Programming language :

A language through which we can input data/instruction into the computer and the computer perform given task according to the given instruction is called programming language.

Types of Computer Languages :

- ***Low-Level Languages:*** *A language that corresponds directly to a specific machine.*
- ***High-Level Languages:*** *Any language that is independent of the machine.*

Low-Level Languages

- *Low-level computer languages are machine codes .*
- *It is the **machine level** language or first generation language .*
- *It is very difficult to written by programmer & very easy to understand by computer.*
- *There is no need to any translator because it directly written in machine language i.e. binary(0 or 1).*

There are two types of low-level languages:

- ***Machine Language***
- ***Assembly Language***

(i) Machine Language

Machine language is the lowest and most elementary level of programming language and was the first type of programming language to be developed.

Advantages	***Disadvantages***
Machine language makes fast and efficient use of the computer	*All operation codes have to be remembered*
It requires no translator .It is directly understood by the computer.	*It is hard to find errors in a program written in the machine language*

(ii) Assembly Language

- *This is another low-level but very important language in which operation codes and operands are given in the form of alphanumeric symbols instead of 0's and l's.*
- *It is the **middle-level language** or second generation language .*
- *These alphanumeric symbols are known as mnemonic codes and can combine in a **maximum of five-letter** combinations e.g. **ADD** for addition, **SUB** for subtraction, START, LABEL etc. Because of this feature, assembly language is also known as 'Symbolic Programming Language.*

Advantages	***Disadvantages***
Assembly language is easier to understand and use .	*Like machine language, it is also machine dependent.*
It is easy to locate and correct errors. It is easily modified.	*It is machine dependent.*

High-Level Languages

- *High-level computer languages are similar to English language.*
- *The purpose of developing high-level languages was to write programs easily by the programmers, But it difficult to understand by the computer.*
- *Compiler and interpreter are used as a translator to translate each instruction in the high-level Language is translated into machine language instructions that the computer can understand.*

Types of High-Level Languages

- *BASIC (Beginners All Purpose Symbolic Instruction Code)*
- *FORTRAN (Formula Translation)*
- *PL/I (Programming Language, Version 1)*
- *ALGOL (Algorithmic Language)*
- *APL (A Programming Language)*
- *COBOL (Common Business Oriented Language)*
- *RPG (Report Program Generator)*
- *LISP (List Processing)*
- *Prolog (Program in Logic)*
- *C , C++ , Java*
- *Visual Java*
- *Visual C*

Advantages	***Disadvantages***
High-level languages are user friendly. They are easier to learn. The language is independent of the machine.	*A high-level language has to be translated into the machine language by a translator, which takes up time*
They are similar to English and use English vocabulary and well-known symbols.	*The object code generated by a translator might be inefficient compared to an equivalent assembly language program.*

C- Language

C is a general-purpose, high-level language that was originally developed by Dennis M. Ritchie to develop the UNIX operating system at Bell Labs in the period of 1969 to 1973. C was originally first implemented on the DEC PDP-11 computer in 1972.

C has now become a widely used professional language for various reasons-

- *Easy to learn*
- *Structured language*
- *It produces efficient programs*
- *It can handle low-level activities*
- *It can be compiled on a variety of computer platforms.*

Facts about C

- *C was invented to write an operating system called UNIX.*
- *C is a successor of B language which was introduced around the early 1970s.*
- *The language was formalized in 1988 by the American National Standard Institute (ANSI).*
- *The UNIX OS was totally written in C.*
- *Today C is the most widely used and popular System Programming Language.*
- *Most of the state-of-the-art software have been implemented using C.*
- *Today's most popular Linux OS and RDBMS MySQL have been written in C.x*

Why use C ?

C was initially used for system development work. C was adopted as a system development language because it produces code that runs nearly as fast as the code written in assembly language. Some examples of the use of C might be –

- *Operating Systems*
- *Language Compilers*
- *Assemblers*
- *Text Editors*
- *Print Spoolers*
- *Network Drivers*
- *Modern Programs*
- *Databases*
- *Language Interpreters*
- *Utilities*

Bug : *In computer technology, a bug is a coding error in a computer program.*

Testing : *Finding the error is known as testing.*

Debugging : *Removing the error is known as debugging.*

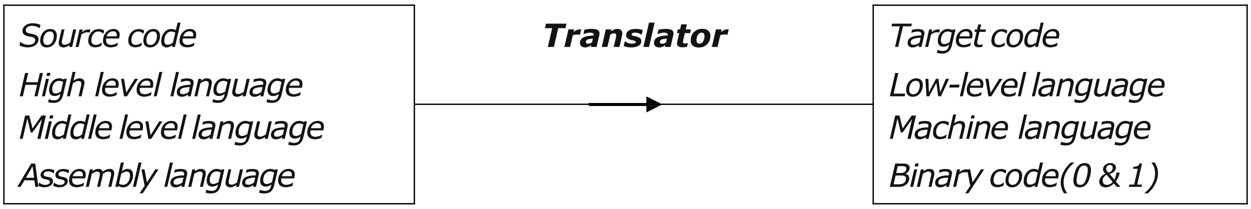
Difference between testing & debugging :-

Testing	Debugging
Finding & locating a defect	*Fixing a defect*
Done by testing team	*Done by development team*
Intension behind is to find as many defect as possible	*Intension is to remove these defects*

What is translator ?

A translator is a program/software that translate one language(source code) to another language(machine code) .

A translator is a converter that convert one language program to another language program.



Source code
High level language
Middle level language
Assembly language

Translator

Target code
Low-level language
Machine language
Binary code(0 & 1)

There are three types of translator:

- ***Compiler***
- ***Interpreter***
- ***Assembler***

- **Compiler :**
  - *A compiler is a program/software that converts high level language program to low level languageprogram.*
  - *It scan whole program at once &generate executable code.*
  - *It is a translator program.*

- **Interpreter :**
  - *An interpreter is a program/software that converts high level language program to low levellanguage program.*
  - *Itscana program **line by line** &generateexecutablecode.*
  - *It is a translator program.*
- **Assembler :**
  - *An assembler is a program/software that converts middle level language program to low levellanguage program.*
  - *It is a translator program.*

Difference between compiler & interpreter :-

Compiler	***Interpreter***
It scan whole program at once & executes	*It scan line by line & executes.*
It convert HLL to LLL	*It is also convert HLL to LLL*
It takes less time	*It takes more time*
It requires more memory	*It requires less memory*

C Programs

*A C program can vary from 3 lines to millions of lines and it should be written into one or more text files with extension "**.c**"; for example, hello.c.*

Algoritham *: The way of writing programming step-by-step is known as algoritham.*

Flow-chart *: The diagrammatic representation of an algoritham is called flowchart.*

Program *: Program is a set of instructions that perform a particular task.*

Software *: Software is a set of programs.*

Note *: By default the extension files for C is **".c"** .*

Rules of developing programming in C- Language

A C program basically consists of the following parts –

- *Preprocessor Commands*
- *Functions*
- *Variables*
- *Statements & Expressions*
- *Comments*

```
#include <stdio.h>
  int main()
    {
        /* my first program in C */
       printf("Govt. Polytechnic, Saharsa\n");
       return 0;
    }
  Output : Govt. Polytechnic, Saharsa
```

Command	**Explanation**
#include <stdio.h>	*This is a preprocessor command that includes standard input output Header file(stdio.h) from the C library before compiling a C program.*
int main()	*This is the main function from where execution of any C program begins.*
{	*This indicates the beginning of the main function.*
/*some comments*/	*Whatever is given inside the command "/* */" in any C program, won't be considered for compilation and execution.*
printf("GP Saharsa");	*printf command prints the output onto the screen.*
getch();	*This command waits for any character input from keyboard.*
return 0;	*This command terminates C program (main function) and returns 0.*
}	*This indicates the end of the main function.*

Note : C language is case sensitive.

