an alphabet or underscore.
c. No commas or blanks are allowed within a variable name.
d. No special symbol other than an underscore can be used in a variable name.

For eg.: si_yr, hra_anm, x1, day_today etc.

Note: These rules remain same for all the types of primary and secondary variables.

2.5.2 Declaration of Variables:

In C, all variables must be declared before they are used in the program, usually at the beginning of the program before any executable statements. A declaration announces the properties of variables; it consists of the data-type of the variable and the name of the variable.

Syntax is:

Data_type varlist semicolon

For example:

```
int x;
float y;
```

We can declare more than one variable in one statement. For example:

```
int lower, upper, step;
float fahr, celsius;
```

In C programming, every statement must be terminated with a symbol known as semicolon (;). Here the type int means that the variables listed are integers; by contrast with float, which means floating point, i.e., numbers that may have a fractional part.

2.5.3 Initialization of Variables:

Initializing a variables means assigning values to variables with the help of assignment operator (=). After declaring a variable we can assign value to a variable.
Syntax is:

Variable_name = value ;

For eg.:

```
a = 10;
ch = 'x';
```

We can assign values to the variables at the time of declaration.

For eg.:

```
int x=5;
float PI= 3.14;
char y= '10';
```

2.6 Delimiters:

A delimiter is a symbol that has a syntactic meaning and significance. But they do not specify any operation to be performed on a value. A C language sentences, labels arrays, etc are separated by special characters, these are called as delimiters. Some of them are given below:

Hash	#	Pre- Processor directive
Comma	,	Variable delimiter in variable list
Curly braces	{ }	used to block C sentences
Square brackets	[ ]	used with arrays
Parenthesis	( )	used in expressions
Colon	:	Label delimiter

Fig: 2.3

2.7 Data types:

Data type determines the type of data a variable will hold. If a variable 'x' is declared as int, this means variable 'x' can hold only integer values. If a variable is declared as float, then it will store floating values. Specifying Data type tells the C compiler about how many bits or bytes, the variable will hold in the memory. So every variable which is to be used in the program must be declared specifying its data type before using in the program.

2.7.1 There are only a few basic data types in C:

1) **char** consuming a single byte, capable of holding one character.
2) **int** can store an integer and holds two bytes.
3) **float** is used for single-precision floating point, uses four bytes of memory.
4) **double** is used for double-precision floating point, and uses eight bytes of memory

2.7.2 User defined data types: These data types can be defined by user. These are of following types:

5) **Arrays**: An array is a collection of similar type of data.
6) **Structures:** A structure is a collection of variables under a single name. These variables can be of different data-types. All the data items are stored in individual memory locations.
7) **Unions:** Unions are quite similar to structure in C. A union is a special data-type that enables us to store deferent data-types in same memory location. All the data items can share unique memory location.
8) **Enumerations:** An Enumeration type allows the programmer to define their own data-types. An Enumeration consists of a set of named integer constants.

Broadly we can categories the data types as shown in figure below:

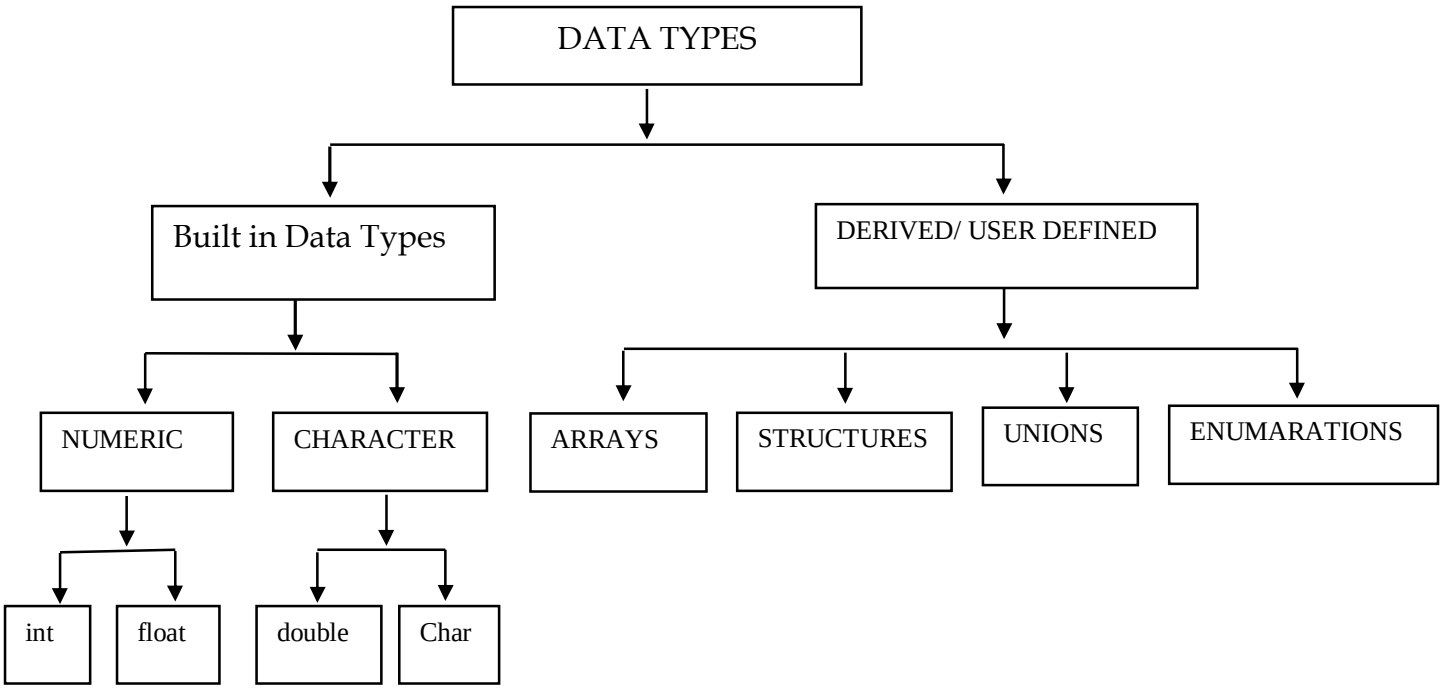


Fig: 2.4

Following table shows the data types, their format-specifier and memory consumption:

Data type	**Memory**	**Format specifier**
int	2 bytes	%d
Char	1 bytes	%c
Float	4 bytes	%f
Double	8 bytes	%f
String	Size of char array	%s
Unsigned int	2 bytes	%u

long unsigned	8 bytes	%lu
long	4 bytes	%ld

Fig: 2.5

2.8 Type modifier or Qualifiers:

The keywords short, long, unsigned, signed etc are called qualifiers. Qualifier is an extra name given to either variable or functions. These are also called TYPE qualifiers. Mostly they are used to modify the basic integer type:

short int <= int <= long int

float <= double <= long double

This means that a 'short int' should assign less than or the same amount of storage as an 'int' and the 'int' should be less or the same bytes than a 'long int'.

2.8.1 Short and long Type data Qualifiers

Example:
```
long int L;            /* short and long apply to integers */
short int si;
long int timer;   /* the word int can be omitted in declarations */
```

2.8.2 Unsigned int:

This qualifier is used for variables that takes positive values only, by prefixing the int declaration (or long int or short int) with the word unsigned, the range of positive numbers can be double. Unsigned numbers are always positive or zero and obey 2n where n is the number of bits. So, for instance, if chars are 8 bits, unsigned char variables have values between 0 and 255. The unsigned char type is designed for an extended character set (usually used for graphics).

2.8.3 Double long:

The type long double specifies extended-precision floating point.

2.9 The void Data type or empty data type:

Every C function has a return data type associated with it. Where returned type is one of the data type that determine the type of the value returned by the function. If a function is not supposed to return any value, its return type is specified as void. Void means empty. Since the main() is the function from where execution begins, it will usually be declared with type void. A pair of parentheses follows the word main. If any data type is contained in these parentheses, it specifies the formal arguments of the function. If there are no

arguments, we write void. The main function header, void main (void) can also be used as main().

Exercise

Fill in the Blanks:

1. _____ should not be used as identifiers.
2. Every word in C program is a _____ or an _____.
3. Real constants are mostly called _____.
4. char data type consumes a _____ byte in the memory.
5. Every C function has a _____ data-type associated with it.

True/ False

1. An integer constant must have a floating Point.
2. An array is a built-in data type.
3. The keywords short, long, unsigned, signed etc are called qualifiers.
4. All variables must be declared before they are used in the program.
5. main() is the function from where execution of the program begins.

Short answer type questions:

1. Define C tokens.
2. What are the rules that are followed for the formulation of identifiers names?
3. What is a keyword in C language?
4. Tell about the character set of C language.
5. What are escape sequences?
6. What do you understand by data-type? Which are the four basic data-types in C.
7. What is a modifier?
8. Define delimiter.

Long Answer type questions:

1. Give the definition of C constant. Explain its types.
2. Define variable? Explain its declaration and initialization in C program.
3. Explain the various data-types used in C.
4. Write the naming conventions for variables in C language.
5. What is an error? Explain its classification?

****

3 Operators and Expressions

Objectives of this lesson

3.1 Introduction

An **expression** is a collection of constants, variables and operators. An expression returns a values or a result. This result can be used in the C program. Expressions can be as simple as a single value and as complex as a large calculation. They are made up of two things, operators and operands.

For eg.: x= 3.5y+ 2.7z

In this example x, 3.5y and 2.7z are **operands** and symbol '=' and '+' are **operators**.

The format for the expression is

Operand1 operator operand2

Here are few examples of expressions: 2 + 5 * 4, -1 +6, (2 + 5) * 6. These expressions are evaluated by moving from left to right.

3.2 Operators:

In order to carry out basic arithmetic operations and logical operations, operators are needed. Operators act as connectors and they indicate what type of operation is being carried out. The values that can be operated by these operators are called operands.

C operators are broadly classified into three main categories:

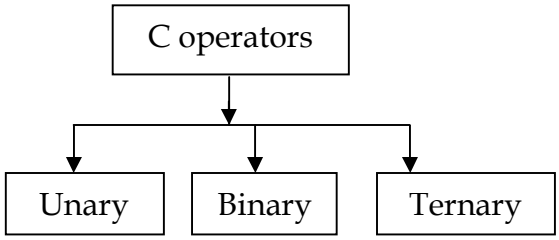


Fig: 3.1

3.2.1 Unary operator:

An operator that acts on only one operand is known as Unary operator. Most important unary operator is unary **minus.** Any positive operand associated with unary minus gets its value changed. For e.g. let a= 10 , b =15 and c = a+(-b), then c = 10+ (-15) = -5. Since b is initially a positive integer variable, when operated by unary minus, gets its value changed

3.2.2 Binary operators:

Binary operators act on two operands. They are further classified into following categories:

1. Arithmetic Operators
2. Relational Operators
3. Logical Operators
4. Assignment Operators
5. Bitwise Operators
6. Other Operators

The following table shows the binary operators and their meaning:

Type	Operators	Meaning
Arithmetic operators	+	Addition
	-	Subtraction
	*	Multiplication
	/	Division
	%	Modulus
Relational operators	>	Greater than
	>=	greater than or equal to
	<	less than
	<=	less than or equal to
	==	equal to(comparison)
	!=	not equal to
Logical operators	&&	Logical AND
	||	Logical OR
	!	Logical NOT

Assignment operators	=, +=, -=, /=, %=	assignment
Bit wise operators	&	Bit wise AND
	|	Bit wise OR
	^	Bit wise XOR
	<<	left shift
	>>	right shift
	~	Compliment
Other operators	++	Increment
	--	decrement
	sizeof	sizeof operator
	? :	ternary conditional

Fig: 3.2

3.2.2.1 Arithmetic Operators:

There are five main arithmetic operators in 'C'. They are '+' for additions, '-' for subtraction, '*' for multiplication, '/' for division and '%'for remainder after integer division.

This '%' operator is also known as modulus operator.

Operands can be integer quantities, floating-point quantities or characters. The modulus operator requires that both operands be integers & the second operand be nonzero. Similarly, the division operator (/) requires that the second operand be nonzero, though the operands need not be integers. Division of one integer quantity by another is referred to as integer division. With this division the decimal portion of the quotient will be dropped. If division operation is carried out with two floating- point numbers, or with one floating-point number and one integer, the result will be a floating-point quotient.

If we take two variables say x and y and their values are 20 and 10 respectively, and apply operators like addition, subtraction, division, multiplication and modulus on them, then their resulted values will be as follows:

x+y 30
x-y 10
x*y 200
x/y 2
x %y 0

If one or both operands represent negative values, then the addition, subtraction, multiplication and division operations will result in values whose signs are determined by the usual rules of algebra. The interpretation of remainder operation is unclear when one of the operands is negative.

Operands that differ in type may undergo type conversion before the expression takes on its final value. The value of an expression can be converted to a different data type if desired. The 'C' programmer also has the choice of explicitly specifying how the

values are to be converted in a mixed mode expression. This feature is known in 'C' as conversion and may be accomplished using what is called the **cast operator** .The name of data type to which the conversion is to be made is enclosed in parentheses and placed directly to the left of the value to be converted; the word "cast" never is actually used. The example of type casting is as follows:

```
int a=17;
float b;
b=(float)a+ 11.0;
```

The cast operator converts the value of int 'a 'to its equivalent float representation before the addition of 11.0 is carried out. The precedence of the cast operator is the same as that of the unary minus operator. The cast operator can be used on a constant or expression as well as on a variable. For e.g.

```
(int) 7.179
(double) (5*3/8)
(float)(a+4)
```

In the second example, the cast is performed only after the entire expression within the parentheses is evaluated.

- **Precedence/ Hierarchical Order of Arithmetical Operators:**

The operators within C are grouped hierarchically according to their order of evaluation known as **precedence**. Operations with a higher precedence are carried out before operations having a lower precedence. The natural order can be altered by making use of parentheses.

Arithmetic operators *,/ and % are under one precedence group and +,- are under another precedence group. The operators *, / and % have higher precedence than + and -. In other words, multiplication, division and remainder operations will be carried out before addition and subtraction. Another important point to consider is the order in which consecutive operations within the same precedence group are carried out. This is known as associativity. Within each of the precedence groups described above, the associativity is left to right. In other sense, consecutive addition and subtraction operations are carried out from left to right, as are consecutive multiplication, division and remainder operations.

