

Module 1

What is computer?

The computer is an electronic device that takes input from the user and processes these data under the control of a set of instructions (called program) and gives the result (output) and saves for future use. It can process both numerical and non-numerical (arithmetic and logical) calculations.

Charles Babbage is called the “Father” of the computer. The First mechanical computer designed by Charles Babbage was called Analytical Engine.

Computer Generations:

Computer generations are based on when major technological changes in computers occurred, like the use of vacuum tubes, transistors, and the microprocessor. As of 2020, there are five generations of the computer.

First generation (1940 - 1956):

- The first generation of computers used vacuum tubes as a major piece of technology.
- Vacuum tubes were widely used in computers from 1940 through 1956.
- Vacuum tubes were larger components and resulted in first-generation computers being quite large in size, taking up a lot of space in a room.
- Some of the first-generation computers took up an entire room.



Second generation (1956 - 1963):

- The second generation of computers saw the use of transistors instead of vacuum tubes.
- Transistors were widely used in computers from 1956 to 1963.
- Transistors were smaller than vacuum tubes and allowed computers to be smaller in size, faster in speed, and cheaper to build.



Third generation (1964 - 1971):

- The third generation of computers introduced the use of IC (integrated circuits) in computers.
- Using ICs in computers helped reduce the size of computers even more than second-generation computers, and also made them faster.



Fourth generation (1972 - 2010):

- The fourth generation of computers took advantage of the invention of the microprocessor, more commonly known as a CPU.
- Microprocessors, with integrated circuits, helped make it possible for computers to fit easily on a desk and for the introduction of the laptop.



Fifth generation (2010 to present):

- The fifth generation of computers is beginning to use AI (artificial intelligence), an exciting technology with many potential applications around the world.
- Leaps have been made in AI technology and computers, but there is still room for much improvement.



Types of Computers:

S.No.	Type	Specifications
1	PC (Personal Computer)	It is a single user computer system having moderately powerful microprocessor
2	Workstation	It is also a single user computer system, similar to personal computer however has a more powerful microprocessor.
3	Mini Computer	It is a multi-user computer system, capable of supporting hundreds of users simultaneously.
4	Main Frame	It is a multi-user computer system, capable of supporting hundreds of users simultaneously. Software technology is different from minicomputer.
5	Supercomputer	It is an extremely fast computer, which can execute hundreds of millions of instructions per second.

PC (Personal Computer):

- A PC can be defined as a small, relatively inexpensive computer designed for an individual user.
- PCs are based on the microprocessor technology that enables manufacturers to put an entire CPU on one chip.



C Programming for Problem Solving – 21CPS13/23

- Businesses use personal computers for word processing, accounting, desktop publishing, and for running spreadsheet and database management applications.
- At home, the most popular use for personal computers is playing games and surfing the Internet.

Workstation Computer:

- Workstation is a computer used for engineering applications (CAD/CAM), desktop publishing, software development, and other such types of applications which require a moderate amount of computing power and relatively high-quality graphics capabilities.
- Workstations generally come with a large, high-resolution graphics screen, large amount of RAM, inbuilt network support, and a graphical user interface.
- Most workstations also have mass storage device such as a disk drive, but a special type of workstation, called diskless workstation, comes without a disk drive.



Minicomputer:

- It is a midsize multi-processing system capable of supporting up to 250 users simultaneously.
- Minicomputers are smaller, less expensive, and less powerful than a mainframe or supercomputer but more expensive and more powerful than a personal computer.
- Minicomputers are used for scientific and engineering computations, business transaction processing, file handling, and database management.



Mainframe Computer:

- Mainframe is very large in size and is an expensive computer capable of supporting hundreds or even thousands of users simultaneously.
- Mainframe executes many programs concurrently and supports many simultaneous executions of programs.



Supercomputer:

- Supercomputers are one of the fastest computers currently available. Supercomputers are very expensive and are employed for specialized applications that require immense number of mathematical calculations (number crunching).



- For example, weather forecasting, scientific simulations, (animated) graphics, fluid dynamic calculations, nuclear energy research, electronic design, and analysis of geological data (e.g., in petrochemical prospecting).

Bit, Byte and Word:

Bit:

- Computer memory stores the data in bits. It is the smallest value which can be stored in computer.
- Each bit stores the value either 0 or 1.
- Inside the Computer, there are electronic switches that store the bit values; either 0 or 1.
- If the switch is OFF means, that represents 0 & the switch is ON means, that represents 1.

Byte:

- A byte is a grouping of consecutive bits.
- 8-bits represents a Byte.
- The most common term we use in Computer's terminology is Bytes.

How internally Bytes store the data?