For example, printf() and scanf() are different from Printf() and Scanf(). All characters in printf() and scanf() functions must be in lower case.

printf() : - *Printf is a predefined function in* ***"stdio.h"*** *header file, by using this function, we can print the data or user defined message on console or monitor. It is an output function.*

scanf() : - *scanf() is a predefined function in* ***"stdio.h"*** *header file. It can be used to read the input value from the keyword.*

clrscr() : - *It is a predefined function in* ***"conio.h"*** *(console input output header file) used to clear the console screen. It is a predefined function, by using this function we can clear the data from console (Monitor). Using of clrscr( ) is always optional but it should be place after variable or function declaration only.*

getch() :- *It is a predefined function in* ***"conio.h"*** *(console input output header file) will tell to the console wait for some time until a key is hit given after running of program.*

By using this function we can read a character directly from the keyboard. Generally getch( ) are placing at end of the program after printing the output on screen.

Format specifier		**Escape Sequence**	
Format specifier	***Type of value***	***Escape Sequence***	***Character***
%d or %i	*Integer*	***\a***	*Bell (speaker beeps)*
%f	*Float*	***\b***	*Backspace (non-erase)*
%lf	*Double*	***\?***	*Question mark*
%c	*Single character*	***\n***	*New line*
%s	*String*		*Carriage Return*
%u	*Unsigned int*	***\t***	*Horizontal Tab*
%ld	*Long int*	***\v***	*Vertical tab*
%lf	*Long double*	\	*Backslash*
\'	*Single quote*	***\"***	*Double quote*

Standard header file in C Language

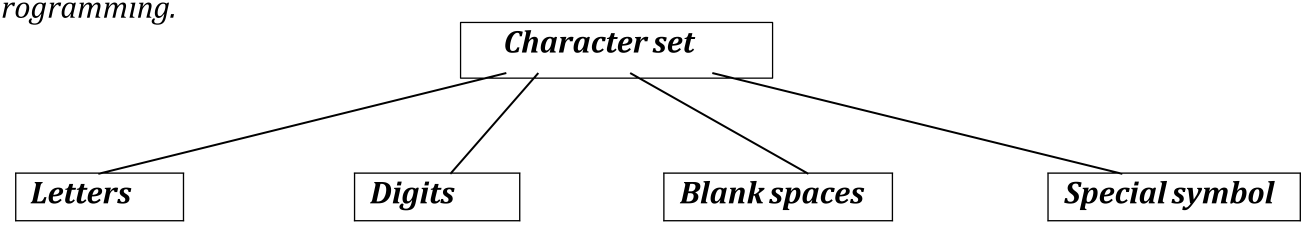
S. No.	Name	Description
1	stdio.h	Input/Output Functions
2	conio.h	console input/output
4	ctype.h	Character Handling Functions
5	math.h	Mathematics Functions
6	string.h	String Functions
7	time.h	Date and Time Functions

CHARACTER SET, KEYWORDS AND DATA TYPES

Character set, keyword and data types are considered the basic building blocks of C program.

Character Set :

The character set of C defines the characters, which can be used by the programmer during programming.



Capital : A to Z
Small : a to z

All decimal numbers(0 to 9)

Blank space
White space
Horizontal space
Vertical space
New line
Form feed

Special symbols

Symbol	Meaning		Symbol	Meaning
~	Tilde			
!	Exclamation mark		{	Left brace
#	Number sign(hash)		}	Right brace
$	Dollar sign		[	Left bracket
%	Percent sign		]	Right bracket
^	Caret		:	Colon
&	Ampersand		"	Quotation mark
*	Asterisk		;	Semicolon
(	Lest parenthesis		<	Opening angle bracket
)	Right parenthesis		>	Closing angle bracket
_	Underscore		?	Question mark
+	Plus sign		,	Comma
|	Vertical bar		.	Period
\	Backslash		=	Equal to sign
`	Apostrophe		-	Minus sign

Delimiters: *The C language pattern uses some special kind of symbol which are called as delimiters. They are given below :*

Delimiters		Use
:	**Colon**	Useful for label.
;	**Semicolon**	Terminates statement.
#	**Hash**	Preprocessor directive.
,	**Comma**	Variable Separator.
( )	**Parenthesis**	used in expression & function
{ }	**Curly Brace**	Scope of statement.
[ ]	**Square Bracket**	Used for array declaration.

Keywords :

- *Keywords are those words whose meaning is already defined by Compiler.*
- *It cannot be used as* ***Variable Name***
- *There are* ***32 Keywords*** *in C*
- *C Keywords are also called as* ***Reserved words*** *.*

32KeywordsinCProgrammingLanguage

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

Identifiers : *Identifiers are names of variables, function and arrays.*
They are user defined names consisting sequence of letter and digits.

Rules for writing an identifier :

a. *First character must be non numeric.*
b. *Lower case letter are preffered.*
c. *Upper case are also preffered.*
d. *Underscore are considered as identifirs.Examples : x2, area_of_circle, _ans ete.*

What is Constants ?

*Constant in C means the content whose value does not change at the time of execution of a program. Constant means "**Whose value cannot be changed.***

C Constants

Numeric constant
1. *Integer*
2. *Real*

Character constant
1. *Single Character*
2. *String character*

1. **Integer** ***:->*** *These are the sequence of number from 0 to 9 without decimal point or fractional part or any symbols.*
   - ***->*** *It may be +ve , -ve or zero.*
   - ***->*** *It requires minimum 2 Bytes or maximum 4 Bytes.*

Examples : 10, +20 , -15 etc.

2. Real constants *: Real constant are often known as floating point constant.*
Examples : 2.5, 5.643, 3.14 etc.

3.Single character constants : *It is a single character represented with single digit or single special symbol or white space enclosed within a pair of single quote marks.*
Examples : ***'a' , 'm' , '5' ' ' etc.***

Note : Character constants have integer values known as ASCII values.

Examples : A = 65 , B = 66 …………… , a = 97, b = 98 ………….etc.

4. String constants: *It is a group of characters enclosed within a double quote. It may be combination of all kinds of symbols.*
Examples : "India" , "5555" , "abc3" etc.

What is variables ?