Let us understand this with the help of one example, say there are 3 variables a,b, and c having values 5,10 and 15 respectively. The different operations on these three variables and their result are as follows:

```
a+b/c=5
```

In this expression, division is done first because division has a higher precedence than addition.

```
b*c-a=145
```

In this expression, the multiplication is done before the subtraction.

```
a*b/c= 3
```

In this expression, the calculation proceeds from left to right, but truncation takes place in the division operation.

(a+c)*b/a=40

Finally in the last case (a+c)*b/a, the contents of the parentheses are evaluated first, and then it is multiplied & finally divided. In this case no truncation takes place.

3.2.2.2 Relational operators

Relational operators are symbols that are used to test the relationship between two variables, or between a variable and a constant. The test for equality, is made by means of two adjacent equal signs with no space separating them. 'C' has six relational operators as follows:

1. > greater than
2. < less than
3. != not equal to
4. >= greater than or equal to
5. == equal to(comparison)
6. <= less than or equal to

These operators all fall within the same precedence group, which is lower than the unary and arithmetic operators. The associativity of these operators is left-to-right.

These relational operator are used to form logical expression represeating condition that are either true or false. The resulting expression will be of type integer, since true is represented by the integer value and false is represented by the value 0.

Let us understand operations of these relational operaters with the help of an example: x=2, y=3, z=4.

x<y	true	1
y>z	false	0
(x+y) >=z	true	1
(y+z)<=(x+7)	false	0
z!=4	false	0
y ==3	true	1

Here, true or false represents logical interpretation and they have values 1 and 0 respectively.

3.2.2.3 Logical operators

There are two logical operators in C language, they are **and or**. Theyar e represented by && and || respectively. These operators are referred to as logical and, logical or, respectively. The result of a logical and operation will be true only if both operands are true, whereas the result of a logical or operation will be true if either operand is true or if both operands are true. The logical operators act upon operands that are themselves logical expressions. Let us understand it, with the help of a following example:

Suppose x=7, y=5.5 , z= 'w'

Logical expressions using these variables are as follows:

Expression	Interpretation	Value
(x>=6) &&(z= =’w’)	true	1
(x>=6) || (y = =119)	true	1
(x<=6) && (z=='w')	false	0

Each of the logical operators falls into its own precedence group. Logical and has a higher precedence than logical or. Both precedence groups are lower than the group containing the equality operators. The associativity is left to right.

'C' also includes the unary operator !, that negates the value of a logical expression. This is known as logical negation or logical NOT operator. The associativity of negation operator is right to left.

Precedence of operators in decreasing order is as follows:

1.	-,	++	- - Operators	! size of (type)
2.	*	/	%	
3.	+	-		
4.	<	<=	>	> =
5.	==	!=		
6.	&&			
7.	||			

If you have trouble remembering all these rules of precedence you can always resort to parentheses in order to express your order explicitly.

3.2.2.4 Assignment operator:

There are several different assignment operators in C. All of them are used to form assignment expression, which assign the value of an expression to an identifier. The most commonly used assignment operator is =. The assignment expressions that make use of this operator are written in the form:

Identifier=expression

Where identifier generally represents a variable and expression represents a constant, a variable or a more complex expression.

Assignment operator = and the equality operator == are distinctly different. The assignment operator is used to assign a value to an identifier, whereas the equality operator is used to determine if two expressions have the same value. These two operators cannot be used in place of one another.

If the two operands in an assignment expression are of different data types, then the value of the expression on the right will automatically be converted to the type of the

identifier on the left. The entire assignment expression will then be of this same data type. For example
A floating point value may be truncated if assigned to an integer identifier, a double-precision value may be rounded if assigned to a floating-point identifier, an integer quantity may be altered if assigned to a shorter integer identifier or to a character identifier.

'C' also contains the five additional assignment operators as follows:

1. +=
2. -=
3. *=
4. /=
5. %=

The format for using assignment operator is as shown below:

expression1+= expression2
is equal to
expression1= expression1 + expression2

Similarly

expression1 - = expression2
is equal to
expression1 = expression1 - expression 2

Assignment operators have a lower precedence than any of the other operators.

For example: x* = -3 *(y+z)/4
is equivalent to x= x*(-3 *(y+z)/4)

In this example first (y+z) will be evaluated, then multiplication will take place. Then division, and finally multiplication will take place.

3.2.2.5 Bitwise Operators

C provides the following operators for handling bitwise operations:

1. << Bit shift left (a specified number or bit positions)
2. >> Bit shift right(a specified number of bit positions)
3. | Bitwise OR
4. ^ Bitwise XOR
5. & Bitwise AND
6. ~ Bitwise one's complement

- **The Meaning of Bitwise Operators**

Bitwise operations are not to be confused with logical operations (&&, ||...). A bit pattern is made up of 0s and 1s and bitwise operators operate individually upon each bit in the operand. Every 0 or 1 undergoes the operations individually.

3.2.2.6 Conditional operator(?:)

There is one conditional operator(?:) in 'C' language. An expression that makes use of the conditional operator is called a conditional expression. This operator is also known as **ternary operator**. A conditional expression is written in the form

Expression ? value1 : value2

When evaluating a conditional expression, expression is evaluated first. If expression is true, then **value1** becomes the value of the conditional expression. If expression is false, then **value2** becomes the value of the conditional expression.

For example (i< 1) ? 0:200

i is integer variable here. The expression (i<1) is evaluated first, if it is true the entire conditional expression takes on the value 0. Otherwise, the entire conditional expression takes on the value 200.

3.2.2.7 Increment and Decrement Operators:

Two other commonly used unary operators are increment operator, ++, and the decrement operator - -.

The increment operator causes its operand to increased by one, whereas the decrement operator causes its operand to be decreased by one. The operand used with each of these operators must be a single variable.

For example, x is an integer variable that has been assigned a value of 10.
The expression

++x,

which is equivalent to x= x+1,
causes the value of x to be creased to 11.
Similarly the expression

--x,

which is equivalent to x=x-1,
causes the original value of x to be decreased to 9.

The increment and decrement operators can each be utilized in two different ways, depending on whether the operator is written before i.e. prefix or after the operand i.e postfix. If the operator precedes the operand, then the value of operand will be altered before it is used for its intended purpose within the program. This is called pre increment/ decrement. If, however the operator follows the operand then the value

of the operand will be changed after it is used. This is called post increment/ decrement.
For example:

If the value of x is initially 10, it can be increased by two methods:

++x then the value of x will become 11 (pre increment)

x++ then the value of x will be 10 for current use. (post increment)

Similarly for decrement operator

--x then the value of x will become 9. (pre decrement)

x-- then the value of x will be 10 for current use. (post decrement)

the current value of x will be 9 if we say -- x.

➢ Size of operator:

Another unary operator is the sizeof operator . This operator returns the size of its operand, in bytes. This operator always precedes its operand. The operand may be an expression, or it may be a cast.

For e.g sizeof (x);

sizeof (y);

If x is of integer type variable and y is of floating point variable then the result in bytes is 2 for integer type and 4 for floating point variable.

Consider an array school[]= "National"

Then, sizeof (school) statement will give the result as 8.

Exercise

Fill in the Blanks:

1. _____ is made up of two things, operators and operands.
2. Integer value _____ is equivalent to false.
3. An operator that acts on only one operand is known as _____ operator.
4. The increment and decrement operators can each be utilized in _____ different ways.
5. The order of evaluation of an operator is known as _____.

True/ False

1. The conditional operator is also known as ternary operator.
2. Any positive operand associated with unary minus doesn't get its value changed.
3. Bitwise operations and logical operations are same in C programming.
4. sizeof operator is a unary operator.

5. The format for the expression is **Operand1 operator operand2**.

Short answer type questions:

1. What is unary operator?
2. What are the 6 assignment operators discussed in this chapter?
3. What is sizeof operator?
4. Describe the difference between increment and decrement operators.
5. Tell us about conditional operator.

Long Answer type questions:

1. What is an operator? Describe different types of operators.
2. What is an operand? What is the relationship between operators & operands?

****

4. Control Structures in C

Objectives of this lesson

4.1 Introduction:

Normally in C programming, a program consists of a number of statements which are usually executed in sequence. This is called **sequential execution**. Programs can be much more powerful if we can control the order in which statements are run. The C language includes a wide variety of powerful and flexible control statements. We can transfer the control point to a desired location with control structures. C provides two styles of flow control:

Branching
Looping

Branching is deciding what actions to take and looping is deciding how many times to take a certain action.

Branching:

Branching is so called because the program chooses to follow one branch or another. Branching has the following control statements:

i. Conditional control Statements

a. if statement
b. if-else statement
c. if-else-if statement
d. nested if statement

ii. Multi-way conditional control Statements
a. switch statement

iii. Jumping Statements
a. goto statement
b. continue statement
c. break statement

Looping:

Loops provide a way to repeat commands and control how many times they are repeated. C provides a number of looping way.

a. while loop
b. for loop
c. do-while loop

4.2 Conditional control structures (Decision Making Statement)

4.2.1 if statement:

if is also called as **conditional statement**. The **if** statement is used to carry out a logical test and then take one of two possible actions, depending on whether the outcome of the test is true or false. In if, statements get executed only when the condition is true, it can terminate when the condition becomes false.

Syntax:

```
if (expression)
 {
  Block of statements;
 }
```

For Example: A program in C Language to find whether the student is "Passed" by entering marks through the Keyboard.

```
#include<stdio.h>
void main()
{
        int marks;
        printf("\n Enter your marks: ");
        scanf("%d", &marks);

                if(marks>=33)
                printf("\n Pass");
}
```

The output of above program is shown in Fig 4.1:

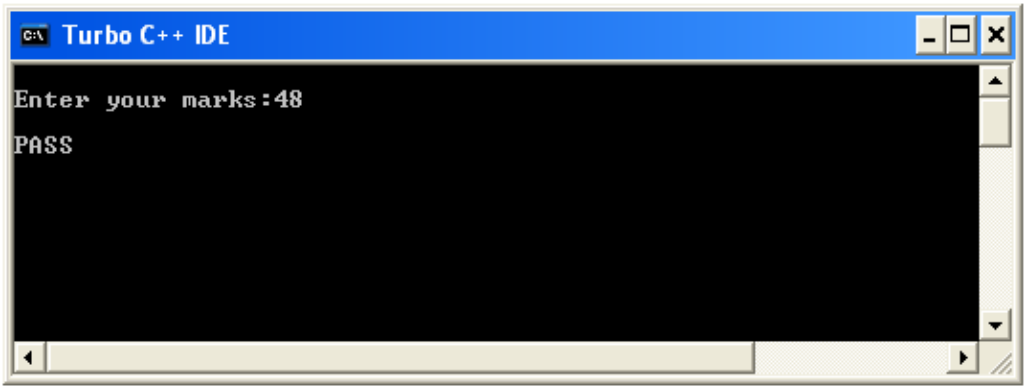


Fig 4.1

4.2.2 if–else statement:

In **if-else** conditional control statement, statements in **if block** gets executed only when the condition is **true** and statements in **else block** gets executed only when the condition is **false**.

Syntax:

```
if (expression)
  {
    Block of statements;
  }
else
  {
    Block of statements;
  }
```

For Example: A program to find whether the student is "Pass" or "Fail" by Entering marks through Keyboard.

```
#include<stdio.h>
void main()
{
        int marks;
        printf("\n Enter your marks: ");
        scanf("%d",&marks);
        if(marks>=33)
        printf("\n Pass");
        else
        printf("\n Fail");
}
```

The output of above program is shown in Fig 4.2:

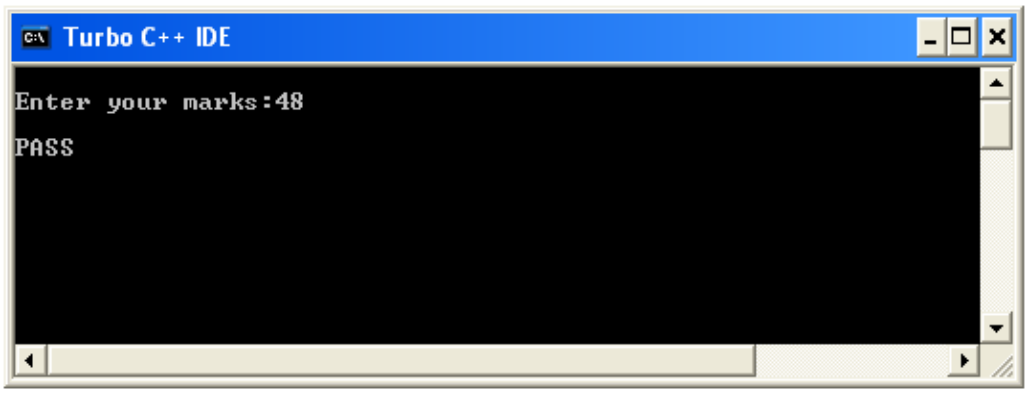


Fig 4.2

4.2.3 if–else-if statement:

In this statement, statements in loops get executed when the corresponding conditions are true. Statements in final else block gets executed when all other conditions are false. We can use any number of else-if blocks between if and else.

