C - Language

Introduction

1) What is a Language?
2) What is a computer language

→ Language is a communication channel b/w two persons

E.g: Eng, tel, hin etc.,

→ Computer language is also a communication channel b/w the person & the system.

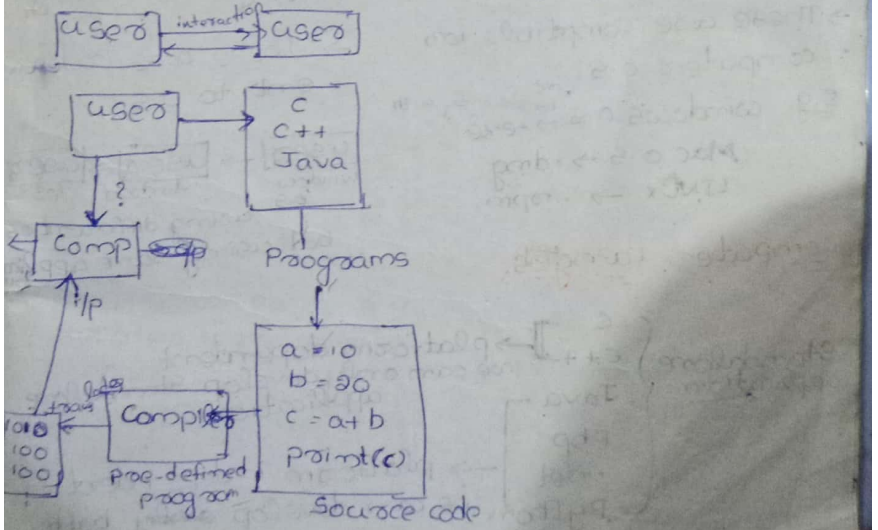
→ Lang ⇒ we have to follow/know Grammar & how to form a sentence

→ computer lang, we have to form some programs

⇒ What is the need of computer language?

user ⟶ High level language

System (0,1) ⟶ Machine language



⇒ What is the purpose of computer languages?

Mainly used for developing computer applications by which users can communicate with machines.

⇒ What is an interface?

is nothing but which interacts b/w users & the computers.

E.g operating system.

⇒ Why to use Computer Languages?

Mainly used to develop computer application.

There are two types of computer applications

Standalone applications

→ They must be installed in the computer to work with that applications.

E.g: Broowsers, MS-office etc.,

→ These are comptible for computers o.s

E.g: windows o.s → .exe (.exe works only with)

Mac o.s → .dmg

LINUX → .rpm

Web applications

→ No need to install to work coith that apps.

E.g: gmail, facebook etc

→ These are independent to o.s.

user (window o.s) → user (Android) → user (IOS)

using different os but using same application

Computer Langes

Standalone application

- C
- C++ → platform dependent, we can only develop Standalone application
- Java
- Php
- .net
- Python → platform independent, we can develop both Standalone & web application

Platform Dependency in C :

extension names of files.

abc.txt ⟶ notepad

xyz.mp3 ⟶ VLC media player

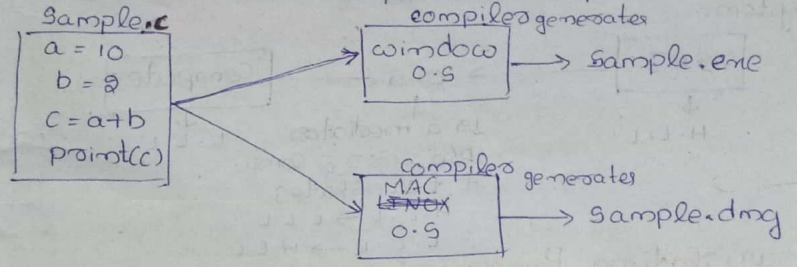
They are platform dependent because they run on their certain platform

Windows O.S → .exe

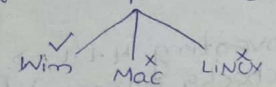
Mac O.S → .dmg

Linux O.S → .rpm, .tar etc..

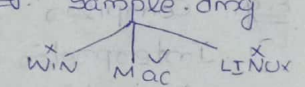
We have to install those extention files on the O.S because these are platform dependent.