Simple, it is Bits. The Byte representation of a character 'A' is: ASCII Character value of 'A' is 65. Bit representation (binary representation) of 'A' in a Byte is below:

Unit	Equivalent to	Remarks
1 kilobyte (KB)	1024 bytes	Space used by 10 lines of text.
1 megabyte (MB)	1024 kilobytes	Memory of the earliest PCs
1 gigabyte (GB)	1024 megabytes	Storage capacity of a CD-ROM
1 terabyte (TB)	1024 gigabytes	Capacity of today's hard disks.
1 petabyte (PB)	1024 terabytes	Space used for rendering of film Avatar

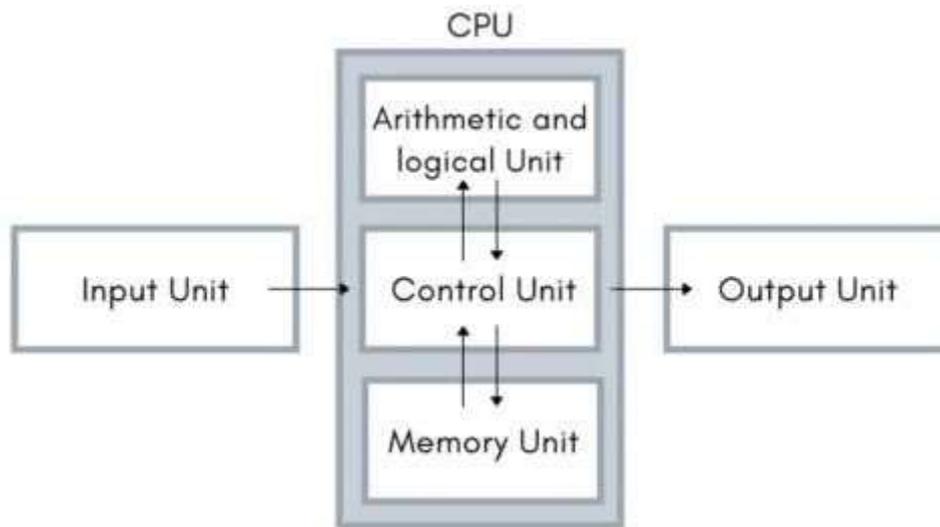


Word:

- WORDs are also consecutive bits or bytes.
- This term is mostly used for CPU registers.
- Generally, each WORD has a length of 16-bits.
- CPU Registers are used to store a small piece of information while doing the calculations or processing the data.
- These are helpful to improve the performance of the system while doing the calculation or processing.

Basic Structure of Computer:

The basic structure of computer system consists mainly of three parts which are the central processing unit (CPU), Input devices, and output devices. Further, the Central processing unit can be divided into two more parts i.e., control unit (CU) and arithmetic logic unit (ALU).



Input Unit:

An input unit is any device that provides input to a computer. The input devices translate the information into a form understandable by the computer.

Examples of input devices are:

➤ Keyboard

- Keyboard is the most common and very popular input device which helps to input data to the computer.
- Keyboards are of two sizes 84 keys or 101/102 keys, but now keyboards with 104 keys or 108 keys are also available for Windows and Internet.

➤ Mouse

- Mouse is the most popular cursor-control device
- Generally, it has two buttons called the left and the right button and a wheel is present between the buttons.
- A mouse can be used to control the position of the cursor on the screen, but it cannot be used to enter text into the computer.

➤ Microphone

- Microphone is an input device to input sound that is then stored in a digital form.

Output Unit:

- Output device receives information from the CPU and presents it to the user in the desired form.
- The processed data, stored in the memory of the computer is sent to the output unit, which then converts it into a form that can be understood by the user.
- The output is usually produced in one of the two ways – on the display device, audio output or on paper (hard copy).

Examples of output device are:

- Monitor:
 - Monitor is often used synonymously with “computer screen” or “display.”
 - Monitor is an output device that resembles the television screen.
 - It may use a Cathode Ray Tube (CRT) to display information.
 - The monitor is associated with a keyboard for manual input of characters and displays the information as it is keyed in.
 - It also displays the program or application output. Like the television, monitors are also available in different sizes.
- Printer:
 - Printers are used to produce paper (commonly known as hardcopy) output.
 - Based on the technology used, they can be classified as Impact or Non-impact printers.
- Sound cards and Speaker:
 - An expansion board that enables a computer to manipulate and output sounds.
 - Sound cards enable the computer to output sound through speakers connected to the board, to record sound input from a microphone connected to the computer, and manipulate sound stored on a disk.

CPU (Central Processing Unit):

CPU is considered as the brain of the computer. CPU performs all types of data processing operations. It stores data, intermediate results, and instructions (program). It controls the operation of all parts of the computer.

CPU itself has the following three components:

1. ALU (Arithmetic Logic Unit)

2. Memory Unit

3. Control Unit

MEMORY:

A memory is just like a human brain. It is used to store data and instructions. Computer memory is the storage space in the computer, where data is to be processed and instructions required for processing are stored.

Memory is primarily of three types -

1. Cache Memory
2. Primary Memory/Main Memory
3. Secondary Memory

Cache Memory

- Cache memory is a very high-speed semiconductor memory which can speed up the CPU.
- It acts as a buffer between the CPU and the main memory.
- It is used to hold those parts of data and program which are most frequently used by the CPU.

Primary Memory (Main Memory)

- Primary memory holds only those data and instructions on which the computer is currently working.
- It has a limited capacity and data is lost when power is switched off.
- It is generally made up of semiconductor device. These memories are not as fast as registers. The data and instruction required to be processed resides in the main memory.
- It is divided into two subcategories RAM and ROM.

RAM (Random Access Memory)

It is the internal memory of the CPU for storing data, program, and program result. It is a read/write memory which stores data until the machine is working. As soon as the machine is switched off, data is erased.

RAM is of two types -

- Static RAM (SRAM)

The word static indicates that the memory retains its contents as long as power is being supplied. However, data is lost when the power gets down due to volatile nature.

- Dynamic RAM (DRAM)

DRAM, unlike SRAM, must be continually refreshed in order to maintain the data. This is done by placing the memory on a refresh circuit that rewrites the data several hundred times per second. DRAM is used for most system memory as it is cheap and small.