Syntax:

```
if (expression)
  {
    Block of statements;
  }
else if(expression)
  {
    Block of statements;
  }
  …….
  …….
```

```
  .....
  .....
else
 {
Block of statements;
}
```

For Example: A program to find Grade of the student if marks>=80 then Grade A, if marks>=60 and marks<80 then Grade B, if marks>=40 and marks<60 then Grade C, otherwise Grade D.

```
#include<stdio.h>
void main()
{
        int marks;
        printf("\n Enter your marks: ");
        scanf("%d",&marks);
        if(marks>=80)
        printf("\n Grade A");
        else
        if(marks>=60 && marks<80)
        printf("\n Grade B");
        else
        if(marks>=40 && marks<60)
        printf("\n Grade C");
        else
        printf("\n Grade D");
}
```

The output of above program is shown in Fig 4.3:

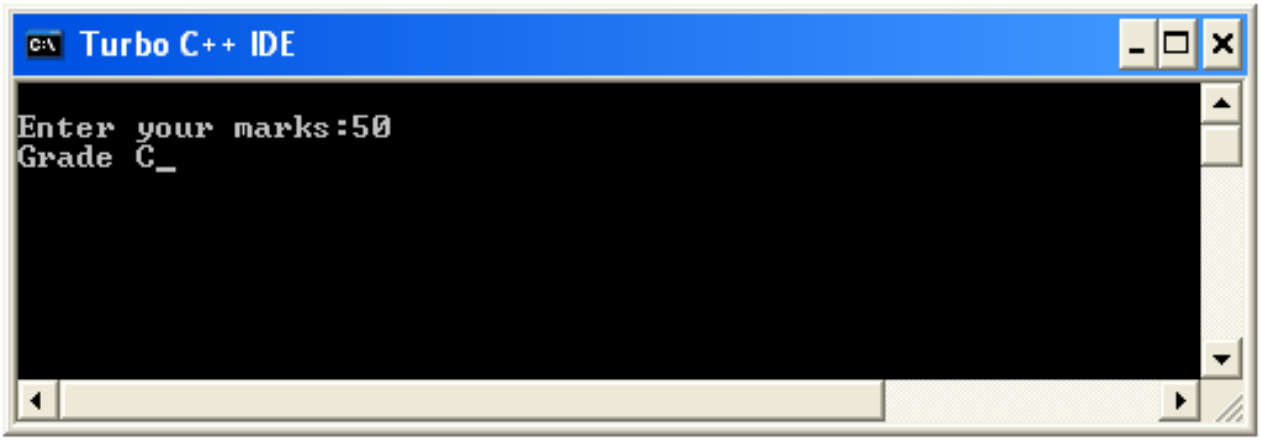


Fig 4.3

4.2.4 nested-if statement:

Writing the if statement with-in another if is called as nested-if. Inner if is processed only when outer if condition is true. Hence statements of inner if gets executed when outer if and inner if conditions are true.

Syntax:

```
if(condition1)
{
  /* Executes when the condition1is true */

if(condition2)
{
    /* Executes when the condition 2 is true */

Statements;
}
}
```

For Example: A program to find greatest number between three numbers.

```
#include<stdio.h>
    void main()
    {
        int a,b,c;
        printf("\n Enter the value of a: ");
        scanf("%d", &a);
        printf("\n Enter the value of b: ", &b);
        scanf("%d", &b);
        printf("\n Enter the value of c: ", &c);
        scanf("%d", &c);
            if(a>b)
            {
            if((a>c)
            {
            printf("A is greatest");
            }
            else
            if(b>c)
            {
            printf("B is greatest");
            }
            else
            printf("C is greater");
            }
    }
```

The output of above program is shown in Fig 4.4:

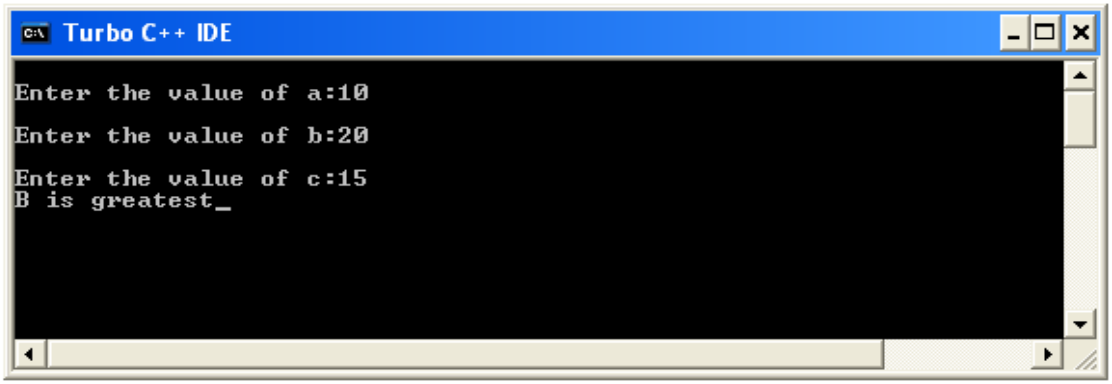


Fig 4.4

4.3 Multi way conditional(case) control structures

4.3.1 switch-case statement:

switch-case is similar to nested else if statement. It's a matter of preference which we use; switch statement can be slightly more efficient and easier to read. In switch-case, statements get executed when corresponding case constants are true only. Control point comes to **default** when all the cases are false.

Syntax:

```
Switch(variable/expression)
{
case constant1:
statements1;
break:
case constant2:
statements2;
break:
case constant3:
statements3;
break:
…
…
…
default:
statements;
break:
}
```

For example: A program to print name of day of week corresponding to the number of the day entered by the user.

```
#include<stdio.h>
void main()
{
        int day;
        printf("\n Enter day number: ");
        scanf("%d", &day);
        switch(day)
        {
        case 1:
                printf("\n Monday");
                break;
        case 2:
                printf("\n Tuesday");
                break;
        case 3:
                printf("\n Wednesday");
                break;
        case 4:
                printf("\n Thursday");
                break;
        case 5:
                printf("\n Friday");
                break;
        case 6:
                printf("\n Saturday");
                break;
        case 7:
                printf("\n Sunday");
                break;
                default:
                printf("\n You have entered a wrong day number");
        }
}
```

The output of above program is shown in Fig 4.5:

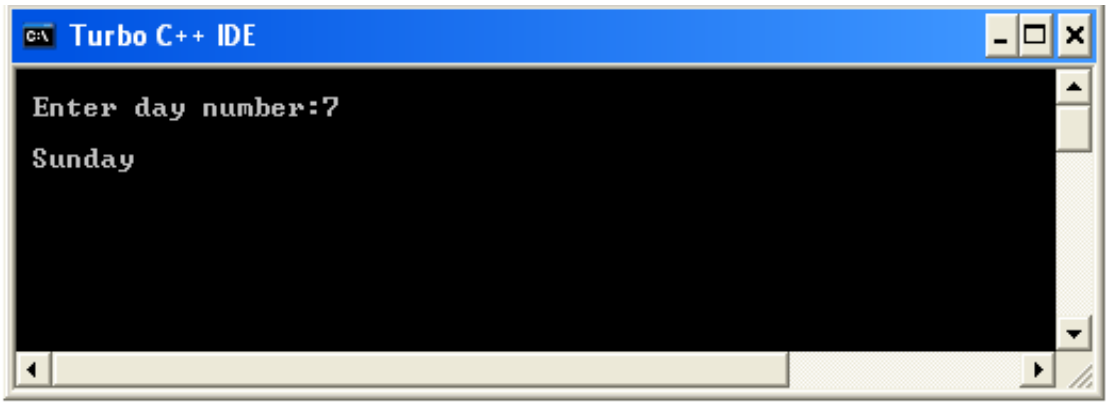


Fig 4.5

4.4 Jumping (branching) Control Structures:

Jumping statements in C programming are used for altering the normal flow of a program.

4.4.1 goto statement:

A goto in C programming provides an unconditional jump from the goto, to a labelled statement in the same function. It transfers control to a label. The give label must reside in the same function and can appear before only one statement in the same function. It is a kind of branching structure. It moves the control flow forward or backward to a label. It is condition dependent.

Syntax:

goto label;
...
…
label: statement;

Here label can be any plain text except C keyword and it can be set anywhere in the C program above or below to goto statement.

```
#include <stdio.h>
int main ()
{
int a = 10;
LOOP:do
        {
         if( a == 15)
        {
        a = a + 1;
        goto LOOP;
        }
        printf("value of a: %d\n", a);
```

```
        a++;
        }
while( a < 20 );
return 0;
}
}
```

When the above code is compiled and executed, it produces the following result:

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

The two more statements built in C programming to alter the normal flow of a program are, break; and continue;. Loops perform a set of repetitive task until text expression becomes false but it is sometimes desirable to skip some statements inside loop or terminate the loop immediately without checking the test expression. In such cases, break and continue statements are used. The break; statement is also used in switch statement to exit switch statement.

4.4.2 Break:

Break statement terminates the loop or switch statement and transfers execution to the statement immediately following the loop or switch. The break statement in C programming language has the following two usages:

1. When the break statement is encountered inside a loop, the loop is immediately terminated and program control resumes at the next statement following the loop.
2. It can be used to terminate a case in the switch statement.

If you we are using nested loops (i.e., one loop inside another loop), the break statement will stop the execution of the innermost loop and start executing the next line of code after the block.

Syntax:

```
break;
```

4.4.3 continue statement:

It is sometimes desirable to skip some statements inside the loop. In such cases, continue statements are used. It causes the loop to skip the remainder of its body and immediately re-test its condition prior to reiterating.

Syntax:

continue;

4.4.4 return:

return is used in functions. It moves the control to calling function from the called function module. Return is also return values to the point where the function is called.

Example:

```
#include<stdio.h>
main( )
{
int a=10,b=20,c;
c=sum(x,y);
printf(“%d”,c);
getch( );
}
int sum(int x, int y)
{
return x+y;
}
```

4.5 Loop (iterative) control structures

There may be a situation, when you need to execute a block of code several numbers of times. A loop statement allows us to execute a statement or group of statements multiple times.

4.5.1 for loop:

A for loop is a repetition or iterative control structure that allows us to write a loop that needs to execute a specific number of times. for loop statements get executed as long as condition is true. For loop contains three expressions, expression1 includes initializing the variables, expression 2 includes condition and expression3 includes increment or decrement of initialized variable.

Syntax:

```
for( expression1(init); expression2 (condition);expression3(++/- -))
        {
        statements;
        }
```

For example: A program to print first ten natural numbers using for loop.

```
#include<stdio.h>
void main()
{
        int i;
        for(i=1;i<=10;i++)
        printf("%d\t",i);
}
```

The output of above program is shown in Fig 4.6:

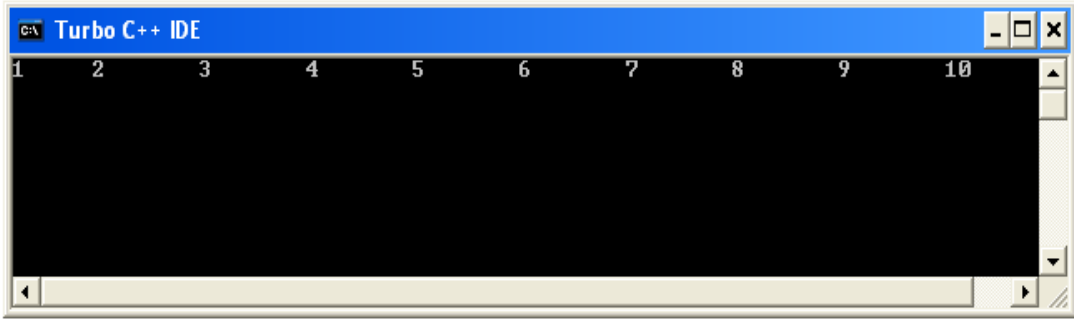


Fig 4.6

4.5.2 While loop:

In while-loop, statements get executed as long as condition is true. In while-loop it checks the condition first and then executes the statements later. So minimum number of execution times of statements is 0.

Syntax:

```
While(condition)
{
statements;
}
```

For example: A program to print odd numbers between 1 to 15 using while loop.

```
#include<stdio.h>
void main()
{
        int i=1;
        while(i<=15)
        {
        printf("\n%d",i);
        i=i+2;
        }
}
```

The output of above program is shown in Fig 4.7:

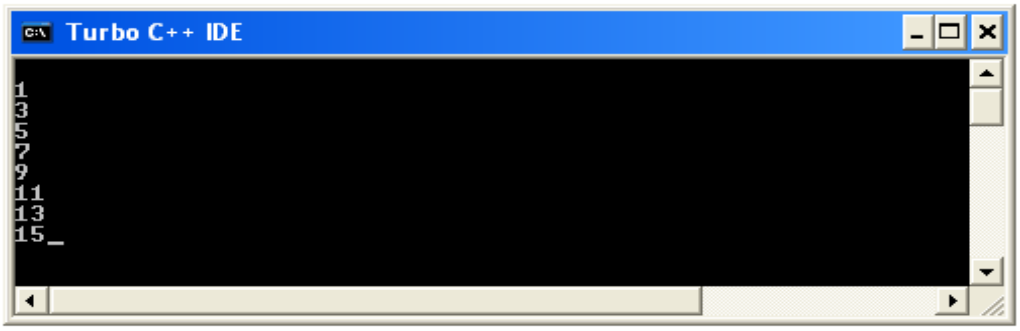


Fig 4.7

4.5.3 do-while loop:

In do-while loop, statements get executed as long as condition is true. In do-while loop, it executes the statements first and checks the condition later. A do-while loop is similar to a while loop, except that a do-while loop is guaranteed to execute at least onetime. So minimum number of execution times of statements is 1.

Syntax:

```
do
{
….
statements;
….
}while(condition)
```

For example: A program to print even numbers from 0 to n (entered by user) using do while loop.

```
#include<stdio.h>
void main()
{
        int i=0,n;
        printf("enter number for upper limit of even no.:");
        scanf("%d",&n);
        do
        {
                printf("\n%d",i);
                i=i+2;
        }
        while(i<=n);
}
```

The output of above program is shown in Fig 4.8:

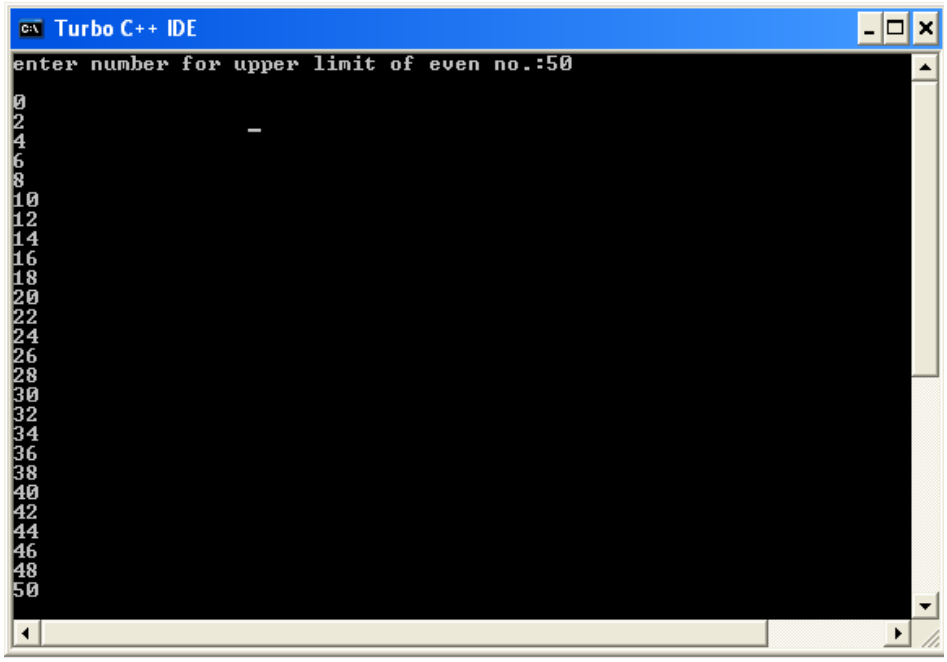


Fig 4.8

Exercise

Fill in the Blanks:

1. In C programming a number of statements which are executed in sequence is called _____ execution.
2. C programming provides two styles of control flow are _____ and _____.
3. In switch statement, control point comes to _____ when all the cases are false.
4. In C programming, _____ statements are used for altering the normal flow of a program.
5. _____ statement transfers control to a label.
6. The _____ statement is used in switch statement to exit.
7. In do-while loop, statements are execute at least for _____ time.