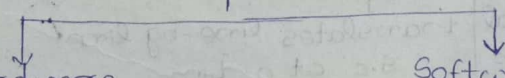
E.g: Sample.exe



E.g: Sample.dmg



Computer



Hardware

→ The physical parts of a computer which can be touchable.

E.g: keyboard, mouse etc.

Software

→ collection of programs

→ programs are set of instructions (or) rules for instructing a computer to perform a task.

E.g: C, C++, Java etc.,

Software

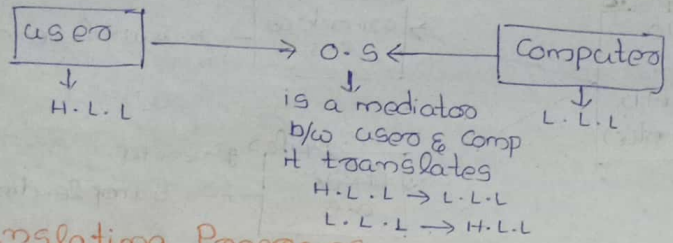
System s/w
E.g: O.S

Application s/w
→ used for specific purpose
E.g: Tally → Accounts
oracle → Database

Without system s/w we cant use application s/w

Operating System

is a s/w [or] collection of programs acts as an interface b/w the users and the system.



Translating Programs

E.g: Compiler
Interpreter } for converting H.L.L → L.L.L & also checks errors in [0, 1] the program
Assembler → Assembly → M.L.L / L.L.L

Compiler	Interpreter
translates total s.c at a time	translates line-by line s.c at a time.

Computer Language

Machine lang [0's & 1's]

Assembly lang
→ Mnemonic codes
E.g: Add, sub, mul, Div, id
→ it is the translating program to convert A.L → M.L.L

High level lang
→ Normal english digits

To convert H.L → M.L we use compiler & interpreter to translate it.

→ What is a C-Language

C is a middle-level, General purpose, compiler based & procedural computer programming language. supporting structured programming

→ C is a middle-level language because it combines the features of both low level & high level languages.

C uses in →
- Web browsers
- System S/w
- Text editors
- Databases
- PC/Mobile Games
- Enterprise Apps
- Embedded devices

'C' → procedural Oriented Programming.

History of C-Language

→ Developed by Dennis Ritchae in 1972 at AT&T Bell Labs of U.S.A

AT&T → American Telephone & Telegraph

→ C is developed to implement UNIX O.S

YEAR	Language	Developed by
1960	ALGOL → Algorithmic Language	International Groups
1963	CPL → Combined Programming language	Christopher Strachey
1967	BCPL → Basic Combined Programming language	Martin Richards
1970	B	Ken Thompson
1972	C	Dennis Ritchae

1978	K & R C	Kernighan & Dennis Ritchie
1989	ANSI C → American National Standard Institute	ANSI Committee
1990	ANSI / ISOC → International Standard organization	ISO Committee
1999	C99	Standard Committee
2011	C11	Standard Committee

Features of C - language :

→ General purpose language
→ Simple
→ Machine independent / portability
→ Middle-level programming language
→ Procedural language
→ Rich library
→ Memory Management
→ Fast & efficient
→ Pointer
→ Case-sensitive
→ Recursion
→ extensible
→ Modularity
→ Popular

Applications of C - language :

i) For developing System software.
ii) For developing Application s/w.
iii) Browsers
iv) Database
v) OS for desktop & Mobile phones
vi) Compilers
vii) computer & Mobile Games.
viii) UNIX O.S
ix) Network devices
x) GUI Applications

Structure of a 'C' program

/* */
// Documentation Section
Link Section (or)
Header file section
Definition, section, declaration
main function definition()
{
main & executable part
}

Functions:

Functions are the main part

Sample Program

```
#include<stdio.h>
void main()
{
printf("Hello");
printf("welcome to C pt");
printf("National computers");
getch();
}
```

Functions:

→ Functions are the main parts of 'C' program
→ Every 'C' program will have one (or) more functions and there will be one mandatory function which is called main() function.