ROM (Read Only Memory):

ROM stands for Read Only Memory. The memory from which we can only read but cannot write on it. This type of memory is non-volatile.

Types of ROMs are:

➤ MROM (Masked ROM)

The very first ROMs were hard-wired devices that contained a pre-programmed set of data or instructions. These kinds of ROMs are known as masked ROMs, which are inexpensive.

➤ PROM (Programmable Read Only Memory)

PROM is read-only memory that can be modified only once by a user. The user buys a blank PROM and enters the desired contents using a PROM program. Inside the PROM chip, there are small fuses which are burnt open during programming. It can be programmed only once and is not erasable.

➤ EPROM (Erasable and Programmable Read Only Memory)

EPROM can be erased by exposing it to ultra-violet light for a duration of up to 40 minutes.

➤ EEPROM (Electrically Erasable and Programmable Read Only Memory)

EEPROM is programmed and erased electrically. It can be erased and reprogrammed about ten thousand times.

Secondary Memory:

This type of memory is also known as external memory or non-volatile. It is slower than the main memory. These are used for storing data/information permanently. CPU directly does not access these memories, instead they are accessed via input-output routines. The contents of secondary memories are first transferred to the main memory, and then the CPU can access it. For example, disk, CD-ROM, DVD, etc.

Control Unit:

➤ The control unit is a very important component of the central processing unit.

➤ The responsibility of the control unit is to take information that is being provided by the memory unit, when the control unit receives instruction from the memory unit then it converts into control signals that are sent to the central processor for further processing.

- The control unit also sends instructions to the arithmetic and logic unit to perform the right operation on the instructed operands and in this way the control unit executes the operation.

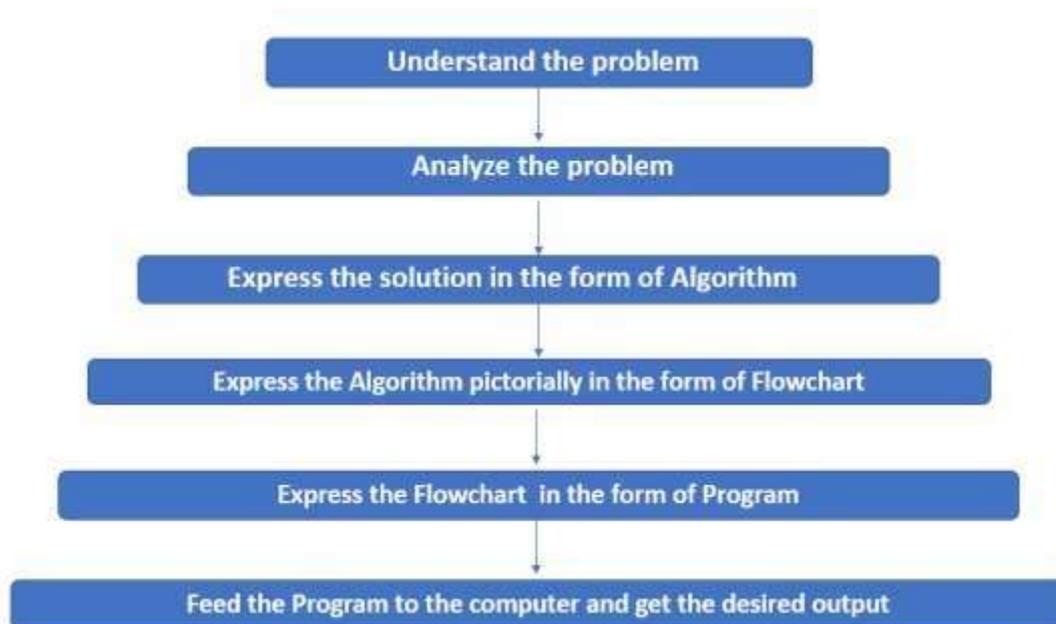
Arithmetic and logic unit:

The work of the arithmetic and logic unit is responsible for performing mathematical instruction (add, subtract, multiply, divide) and logical instruction (greater than, less than, equal to, and, or, not) on the information that is in the form of binary from the control unit and on performing the set of instruction the results are returned to the control unit.

History of C:

- C programming language was developed in 1972 by Dennis Ritchie at bell laboratories of AT&T (American Telephone & Telegraph), located in the U.S.A.
- Dennis Ritchie is known as the founder of the c language.
- It was developed to overcome the problems of previous languages such as B, BCPL, etc.
- Initially, C language was developed to be used in UNIX operating system. It inherits many features of previous languages such as B and BCPL.

How to Solve a C Problem?



Algorithm:

Algorithm is a step-by-step procedure, which defines a set of instructions to be executed in a certain order to get the desired output.

Characteristics of an Algorithm:

Not all procedures can be called an algorithm. An algorithm should have the following characteristics

- Unambiguous – Algorithm should be clear and unambiguous. Each of its steps (or phases), and their inputs/outputs should be clear and must lead to only one meaning.
- Input – An algorithm should have 0 or more well-defined inputs.
- Output – An algorithm should have 1 or more well-defined outputs, and should match the desired output.
- Finiteness – Algorithms must terminate after a finite number of steps.
- Feasibility – Should be feasible with the available resources.
- Independent – An algorithm should have step-by-step directions, which should be independent of any programming code.

How to Write an Algorithm?

There are no well-defined standards for writing algorithms. Rather, it is problem and resource dependent.