True/ False

1. Loops provide a way to repeat commands.
2. Sometimes it is desirable to skip some statements inside the loop. In such cases, break statement is used.
3. In do-while loop minimum number of execution of statements is 1.
4. return statement is used in functions.
5. Writing the if statement with-in another if is called as nested-if.

Short answer type questions:

1. What is sequential execution?
2. Define branching and looping?
3. Write names of conditional control statements?
4. How break statement is used in switch-case?
5. Define jumping.
6. Give syntax of for statement and explain its working.

Long Answer type questions:

1. Write a program in C to print names of months on entering the number of the month using switch- case statement.
2. What are loops? Explain.
3. Write a program in C to print even numbers upto 100 using do-while statement.
4. Explain conditional control statements.
5. Explain jumping statements.
6. What is the difference between while and do-while statements.

****

5. ARRAYS

Objectives of this lesson

5.1 Introduction:

An array is set of elements of similar type. An array is used to store a collection of data, but it may be said that an array as a collection of variables of the same type. An array is an ordered collection of elements that share the same name in sequential or consecutive memory locations. To refer to the elements of array we use **index**, array index starts from zero(0).

Arrays are the data types and allow us to use single variable to store a number of values with same data type. Every value of the array is called element. Each element of an array has its own storage location, which can be referenced like any other variable. The elements of arrays are stored in consecutive storage locations in memory and identified by the name of the array followed by a numbers in brackets; these numbers starts from 0 and increases each time by 1. Say each number 0, 1, 2, 3, 4,…. is called **subscript or an index**, individual elements of the array are called **subscript variables.** A specific element in an array is accessed by an index.

All arrays consist of contiguous memory locations. The lowest address refers to the first element and the highest address to the last element. Let us consider an example of an array contains integer data, named age and having capacity of eight elements. The starting index of the array is 0 and the highest index of array is 7 as shown below:

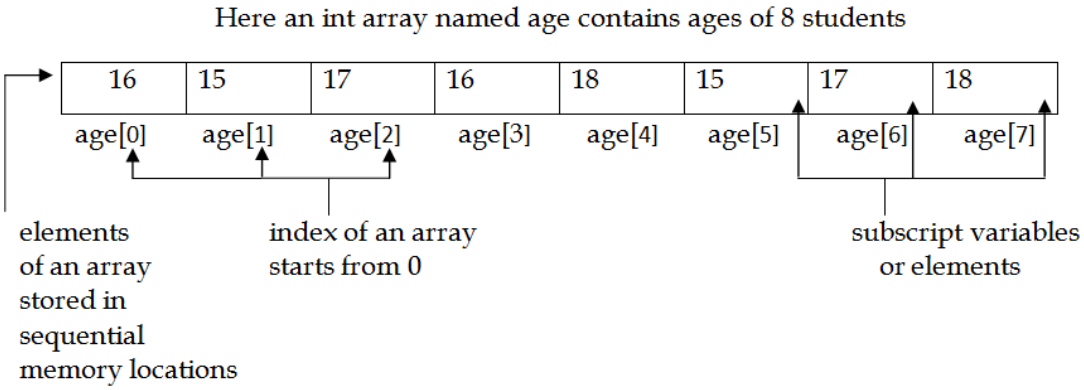


Fig 5.1

The lowest index value of an array is also known as **lower bound** of an array and the highest index value of an array is called **upper bound** of an array. The lower bound starts at 0 and upper bound is equal to n-1, where n is number of elements of an array.

- **Advantages of arrays:**
  - It is capable of storing many elements of same type under one name.
  - It allows random access of elements.

- **Disadvantages of arrays:**
  - Predetermining the size of array is must.
  - Memory wastage is there, if we don't know the exact size of an array.
  - To delete an element of an array we need to traverse throughout whole array.

5.2 Declaration of an array:

Like any other variable, an array must be declared before it is used in a C program.

Syntax for declaration

Data_type array_name [number of elements in array];

Here type is a valid data type (like int, float...), name is a valid identifier and the number of elements (enclosed in square brackets [ ]), specifies how many elements the array will store.
For example:

```
int age[8];
double balance[20];
```

int salary[50]

5.3 Initializing arrays:

Initialization means assigning some values to the variable that undergoes processing. Just like any ordinary variable initialization, the individual elements of an array can be initialized. All the initial values must be constants. Initialization can also be made to all array elements during the time of declaration.

For example:

Data type array_name[size] = {element1, element2, ……. element n};

For example:

int salary [5] = { 1000, 2500, 5000, 10000, 20000 };

The number of values between curly braces { } cannot be larger than the number of elements that we declare for the array between square brackets [ ].
Above declaration will create an array in the memory like this:

	0	1	2	3	4
salary	1000	2500	5000	10000	20000

Fig 5.2

5.4 Accessing Array Elements:

An element is accessed by using index of the array. This is done by placing the index of the element within square brackets after the name of the array. The syntax is:

name[index]

For example:

n = age[3];

The above statement will take 4th element from the array and assign the value to n variable. Notice that the fourth element of age is specified age[3], since the first one is age[0], the second one is age[1], and therefore, the third one is age[2] and so on. Let us try to understand three concepts of declaration, assignment and accessing array elements with following example:

```
#include<stdio.h>
void main()
{
        int salary[5];
        int i;
        for(i=1;i<=5;i++)
        {
                printf("enter the salary of %d employee:",i);
                scanf("\n%d",&salary[i]);
        }
        for(i=1;i<=5;i++)
        {
                printf("\nsalary of %d employee is Rs.%d/-",i,salary[i]);
        }
}
```

The output of above program is shown in Fig 5.3:

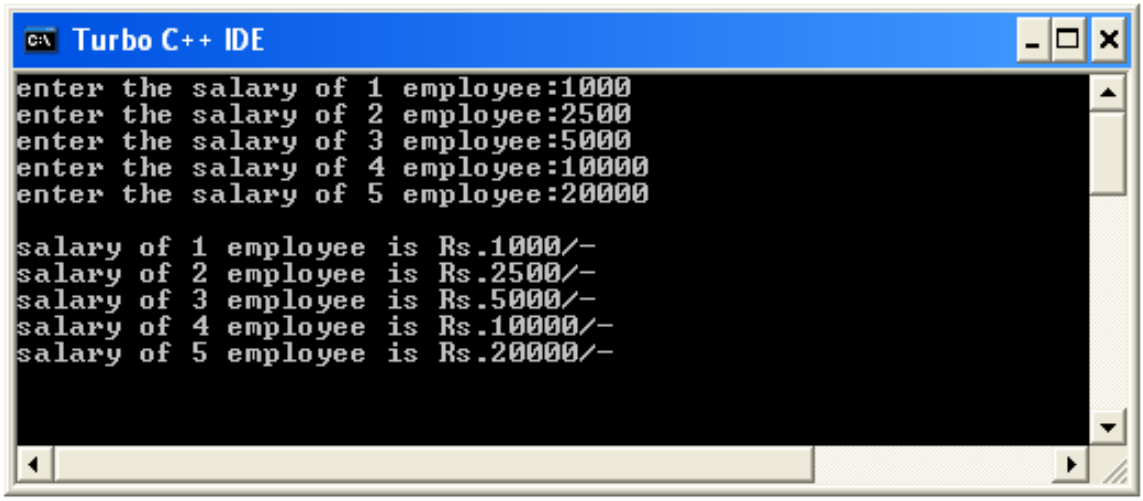


Fig 5.3

5.5 Entering Data into an Array:

For inputting data into the individual elements of the array, use of looping statements is preferred. Since array represents a collection of values, If the actual size of the array is known in advance, then for loop, is preferred over while loop or do – while loop.

As shown in above program the for loop causes the process of asking for and receiving the employee's salary from the user to be repeated 5 times. The first time through the loop, i has a value 0, so the scanf() statement will cause the value type to be

stored in the array element marks[0], the first element of the array. The loop will be continue to be executed until i becomes 4 (i<5). In the scanf() statement, we have used the "address of" operator (&) on the element marks[i] of the array. In doing so, we are passing the address of this particular array element to the scanf() function, rather than its value, which is required by scanf().

5.6 Manipulation of array elements:

C does not support performing any operation on the entire array. That is, the whole array cannot be processed as a single element. But, it allows the programmer to perform certain operations on a element by element basis. Array elements can be used for mathematical operations such as adding the different values of array element, finding the product of array elements. There are two ways to do these operation one by using the values given for the initialization and second by supplying values by the user through keyboard.

Here an array age has been initialized by five values and these are added to give sum of ages. for loop will proceed for 5 times to accept the assigned values. The variable result gives the output as the sum of the ages given in initialization.

Sum of Elements: let's write a program to add the array element supplied via user through keyboard

A program to find sum and average of array elements.

```
#include<stdio.h>
void main()
{
        int a[5],i;
        int sum=0,avg;
        for(i=1;i<=5;i++)
        {
                printf("enter the number:");
                scanf("%d",&a[i]);
        }
        for(i=1;i<=5;i++)
        sum=sum+a[i];
        avg=sum/5;
        printf("\naverage of array elements: %d",avg);
}
```

The output of above program is shown in Fig 5.4:

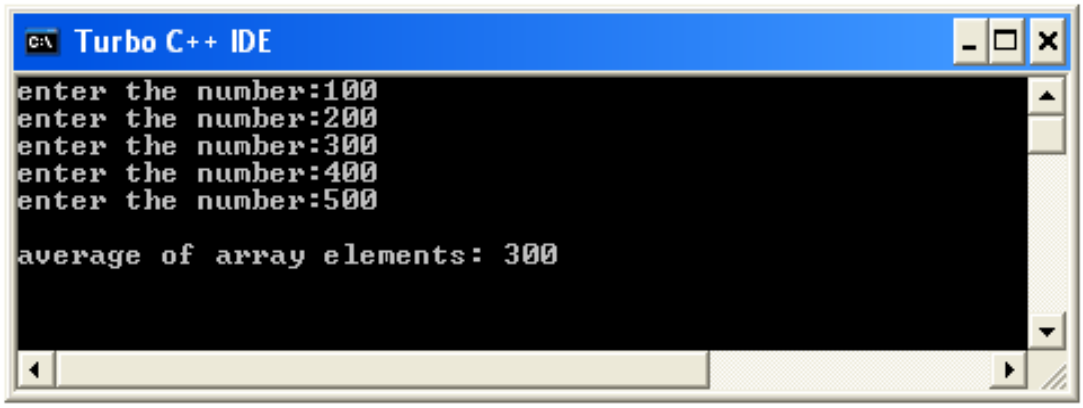


Fig 5.4

5.7 Types of arrays:

So far we have studied about single dimensional array as it contains only one dimension such as

int x[5];

Here array x is having single dimension defined in square bracket as 5.

C programming allows programmer to create array of arrays known as multi dimensional arrays. It is also possible for arrays to have dimensions more than one. Arrays can be mainly of three types:

a. Single dimensional arrays
b. Double dimensional arrays
c. Multi-dimensional arrays

5.7.4 Single dimensional arrays:

Single dimensional arrays can represent either one row or one column.

Syntax:

data_type array name[size];
data_type array name[ ]={ values list};

For example:

int x[5];

Here array x can hold maximum 5 integer elements at a time and 10 bytes of memory is allocated for the array.

int a[ ]={10,20,30,40,50,60,70 };

Here array size is 7.

Reading one dimensional array: To read a single dimensional array from the terminal we can use for loop.

```
For(i=0;i< size;i++)
Scanf("%d",&a[i]);
```

Writing one dimensional array: To write a single dimensional array to the terminal again for loop is used.

```
For(i=0;i< size;i++)
printf("%d",a[i]);
```

5.7.5 double (2-D) dimensional arrys:

A double dimensional array can represent the data in the form of rows and columns which means it can take two indexes. The two-dimensional or 2-D array is also called a matrix .

Syntax:

data_type array name[row size][col. Size]

For example:

int y[2][3];

Here array y can hold maximum of 6 elements (row x col) and 12 bytes of memory allocated.

int x[ ][ ]={(10,20,30),(40,50,60)};

Here the size is 2 x 3=6.