Types → i) Library functions
ii) user-defined-functions

i) Library functions:

It is a pre-defined function
→ Supplies as a part of standard input output library.

i) Standard i/o:

i) printf(): It is used to prints the output in the memory.

ii) Scanf() It is used to read & store the input data. Hence it is declared as '&' as the address of a variable.
so we declare #include <stdio.h>
↓ pre-processor directive ↓ standard i/o

```
#include <stdio.h>
void main()
{
  int x,y,z;
  x=10; y=20;
   z=x+y;
  printf("addition=%d",z);
  getch();
}
```

```
#include <conio.h>
void main()
{
  int x,y,z;
 printf("enter two numbers");
 scanf("%d%d",&x,&y);
 z=x+y;
 printf("addition=%d",z);
 getch();
 }
```

ii) Console i/o functions:

i) getchar() : It is used to read the single character

E.g: char c;
c=getchar();

ii) putchar() : It is used to print the single character

E.g: putchar(c);

iii) getch()
iv) putch()
v) gets()
vi) puts()
vii) clrscr();

```
#include <stdio.h>
#include <conio.h>
void main()
{
  char ch; clrscr();
printf("enter a character");
ch=getchar();
putchar(ch);
getch();
}
```

Keywords:

There are 32 keywords in 'c' - language.
It is also pre-defined.
Each keyword has some fixed meaning.
keywords are always in lower case.

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

Constants:

Constants are the data values (or) fixed values that can never change during the execution of a program.

Types: Integers :- 0-9
Floating Point :- 1.18, 3.14 etc.,
characters :- 'c', 'a', 'b'.
String :- "National computers".

Identifiers

→ refers to identify a value.
→ are the names given to various data items used in a program.

Rules:

1. can contain letters [a-z], digits [0-9] and underscore [_].
2. They can't start with a digits.
3. special Symbol like +, -, $, @, # are not allowed.

Variables Is used to identify the stored value in memory.

Syntax: **E.g.:** a=10; b=20; c=30;

Datatypes The type of data stored in memory.

Syntax: Datatype variables;

Datatypes

- Built-in data types
  - i) integer
  - ii) character
  - iii) float
  - iv) void
- Derived Datatypes
  - Arrays
  - pointers
  - functions
  - Strings
- user-defined datatypes
  - structure
  - unions
  - enum
  - typedef

Built-in Datatypes:

	Keyword	Size	
i) Integer :-	int	2 bytes	%d
ii) character :-	char	1 byte	%c
iii) floating Point :-	float	4 bytes	%f
iv) void :-	void		

- double :- 8 bytes %

i) Integer:

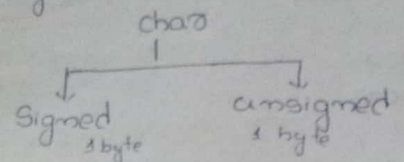
The keyword is int

Integer

- short
  - Signed (+ve/-ve)
  - unsigned (+ve)
  - 2 bytes
- int
  - Signed
  - unsigned
  - 2/4 bytes
- long
  - Signed
  - unsigned
  - 4 bytes
- long long
  - Signed
  - 8 byt

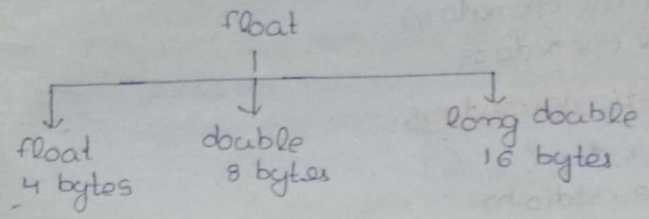
ii) Characters

The keyword is char



iii) floating Point

The keyword is float.




```
#include <stdio.h>
#include <conio.h>
void main()
{ int a,b; clrscr();
printf("enter a number");
scanf("%d", &a);
b= a*a;
printf(" square of a is %d", b);
 getch();
}
```

```
#include<stdio.h>
#include<conio.h>
void main()
{ int m,y,d; clrscr();
  m = 1;
  y = 8;
  d = 3;
  printf("%d %d %d",m,y,d);
} getch();
```

Operators An operator is a symbol that is used to manipulate data and variables. in c.