A variable is a data name used for storing a data value. Its value may be changed during the program execution.

Rules for defining variables :

1. *They must begin with a character without space but underscore is permitted.*
2. *Length of variable varies from compiler to compiler. Generally must of the compiler support 8 character. ANSI standard support upto 31 characters.*
3. *It should not be a C keyword.*
4. *It may be combination of upper & lower case letters.*
5. *It should not be start with a digit.*

Examples : sum, Sum , average , height etc.

There are two types of variables :

(1) Local variable
(2) Global variable

Local variable :

- *The scope of local variables will be within the function only.*
- *These variables are declared within the function and can't be accessed outside the function.*

Global variable :

- *The scope of global variables will be throughout the program. These variables can be accessed from anywhere in the program.*
- *This variable is defined outside the main function. So that, this variable is visible to main function and all other sub functions.*

Data types in C language :

In C programming the data types is a keyword.
It defines the size of data in a program.
It also defines which type of data can be stored in a memory.
It is used to create a variable.
There are two types of data :-

Data types

Primary data types
Char
Int
Float
Double
void

Secondary data types
Array
Pointer
Structure
Enum

All C compilers support a variety of data types. The different compiler support the different data types . They are given below :-

Entire data types in C

Data type	Size(Bytes)	Range	Format string
Char	1	-128 to 127	%c
unsigned char	1	0 to 255	%c
int	2	-32768 to 32767	%d or %i
unsigned int	2	0 to 65535	%u
long	4	-2147483648 to 2147483647	%ld
unsigned long	4	0 to 2147483847	%lu
float	4	$3.4e^{-38}$ to $3.4e^{+38}$	%f or %g
double	8	$1.7e^{-308}$ t0 $1.7e^{+308}$	%lf
long double	10	$3.4e^{-4932}$ to $1.1e^{+4932}$	%ld

We can used modifiers with data types to change the meaning of the data types. The modifiers are used to alter the sign & size of the keywords.
Modifiers are classified into two categories :

Modifiers

Sign modifiers
Signed
Unsigned

Size modifiers
Short
Long

Integer

Short int
Occupies 2 byte
Program runs faster

long int
Occupies 4 byte
Program runs slower

signed int
Occupies 2 byte

unsigned int
Occupies 2 byte

Note : By default the C compiler take short signed integer.

Operators in C programming :

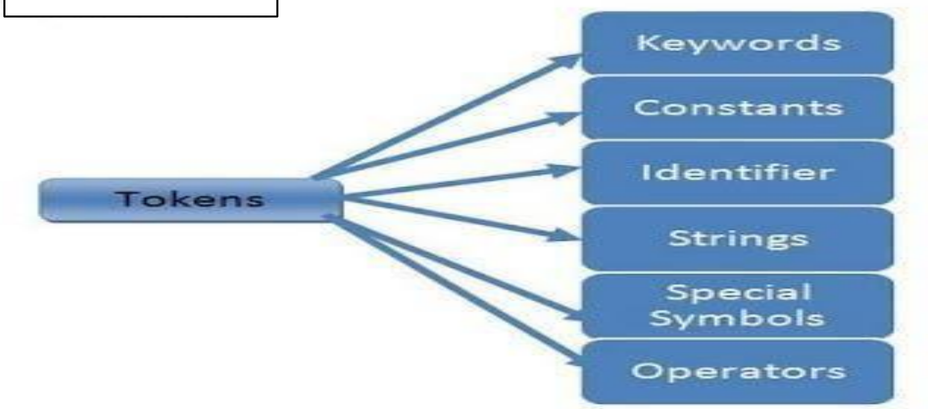
- *An operator is a symbol which operates on a value or a variable.*
- *An operator is a symbol that tells the compiler to perform certain mathematical or logical manipulations. Operators are used in program to manipulate data and variables.*
  For example: + is an operator to perform addition.

C programming has wide range of operators to perform various operations. For better understanding of operators, these operators can be classified as:

Types of Operators		***Description***
Arithmetic operators	***+, -, *, /, %***	*These are used to perform mathematical calculations like addition, subtraction, multiplication, division and modulus*
Assignment operators	***=, +=, -=, *=, etc.***	*These are used to assign the values for the variables in C programs.*
Relational operators	***<, >, <=, >=, != etc.***	*These operators are used to compare the value of two variables.*
Logical operators	***&&, ||, ! etc.***	*These operators are used to perform logical operations on the given two variables.*
Bit wise operators	***&, |, ^, >>, << etc.***	*These operators are used to perform bit operations on given two variables.*
Conditional (ternary) operators	***?***	*Conditional operators return one value if condition is true and returns another value is condition is false.*
Increment / decrement operators	***++, --***	*These operators are used to either increase or decrease the value of the variable by one.*
Special operators	***&, *, sizeof***	*&, *, sizeof( ) and ternary operators.*

C Tokens Chart

- *In C Programming **punctuation,individual words,characters** etc are called **tokens**.*
- *Tokens are **basic building blocks** of C Programming.*



Token	*Meaning*
Keyword	*Keywords are those words whose meaning is already defined by Compiler.*
Constant	*Constants are expressions with a fixed value*
Identifier	*The term identifier is usually used for variable names*
String	*Group of characters*
Special Symbol	*Symbols other than the Alphabets and Digits and white-spaces*
Operators	*A symbol that represent a specific mathematical or non mathematical action*

Storage classes in c language :

*Each variable has a **storage class** which decides scope, visibility and lifetime of that variable. The following storage classes are most oftenly used in C .*

1. ***Automatic variables***
2. ***External variables***
3. ***Static variables***
4. ***Register variables***

Automatic variables

- *A variable declared inside a function without any storage class specification, is by default an **automatic variable**.*
- *They are created when a function is called and are destroyed **automatically** when the function exits.*
- *Automatic variables can also be called local variables because they are local to a function.*

Note : By default they are assigned garbage value by the compiler.

External variables

- *The **extern** keyword is used before a variable to inform the compiler that this variable is declared somewhere else.*
- *The **extern** declaration does not allocate storage for variables..*

Static variables

- *A **static** variable tells the compiler to persist the variable until the end of program.*
- *Instead of creating and destroying a variable every time when it comes into and goes out of scope, **static** is initialized only once and remains into existence till the end of program.*
- *A static variable can either be internal or external depending upon the place of declaraction.*

Note : By default they are assigned 0 (zero) value by the compiler.

Register variables

- ***Register** variable inform the compiler to store the variable in register instead of memory.*

- ***Register*** *variable has faster access than normal variable.*
- *Frequently used variables are kept in register. Only few variables can be placed inside register.*

NOTE : We can never get the address of such variables

Syntax :

```
register int number;
```

Control statements

Branching

- *If-statement*
- *If-else-statement*
- *Nested If-else-statement*

Looping

for loop
while loop
do-while loop

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.