Reading two dimensional arrays: Following method is used for reading a 2-D array:

```
for(i=0;i<row;i++)
for(j=0;j<col;j++)
scanf(""d",&a[i][j]);
```

Writing two dimensional arrays: Following method is used for writing a 2-D array:

```
for(i=0;i<row;i++)
{
printf('\n');
for(j=0;j<col;j++)
scanf("\t%d",&a[i][j]);
}
```

A program to find sum of elements of two arrays using two-dimensional arrays.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int a[2][2],b[2][2],c[2][2],i,j;
        clrscr();
        printf("enter the value of 1st array:");
        for(i=1;i<=2;i++)
        for(j=1;j<=2;j++)
        scanf("%d",&a[i][j]);
        printf("enter the value of 2nd array:");
        for(i=1;i<=2;i++)
        for(j=1;j<=2;j++)
        scanf("%d",&b[i][j]);
        for(i=1;i<=2;i++)
        for(j=1;j<=2;j++)
        c[i][j]=a[i][j]+b[i][j];
        for(i=1;i<=2;i++)
        for(j=1;j<=2;j++)
        printf("\nsum of elements of two arrays %d\n",c[i][j]);
        getch();
}
```

The output of above program is shown in Fig 5.5:

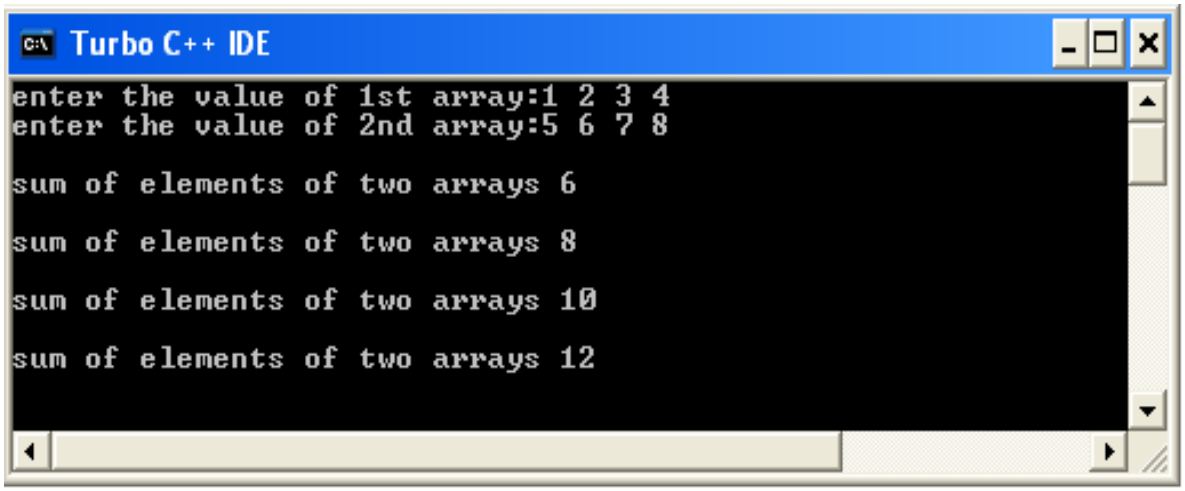


Fig 5.5

5.7.6 multi (3-D) dimensional arrays:

A multi dimensional array can represent the data in the form of number of arrays, rows and columns by taking two or more indexes. Three dimensional or 3-D arrays use three indexes. A three-dimensional array can be thought of as an array of arrays of arrays.

For example:

```
int arr[2][4][3]={
                    {
                        { 1, 5, 4},
                        { 2, 8, 6 },
                        { 3, 7, 2 },
                        { 4, 6, 1 }
                     },
                    {
                        { 5, 6, 9 },
                        { 6, 4, 3 },
                        { 7, 1, 5 },
                        { 8, 3, 4 }
                    },
                    } ;
```

Here the outer array consists of two arrays, each array is a two-dimensional array which contain four rows and three columns.

Exercise

Fill in the Blanks:

1. _____ is a set of similar type of elements.
2. The index of an array starts from _____.
3. Individual elements of the array are called _____ variables.
4. A two-dimensional array is also known as _____.
5. To refer to the elements of array we use _____.

True/ False

1. Reading and writing of arrays is done by using for Loop.
2. Initialization cannot be made to array elements during the time of declaration.
3. The whole array can be processed as a single element.
4. The number of elements of array which it can store is enclosed in square brackets [ ].
5. The three dimensional array uses three indexes.

Short answer type questions:

1. Define arrays. Write its types.
2. What do you understand by array index?

3. How the array elements are accessed?
4. What are upper and lower bound of an array?
5. How can an array be declared and initialized?

Long Answer type questions:

1. Explain different types of arrays.
2. What are the advantages and disadvantages of arrays?
3. Write a program in C to calculate sum and average of ten real numbers stored in float array?
4. How will you enter data into an array?

****

6. String Handling Functions

Objectives of this lesson

6.1 Introduction

String is a combination of characters. Strings are used to hold text data, consisting of letters, numerals, punctuation marks, and other symbols within double quotation symbols .It may be considered as a one dimensional array of characters. Every string is terminated by null character '\0' whose equivalent integer value is zero. Let us consider a string of 6 characters ''punjab'' which is stored in the system memory as an array of characters as shown in figure below:

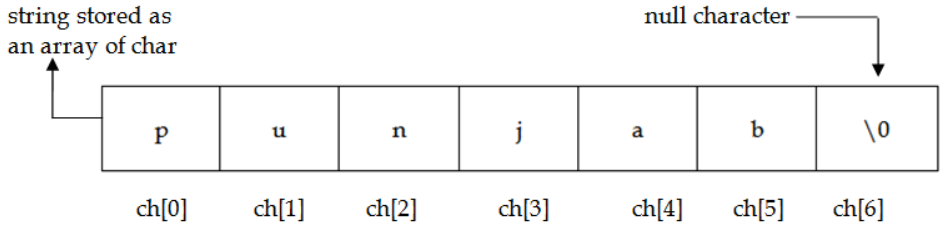


Fig: 6.1

6.2 Declaration and Initializing String Variables:

The initialization of a String can be done in two different ways:

```
char x[ ]={'a','m','a','r','\0'};
        (or)
char x[ ]=''amar'';
```

It is also possible to write set of strings:

```
char x[5][10];
```

Here first size 5 indicates no. of strings and second size 10 indicates no. of characters in each string.

```
char x[ ][ ]={ ''amar'',''vijay'',''kiran'',''ravi''};
```

6.3 READING and WRITING STRINGS:

Strings can be read in or written out using the standard input/output statements provided in C language. Standard input device is termed as "standard input" and output going to standard output device is termed as "standard output". In C standard input device is keyboard and standard output device is monitor. Standard input and output is also termed as console input and output. The following figure illustrates the concept of I/O statements used in C.

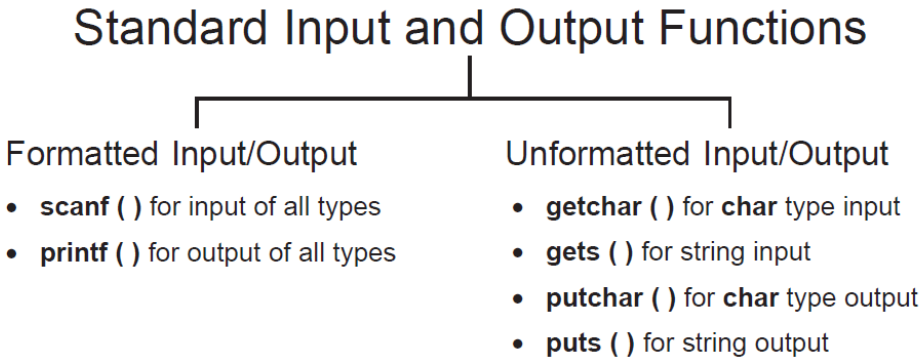


Fig: 6.2

6.3.1 Formatted Input/ Output

There are two function which are used as formatted I/O functions: scanf() and printf(). We have used these functions in previous chapters for reading and writing purposes, from and to the terminal.

Scanf(): It is used for reading from standard input device that is keyboard.

Syntax:

```
Scanf(format_string, list_of_variable_addresses);
```

For example:

```
Scanf("%d",&a);
```

Printf(): It writes to strandard output device that is monitor.

Syntax:

printf(format_string, list_of_ variables);

For example:

printf("%d",a);

6.3.2 Unformatted Input/ Output

Unformatted Input/ Output functions for single character is getchar() and putchar() functions. Unformatted I/O functions for strings are gets() and puts() respectively. Let us now discuss in detail these functions which are known as unformatted I/O functions:

6.3.2.1 getchar():

Function **getchar()** is used for char type input. When it reaches the end of file it is reading, thereafter it returns the character represented by '\0'

Syntax:

ch = getchar();

Here ch is char variable, and function getchar() picks up a character sent by keyboard and returns it, which is assigned to char variable ch. The characters typed by the user are processed by getchar() function only after pressing the enter key.

6.3.2.2 gets():

Function **gets()** is used to accept a multi-word line of text from keyboard that is terminated by pressing enter key. We can include punctuation symbols and whitespace characters except newline character, in this line of text.

Syntax:

gets(line);

Here line is an array of char which is passed to function gets() as an argument.

6.3.2.3 putchar():

Function **putchar()** is used for char type output. This function puts one character out on the standard output (usually the terminal) each time it is called. We can think of function putchar() as a counterpart of function getchar()

Syntax:

putchar(ch);

This will display the character stored in ch on the screen.

6.3.2.4 puts():

The function **puts()** is a counterpart of function gets(). Function gets() is meant for input the string of characters, whereas function **puts()** is meant for output of strings of characters.

Syntax:

puts(line);

Here array named line is passed to function puts() as an argument. Function puts() accept only **char** type array as an argument.

6.4 Reading Strings using scanf() function:

The function scanf() with %s format specification is used to read the character string from the terminal. The following is the format for scanf():

```
char address[20];
scanf(%s,address);
```

6.4.1 Drawback of using scanf() function for reading string:

There is drawback of using scanf() statement for inputting a string. scanf() statement just terminates the statement as soon as it finds a blank space in the string. scanf() is incapable of receiving multi-word strings, So instead of using scanf() statement we can use gets() statement for inputting the multi-word string.

For eg. "computer language" is a multi word string, but as soon as the blank space comes the string will be terminated and only one word "computer" will be received. So in such cases, gets() statement must be used to avoid the termination of string in-between.

6.5 Arithmetic operations on characters:

C allows us to manipulate the characters just as numbers. Whenever the system encounters the character data it is automatically converted into a integer value by the system. All the characters are stored in their corresponding ACSII codes in the system. We can represent a character as an integer by using the following method.

```
X='a';
printf("%d\n",x);
```

Above printf() statement will display integer value 97 on the screen. So, we can also perform arithmetic operations on characters.

for example x='y'-1; is a valid statement.

Figure shown below shows the corresponding ASCII values of the characters and symbols:

032	space	052	4	072	H	092	\	112	p
033	!	053	5	073	I	093	]	113	q
034	"	054	6	074	J	094	^	114	r
035	#	055	7	075	K	095	_	115	s
036	$	056	8	076	L	096	`	116	t
037	%	057	9	077	M	097	a	117	u
038	&	058	:	078	N	098	b	118	v
039	'	059	;	079	O	099	c	119	w
040	(	060	<	080	P	100	d	120	x
041	)	061	=	081	Q	101	e	121	y
042	*	062	>	082	R	102	f	122	z
043	+	063	?	083	S	103	g	123	{
044	,	064	@	084	T	104	h	124	|
045	-	065	A	085	U	105	i	125	}
046	.	066	B	086	V	106	j	126	~
047	/	067	C	087	W	107	k	127	delete
048	0	068	D	088	X	108	l		
049	1	069	E	089	Y	109	m		
050	2	070	F	090	Z	110	n		
051	3	071	G	091	[	111	o		

Fig: 6.3

6.6 STRING FUNCTIONS:

C complier provides string handling functions which are defined in the header file<string.h>. We must include this header file in our program by using #include directive, in order to perform these operations on strings. Various string functions are as follows:

strcpy( ), strcat( ), strlen( ), strrev( ), strlwr( ), strupr( ), strcmp( ), etc.

6.6.1 strlen() function:

This function counts and returns the number of characters in a particular string. The length of the string does not include a null character.

The syntax of strlen() is as follows:

n=strlen(string);

Where n is the integer variable which receives the value of length of the string.

/*write a C program to find the length of the string using strlen() function*/

```
#include<stdio.h>
#include<string.h>
main ( )
{
char name[100];
int length;
printf(''enter a string:'');
gets(name);
length=strlen(name);
printf(''\n Number of characters in the sting is =%d'',length);
}
```

6.6.2 strcat() function:

When we combine two strings, we add the characters of one string to the end of the other string. This process is called as concatenation. The strcat() function is used to joins 2 strings together.

The syntax of strcat() is as follows:

strcat(string1,string2);

string1 & string2 are the character arrays. When the function strcat is executed string2 is appended to the string1.

/*write a c program to concatenate two strings using strcat() function*/

```
#include<stdio.h>
#include<string.h>
void main()
{
char x[10],y[10];
clrscr();
printf(''enter two strings to x,y'');
scanf(''%s%s'',&x,&y);
strcat(y,x)          /*here y characters added to x.*/
printf(''\n x string is =%s'',x);
printf(''\n y string is =%s'',y);
getch( );
}
```

6.6.3 Strcmp() function:

This string function is used to compare two strings. It returns the value 0 when both strings are equal, returns 1(positive value) when string1 is greater than string 2, and returns negative when the string 1 is less than string 2.

The syntax of strcmp() is as follows :

strcmp(string1,string2);

Here string1 is compared with string2 to know whether they are same or not.

/*write a c program to compare two strings using strcmp() function*/

```
#include<stdio.h>
#include<string.h>
void main ( )
{
char x[10],y[10];
int L;
clrscr();
printf(''enter two strings to x,y'');
scanf(''%s%s'',&x,&y);
L=strcmp(x,y);
if(L>0)
printf('' x is big'');
else if(L<0)
printf ('' y is big'');
else
printf(''both are equal'');
getch( );
}
```

6.6.4 strcpy( ) function:

This string function is used to copy the one string characters to another string. This function copies second string to first string.

The syntax for strcpy() is as follows:

strcpy(string1,string2);

Here string1 & string2 are the character arrays. When the function strcpy is executed string2 is copied to the string1.

/*write a c program to copy one string to another using strcpy() function*/

```
#include<stdio.h>
#include<string.h>
void main ( )
{
char x[10],y[10];
clrscr();
printf (''enter a string to x'');
scanf(''%s'',&x);
strcpy(y,x)              /*here x characters copied to y.*/
printf(''\n y string is =%s'',y);
getch( );
}
```

6.6.5 strrev( ) function:

This string function is used to convert the given string characters into reverse order.

The syntax for strrev() is as follows:

strrev(string);

Here string is character arrays. When the function strrev is executed string gets reversed.

/*write a c program to reverse a string by using strrev() function*/

```
#include<stdio.h>
#include<string.h>
void main ( )
{
char x[10];
clrscr();
printf(''enter a strings to x'');
scanf(''%s'',&x);
printf(''\n x reverse string of x is =% s'',strrev(x));
getch( );
}
```

6.6.6 strlwr( ) function:

This string function is used to convert the given string characters in to lower-case format.

The syntax for strlwr() is as follows:

strlwr(LOWER);

Here LOWER is a string. When the function strlwr is executed string LOWER is converted to lower-case letters "lower".

6.6.7 strupr( ) function:

This string function is used to convert the given string characters in to uppercase format.

The syntax for strupr() is as follows:

strupr(upper);

Here upper is a string. When the function strupr is executed string upper is converted to upper-case letters "UPPER".

/*write a c program to convert a string to uppercase and lowercase by using strupr() and strlwr() functions respectively*/