Types of operators

1) Arithmetic operators
2) Relational operators
3) Logical operators
4) Assignment operators
5) Increement & Decreement operators
6) Bitwise operators
7) Conditional operators
8) Special operators

1) Arithmetic operators :-

+ , - , * , / , %

```
#include <stdio.h>
#include <conio.h>
void main()
{
 int a,b,c,d;
printf("enter three numbers");
Scanf("%d%d%d", &a, &b, &c);
d = a+b+c;
printf ("addition = %d", d);
getch();
}
```

2) Relational operators :-

< , <= , > , >= , == , !=

```
#include <stdio.h>
#include <conio.h>
void main()
{
 int  a,b;
 printf("enter two numbers");
 Scanf("%d%d", &a, &b);
 if(a > b)
 {
  printf(" a is big");
 }.
```

```
else
{
  printf("b is big");
}
  getch();
}
```

3) Assignment operators:-

is used to assign the value of expression to a variable. It is declared with `=` symbol.

e.g: `x = 10;`

`+=` → adds left side variable to the right side variable

`-=` → Substracts left side variable to the right side variable

`*=` → multiply left side variable to the right side variable

`/=` → divides left side variable to the right side variable

`%=` → modulo left side variable to right side variable.

```
#include <stdio.h>
#include <conio.h>
void main()
{
  int x,y;
printf("enter two numbers");
scanf("%d %d", &x, &y);
x += y;
printf("x = %d", x);
 getch();
}
```

4) Increement & Decreement:-

It is used to add one to its operand. Hence it is declared with `++` symbol. There are two types of increement operators in c.

i) pre increement e.g: `++a`

ii) post increement e.g: `a++`

```
#include <stdio.h>
void main()
{
  int a,b;
```

```
printf("enter two num");
scanf("%d%d", &a, &b);
a++;
++b;
printf("a=%d", a++);
printf("b=%d", b);
getch();
}
```

⇒ Decreement operator is used to substract from its operand.

i) pre-decreement : e.g --a

ii) post-decreement e.g. a--

```
#include <stdio.h>
#include <conio.h>
void main()
{
  int x, y;
  printf("enter two num");
  scanf("%d%d", &x, &y);
  x--;
  --y;
  printf("x=%d", x--);
  printf("y=%d", y);
  getch();
}
```

5) Conditional operators : also called as ternary operator

(? :)

Syntax :

cond1 ? stat1 : stat2;

used to express the value of expression to a variable.

```
#include <stdio.h>
#include <conio.h>
void main()
{ int x,y;
  printf("enter two num");
  scanf("%d%d", &x, &y);
```

```
(x>y)? printf("x is big"): printf("y is big");
getch();
}
#include<stdio.h>
#include<conio.h>
void main()
{
 int age;
printf("enter your age");
scanf("%d", &age);
(age>=18)? printf("eligible to vote"): printf("not
                    eligible to vote");
getch();
}
```

6) Logical operators: is used to combine two (or) more expressions logically

i) Logical AND (&&)
ii) Logical OR (||)
iii) Logical NOT (!)

(&&)

cond1	cond2	cond1 && cond2
T	T	T
T	F	F
F	T	F
F	F	F

(||)

cond1	cond2	cond1 || cond2
T	T	T
T	F	T
F	T	T
F	F	F

(!)

T	→ F
F	→ T

```
#include <stdio.h>
#include <conio.h>
void main()
{
 int a, b, c;
printf("enter three num");
scanf("%d %d %d", &a, &b, &c);
```

```
if(a<b && a<c)
{printf(" a is small");
}
if(b<c && b<a)
{printf(" b is small");
}
if(c<a && c<b)
{printf(" c is small");
}
getch();
}
```

```
#include<stdio.h>
void main()
{int x,y;
printf("enter two num");
scanf("%d%d",&x,&y);
if(!(x>y))
{
printf(" x is less than y");
}
else
{
printf(" x is greater than y");
}
getch();
}
```

Bitwise operators

Bitwise operators are used to perform/operate on binary form (or) bits [0, 1]

i) Bitwise AND (&)
ii) Bitwise OR (|)
iii) Bitwise XOR (^)
iv) Bitwise complement (~) Unary operator
v) Left shift (>>)
vi) Right shift (<<)

*) #include <stdio.h>
void main()
{ int a,b,c;
printf("enter two num");
scanf("%d %d", &a, &b);
c = a & b;
printf("c = %d", c);
getch();
}

*) #include <stdio.h>
void main()
{ int a, b, c;
printf("enter two num");
scanf("%d %d", &a, &b);
c = a | b;
printf("c = %d", c);
getch();
}