Looping

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

If-statement : *Syntax :*
If(condition)
{
statements;
}
	for loop : Syntax :
	for(initialization; condition; steps)
	{
	Block of statements;
	}

If-else-statement :
Syntax : If(condition)
{
Block of statements;
}
else
{
statements;
}
	While loop : Syntax :
	While(condition)
	{
	Block of statements;
	Increment/decrement;
	}

Nested If-else-statement
If(condition)
{
If(condition)
{ statements; }
else
{ statements; }
}
else
{
statements;
}
	Do While loop : Syntax :
	do
	{
	Block of statements;
	Increment/decrement;
	} While(condition);

Some Statements :

- *break Statement*
- *continue Statement*
- *switch Statement*

Break Statement :

The break statement terminates the loop immediately when it is encountered. The break statement is used with decision making statement such as if...else.

Syntax of break statement : break ;

In C programming, break statement is also used with switch...case statement.

Continue Statement :

The continue statement skips some statements inside the loop. The continue statement is used with decision making statement such as if...else.

Syntax of continue Statement : continue ;

- **Difference Between break and continue :**

break	***continue***
A break can appear in both switch and loop statements.	*A continue can appear only in loop statements.*
It terminate the statements.	*It doesn't terminate the statements.*
The break statement can be used in both switch and loop statements.	*The continue statement can appear only in loops. You will get an error if this appears in switch statement.*
Break is used to terminate the statements.	*Continue is used to skip the statements.*

Switch Statement :

***A switch statement** is a type of selection control mechanism used to allow the value of a variable or expression to change the control flow of program execution via a multiway branch.*

Syntax :

```
Switch(variable)
 {
  Case value :
              Statements ;
               Break;
  Case value :
              Statements ;
               Break;
   ........................
   ....................
   ....................
   ....................
   ....................

   Default :
                       Statements ;
  }
```

Array

It is a group of similar type of data which can store data in continous manner.

- *Arrays is the **Collection of Elements.***
- *Arrays is collection of the Elements of the **same data type.***
- *All Elements are stored in the Contiguous memory.*
- *All elements in the array are accessed using the subscript variable.*
- *It is a data structure.*

Array element	*4*	*5*	*33*	*13*	*1*
Location	*0*	*1*	*2*	*3*	*4*

4,5,33,13,1 are actual data items or array element.
0,1,2,3,4 are index variables ..(Subscript variable)

Index or subscript variable :

1. *Individual data items can be accessed by the name of the array and an integer enclosed in square bracket called subscript variable / index*
2. *Subscript Variables helps us to identify the item number to be accessed in the contiguous memory.*

There are two types of array :

1. ***Single Dimensional Array***
2. ***Multi Dimensional Array***

1. Single Dimensional Array :

Single Dimensional array is used to represent and store data in a linear form.
*Array having only one subscript variable is called **One-Dimensional array.***
*It is also called as **Single Dimensional Array** or **Linear Array***

```
Syntax :       data-type array_name [size];
Example :      int  a[5];
               char  c[20];
               float  f[3];
```

2. Multi Dimensional Array :

1. Array having more than one subscript variable is called Multi-Dimensional array.
*2. Multi Dimensional Array is also called as **Matrix**.*

Syntax :

```
data-type array_name [row_subscripr] [column_subscripr];
                int a[3][3];
```

Array declaration :

Array has to be declared before using it in C Program. Array is nothing but the collection of elements of similar data types.

Syntax :

```
            data-type  array_name [size1][size2]...........[sizen];
```

Array declaration tells to the compiler :

1. *Type of the Array*
2. *Name of the Array*
3. *Number of Dimension*
4. *Number of Elements in Each Dimension*

Initialization of an Array :

Initializing 1-D Array is called as compiler time initialization if and only if we assign certain set of values to array element before executing program. i.e at compilation time.

Ways Of Array Initializing 1-D Array :

1. ***Size is Specified Directly :*** *At the time of compilation all the elements are at Specified Position So This Initialization Scheme is Called as "**Compile Time Initialization**".*

   Example : `int num[5] = {2,8,7,6,0}`

2. ***Size is Specified Indirectly :*** *In this scheme of compile time Initialization, We does not provide size to an array but instead we provide set of values to the array.*

   Example `int num[ ] = {2,8,7,6,0}`

   - *Compiler Counts the Number Of Elements Written Inside Pair of Braces and Determines the Size of An Array.*
   - *After counting the number of elements inside the braces, The size of array is considered as 5 during complete execution.*
   - *This type of Initialization Scheme is also Called as "**Compile Time Initialization**"*

Ways Of Array Initializing 2-D Array : *For initializing 2D Array we can need to assign values to each element of an array using the below syntax.*

```
int a[3][2] = {  { 1, 4 }
                 { 5, 2 }
                 { 6, 5 }
              } ;
```

Two-Dimensional array :

- *Two Dimensional Array requires **Two Subscript Variables***
- *Two Dimensional Array stores the values in the form of matrix.*
- *One Subscript Variable denotes the "**Row**" of a matrix.*
- *Another Subscript Variable denotes the "**Column**" of a matrix.*

Example : `int a[3][4];`

What are Advantages and Disadvantages of arrays?

Advantages:	***Disadvantages:***
1. It is used to represent multiple data items of same type by using only single name.	*1. We must know in advance that how many elements are to be stored in array.*
2. It can be used to implement other data structures like linked lists, stacks, queues, trees, graphs etc.	*2. Array is static structure. It means that array is of fixed size. The memory which is allocated to array can not be increased or reduced.*
3. 2D arrays are used to represent matrices.	*3. Since array is of fixed size, if we allocate more memory than requirement then the memory space will be wasted. And if we allocate less memory than requirement, then it will create problem.*
	4. The elements of array are stored in consecutive memory locations. So insertions and deletions are very difficult and time consuming.

Pointer

1. *Pointer is a variable which stores the address of another variable.*
2. *Since Pointer is also a kind of variable , thus pointer itself will be stored at different memory location.*

Variable name →	*i*	*j*	*k*
Value of variable →	*5*	*65422*	*65524*
Address of variable →	*65422*	*65524*	*65530*

1. ***i** is the name given for particular memory location of ordinary variable.*
2. *Let us consider it's Corresponding address be 65422 and the Value stored in variable **'i'** is 5.*
3. *The address of the variable **'i'** is stored in another integer variable whose name is **'j'** and which is having corresponding address 65524.*

thus we can say that – *j = &i;*
j = address of i;

Here j is not ordinary variable , It is special variable and called pointer variable as it stores the address of the another ordinary variable. We can summarize it like –

Variable Name	***Variable Value***	***Variable Address***
i	*5*	*65522*
j	*65422*	*65524*

Address operator (&) :

1. *Pointer address operator is denoted by '&' symbol.*
2. *When we use ampersand symbol as a prefix to a variable name '&', it gives the address of that variable.*

Example : ***&n → it gives an address on variable n.***

```
#include<stdio.h>
void main( )
{
    int n=10;
   printf("\n value of n is : %d", n);
   printf("\n value of &n is : %u", &n);
}
```

Output :-

```
value of n is : 10
 value of &n is : 1002
```

Note : print the address of the variable using ampersand operator.In order to print the variable we simply use name of variable while to print the address of the variable we use ampersand along with `%u`*.*

Pointer Declaration in C :

Syntax:

data_type *pointer_name;

Explanation:

Data_type

- *Type of variable that the pointer **points to.***
- *OR data type whose address is stored in **pointer_name.***

Asterisk (*)

- *Asterisk is called as **Indirection Operator***
- *It is also called as **Value at address Operator***
- *It Indicates **Variable declared is of Pointer type.***

Pointer_name

- *Must be any **Valid C identifier.***
- *Must follow all Rules of Variable name declaration.*

Example : ***int *a ;***
char *p ; ***etc.***

Whitespace while writing pointer :

pointer variable name and asterisk can contain whitespace because whitespace is ignored by compiler.

```
int *ptr;
int *     ptr;
int          *ptr;
```

All the above syntax are legal and valid. We can insert any number of spaces or blanks inside declaration. We can also split the declaration on multiple lines.