```
#include<stdio.h>
#include<string.h>
void main ( )
{
char x[10];
clrscr( );
printf(''enter a string'');
scanf(''%s'',&x);
strupr(x);
printf(''\n given string in upper case=%s'',x);
strlwr(x);
printf(''\n given string in lower case=%s'',x);
getch( );
}
```

Exercise

Fill in the Blanks:

1. A _____ is a combination of characters.
2. Every string is terminated by null character whose equivalent integer value is _____.
3. The initialization of the strings can be done in _____ different ways.
4. String handling functions are defined in _____ header file.
5. The function scanf() with _____ format specifier is needed to read the character.

True/ False

1. Every string is terminated by null character '\0'.
2. Standard input and output is also termed as console input and output.
3. The characters typed by the user are processed by getchar() function only after pressing the enter key.
4. Whenever the system reads the character data it is automatically converted into its corresponding ASCII value by the system.
5. We cannot manipulate the characters as numbers in C.

Short answer type questions:

1. Define string. How can it be represented in the system's memory?
2. How to declare and initialize a string?
3. What is drawback of reading a multi-word string with scanf() function?
4. Explain strlwr() and strupr() function?
5. What is the difference between strcat() and strcpy() functions?
6. Which header file is included in the C program to perform operations on strings? List the different operations which can be performed on strings.

Long Answer type questions:

1. Explain the standard I/O functions for reading and writing stings.
2. With which function can you find the length of a string? Write a program in C to find the length of a string.
3. How the comparison of two strings is done? Write a program to compare two strings entered through keyboard by the user.
4. Write a program to reverse the string "computer"?
5. How the manipulation of characters is possible as numbers? Explain.

****

7. User defined functions

Objectives of this lesson

7.1 Introduction

In earlier chapters we have read that C supports the use of library functions, which are used to carry out a number of commonly used operations or calculations. C also allows programmers to define their own functions for carrying out various individual tasks.

7.2 What is a function?

A function is a self-contained program segment that carries out some specific well-defined task. A function is a reusable block of statements that gets executed on calling. It can be treated as sub program.

Types of Functions: In C Language function can be divided into two categories:

7.2.1 Built-in or Library functions
7.2.2 User defined functions

7.2.1 Library functions: These are also called as built in functions. Library functions carry out various commonly used operations or calculations. Some functions return a data item to main program, others indicate whether a condition is true or false by returning 1 or 0 respectively, still others carry out specific operations on data items but do not return anything. Functionally similar library functions are usually grouped together as object programs in separate library files. These library files are supplied as a part of each C compiler and are included in header files.

In order to use a library function it may be necessary to include certain specific information within the main portion of the program. This information is

generally stored in special files supplied with the compiler. Thus, the required information can be obtained simply by accessing these special files. This is accomplished with the preprocessor statement.

 # include <filename>

The list of some commonly used library functions are:

i) abs(i) to determine absolute value of i.
ii) exp(i) raise e to the power i.
iii) log(d) determine natural logarithm of d.
iv) pow(d1,d2) returns d1 raised to the d2 power
v) putchar(c) send a character to the standard output device
vi) sqrt(d) return the square root of d.

These functions can be included in C programs with the help of header files .For eg. all input/ output functions be used by the preprocessor directive like #include<stdio.h> and all the mathematical functions can be used by the preprocessor directive like # include <math.h>.

7.2.2 User defined functions: These Functions are defined by the user to perform a task according to the user's requirement. Such functions are called as user-defined functions. User can include any number of functions in the program, for which declaration and definition must be provided.

7.3 Need of functions: The User defined functions makes the programming easier. A large program can be divided into number of modules or sub programs and each module can be treated as an independent part of main program. The following are the advantages of functions:

1. **MODULAR APPROACH:** The use of user-defined functions allows a large program to be broken down into a number of smaller, self-contained components, each of which has some unique purpose. Thus a C program can be modularized through the intelligent use of such functions. For example many programs require a particular group of instructions to be accessed repeatedly from several different places within a program. The repeated instruction can be placed within a single function, which can then be accessed whenever it is needed.
2. **REDUCE REDUNDANCY:** The use of a function avoids the need for repeating programming of the same instructions. The repeating of programming of the same instructions is called redundancy. With the help of functions it is reduced.
3. **REUSEABILITY:** We can use functions any no. of times whenever required in the program. It is called reusability.

4. **GOOD PROGRAMMING APPROACH:** The decomposition of a program into individual program modules is generally considered to be an important part of good programming.
5. **EASY DEBUGGING**: As the program is divided into no. of modules which are smaller and independent, the error or bug detection is easy. So the debugging (means correcting errors) becomes an easy process.
6. **BUILDING LIBRARY:** It provides the facility to the programmer to create a library of the commonly used functions. This reduces time and space complexity. It facilitate with the feature of portability.

7.4 Defining Function

In order to use a user defined function in the program, programmer must follow the following three simple steps:

1. Define the function
2. Declare the function
3. Use the function in the main code

The definitions of functions may appear in any order in a program file because they are independent of one another. A function can be executed from anywhere within a program. Once the function has been executed, control will be returned to the point from which the function was accessed. A function contains parameters or arguments through which information is passed to the function and from functions information is returned via the return statement. It is not necessary that every function must return information or any value to the main program. There are some functions which do not return any information, For example the system defined function **printf().**

Before using any function it must be defined in the program:

7.5 Structure / Form of a function:

The structure or form of a user defined function is shown below in fig 7.1.

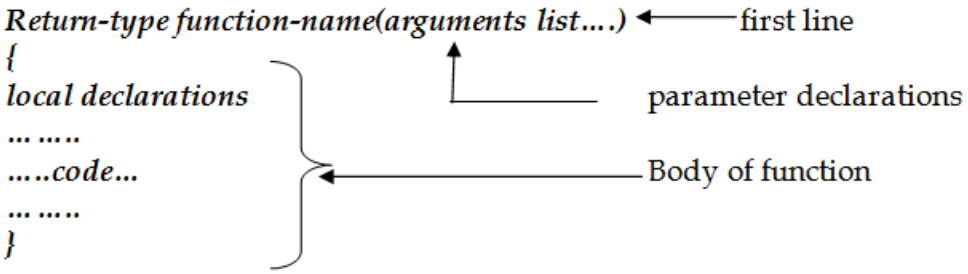


Fig: 7.1

Function definition has three principal components:

1. the first line,
2. the parameter declarations and
3. the body of the functions.

The first line of a function definition contains:

Return-type function-name(arguments list….)

1. the data type of the information return by the function,
2. followed by function name,
3. a set of arguments or parameters, separated by commas and enclosed in parentheses.

An empty pair of parentheses must follow the function name if the function definition does not include any argument or parameters.

The general term of first line of functions can be written as:

data-type function-name (formal argument 1, formal argument 2…formal argument n)

The formal arguments allow information to be transferred from the calling portion of the program to the function. They are also known as parameters or formal parameters. These formal arguments are called actual parameters when they are used in function reference. All formal arguments must be declared after the definition of function.

The remaining portion of the function definition is a compound statement that defines the action to be taken by the function. This compound statement is sometimes referred to as the **body of the function**. This compound statement can contain expression statements, other compound statements, control statements etc.

Information is returned from the function to the calling portion of the program via the return statement. The return statement also causes control to be returned to the point from which the function was accessed. In general terms, the return statement is written as

return expression;

The value of the expression is returned to the calling portion of the program. The return statement can be written without the expression. Without the expression, return statement simply causes control to revert back to the calling portion of the program without any information transfer. The point to be noted here is that only one expression can be included in the return statement. Thus, a function can return only one value to the calling portion of the program via return. But a function definition can include multiple return statements, each containing a different expression. Functions that include multiple branches often require multiple returns.

It is not necessary to include a return statement altogether in a program. If a function reaches the end of the block without encountering a return statement, control simply reverts back to the calling portion of the program without returning any information.

7.6 Arguments and Parameters

In function a definition a set of arguments or parameters are enclosed in parentheses, and are separated by commas.

data-type function-name (formal argument 1, formal argument 2…formal argument n)

These variables used in the function definition are called parameters. They are also referred to as formal parameters, because they are not the accepted values. They receive values form the calling function. Parameters must be written within the parentheses followed by the name of the function, in the function definition.

The formal arguments allow information to be transferred from the calling portion of the program to the function. They are also known as parameters or formal parameters. These formal arguments are called actual parameters when they are used in function reference. The names of actual parameters and formal parameters may be either same or different but their data type should be same. All formal arguments must be declared after the definition of function.

7.6.1 Argument List

The argument list may be defined as valid variable names separated by commas in the function definition. This list must be enclosed by parenthesis. These argument variables receive values from the calling function. They provide the means for data communication from the calling function to the called function. All argument variables must be declared with their types before the body of the function.

(formal argument 1, formal argument 2..…formal argument n)

7.7 USING A FUNCTION IN PROGRAM:

A function can be used by specifying its name, followed by a list of parameters or arguments enclosed in parentheses and separated by commas. If the function call does not require any arguments, an empty pair of parentheses must follow the function's name.

The parameters in the body of the functions are called **actual parameters or arguments**. They may be expressed as constants, single variables or more complex expressions.

The User defined functions can be called in a C program by two methods:

Call by value
Call by reference

7.7.1 call by value :

In call by value simple variables are sent to the functions in the form of arguments. In this case only values of the arguments are copied to the corresponding function parameters. When we make some changes to variables in the function, it does not affect the variables of the calling function.

7.7.2 call by reference :

In call by reference address of variables are sent to the functions in the form of arguments. In this case addresses of the argument variables are copied to the corresponding function as pointer type parameters. When we make some changes to variables in the function, it affects the variables of the calling function.

Let us consider an example of function.