*) #include <stdio.h>
#include <conio.h>
void main()
{ int a, b, c;
printf("enter two num");
scanf("%d %d", &a, &b);
c = a ^ b;
printf("c = %d", c);
getch();
}

*) #include <stdio.h>
#include <conio.h>
void main()
{ int a, b, c, d;
printf("enter two num");
scanf("%d %d", &a, &b);

```
  c = ~a;
  d = ~b;
 printf("a=%d", c);
 printf("b=%d", d);
 getch();
 }
```

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int x;
 printf("enter a num");
 scanf("%d", &x);
 x>>2;
 printf("x=%d", x);
 getch();
 }
```

```
*) #include<stdio.h>
 #include<conio.h>
 void main()
 {
  int y;
 printf("enter a num");
 scanf("%d", &y);
 y<<2;
 printf("y=%d", y);
 getch();
}
```

Special operators (,) (sizeof) (.)

```
#include<stdio.h>
#include<conio.h>
void main()
{ int a;
  char b;
  float c;
  double d;
 printf("a=%d\n", sizeof(a));
 printf("b=%d\n", sizeof(b));
 printf("c=%d\n", sizeof(c));
 printf("d=%d\n", sizeof(d));
 getch();
}
```

Decision making & Branching Statements

When a program skips the block of code and jumps into another part of code is called branching statements

i) if
ii) if else
iii) else if ladder
iv) Nested if else

i) if : Powerful decision making statements

Syntax:

```
if (cond)
{
  Block of statement
}
```

```
#include <stdio.h>
#include <conio.h>
void main()
{
 int x,y;
 printf("enter two num");
 Scanf("%d%d", &x, &y);
 if (x >= y)
 {
  printf("x is big");
 }
}
```

ii) if else :

Syntax:

```
if (cond)
{
  stat1;
}
else
{
  stat2;
}
```

if cond1 is true statement1 will be executed. if cond1 is false statement2 will be executed

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int a,b;
 printf("enter two num");
 scanf("%d%d", &a, &b);
 if(a>=b)
 {
  printf("a is big");
 }
  else
  {
   printf("b is big");
  }
   getch();
  }
```

*) else if ladder: Multi way branching statements

```
if(cond1)
{
  stat1;
}
else if(cond2)
  {
    stat2;
  }
else if(cond3)
  {
    stat3;
  }
  ——
  ——
  ——
  ——
  ——
else {
      stat n;
    }
```

```c
#include <stdio.h>
#include <conio.h>
void main()
{
 int a,b,c;
printf("enter three num");
scanf("%d%d%d", &a, &b, &c);
if(a>b)
 {
  printf("a is big");
 }
else if(b>c)
 {
  printf("b is big");
 }
 else
 {
  printf("c is big");
 }
 getch();
}
```

```c
#include <stdio.h>
#include <conio.h>
void main()
{
 int num;
printf("enter a num");
scanf("%d", &num);
if(num>=85)
{
 printf("distinction");
}
else if(num>=75)
{
 printf("first class");
}
else if(num>=60)
 {
  printf("Second class");
 }
```

```
else if (num>=45)
  {
   printf("third class");
  }
else
  {
   printf("fail");
  }
  getch();
}
```

*) Nested if else :

Syntax:

```
if (cond1)
{
 if (cond2)
 {
   stat;
 }
}
if (cond3)
{
 if (cond4)
 {
   stat;
 }
}
```

```
#include <stdio.h>
#include <conio.h>
void main()
{ int a,b,c;
printf("enter 3 num");
scanf("%d%d%d", &a, &b, &c);
if (a>b)
{
 if (a>c)
 {
  printf("a is big");
 }
```

```
if(b>c)
{
if(b>a)
{printf("b is big");
}
}
if(c>a)
{
if(c>b)
{printf("c is big");
}
}
getch();
}
```

Decision Making & Looping Statement:-

The process of repeatedly executing a block of statements is called looping statements.