1. Name of the algorithm must be specified.
2. Step numbers must be given to each step for the better identification.
3. Each step must have an explanatory note provided with the square bracket followed by corresponding operations.
4. Completion of an algorithm must be specified towards the end using STOP statement.
5. In the same way beginning of an algorithm must be specified with START statement.

Examples:

1. Write an Algorithm to add two numbers.

Algorithm Name: To add two numbers

Step 1: START

Step 2: [Initialize Variables]

C Programming for Problem Solving – 21CPS13/23

Read a, b

Step 3: [Compute the process]

res = a+b

Step 4: [Display the result]

Display res

Step 5: STOP

2. Write an Algorithm to multiply two numbers.

Algorithm Name: To multiply two numbers

Step 1: START

Step 2: [Initialize Variables]

Read a, b

Step 3: [Compute the process]

res = a*b

Step 4: [Display the result]

Display res

Step 5: STOP

3. Write an Algorithm to find the area of a triangle.

Algorithm Name: To find the area of a triangle

Step 1: START

Step 2: [Initialize Variables]

Read b, h

Step 3: [Compute the process]

res = 0.5*b*h

Step 4: [Display the result]

Display res

Step 5: STOP

4. Write an algorithm to swap two numbers using temporary variables?

Algorithm Name: To swap two numbers using temporary variable

Step 1: START

Step 2: [Initialize Variables]

Read a, b

Step 3: [Compute the process]

temp = a

a = b

b = temp

Step 4: [Display the result]

Display a, b

Step 5: STOP

5. Write an algorithm to swap two numbers without using temporary variables?

Algorithm Name: To swap two numbers without using temporary variable

Step 1: START

Step 2: [Initialize Variables]

Read a, b

Step 3: [Compute the process]

a = a + b

b = a - b

a = a - b

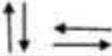
Step 4: [Display the result]

Display a, b

Step 5: STOP

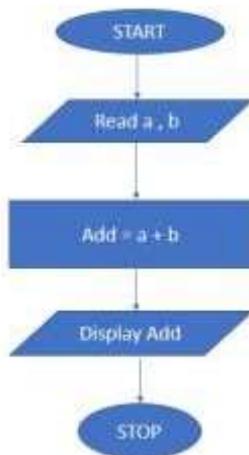
Flowchart:

- Flowchart in C is a diagrammatic representation of a sequence of logical steps of a program.
- Flowcharts use simple geometric shapes to depict processes and arrows to show relationships and process/data flow.
- A flowchart in C language is a graphical representation of an algorithm.

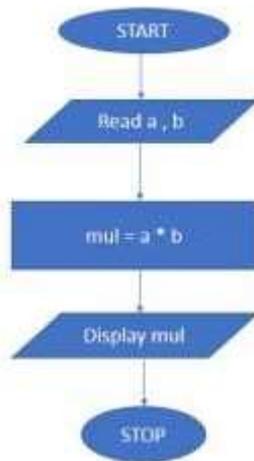
Symbol	Name	Function
	Process	Indicates any type of internal operation inside the Processor or Memory
	input/output	Used for any Input / Output (I/O) operation. Indicates that the computer is to obtain data or output results
	Decision	Used to ask a question that can be answered in a binary format (Yes/No, True/False)
	Connector	Allows the flowchart to be drawn without intersecting lines or without a reverse flow.
	Predefined Process	Used to invoke a subroutine or an Interrupt program.
	Terminal	Indicates the starting or ending of the program, process, or interrupt program
	Flow Lines	Shows direction of flow.

Examples:

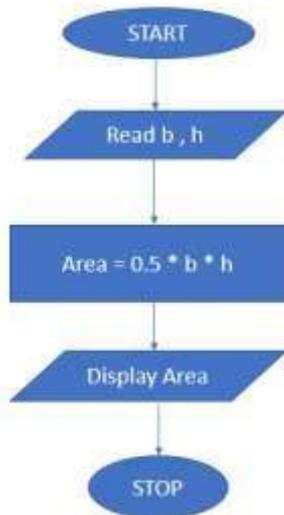
1. Write a flowchart to add two numbers.



2. Write a flowchart to multiply two numbers



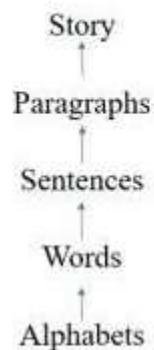
3. Write a flowchart to find the area of a triangle.



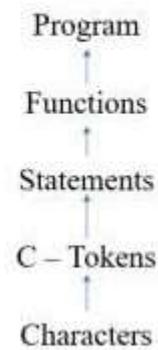
PROGRAM:

A group of instructions given by the programmer to the computer to do specific task.