```
#include <stdio.h>
main()
        {
                int a,b,c;
                printf("Enter two numbers");
                scanf("%d%d", &a,&b);
                c=sum_v(a,b,);
                printf("\n The sum of two variables is %d\n,",c);
        }
        sum_v(a,b)
                {
                int a,b
                {
                int d;
                d=a+b;
                return d;
                }
                }
```

This program returns the sum of two variables a and b to the calling program from where sum_v is executing. The sum is present in the variable c through the 'return d;' statement. There may be several different calls to the same function from various places within a program. The actual parameters may differ from one function call to another. Within each function call, the actual arguments must correspond to the formal arguments in the function definition, i.e. the

number of actual arguments must be same as the number of formal arguments and each actual argument must be of the same data type as its corresponding formal argument.

Exercise

Fill in the Blanks:

1. There are two types of functions in C language, _____ and _____.
2. Built-in functions are also called _____.
3. The _____ statement causes control to be returned to the point from which the function was accessed.
4. The parameters in the body of the functions are called _____.
5. An _____ pair of parentheses in function shows that function does not include any argument or parameters.

True/ False

1. A function is a reusable block of statements.
2. The return statement is written as **return expression;.**
3. Use of functions increases redundancy.
4. Use of functions makes debugging an easy process .
5. The number of actual arguments must be same as the number of formal arguments.

Short answer type questions:

1. What is a function?
2. How many types of functions are there? Write names.
3. What are library functions?
4. Define the structure of a function?
5. What is the difference between parameters and arguments?
6. What do you understand by modular approach?

Long Answer type questions:

1. Write the advantages of using functions.
2. What are user-defined functions? Why we need user-defined functions?
3. How can we use a user-defined function in a C program?
4. Write a program to calculate simple interest using a user defined function.
5. State the difference between library functions and user-defined functions.

****

8. Internet & E-Governance

Objectives of this lesson

8.1 Introduction:

The **internet** is the global network of computers. It is a world-wide system of interconnected **computer networks** that use the standard **Internet rules** (TCP/IP) to link several devices worldwide. It is a network of networks that consists of number of private, public, academic, business, and government networks. These are inter-connected by means of electronic devices, such as modems, hubs, routers and switches etc and wireless and optical networking technologies. The Internet possesses a wide

range of information resources and services, such as the inter-linked web pages and **applications** of the **World Wide Web** (WWW), such as email, and **file sharing.** The computers can be linked in several ways such as hard wired, fibre optics, wireless technology or by means of satellite links. The computers that make up the internet are for private, public, academic and business use. The Internet is a powerful communications tool. It empowers its users with instantaneous access to data, tools and the power to accomplish many different tasks.

The World Wide Web (WWW) requires a tool (software) called a **browser** to locate resources on the WWW and to take advantage of its special searching and multimedia features. It may be possible to use a browser for all the communication functions of e-mail, newsgroups, and downloading files.

8.2 Applications of Internet

In today's world, the internet is treated as one of the biggest invention. It has a large number of uses:

1. **Search:** A large number of people are using internet for research purposes to download any kind information by using internet.
2. **Communication**: It is used for sending and receiving message from one another through internet by using electronic mail. Some of the web sites providing these services are gmail.com, yahoomail.com, Hotmail.com, rediffmail.com, and many more.
3. **Finding books and study material**: Books and other study material stored around the world can be easily located through internet. Latest encyclopaedias are also available online.
4. **Online Examination and Results**: You can fill online forms for any board, university and competitive exams on internet and can view their results. You can also apply online for various scholarships schemes.
5. **Health and medicine:** Internet provides information and knowledge about field of health and medicine. People can have information about various diseases and can receive help.
6. **Travel:** One can use internet to gather information about various tourist places. It can be used for booking Holiday tours, hotels, train and flights. Some of the web sites providing this service are indiatravelog.com, rajtravel.com, makemytrip.com etc.
7. **Entertainment**: One can download jokes, songs movies, and latest sports updates through internet. Many web sites provide these services.
8. **Shopping:** Internet is also used for online shopping. By just giving accounts details you can perform the sale and purchase of any new and old product.

9. **Online Banking:** You can perform various bank related transactions through online banking, such as transferring funds and depositing fee etc.
10. **Payment of Bills and Taxes:** You can even pay your water, electricity and telephone bills and taxes also.
11. **Job searches**: Getting information regarding availability of job in different sectors and areas. You can publish your resume in online for prospective job. Some of the web sites providing this service are naukri.com, monster.com, summerjob.com, recuritmentindia.com etc.
12. **Business use of internet:** Different ways by which internet can be used for business are:
    a. Information about the product can be provided online to the customer.
    b. Provide market information to the business.
    c. It helps business to recruit talented people.
    d. Help in locating suppliers of the product.
    e. Fast information regarding customers view about company's product.
    f. Eliminate middle man and have a direct contact with contact with customer.
    g. Providing information to the investor by providing companies back ground and financial information on web site.

13. **Stock market updates:** You can sell or buy shares while sitting on computer through internet. Several websites like ndtvprofit.com, moneypore.com, provide information regarding investments.

8.3 History of Internet

The Internet was originated in the late 1960s when the United States Defense Department developed ARPAnet (Advanced Research Projects Agency network). It was an experimental network of computers designed to exchange information about research mostly military research. In the mid-1980s, when desktop computer workstations became popular, organizations wanted to connect their local area networks (LANs) to ARPAnet, as if computers could link together and share resources, everyone would benefit. In 1986, the National Science Foundation (NSF) established its network, NSFnet, to allow researchers across the country to share access to a few expensive supercomputers, creating the NSFnet backbone.

In 1990, the original Defence Department network was retired, its work having been taken over by NSFnet. 1991 was a big year for the Internet: The National Research and Education Network (NREN) were founded and the World Wide Web was released. The Internet is still dominated by scientists and other academics, but begins to attract public interest. With the release of the Mosaic Web browser in 1993 and Netscape in 1994, interest in and use of the World Wide Web exploded. More and more communities become wired, enabling direct connections to the Internet. Internet is a global collection

of interconnected network of computers. The internet is made up of millions of computers linked together around the world.

The internet is often described as "a network of networks" because all the smaller networks of organizations are linked together into the one giant network called the internet.

8.4 Facilities of Internet

The Internet is a valuable tool for accessing a wide variety of on-line materials. With the recent development of tools such as the Hyper-Text Transfer Protocol (HTTP), Mosaic, low-cost access, and high-speed modems, the Internet has become more intuitive and accessible to millions of people. The Internet is a group of thousands of computer networks that are tied together via leased lines, or radio frequencies that share a common protocol called the Internet Protocol or IP.

Access to the Internet is available almost everywhere on the globe. Each one on the globe can use the resources available on the Internet 24 hours a day. Several new tools have made the Internet easier to navigate and access. The concept of a Uniform Resource Locator (URL), essentially an address, a URL contains information such as the protocol, the site, and the directory and file name that are needed to retrieve a document. With this tremendous growth in technology and ease of access, there has been an explosion in resources. One of the basic advantages of the Internet is that it supports the bi-directional flow of information. The Internet is flexible and cost effective. The various facilities of internet are discussed in following sections:

8.4.1 Electronic mail (E-Mail):

E-mail stands for electronic mail. It is a method of exchanging digital messages from an author to one or more recipients. Email is used for composing, sending and receiving messages over Electronic communication systems like Internet. This internet resource is the most widely used. An electronic mail message can be sent to one or more e-mail addresses. It delivers a message almost instantaneously to anyone in the world. The messages go into electronic mailboxes for users to read at their convenience. There are many software platforms available to send and receive. Popular email platforms include Gmail, Hotmail, Yahoo! Mail, Outlook, and many others.

Whoever has access to Internet can have a unique e-mail address. E-mail address always have an @ symbol in it. For example ***user@yahoo.com*** is an e-mail address. The symbol @ separates the username from the location where their mail is stored.

8.4.1.1 Properties of E-mail:

i) **Cost:** It is the cheapest mode of communication on Internet. Users save not only postage cost, but it's also typically cheaper than faxing, saving the cost of fax paper and long-distance telephone charges. Long

messages cost the same as short ones, and sending a message to Switzerland costs the same as sending one across town.

ii) **Speed:** Most e-mail messages are delivered within a few minutes or even seconds. You could correspond back and forth with another several times in the course of one day.

iii) **Convenience:** With e-mail, User writes electronically on the computer and then sends the message on its way with a few keystrokes. There is no use of paper and no postage.

iv) **Automatic reply:** User can automatically reply messages and send copies of a message to many people

v) **Address Book:** User can manage addresses in an address book and can retrieve them when needed.

vi) **Automatic date and time:** Emails are automatically date and time stamped. User can attach electronic signatures also with the email. Files, graphics or sound can also be sent as attachments, often in compressed formats with message.

8.4.2 Website:

A website is a collection of web pages (documents that are accessed through the Internet). A web page is what you see on the screen when you type in a web address, click on a link, or put a query in a search engine. A web page can contain any type of information, and can include text, colour, graphics, animation and sound. You locate information on the Web through Web sites, which are made up of one or more Web pages. Web sites contain information about businesses, educational institutions, research facilities, or the personal interests of an individual user. Generally, people look at websites for two primary reasons:

1. To find information they need. This could be anything from a student looking for pictures of frogs for a school project, to finding the latest stock quotes, to getting the address of the nearest restaurant or anything else.

2. To complete a task. Visitors may want to buy the latest best-seller, download a software program, or participate in an online discussion about a favorite hobby.

Finally we can say, a website is a collection of Web pages, images, videos or other digital assets that is hosted on one or several Web server(s), usually accessible via the Internet, cell phone or a LAN.

8.4.3 Chatting:

Online chatting is a way of communication over the Internet that offers a real-time transmission of short text messages from sender to receiver. As the massages are short and quickly responded, Thereby, a feeling of a spoken

conversation is created. This distinguishes chatting from other text-based online communication such as email. The word **online chat** comes from the word chat which means "informal conversation". Online chat includes web-based applications that allow communication, by using tools such as instant messengers, Internet Relay Chat (IRC), etc. Chatting is similar to a telephone conversation - except it's text over the internet. When you chat with someone over a network you type into the computer instead of speaking through the telephone.

Online chat may be text-based or video-based (webcams), one-on-one chat or one-to-many that is group chat.

8.4.4 Voice Chatting:

Voice chat is a modern mode of communication used on the Internet. In voice chat, you can speak and hear during chat rather than entering in text through keyboard.

8.4.5 Video Conferencing:

Conducting a conference between two or more participants at different sites by using computer networks to transmit audio and video data is known as video conferencing. It is a technology that allows users in different locations to hold face-to-face meetings without having to move to a single location. This technology is particularly convenient for business users in different cities or even different countries because it saves the time and expense associated with business travel. Uses for video conferencing include holding routine meetings, negotiating business deals and interviewing job candidates.

In point-to-point video conferencing, each participant has a video camera, microphone, and speakers mounted on her/his computer. As the two participants speak to one another, their voices are carried over the network and delivered to the other's speakers, and whatever images appear in front of the video camera appear in a window on the other participant's monitor.

Multipoint video conferencing allows three or more participants to sit in a virtual conference room and communicate as if they were sitting right next to each other.

8.3.5.1 Benefits of Video Conferencing

1. **Support for Environmental Initiatives:** Video conferencing is an obvious "green" technology, allowing organizations to save energy use by dramatically reducing the need to travel. This rich communications technology offers new possibilities for schools, colleges, and libraries, who are now using video-conferencing systems for a variety of purposes, including formal instruction (courses, lessons, and tutoring), connection with guest speakers and experts, multi-school project collaboration,

professional activities, and community events across distance at any time, from wherever they are.

2. **Reduced Travel Costs:** With videoconferencing travelling cost has reduced to a greater extent. Videoconferencing is a direct replacement of business trips, and there is no cost to add additional employees to a virtual meeting.