D. Key points for Pointer :

1. *pointer is special type of variable which stores the address of ordinary variable.*
2. *Pointer can only store the whole or integer number because address of any type of variable is considered as integer.*
3. *It is good to initialize the pointer immediately after declaration.*
4. *& symbol is used to get address of variable.*
5. ** symbol is used to get value from the address given by pointer.*

Pointer storing the address of following Data	*Pointer is called as*
Integer	*Integer Pointer*
Character	*Character Pointer*
Double	*Double Pointer*
Float	*Float Pointer*

Types of variable :

1. *Simple Variable that stores a value such as integer,float,character*
2. *Complex Variable that stores address of simple variable i.e pointer variables. Examples :*

```
#include<stdio.h>
  Void main( )
    {
         int   a=5;
         int   *ptr;
         ptr = &a;
     }
```

ExplanationofExample:

Point	*Variable 'a'*	*Variable 'ptr'*
Name of Variable	*a*	*ptr*
Type of Value that it holds	*Integer*	*Address of Integer 'a'*
Value Stored	*3*	*2016*
Address of Variable	*2016 (Assumption)*	*4016 (Assumption)*

Simpleexamples:

```
                        #include<stdio.h>
                          Void main( )
                            {
                                 int a=5;
                                 int *ptr, **pptr;
                                 ptr = &a;
                                 pptr= &ptr;
                             }
```

ExplanationofExample

With reference to above program –

Point	***Variable 'a'***	***Variable 'ptr'***	***Variable 'pptr'***
Name of Variable	*a*	*ptr*	*pptr*
Type of Value that it holds	*Integer*	*Address of 'a'*	*Address of 'ptr'*
Value Stored	*3*	*2001*	*4001*
Address of Variable	*2001*	*4001*	*6001*

What is Null Pointer ?

1. *NULL Pointer is a pointer which is pointing to nothing.*
2. *NULL pointer points the base address of segment.*
3. *In case, if you don't have address to be assigned to pointer then you can simply use NULL*
4. *Pointer which is initialized with NULL value is considered as NULL pointer.*
5. *NULL is macro constant defined in following header files –*

   - *stdio.h*
   - *alloc.h*
   - *mem.h*
   - *stddef.h*
   - *stdlib.h*

Void Pointer Basics :

1. *In C **General Purpose Pointer** is called as void Pointer.*
2. *It does not have any data type associated with it*
3. *It can store address of any type of variable*
4. *A void pointer is a C convention for a raw address.*

5.The compiler has no idea what type of object a void Pointer really points to ?

Pointer Arithmetic

Pointer arithmetic is very important to understand, if you want to have complete knowledge of pointer. In this topic we will study how the memory addresses change when you increment a pointer.

16 bit Machine (Turbo C)

In a 16 bit machine, size of all types of pointer, be it int, float*, char* or double* is always **2 bytes**. But when we perform any arithmetic function like increment on a pointer, changes occur as per the size of their primitive data type.*

Size of datatypes on 16-bit Machine :

Type	***Size(bytes)***
Char	*1*
Int or signed int	*2*
float	*4*
long	*4*
double	*8*
Long double	*10*

Examples for Pointer Arithmetic :

Now lets take a few examples and understand this more clearly.

```
int* i;
i++;
```

*In the above case, pointer will be of 2 bytes. And when we increment it, it will increment by 2 bytes because **int** is also of 2 bytes.*

```
float* i;
i++;
```

*In this case, size of pointer is still 2 bytes. But now, when we increment it, it will increment by 4 bytes because **float** is of 4 bytes.*

```
double* i;
i++;
```

*Similarly, in this case, size of pointer is still 2 bytes. But now, when we increment it, it will increment by 8 bytes because its data type is **double**.*

What are the advantages of using pointers in a program?

Major advantages of pointers are:

- *Pointers provide direct access to memory.*
- *Pointers provide a way to return more than one value to the functions*
- *Reduces the storage space and complexity of the program*
- *Reduces the execution time of the program*
- *Provides an alternate way to access array elements.*
- *Pointers allows us to perform dynamic memory allocation and deallocation.*
- *Pointers helps us to build complex data structures like linked list, stack, queues, trees, graphs etc.*

Structures and Union

Why Use Structure in C :

In C language array is also a user defined data type but array hold or store only similar type of data, If we want to store different-different type of data in then we need to defined separate variable for each type of data.

Structure

It is a group of dis-similar type of data which can store data in hetrogenious manner. structure can store collection of any type of data.

Or

Structure is composition of the different variables of different data types , grouped under same name.

Example :

```
typedef struct {
      char name[64];
      char course[128];
      int age;
      int year;
 } student;
```

Some Important Definitions of Structures :

1. ***Each member declared in Structure is called member.***

```
                char name[64];
                char course[128];
                int age;
                int year;
```

are some examples of members.

2. ***Name given to structure is called as tag***

Student

3. ***Structure member may be of different data type including user defined data-type also***

```
typedef struct
  {
     char name[64];
     char course[128];
     book b1;
     int year;
  } student;
```

Here book is user defined data type.

Note1: *Minimum size of Structure is one byte and Maximum size of Structure is sum of all members variable size.*

Note2: *Empty Structure is not possible in C Language.*

Defining a structure

struct *keyword is used to define a structure.* ***struct*** *define a new data type which is a collection of different type of data.*

Syntax :

```
struct  structure_name
   {
      //Statements
   };
```

At end of the structure creation (;) must be required because it indicates that an entity is constructed.

```
struct tagname
{
Datatype1  member1;
Datatype2  member2;
Datatype3 member3;
..........
};
```

Example of Structure

```
struct Book
  {
     char name[15];
     int price;
     int pages;
  };
```

Initialization of structure :

1. *When we declare a structure, memory is not allocated for un-initialized variable.*

```
struct student
{
  char name[20];
  int roll;
  float marks;
}std1 = { "Pritesh",67,78.3 };

Std1 = {“Paritesh”,67,78.3};
```

This is the code for initializing structure variable in C programming.

ASSIGNING VALUES TO STRUCTURE ELEMENTS

To assign todays date to the individual elements of the structure todays_date, the statement.

```
todays_date.day = 21;
todays_date.month = 07;
todays_date.year = 1985;
```

is used.