Process of learning language



Process of learning C program



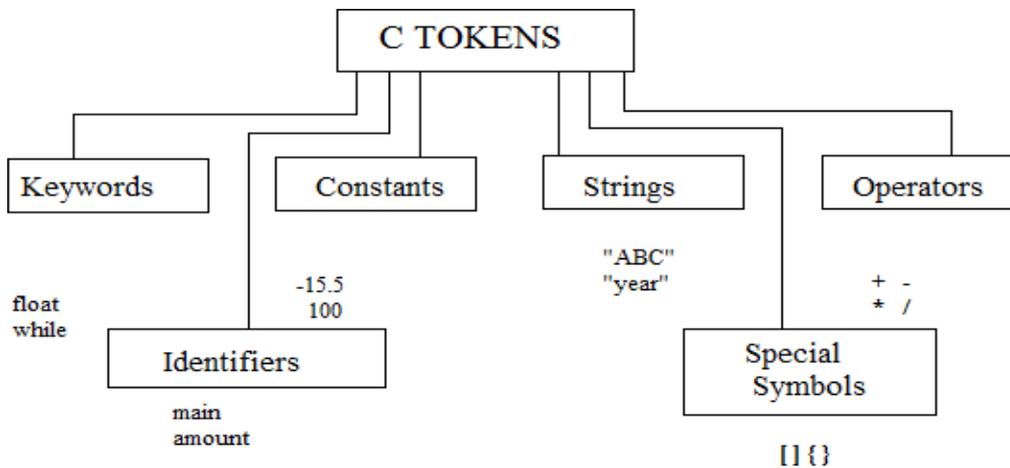
Characters:

The C character set consists of upper and lowercase alphabets, digits, special characters and white spaces.

Alphabets	A, B,, Y, Z a, b,, y, z
Digits	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Special symbols	~ ^ ! @ # % ^ & * () _ - + = \ { } [] ; : " ' < > , . ? /

C – Token:

One or more character of the character set is combined to form meaningful word are called C-Token



Keywords:

- Keywords in C can be defined as the pre-defined or the reserved words having its own importance, and each keyword has its own functionality.
- Since keywords are the pre-defined words used by the compiler, so they cannot be used as the variable names. If the keywords are used as the variable names, it means that we are assigning a different meaning to the keyword, which is not allowed.
- C language supports 32 keywords given below:

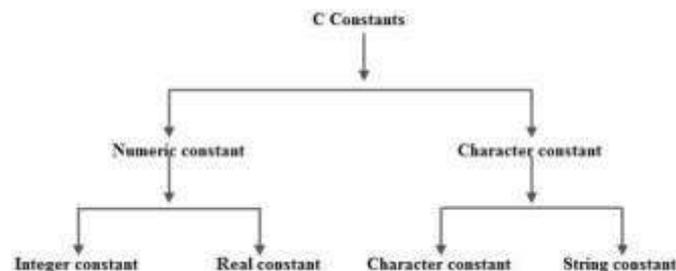
auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
continue	for	signed	void
do	if	static	while
default	goto	sizeof	Volatile
const	float	short	Unsigned

Identifiers:

- Identifiers in C are used for naming variables, functions, arrays, structures, etc.
- Identifiers in C are the user-defined words.
- It can be composed of uppercase letters, lowercase letters, underscore, or digits, but the starting letter should be either an underscore or an alphabet. Identifiers cannot be used as keywords.
- Rules for constructing identifiers in C are given below:
 - The first character of an identifier should be either an alphabet or an underscore, and then it can be followed by any of the character, digit, or underscore.
 - It should not begin with any numerical digit.
 - In identifiers, both uppercase and lowercase letters are distinct. Therefore, we can say that identifiers are case sensitive.
 - Commas or blank spaces cannot be specified within an identifier.
 - Keywords cannot be represented as an identifier.
 - The length of the identifiers should not be more than 31 characters.
 - Identifiers should be written in such a way that it is meaningful, short, and easy to read.

Constants:

Constants refer to fixed values that the program may not alter during its execution. These fixed values are also called literals.



Data Type:

A data type specifies the type of data that a variable can store such as integer, floating, character, etc.

Data type	Memory Size (In Bytes)	Range
char	1	-128 to +127
int	2	-32768 to +32767
float	4	3.4exp(-38) to 3.4exp(+38)
double	8	1.7exp(-308) to 1.7exp(+308)
void	0	---

char

1. It is a keyword that is capable of holding a character
2. It occupies 1 byte [8 bits] of memory
3. It may contain alphabets, digits, special symbols that should be enclosed with in a single quote.

int

1. It is used to store the whole numbers
2. It occupies two bytes of memory

float

1. It is used to store floating point numbers
2. It occupies 4 bytes of memory
3. It provides 6 digits after decimal points

double

1. It is used to store real numbers that have double-precision floating point
2. It occupies 8 bytes of memory
3. It provides 16 digits after decimal points.

void

1. It does not store any value
2. We cannot store any operation on a variable declared as void
3. It has no range

Variable Declaration:

In C programming, variables which are to be used later in different parts of the functions have to be declared. Variable declaration tells the compiler two things:

The name of the variable. The type of data the variable will hold.

Syntax:

datatype variable_name;

or

datatype variable1, variable2, ..., variablen;

where:

datatype is the datatype supported by c language

variable1, variable2 etc. are the names given to the memory location by following the rules.

Example:

int a;

float j;

Input / Output Statements:

- C programming language provides many built-in functions to read any given input and to display data on screen when there is a need to output the result.
- Input output built-in functions in C falls into two categories, namely, formatted input output (I/O) functions and unformatted input output (I/O) functions.
- printf () and scanf () are examples for formatted input and output functions
- getch (), gets (), puts (), putchar () etc. are examples of unformatted input output functions.

Input statement:

scanf()

To enter the input through the input device like keyboard we make use of scanf () statement.