- While loop
- do.... while loop
- for loop.

i) While:-

```
while(cond)
{ stat;
 i/d;
}
```

```
#include<stdio.h>
#include<conio.h>
void main()
{ int i;
 i=1;
 while(i<=5)
 { printf("i=%d\n",i);
  i++;
 }
 getch();
}
```

```
#include<stdio.h>
#include<conio.h>
void main()
{ int i;
 i=5;
 while(i>=0)
 {printf("i=%d\n",i)
 i--;
 } getch();
}
```

*) do... while :-

```
do
{
 stat;
 i/d;
}
while (cond);
```

```
#include <stdio.h>
#include <conio.h>
void main()
 {
   int i;
   i=0;
  do
  {
   printf("i=%d", i);
   i++;
  } while (i<=5);
   getch();
 }
```

*) for loop :-

Entry controlled loop

```
#include<stdio.h>
#include<conio.h>
void main()
 {
   int i;
  for(i=0; i<=10; i++)
  {
   printf("i=%d\n", i)
  }
   getch();
 }
```

*) Nested for loop

```
#include <stdio.h>
#include<conio.h>
void main()
{
  int i, j;
 for(i=0; i<3; i++)
 {
  for(j=0; j<3; j++)
  {
   printf("%d\n", i*j);
  }}
 getch();
}
```

Pattern Programs

```
#include<stdio.h>
#include<conio.h>
void main()
{ int i,j;
 for(i=1; i<=5; i++)
 {
 for(j=1;j<=i;j++)
 {
  printf("* ");
 }
  printf("\n");
 }
 getch();
}
```

O/P →

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

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int i,j;
 for(i=5; i>=1; i--)
 {
 for(j=1; j<=i; j++)
 {
   printf("* ");
 }
```

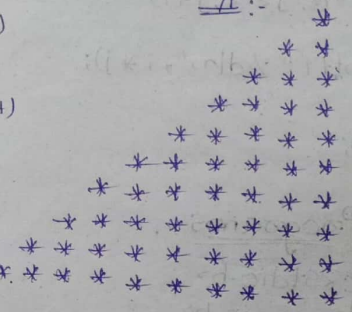
O/P →

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

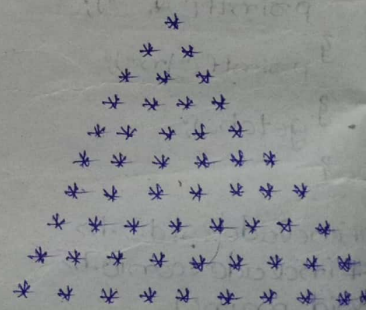
```
    printf("\n");
  }getch();
 }
```

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int n=10, i, j;
 for(i=1; i<=n; i++)
 {
 for(j=1; j<n-i; j++)
 {
   printf(" ");
 }
 for(j=1; j<=i; j++)
 {
   printf(" * ");
 }
   printf("\n");
 }
   getch();
 }
```

o/p :-



```
#include<stdio.h>
#include<conio.h>
void main()
{
  int n=10, i, j;
 for(i=1; i<=n; i++)
 {
 for(j=1; j<n-i; j++)
 {
   printf(" ");
 }
  for(j=1; j<=i; j++)
 {
   printf(" * ");
 }
   printf("\n");
 }getch();
 }
```



```
#include<stdio.h>
#include<conio.h>
void main()
{ int n=5, num=1, i, j;
for(i=0; i<n; i++)
{
for(j=0; j<=i; j++)
{
printf("%d", num);
num = num+1;
}
printf("\n");
}
getch();
}
```

O/P :-

```
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
```

To print Tables

```
#include<stdio.h>
#include<conio.h>
void main()
{ int i, n;
printf("enter a num");
scanf("%d", &n);
for(i=1; i<=10; i++)
{
printf("%d x %d = %d", n, i, n*i);
}
getch();
}
```

Switch Statement:

Multi-way branching statements

```
#include <stdio.h>
#include <conio.h>
void main()
{
  int i;
 printf("enter a num");
  scanf("%d", &i);
  switch(i)
  {
   case 1: printf("you entered one");
           break;
   case 2: printf("you entered two");
           break;
   case 3: printf("you entered three");
           break;
   case 4: printf("you entered four");
           break;
  default: printf("wrong choice");
  getch();
}
```

Jump Statement:

i) Break
ii) continue
iii) goto.