NOTE *: the use of the . element to reference the individual elements within todays_date.*

Array of Structure :

Structure is used to store the information of One particular object but if we need to store such 100 objects then Array of Structure is used.

Example:

```
struct Bookinfo
{
    char[20] bname;
    int pages;
    int price;
}Book[100];
```

Explanation:

1. *Here Book structure is used to Store the information of one Book.*
2. *In case if we need to store the Information of 100 books then Array of Structure is used.*
3. *b1[0] stores the Information of 1st Book , b1[1] stores the information of 2nd Book and So on We can store the information of 100 books.*

Difference Between Array and Structure

	Array	***Structure***
1	*Array is collection of homogeneous data.*	*Structure is the collection of heterogeneous data.*
2	*Array data are access using index.*	*Structure elements are access using . operator.*
3	*Array allocates static memory.*	*Structures allocate dynamic memory.*
4	*Array element access takes less time than structures.*	*Structure elements takes more time than Array.*

What is Union ?

- *Unions are conceptually similar to structures.*
- *The syntax of union is also similar to that of structure.*
- *The only differences is in terms of storage.*
- *A union is declared using union keyword.*
- *, it cannot handle all the members at same time.*
- *A union may contain many members of different types.*
- *In structure each member has its own storage location, whereas all members of union uses a single shared memory location which is equal to the size of its largest data member.*

Note #1 : Union and Structure are Almost Similar

```
union stud
{
  int roll;
  char name[4];
  int marks;
}s1;
```

```
struct stud
{
  int roll;
  char name[4];
  int marks;
}s1;
```

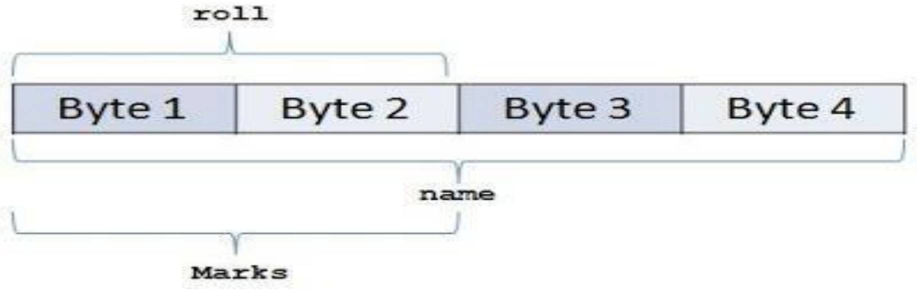
we can say that both structure and union are same except Keyword.

Note #2 : Multiple Members are Collected Together Under Same Name

```
int roll;
char name[4];
int marks;
```

We have collected three variables of different data type under same name .

Note #3 : All Union Members Occupy Same Memory Area



For the union maximum memory allocated will be equal to the data member with maximum size. In the example character array 'name' have maximum size thus maximum memory of the union will be 4 Bytes.

Maximum Memory of Union = Maximum Memory of Union Data Member

Note #4 : Only one Memberwill be active at atime.

*Suppose we are accessing one of the data member of union then we cannot access other data member since we can access single data member of union because <u>each data member shares same memory.</u> By Using Union we can **Save Lot of Valuable Space.***

Function

- *A function is a block of code that performs a specific task.*

  Or
- *A function is a group of statements that together perform a task.*
- *Every C program has at least one function, which is **main( )**.*
- *All the most trivial programs can define additional functions.*

There are two types of function:

1. ***Pre-defined function***
2. ***User-defined function***

Pre-defined functions :

- *These functions are defined in the header file.*
- *When you include the header file, these functions are available for use. For example: The **printf( )** is a pre-defined function to send formatted output to the screen (display output on the screen). This function is defined in "**stdio.h**"header file.*
- *There are other numerous library functions defined under "**stdio.h**", such as **scanf( )**, **printf( )**, **getchar( )** etc. Once you include **"stdio.h"** in your program, all these functions are available for use.*

User-defined functions :

- *As mentioned earlier, C language allows programmer to define functions.*
- *Such functions created by the user are called user-defined functions.*
- *Depending upon the complexity and requirement of the program, you can create as many user-defined functions as you want.*

Declaration of function :

Syntax :

```
Return_data-type  function name(argument)
                                    {
                                      …………….
                                      …………….statements;
                                    }
```

*A function **declaration** tells the compiler about a function's name, return type, and parameters. A function **definition** provides the actual body of the function.*

Advantages of user-defined function

1. *The program will be easier to understand, maintain and debug.*
2. *Reusable codes that can be used in other programs.*
3. *A large program can be divided into smaller modules. Hence, a large project can be divided among many programmers.*

Top down approach of problem solving :

1. *Every C program starts **from main function**.*
2. *Every function is **called directly or indirectly through main**.*
3. ***Top down approach**. (functions are executed from top to bottom)*

\Modular programming/advantage of function:

1. *We can divide c program in **smaller modules**.*
2. *We can call module whenever require. e.g suppose we have written calculator program then we can write 4 modules (i.e add,sub,multiply,divide)*
3. *Modular programming **makes C program more readable**.*
4. *Modules once created , **can be re-used in other programs**.*

Recursion

A function that calls itself is known as recursive function. And, this technique is known as recursion.

How recursion works?

```
void recurse( )
    {
        ... .. ...
        recurse( );
        ... .. ...
    }
int main( )
    {
        ... .. ...
        recurse( );
        ... .. ...
    }
```

The recursion continues until some condition is met to prevent it. To prevent infinite recursion, if...else statement (or similar approach) can be used where one branch makes the recursive call and other doesn't.

Advantages and Disadvantages of Recursion

Advantages	*Disadvantages*
Avoidance of unnecessary calling of functions.	*A recursive function is often confusing.*
A substitute for iteration where its solution is very complex(reduce the code size)	*The exit point must be explicitly coded.*
for Tower of Honai application, a recursive function is bet suited.	*It is difficult to trace the logic of the function.*
Extremely useful when applying the same solution.	*Recursions use more memory and are generally slow.*

The Standard Library Functions

Some of the "commands" in C are not really "commands" at all but are functions. For example, we have been using printf and scanf to do input and output, and we have used rand to generate random numbers - all three are functions.