General Syntax:

scanf (“format specifier”, Address list);

Where:

format specifier consists of %d for int, %f for float, %c for char etc. based on the datatype

Address list consists of variable name preceded with & symbol (address operator)

Example:

```
int a;  
float b;  
scanf ("%d%f", &a, &b);
```

Rules for scanf ():

- No blank spaces should be specified in the format specifier
 - Ex: scanf ("%d %f", &a, &b); // invalid
- & Must read the values, if not the entered value will not be stored in the variable specified.
 - Ex: scanf ("%d%f", a, b); //invalid

Output statement:

printf()

printf is an output statement in C used to display the content on the screen.

print: Print the data stored in the specified memory location or variable.

Format: The data present in memory location is formatted in to appropriate data type.

There are various forms of printf statements

Method 1:

printf ("Display message \n");

Display message may be any character. The characters included within the double quotes will be displayed on the output screen

➤ Example: **printf("Welcome to India");**

Output:

Welcome to India

Method 2:

Printf ("format specifier", variable list);

format specifier consists of %d for int, %f for float, %c for char etc. based on the datatype
format specifiers starts with % sign followed by conversion code.

variable list are the variable names separated by comma.

Example:

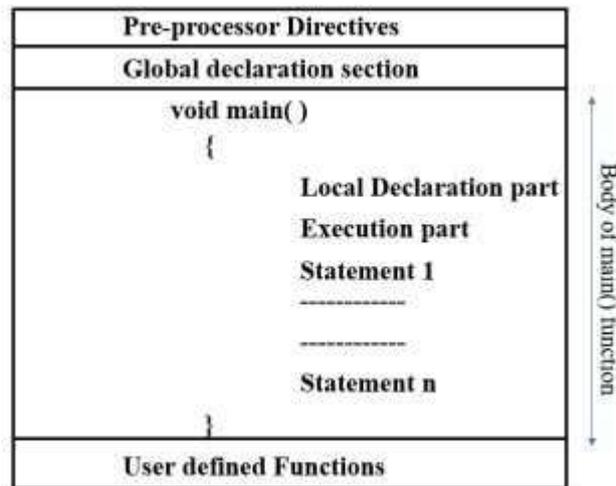
```
int a=10;  
float b=20;
```

```
printf(" integer =%d, floating=%f",a,b);
```

Output:

Integer=10, floating=20.000000

Structure of C Program:



Pre-processor Directives:

- Preprocessor Directives begins with a # symbol
- Provides instructions to the compiler to include some of the files before compilation starts.
- Some examples of header files are

```
#include <stdio.h>  
#include <conio.h>  
#include <string.h>  
#include <math.h>  
#include <stdlib.h> Etc...
```

Global Declaration Section:

- Variables are declared in this section
- Those variables are used in throughout the program

The Program Header (main function)

- Every program must contain a main () function.
- The execution always starts from main function.
- Every C program must have only one main function.
- We can also use int or main with the main ()

C Programming for Problem Solving – 21CPS13/23

- The void main () specifies that the program will not return any value.
- The int main() specifies that the program can return integer type data

Body of the program

- Series of statements that performs specific task will be enclosed within braces {and}
- It is present immediately after program header.
- It consists of two parts
 - Local Declaration part: Describes variables of program
 - Ex: int sum = 0;
Int a, b;
float b;
 - Executable part: Task done using statements, all statements should end with semicolon (;)

User-defined function

- Functions defined by the user are called as user-defined function

Examples:

1. Write a C program to add two numbers?

```
#include<stdio.h>
void main()
{
    int a, b, res;
    printf("enter two numbers\n");
    scanf("%d%d",&a,&b);
    res = a+b;
    printf("Result = %d\n",res);
}
```

2. Write a C program to multiply two numbers?

```
#include<stdio.h>
void main()
{
```

C Programming for Problem Solving – 21CPS13/23

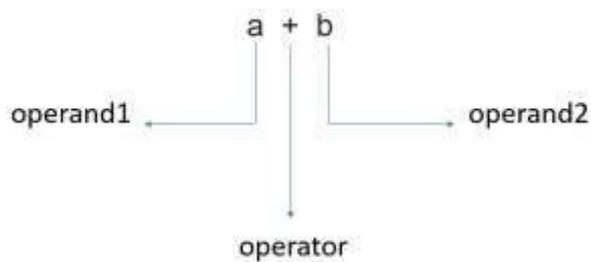
```
int a, b, mul;
printf("enter two numbers\n");
scanf("%d%d",&a,&b);
mul = a*b;
printf("Result = %d\n",mul);
}
```

3. Write a C program to find the area of a triangle?

```
#include<stdio.h>
void main()
{
    int base, height;
    float area;
    printf("enter base and height value\n");
    scanf("%d%d",&base,&height);
    area = 0.5 * base * height;
    printf("Result = %f\n",area);
}
```