i) Break:

```
#include <stdio.h>
#include <conio.h>
void main()
{   int i;
for(i=0; i<=10; i++)
{
printf("i=%d\n", i);
 if(i==5)
  {  break;
  }
} getch();
}
```

ii) continue :-

```
#include <stdio.h>
#include <conio.h>
void main()
{
  int i;
  for(i=0; i<=10; i++)
  {
   printf("i = %d\n", i);
   if (i==5)                    (i==5 || i==2 || i==10)
   {
     continue;
   }
  } getch();
}
```

iii) goto :-

```
#include <stdio.h>
#include <conio.h>
void main()
{ int age;
 ineligible:
  printf("you are not eligible to vote");
  printf("enter your age");
  scanf("%d", &age);
  if(age >= 18)
  { printf("eligible to vote");
  }
   else
   {
    goto ineligible;
  }
   getch();
  }
}
```

Arrays:- An Array is a collection of similar data items. We can store 100 of values of same type with a single array name.
All must be of same type.

Declaration:

Syntax:- datatype arrayname[size];

E.g:- int arr[10];

arr → array name

size → size of the array.

Initialization:-

```
int arr[3];
int arr[3] = {10,20,30};
arr[0] = 10;
arr[1] = 20;
arr[2] = 30;
```

index must be zero

Types:- *) for loop is used to print array values

i) Single/one dimensional Array

ii) Multi/Two/double dimensional Array

i) Single/One Dimensional Array:-

```
int arr[3];
```

```
#include <stdio.h>
#include <conio.h>
void main()
{
  int a[3] = {10,20,30};
  int i;
  printf("elements of array are");
  for(i=0; i<3; i++)
  {
    printf("%d\n", a[i]);
  }
  getch();
}
```

```
#include <stdio.h>
 void main()
 { int a[5], i;
 printf("enter 5 elements");
  for(i=0; i<5; i++)
   {
    scanf("%d", &a[i]);
   }
 printf("elements of array are:");
  for(i=0; i<5; i++)
   { printf("%d\n", a[i]);
   }getch();
  }
```

Two/Multi/Double Dimensional Array:-

Syntax:-

datatype arrayname[rowsize][column size];

E.g:- int arr[3][3];

3*3 = 9 elements

```
arr[0][0] = 10
arr[0][1] = 20
arr[0][2] = 30
arr[1][0] = 40
arr[1][1] = 50
arr[1][2] = 60
arr[2][0] = 70
arr[2][1] = 80
arr[2][2] = 90
#include<stdio.h
#include<conio.h>
void main()
{
 int a[3][3]={{10,20,30},{40,50,60},{70,80,90}};
 int i,j;
 printf("elements of array are");
 for(i=0;i<3;i++)
 {
  for(j=0;j<3;j++)
```

```
    {
     printf("%d\t", a[i][j]);
    }
    printf("\n");
   }
   getch();
  }
#include<stdio.h>
void main()
{
 int a[3][3], i, j;
 printf("enter elements of array");
 for(i=0; i<3; i++)
 {
  for(j=0; j<3; j++)
  {
   scanf("%d", &a[i][j]);
  }
 }
 printf("elements of array are");
 for(i=0; i<3; i++)
 {
  for(j=0; j<3; j++)
  {
   printf("%d\t", a[i][j]);
  }
  printf("\n");
 }
 getch();
}
```

Strings:-

Group of characters is called a String. There are some pre-defined functions designed by handling strings which are available in the library string.h.

Syntax:-

```
char a[10] = "hello";
```

We can declare string in two ways

1) using single character constant

```
char a[10] = {'h','e','l','l','o','\0'};
```

↑ end of the string

2) using String constant

```
char b[10] = "hello".
```

```
#include <stdio.h>
#include <string.h>
void main()
{ char s[5] = "hello";
  char s1[10] = {'h','e','l','l','o','\0'};
  printf("you entered string is %s", s);
  printf("second string = %s", s1);
  getch();
}
```

```
#include <stdio.h>
#include <string.h>
void main()
{ char s1[5];
printf("enter a string");
scanf("%s", s1);
printf("you entered string = %s", s1);
getch();
}
```

String functions:

i) strlen() : It is used to find the length of the string.

```
#include <stdio.h>
#include <string.h>
void main()
{
 char s1[10]; int l;
printf("enter a string");
scanf("%s", s1);
l= strlen(s1);
printf("l= %d", l);
getch();
}
```

ii) strcpy() : This function is used for copying source string into destination string.