The most common standard libraries and a brief description of the most useful functions they contain follows:

1. ***stdio.h: I/O functions:***
   a. ***getchar()*** *returns the next character typed on the keyboard.*
   b. ***putchar()*** *outputs a single character to the screen.*
   c. ***printf()*** *as previously described*
   d. ***scanf()*** *as previously described*
2. ***string.h: String functions***
   a. ***strcat()*** *concatenates a copy of str2 to str1*
   b. ***strcmp()*** *compares two strings*
   c. ***strcpy()*** *copys contents of str2 to str1*

3. ***ctype.h: Character functions***
    a. ***isdigit()*** *returns non-0 if arg is digit 0 to 9*
    b. ***isalpha()*** *returns non-0 if arg is a letter of the alphabet*
    c. ***isalnum()*** *returns non-0 if arg is a letter or digit*
    d. ***islower()*** *returns non-0 if arg is lowercase letter*
    e. ***isupper()*** *returns non-0 if arg is uppercase letter*

4. ***math.h: Mathematics functions***
    a. ***acos()*** *returns arc cosine of arg*
    b. ***asin()*** *returns arc sine of arg*
    c. ***atan()*** *returns arc tangent of arg*
    d. ***cos()*** *returns cosine of arg*
    e. ***exp()*** *returns natural logarithim e*
    f. ***fabs()*** *returns absolute value of num*
    g. ***sqrt()*** *returns square root of num*

5. ***time.h: Time and Date functions***
    a. ***time()*** *returns current calender time of system*
    b. ***difftime()*** *returns difference in secs between two times*
    c. ***clock()*** *returns number of system clock cycles since program execution*

6. ***stdlib.h:Miscellaneous functions***
    a. ***malloc()*** *provides dynamic memory allocation, covered in future sections*
    b. ***rand()*** *as already described previously*
    c. ***srand()*** *used to set the starting point for rand()*

Function Prototypes

A function prototype is a function declaration that specifies the data types of its arguments in the parameter list.

- *Our program starts from main function. Each and every function is called directly or indirectly through main function.*
- *Like variable we also need to declare function before using it in program.*
- *In C, declaration of function is called as prototype declaration.*
- *Function declaration is also called as function prototype.*

Important Points :

1. *It tells name of function,return type of function and argument list related information to the compiler*
2. *Prototype declaration always ends with semicolon.*
3. *Parameter list is optional.*
4. *Default return type is integer.*

What is Parameter ?

- *In C Programming Function Passing Parameter is Optional.*
- *We can Call Function Without Passing Parameter .*

Function With Parameter :

add(a,b);

Here Function add( ) is Called and two Parameters a & b are Passed to Function.

Function Call Without Passing Parameter :

Display( );

What is Actual Definition of Parameter ?

*In computer programming, a **parameter** is a special kind of variable, used in a subroutine to refer to one of the pieces of data provided as input to the subroutine.These pieces of data are called **arguments**.*

1. *Parameter Means Values Supplied to Function so that Function can Utilize These Values.*
2. *Parameters are Simply Variables.*
3. *Difference between Normal Variable and Parameter is that "These Arguments are Defined at the time of Calling Function".*
4. *Syntactically We can pass any number of parameter to function.*
5. *Parameters are Specified Within Pair of Parenthesis .*
6. *These Parameters are Separated by Comma (,)*

```
Display(a,b,c,d,e);
```

- ***Parameter*** *: The names given in the function definition are called Parameters.*
- ***Argument :*** *The values supplied in the function call are called Arguments.*

Formal Parameter :

- *Parameter Written In Function Definition is Called "Formal Parameter".*

```
void main()
{
int num1;
display(num1);
}

void display(int para1)
{
------------
------------
}
```

- *Para1 is "**Formal Parameter**"*

Actual Parameter :

- *Parameter Written In Function Call is Called "Actual Parameter".*

```
void main()
{
int num1;
display(num1);
}

void display(int para1)
{
    ------------
    ------------

}
```

- *num1 is "**Actual Parameter**"*

Algorithm in Programming

What is an algoritham ?

- *An algorithm is the way of writing a programming language step by steps.*
- *In programming, algorithm are the set of well defined instruction in sequence to solve a program.*
- *An algorithm should always have a clear stopping point.*

Qualities of a good algorithm

1. *Inputs and outputs should be defined precisely.*
2. *Each steps in algorithm should be clear and unambiguous.*
3. *Algorithm should be most effective among many different ways to solve a problem.*
4. *An algorithm shouldn't have computer code. Instead, the algorithm should be written in such a way that, it can be used in similar programming languages.*

Examples Of Algorithms In Programming

Write an algorithm to add two numbers entered by user.

```
Step 1: Start
Step 2: Declare variables num1, num2 and sum.
Step 3: Read values num1 and num2.
Step 4: Add num1 and num2 and assign the result to sum.
    $sum \leftarrow num1+num2$
Step 5: Display sum
Step 6: Stop
```

Write an algorithm to find the largest among three different numbers entered by user.

```
Step 1: Start
Step 2: Declare variables a,b and c.
Step 3: Read variables a,b and c.
Step 4: If $a>b$
        If $a>c$
            Display a is the largest number.
        Else
            Display c is the largest number.
    Else
        If $b>c$
            Display b is the largest number.
        Else
            Display c is the greatest number.
Step 5: Stop
```

Write an algorithm to find all roots of a quadratic equation $ax^2+bx+c=0$.

```
Step 1: Start
Step 2: Declare variables a, b, c, D, x1, x2, rp and ip;
Step 3: Calculate discriminant
    $D \leftarrow b2-4ac$
Step 4: If $D \geq 0$
        $r1 \leftarrow (-b+\sqrt{D})/2a$
        $r2 \leftarrow (-b-\sqrt{D})/2a$
        Display r1 and r2 as roots.
    Else
        Calculate real part and imaginary part
        $rp \leftarrow b/2a$
        $ip \leftarrow \sqrt{(-D)}/2a$
        Display rp+j(ip) and rp-j(ip) as roots
Step 5: Stop
```

Write an algorithm to find all roots of a quadratic equation $ax^2+bx+c=0$.

```
Step 1: Start
Step 2: Declare variables a, b, c, D, x1, x2, rp and ip;
Step 3: Calculate discriminant
        D←b2-4ac
Step 4: If D≥0
            r1←(-b+√D)/2a
            r2←(-b-√D)/2a
            Display r1 and r2 as roots.
        Else
            Calculate real part and imaginary part
            rp←b/2a
            ip←√(-D)/2a
            Display rp+j(ip) and rp-j(ip) as roots
Step 5: Stop
```

Write an algorithm to find the factorial of a number entered by user.

```
Step 1: Start
Step 2: Declare variables n,factorial and i.
Step 3: Initialize variables
         factorial←1
         i←1
Step 4: Read value of n
Step 5: Repeat the steps until i=n
    5.1: factorial←factorial*i
    5.2: i←i+1
Step 6: Display factorial
Step 7: Stop
```

Write an algorithm to find the Fibonacci series till term≤1000.

```
Step 1: Start
Step 2: Declare variables first_term,second_term and temp.
Step 3: Initialize variables first_term←0 second_term←1
Step 4: Display first_term and second_term
Step 5: Repeat the steps until second_term≤1000
    1.  : temp←second_term
    2.  : second_term←second_term+first term
    3.  : first_term←temp
4.  : Display second_term
Step 6: Stop
```

Note : *Algorithm is not the computer code. Algorithm are just the instructions which gives clear idea to write the computer program.*

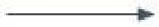
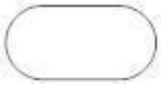
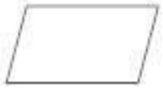
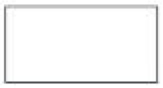
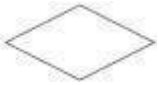
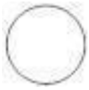
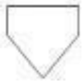
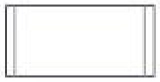
Flowchart In Programming

What is flow-chart ?