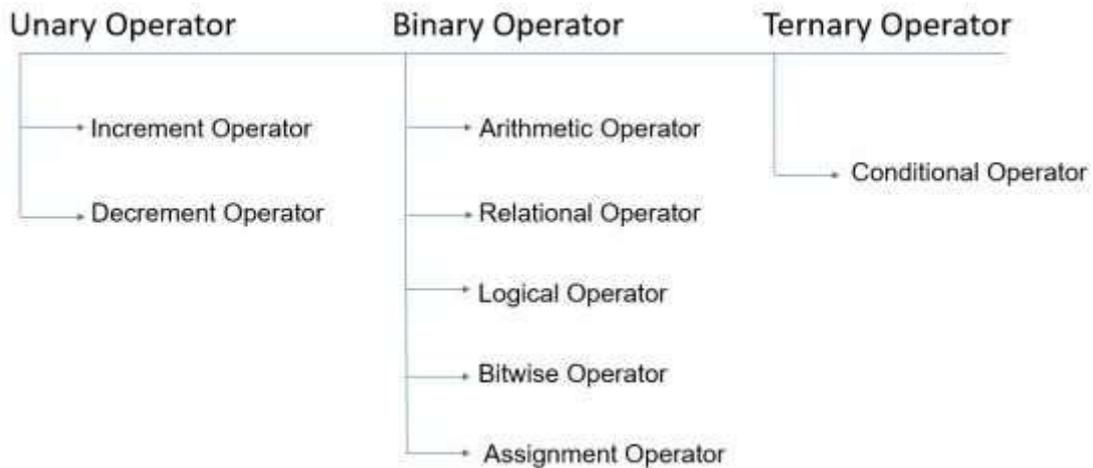
OPERATORS:

An operator is a symbol that operates on a value or a variable.



Types of operators:

1. **Unary Operator:** Operator which works with one operand
2. **Binary Operator:** Operator which works with two operands
3. **Ternary Operator:** Operator which works with three operands



Increment Operator:

In programming, the increment operator (++) increases the value of a variable by 1.

Two types of Increment Operator

➤ Pre-Increment

A pre-increment operator is used to increment the value of a variable before using it in an expression. In the Pre-Increment, value is first incremented and then used inside the expression.

➤ Post-Increment

A post-increment operator is used to increment the value of variable after executing expression completely in which post increment is used. In the Post-Increment, value is first used in a expression and then incremented.

The following is the syntax of pre and post increment.

```
++variable_name; // Pre-Increment
```

```
variable_name++; //Post-Increment
```

Example:

1. Solve $++a + a++ - ++b * ++b$, where $a=2, b=4$

```
++a + a++ - ++b * ++b
```

```
3 + a++ - 5 * ++b //a=3 b=5
```

```
3 + 3 - 5 * 6
```

```
3 + 3 - 30
```

```
6 - 30
```

```
-24
```

Decrement Operator:

In programming, the decrement operator (--) decreases the value of a variable by 1.

Two types of decrement Operator

➤ Pre-decrement

A pre-decrement operator is used to decrement the value of a variable before using it in an expression. In the Pre-decrement, value is first decremented and then used inside the expression.

➤ Post-decrement

A post-decrement operator is used to decrement the value of variable after executing expression completely in which post decrement is used. In the Post-decrement, value is first used in a expression and then decremented.

The following is the syntax of pre and post decrement.

--variable_name; // Pre-decrement

variable_name--; //Post-decrement

Example:

1. Solve $--a + a-- - -b * --b$, where $a=2, b=4$

```
--a + a-- - -b * --b
1 + a-- - 3 * --b    //a=1 b=3
1 + 1 - 3 * 2
1 + 1 - 6
2 - 6
-4
```

Arithmetic Operator:

An arithmetic operator performs mathematical operations such as addition, subtraction, multiplication, division etc. on numerical values (constants and variables).

Operator	Description	Usage	Priority
+	To perform addition	a+b	2
-	To perform subtraction	a-b	2
*	To perform multiplication	a*b	1
/	To perform division	a/b	1
%	To get the remainder	<u>a%b</u>	1

Example:

1. Solve $a + b * c / d \% a - c$ where $a=2$ $b=3$ $c=4$ $d=5$

$$\begin{aligned} &\rightarrow a + b * c / d \% a - c \\ &\rightarrow 2 + 3 * 4 / 5 \% 2 - 4 \\ &\rightarrow 2 + 12 / 5 \% 2 - 4 \\ &\rightarrow 2 + 2 \% 2 - 4 \\ &\rightarrow 2 + 0 - 4 \\ &\rightarrow 2 - 4 \\ &\rightarrow -2 \end{aligned}$$

Relational Operator:

A relational operator checks the relationship between two operands. If the relation is true, it returns 1; if the relation is false, it returns value 0.

Operator	Description	Usage	Priority
>	Greater than	$a > b$	1
<	Lesser than	$a < b$	1
>=	Greater than or equal to	$a >= b$	2
<=	Lesser than or equal to	$a <= b$	2
==	Equal to	$a == b$	3
!=	Not equal to	$a != b$	3

Example:

1. Solve $a < b >= c == d > 1 != 0$ where $a=10$ $b=20$ $c=30$ $d=40$

$$\begin{aligned} &\rightarrow a < b >= c == d > 1 != 0 \\ &\rightarrow 10 < 20 >= 30 == 40 > 1 != 0 \\ &\rightarrow 1 >= 30 == 40 > 1 != 0 \\ &\rightarrow 1 >= 30 == 1 != 0 \\ &\rightarrow 0 == 1 != 0 \\ &\rightarrow 0 != 0 \\ &\rightarrow 0 \end{aligned}$$

Logical Operator:

An expression containing logical operator returns either 0 or 1 depending upon whether expression results true or false.