Syntax: strcpy(destination string, source string)

```
#include <stdio.h>
#include <string.h>
void main()
{
 char a[50], b[50];
printf("enter a source string");
scanf("%s", a);
 strcpy(b,a);
printf("copied string is %s", b);
getch();
}
```

iii) strcat() : It is used to combine both the strings. The resultant string will be the source string.

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

```
    char a[15];
    char b[15];
  printf("enter two strings");
  scanf("%s%s",a,b);
   strcat(a,b);
  printf("string is %s", a);
   getch();
}
```

iv) strcmp().

This function is used to compare both the string

```
#include <stdio.h>
#include <string.h>
void main()
{
   char a[10];
    char b[15];
    int d;
  printf("enter two strings");
  scanf("%d%d", a, b);
  d = strcmp(a,b);
  if(d>0)
  {
   printf("%s is greather than %s", a, b);
  }
else if(d<0)
{
  printf("%s is less than %s", a, b);
}
else
{
  printf("%s is equal to %s", a, b);
}getch();
}
```

v) strupr(), strlwr():

```
#include<stdio.h>
#include<string.h>
void main()
{
  char a[20], b[20];
printf("enter two strings");
scanf("%s%s",a,b);
 strupr(a);
 strlwr(b);
printf("a=%s  b=%s", a, b);
 getch();
}
```

Pointers: A pointer is a variable which is used to store the address of a variable

*p → Pointer variable.

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int *p;
  int num=10;
  p = &num;
printf("num=%d\n", num);
printf("p=%d\n", *p);
 printf("p=%u", p);
 } getch();
```

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int *p, m;
printf("enter a num");
 scanf("%d", &m);
  p = &m;
printf("m=%d\n", m);
printf("p=%d\n", *p);
 printf("p=%u", p);
```

```
void main getch();
}
```

Pointers to pointers:

is also a variable which is used to store the address of another pointer variable. Hence it is declared as **

```
**p;
#include <stdio.h>
#include <conio.h>
void main()
{
  int *p1, **p2, m;
 printf("enter a number");
 scanf("%d", &m);
  p1 = &m;
  p2 = &p1;
 printf("%d\m", m);
 printf("p1 = %d\m", *p1);
 printf("p1 = %u\m", p1);
 printf("p2 = %d\m", *p2);
 printf("p2 = %x", p2);
 getch();
}
```

Mathematical functions:

There are some mathematical functions in C language those are pre-defined functions. To use these functions we have to use (or) include #include <math.h>

```
#include <stdio.h>
#include <math.h>
void main()
{
  float x,y;
 printf("enter two numbers");
 scanf("%d%d", &x, &y);
  ceil printf("%f\m", ceil(x));
       printf("%f", ceil(y));
```

```
    getch();
}
*)#include <stdio.h>
 #include <math.h>
 void main()
{ int float a,b;
 Printf("enter a num");
 Scanf("%f%f", &a, &b);
 Printf("%d\n", floor(a));
 Printf("%d", floor(b));
 getch();
}

*)#include <stdio.h>
 #include <math.h>
void main()
{ int a;
 printf("enter a num");
 Scanf("%d", &a);
 print("a = %d", sqrt(a));
 getch();
}

*)#include <stdio.h>
 #include <math.h>
void main()
{ int x,y;
 printf("enter two num");
 Scanf("%d%d", &x, &y);
 printf("pow = %d", pow(x,y));
 getch();
}

*)#include<stdio.h>
 #include <math.h>
 void main()
 { int x;
   printf("enter a num");
```

```
    scanf("%d", &x);
    printf("abs = %d", abs(x));
    getch();
}
```

```
#include<stdio.h>
#include<math.h>
void main()
{
    float a = 10;
    printf("sin(a) = %f\n", sin(a));
    printf("cos(a) = %f\n", cos(a));
    printf("tan(a) = %f\n", tan(a));
    printf("exp(a) = %f\n", exp(a));
    printf("log(a) = %f\n", log(a));
    printf("log(10) = %f\n", log10(a));
    printf("sqrt(a) = %f\n", sqrt(a));
    printf("cbrt(a) = %f\n", cbrt(a));
    getch();
}
```