- *Flowchart is a diagrammatic representation of an algorithm.*
- *Flowchart are very helpful in writing program and explaining program to others.*

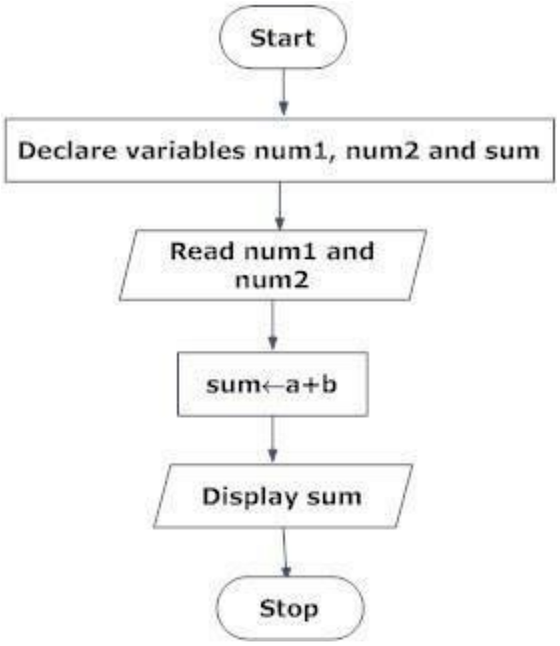
Symbols Used In Flowchart

Different symbols are used for different states in flowchart, For example: Input/Output and decision making has different symbols. The table below describes all the symbols that are used in making flowchart

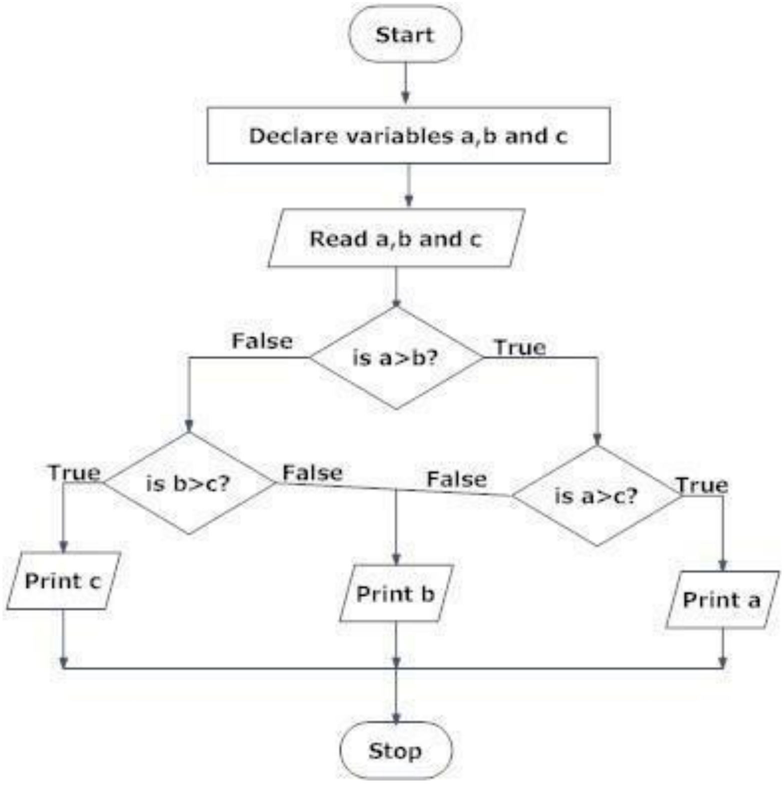
Symbol	Purpose	Description
	*Flow line*	*Used to indicate the flow of logic by connecting symbols.*
	*Terminal(Stop/Start)*	*Used to represent start and end of flowchart.*
	*Input/Output*	*Used for input and output operation.*
	*Processing*	*Used for airthmetic operations and data-manipulations.*
	*Desicion*	*Used to represent the operation in which there are two alternatives, true and false.*
	*On-page Connector*	*Used to join different flowline*
	*Off-page Connector*	*Used to connect flowchart portion on different page.*
	*Predefined Process/Function*	*Used to represent a group of statements performing one processing task.*

Examples of flowcharts in programming

Draw a flowchart to add two numbers entered by user.



Draw flowchart to find the largest among three different numbers entered by user.



QUESTIONS

Type of theory questions:-

1. *What do you mean by testing & debugging ?*
2. *What is union ? Differentiate union & structure .*
3. *What is structure ? How it declares ?*
4. *What do you mean by call by value & call by reference.*
5. *What do you mean by bottom-up & top-down programming ?*
6. *Define function prototype & function declaration.*
7. *Write the syntax of for , while & do-while.*
8. *Define operator . Explain arithmetic operators.*
9. *Explain machine language, assembly language & high level language.*
10. *Define modular programming. Explain its type.*
11. *What is pointer ? How it store the address of another variable.*
12. *What is difference between compiler & interpreter explain briefly ?*
13. *How array can declare & initialize.*
14. *Write the generation of programming language.*

Programs :-

1. *Write a program to calculate simple interest.*
2. *Write a program to swap two variables.*
3. *Write a program to fine the entered number is even or odd.*
4. *Write a program to calculate multiplication of two matrices.*
5. *Write a program to generate Fibonacci series.*
6. *Write a program to search an array element.*
7. *Write a program to calculate the sum of two numbers using pointer.*
8. *Write a program to check whether the entered number is prime or not.*
9. *Write a program to convert lower-case to upper-case.*

10. *Write a program to check whether the given number is palindrome or not.*
11. *Write a program to calculate the average temperature of 5 Sunday.*
12. *Write a program to calculate the GCD of a number.*
13. *Write a program to generate pyramid.*
14. *Write a program to generate first 15 natural number.*
15. *Write a program to find the roots of quadratic.*
16. *Write a program to find the factorial of a number.*

Algorithms :-

1. *Write an algorithm for prime number.*
2. *Write an algorithm for binary search.*
3. *Write an algorithm to solve towers of honoi problems.*
4. *Write an algorithm for simple interest.*
5. *Write an algorithm to find factorial of a number.*

Flow-chart :-

1. *Drow a flow-chart to find simple interest.*
2. *Drow a flow-chart to find the roots of the quadratic equation.*
3. *Drow a flow-chart to find the largest between three numbers.*
4. *Drow a flow-chart to find the area of the circle.*
5. *Drow a flow-chart to find the positive integers.*

Write short notes on :-

1. *Keywords*
2. *Datatypes*
3. *Void function*
4. *Bitwise/Relational operator*
5. *Types of function*
6. *Linking & loading*
7. *Function call*
8. *Break & Continou*
9. *Switch statements*
10. *Recursion*
11. *Debbugger*
12. *Pointer*
13. *Template in C.*