Operator	Usage	Priority
&& [AND]	0 && 0 = 0 0 && 1 = 0 1 && 0 = 0 1 && 1 = 1	2
 [OR]	0 0 = 0 0 1 = 1 1 0 = 1 1 1 = 1	3
! [NOT]	!1 = 0 !0 = 1	1

Example:

1. Solve $1 \ \&\& \ 1 \ || \ !0 \ \&\& \ 1$

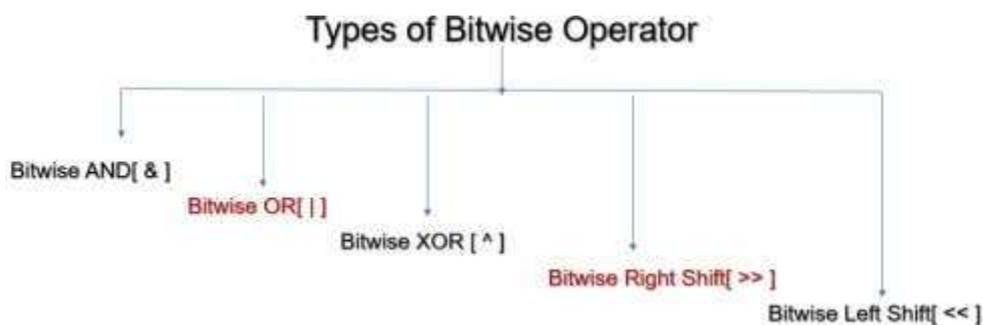
→ $1 \ \&\& \ 1 \ || \ \underline{!0 \ \&\& \ 1}$

→ $1 \ \&\& \ 1 \ || \ \underline{1 \ \&\& \ 1}$

→ $\underline{1 \ || \ 1}$

→ 1

Bitwise Operator:



Bitwise AND [&]:

The output of bitwise AND is 1 if the corresponding bits of two operands is 1. If either bit of an operand is 0, the result of corresponding bit is evaluated to 0.

Let us suppose the bitwise AND operation of two integers 12 and 25.

12 = 00001100 (In Binary)

C Programming for Problem Solving – 21CPS13/23

25 = 00011001 (In Binary)

Bit Operation of 12 and 25

00001100

& 00011001

00001000 = 8 (In decimal)

Bitwise OR [|]:

The output of bitwise OR is 1 if at least one corresponding bit of two operands is 1. In C Programming, bitwise OR operator is denoted by |.

12 = 00001100 (In Binary)

25 = 00011001 (In Binary)

Bitwise OR Operation of 12 and 25

00001100

| 00011001

00011101 = 29 (In decimal)

Bitwise XOR [^]:

The result of bitwise XOR operator is 1 if the corresponding bits of two operands are opposite. It is denoted by ^.

12 = 00001100 (In Binary)

25 = 00011001 (In Binary)

Bitwise XOR Operation of 12 and 25

00001100

^ 00011001

00010101 = 21 (In decimal)

Bitwise Right Shift[>>]:

Right shift operator shifts all bits towards right by certain number of specified bits. It is denoted by >>.

212 = 11010100 (In binary)

212>>2 = 00110101 (In binary) [Right shift by two bits]

Mr. Suhas A Bhyratae, Dept. of CSE, SCEM, Mangalore

C Programming for Problem Solving – 21CPS13/23

$212 \gg 7 = 00000001$ (In binary)

$212 \gg 8 = 00000000$

$212 \gg 0 = 11010100$ (No Shift)

Bitwise Left Shift [<<]:

Left shift operator shifts all bits towards left by a certain number of specified bits. The bit positions that have been vacated by the left shift operator are filled with 0. The symbol of the left shift operator is <<.

$212 = 11010100$ (In binary)

$212 \ll 1 = 110101000$ (In binary) [Left shift by one bit]

$212 \ll 0 = 11010100$ (Shift by 0)

$212 \ll 4 = 110101000000$ (In binary) = 3392 (In decimal)

Assignment Operator:

Assignment operators are used to assigning value to a variable. The left side operand of the assignment operator is a variable and right-side operand of the assignment operator is a value.

Operator	Description	Example
=	Simple assignment operator. Assigns values from right side operands to left side operand	$C = A + B$ will assign the value of $A + B$ to C

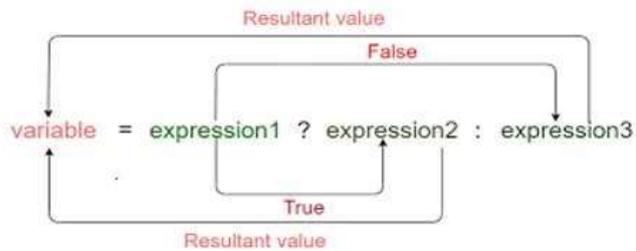
Shorthand Assignment:

Shorthand assignment operator	Example	Meaning
+=	$a += 10$	$a = a + 10$
-=	$a -= 10$	$a = a - 10$
*=	$a *= 10$	$a = a * 10$
/=	$a /= 10$	$a = a / 10$
%=	$a %= 10$	$a = a \% 10$
&=	$a \&= 10$	$a = a \& 10$
=	$a = 10$	$a = a 10$
^=	$a ^= 10$	$a = a ^ 10$
~=	$a ~= 10$	$a = a ~ 10$
<<=	$a \ll= 10$	$a = a \ll 10$
>>=	$a \gg= 10$	$a = a \gg 10$

Conditional Operator:

The conditional operator is also known as a ternary operator. The conditional statements are the decision-making statements which depends upon the output of the expression.

Syntax:



Conditional Operators Example

```
#include<stdio.h>
void main()
{
    int a, b, x;
    printf("Enter the values of a add b : ");
    scanf("%d %d", &a, &b);
    x=(a>b)?a:b;
    printf("Biggest Value is :%d",x);
}
```