3. **Improved Communication:** During a videoconference you can see the facial expressions and body language of conference participants, which leads to faster and more effective collaboration.

8.4.6 E-commerce

Facilitating business on the Internet is called **ecommerce**. Ecommerce is short for "electronic commerce." It is a type of business model that enables a firm or individual to conduct business over an electronic network, typically the internet. Electronic commerce operates in all four of the major market segments: business to business, business to consumer, consumer to consumer and consumer to business. It can be thought of as a more advanced form of mail-order purchasing through a catalogue. Almost any product or service can be offered via ecommerce, from books and music to financial services. Ecommerce revolve around buying and selling online. Any form of business transaction conducted electronically is ecommerce.

Following are the uses of ecommerce:

1. **Online Shopping**: Buying and selling goods on the Internet is one of the most popular examples of ecommerce. Sellers create store-fronts that are the online equivalents of retail outlets. Buyers browse and purchase products with mouse clicks.
2. **Electronic Payments**: When you are buying goods online, there is a need to pay online too. Electronic payments are done to reduce the inefficiency associated with writing and mailing checks.
3. **Online Auctions:** Online auction, made auctions accessible to a large number of buyers and sellers. Online auctions are an efficient mechanism.
4. **Internet Banking**: Today it is possible to perform the entire banking operations without visiting a physical bank branch. Interfacing of websites with bank accounts, and by using credit cards, is the biggest driver of ecommerce.
5. **Online Ticketing**: Air tickets, movie tickets, train tickets, play tickets, tickets to sporting events, and just about any kind of tickets can be booked online. Online ticketing need not to queue up at ticket counters.

8.4.7 E-governance

E-governance refers to any government functions or processes that are carried out in digital form over the internet. It may be said that it is a use of information technology (such as WANs, the internet and the mobile technology) by the government agencies. These technologies can serve a variety of users for better delivery of government services to the citizens. Local, state or central government, setup sites from which the public can find information, download forms and can contact government officials or representatives.

8.4.7.1 The benefits of e-governance are:

Less corruption
Increased transparency
Greater convenience
Cost reduction.

For example, filing taxes online on government sites, applying for passport and many more services are provided by government official sites.

8.4.8 Surfing

Surfing is exploration of the World Wide Web by clicking one link to another, usually without a definite objective and search strategy. In comparison, browsing is exploration with a definite objective but without a planned search strategy exploration on internet is surfing.

8.5 Search engine

A search engine is a tool that searches web pages, indexes them, and identifies web pages that are related to certain keywords and topics that you ask for. A web search engine is a software system that is designed to search information on the World Wide Web. The search results, appearing as hyperlinks, are generally presented in a line of results often referred to as search engine results pages. This information may be mixture of web pages, images, and other types of files.

The World Wide Web is the one of the most popular facility provided by Internet to link you to information all over the world. The following are the URLs of some of the popular search engines**.**

www.google.com
www.yahoo.com
www.about.com
www.links2go.com
www.infoseek.com
www.khoj.com
www.altavista.com

The search-textbox in a Google website is often looks like Fig 8.1:

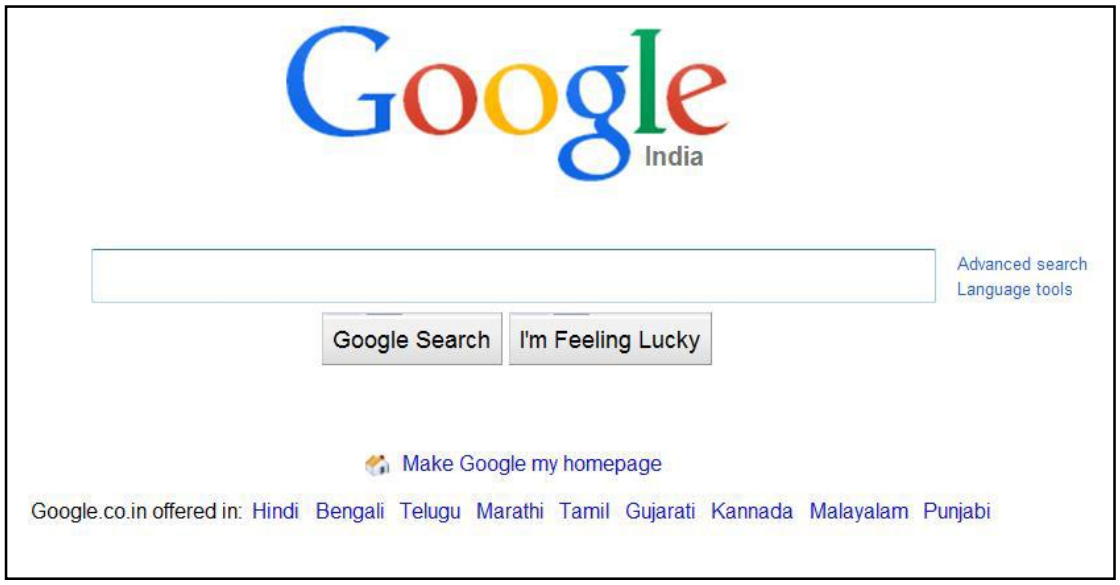


Fig 8.1

Let's take a look at some of the basics to help you in your search. To begin with, you just need to enter some keywords into the text box of the search engine. In Advanced Search, you can set options that let the search engine know whether you want an exact word match, an exact phrase match, or a match on any entered keyword.

8.6 Internet Explorer

Internet Explorer is one of the first graphical-based Web browsers used as browsing software manufactured by Microsoft Corporation. This software allows users to view and navigate web pages on the Internet. Internet Explorer is the most widely used browser in the world. It enables a user to use the hypertext, photographs, sound, video, etc. that are available on the World Wide Web. Internet Explorer utilizes "point-and-click" technology to select hypertext links and uses drop-down menus and toolbar buttons to navigate and access resources on the Internet.

8.6.1 Starting an Internet Explorer: To open window of internet explorer follow steps:

1. Click on start button.
2. Click on programs.
3. Click on Internet Explorer.

OR

You can also open internet explorer by using icon on taskbar or desktop. Window shown in above figure will be displayed. Let us study in detail the contents of internet explorer window:

8.6.2 Parts of Internet Explorer: The Parts of Internet Explorer are shown in below (Fig 8.2)

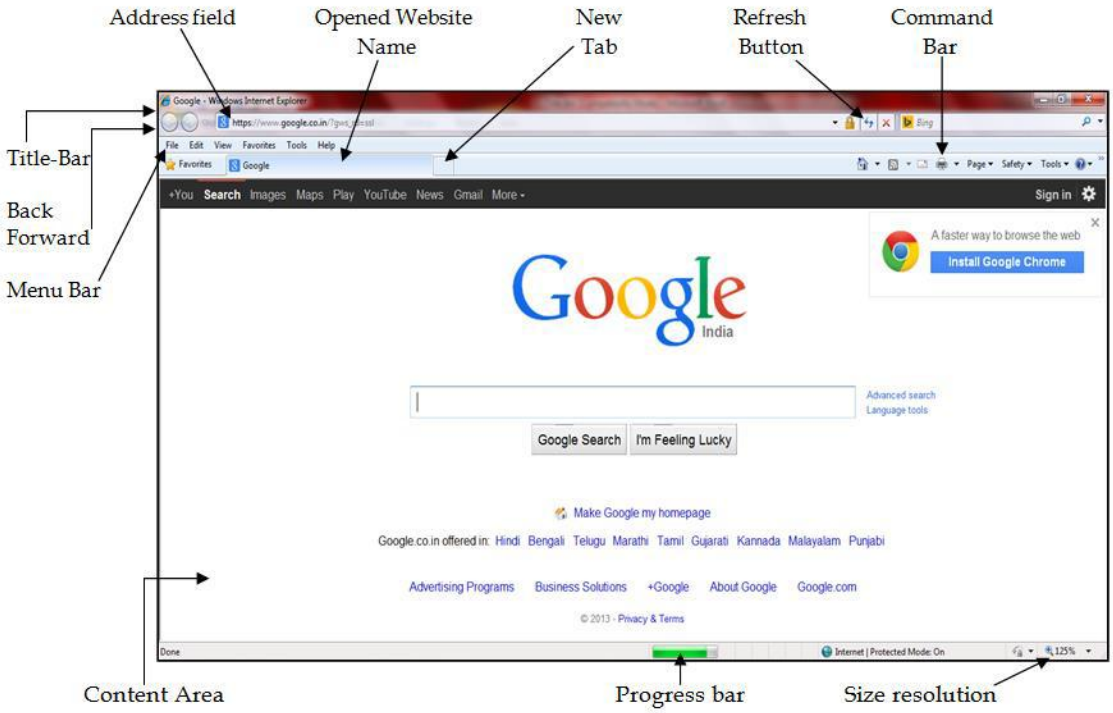


Fig 8.2

1. **Title Bar**- On the left side of the title bar, icon of Internet Explorer with the name of the website is shown. Three buttons of minimize, maximize and close are present on the right side of the title bar.
2. **Menu bar-** In menu bar, different menus are present which contains different options. Menus are File, Edit, View, Favorites, Tools and Help.
3. **Address field-** Here the name of the website is typed which we want to open. After entering the name, press enter key from the keyboard or click on go button to open the home page of the website.
4. **Back-** It is used to go to previous pages which we have opened before.
5. **Forward-**It is used to navigate between pages in forward direction, which we have opened before.

6. **Opened website name-** This tab shows the name of the current web site whose page is displayed on the screen.
7. **New tab –** It is used to open new tab in the same window. To open a new tab you can also use the shortcut ctrl+T.
8. **Refresh button-** It is used to reload the site again if it doesn't respond or become inactive.
9. **Command bar –** it consists of icons of options which are frequently used such as Home, Read mail, Print, Page, Safety, Tools and help. We can add or remove these options as per our requirement.
10. **Content area –** This is the area where the content of the web page is displayed.
11. **Progress bar –** This bar is present in the task bar in lower portion of the window. It shows the progress of loading of a webpage.
12. **Size resolution –** This tab shows the zoom level of the current web page and from this option you can change the resolution of web page.

8.7 Electronic mail (E-Mail)

Electronic mail, or e-mail,is a method of exchanging digital messages from one person to one or more recipients. Modern email operates across the Internet or other computer networks. One must possess an email address to send email to one or more e-mail addresses at a time.

8.7.1 E-Mail Address

Every user who wants to communicate over Internet through e-mail will have a unique e-mail address. An e-mail address identifies a location to which e-mail can be delivered. An e-mail address is a string of the form myname@sitename.com. An email address is made up of several parts. The first part of the address is the username, identifies a unique user on a server. The symbol @ separates the username from the host name or the domain name. The host name uniquely identifies the server computer and is the last part of the Internet email address. The three-letter suffix in the host name identifies the kind of organization operating the server. The most common suffixes are: .com (commercial), .edu (educational), .gov (government) etc.
An e-mail address is read from left to right. For example: rd_barman@gmail.com is an email address, here rd_barman is the name of the person sending or receiving the message; this is referred to as the username.

8.7.2 Step to create an email account:

To create an email account, steps written below should be followed:

1. Open the window of Internet explorer.

2. Now open the website on which you want to create an email id like, gmail, yahoo, rediffmail, hotmail, live etc. for example here we will create an email account using www.gmail.com.
3. Type www.gmail.com in address field of internet explorer window. The window shown below will appear (Fig 8.3)

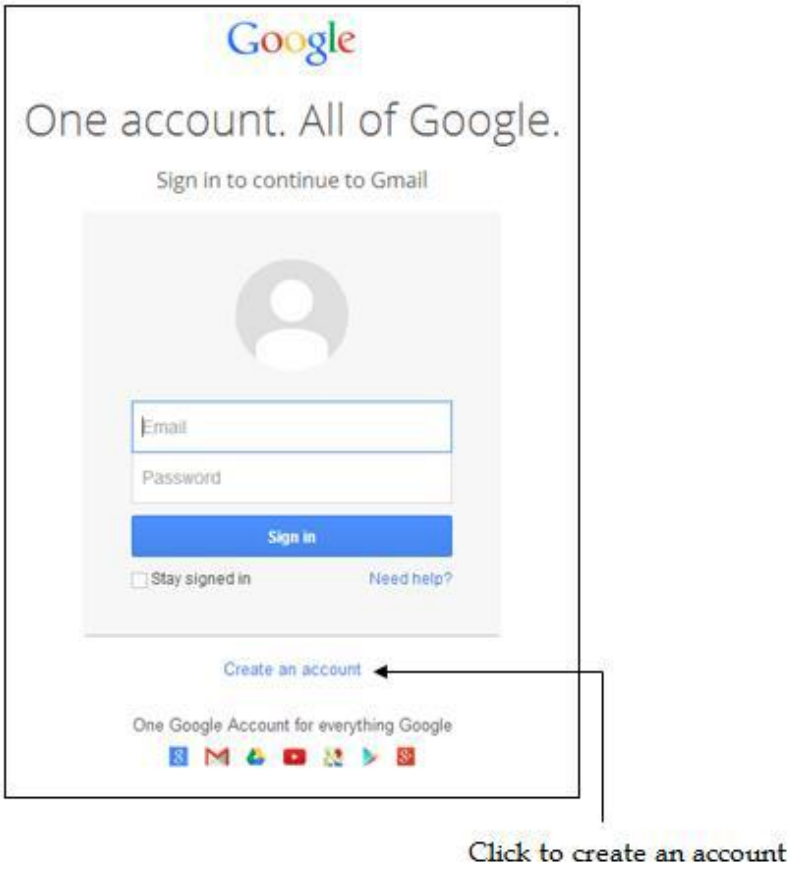


Fig 8.3

4. Here click on create an account as shown in figure. The form shown in figure below will be displayed. (Fig 8.4)

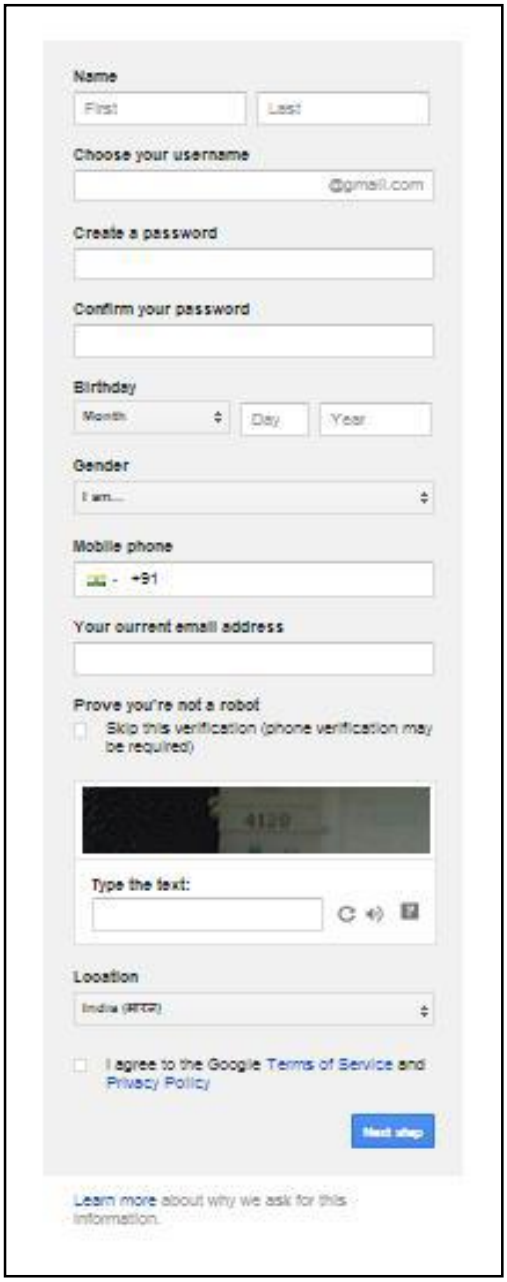


Fig 8.4

5. Fill in the necessary information needed in this form. Check the terms of service and privacy policy, and click on next step.
6. Sometimes, it will ask for your mobile number for security purpose and send a security code on that number. When you submit that security code in the form, your email id will be created.
7. Now your inbox will be opened.

Note: Remember your email id and password in order to use your account in future.

8.7.3 Receiving E-mail:

In order to read emails in your account, you need to open your account by entering your email ID and password on the site. All the emails to be read are in the inbox of your account just like the inbox of your mobile phone, where your text messages are received. By clicking on the new email, you can read your email.

8.7.4 Sending E-mail:

In order to send an email, first you need to compose it. For this click on compose button, window for composing an email will appear as shown in figure below.

Here you have to fill the recipient's email Id, in the space provided against the 'TO' label, to whom you wanted to send email, then enter the subject of your email in the space provided against the 'SUBJECT' label, and in the space provided for the message, type your message. After completing your task, you can send your mail by clicking on the send button. Your mail will be delivered in few seconds to the recipient.

Email facility provides some more features for the convenience of the user. They are as follows:

13. **Address book:** It contains the list of personal contacts with their e-mail addresses. We can add detailed information about the persons in this address book.
14. **Forward:** This option is used for sending the original message to someone else.
15. **Reply:** It is used to send a reply message. It copy the senders e-mail address as well as the text from the message, you can type your additional comments if necessary and send the message.
16. **CC (Carbon copy):** It sends the same message to several people. The addresses are separated with commas or semi-columns.
17. **BCC (Blind carbon copy):** It sends the message to several addresses without showing the addresses to each other.
18. **Attachments:** These are used to send the various types of files along with your e-mail as an attachment. The attachment could be word document, a spreadsheet, image, an audio/video file or an application. These attachments can be downloaded on the other end and automatically saved in download folder of the system.

8.8 Computer Virus

Computer virus is a piece of programming-code that attacks computer and network systems through infected data files, introduced into a system via disks such as floppy disks, compact disks and pen-derives or through internet. It gets loaded onto your computer without your knowledge and runs against your wish. It attaches itself to the target computer's operating system or other programs, and automatically replicates itself to spread to other computers or networks. All computer viruses are man-made. Few viruses are harmless but most are malicious and cause widespread and severe damage and may bring down entire communication-networks or websites. Some are immediately active, others remain latent for weeks or months, or work slowly to avoid detection and cause destruction over long periods.

8.8.1 Symptoms of virus in the system:

Following are some symptoms of virus in your system:

1. A computer virus might corrupt or delete data on a computer, or even delete everything on the hard disk.
2. The processing speed of your system becomes slow.
3. It can use whole memory of your system and bring the system to a halt.
4. It can make copies of files and itself over and over again.

8.8.2 Solution:

In order to protect the system from virus we should follow some precautions while using internet and other file sharing systems:

1. Don’t open unknown email attachments.
2. Don’t run unknown programs.
3. Turn off your computer or disconnect from the network when not in use.
4. Install anti-virus and anti-spyware software and ensure it is configured to automatically update when you are connected to the Internet.
5. Ensure the anti-virus software scans all e-mail attachments, pen-derives and other external derives when you mount them to your system.
6. Make regular backups of data in your system.

8.9 On-line railway & air ticketing

Now where everything is getting on-line, Railways and Airlines organization had also initiated in these particular areas and are providing/offering its services through the internet to its customers.

We can now book the tickets of any railways and airlines through the use of internet. Here for making payments, lots of options are also available. All information about all the trains and air services can be obtained in advance. The tickets can be booked by getting the information regarding availability of reservation in advance.

8.9.1 E-ticketing:

An **electronic ticket** (commonly abbreviated as e-ticket) is a **digital** ticket. The term is most commonly associated with airline issued tickets.

E-ticketing is booked according to the time table of an airline/train. When a reservation is confirmed, the airline keeps a record of the booking in its computer reservations system. Customers can print out or are provided with a copy of their e-ticket itinerary receipt which contains the record locator or reservation number and the e-ticket number. It is possible to print multiple copies of an e-ticket itinerary receipt.

Besides providing itinerary details, an e-ticket itinerary receipt also contains:

An official ticket number.
Carriage terms and conditions.
Fare and tax details, the exact cost might not be stated, but a "fare basis" code will always identify the fare used.
Form of payment.
Issuing office.

For booking of e-ticketing, we will have to provide the detail of identity proof e.g. PAN card, Voter ID card, Driving License, Passport or Ration card. Thereafter the printout and identity proof should be kept with you for performing the journey. For payment of tariff, many banks are offering the on-line services.

Exercise

Fill in the Blanks:

1. The internet is the global network of _____.
2. The World Wide Web (WWW) requires a tool (software) called a _____
3. E-mail stands for _____
4. A _____ is a collection of web pages
5. Facilitating business on the Internet is called _____.

True/ False

1. Google is an example of search engine.
2. Address field is a part of internet explorer.
3. To send an email, you need not to compose it.

4 A computer virus might corrupt or delete data on a computer.
5 Using computer networks to transmit audio and video data is known as video conferencing

Short answer type questions:

1 What is Internet? Write its berief History?
2 Describe E-mail? Write the properties of E-mail?
3 Define E-commerce? Write about their uses?
4 Define E-Governance? Write the benefit of it?
5 What is a search engine? Write names of some popular search engines.

Long Answer type questions:

1 Write the applications of Internet.
2 Write about facilities of Internet.
3 Describe the Internet Explorer? Explain the Parts of Internet Explorer window?
4 How to create an E-mail account? Write the Steps?
5 What is Computer Virus? Write the Symptoms of virus in the system:
6 Give Solutions for computer virus?
7 Write about e-ticketing.

****