C-LANGUAGE

BY:- ABHISHEK MAURYA

Introduction.

- C is a general-purpose high level language .
- originally developed by Dennis Ritchie for the Unix operating system.
- It was first implemented on the Digital Equipment Corporation PDP-11 computer in 1972.
- From beginning C was intended to be useful for busy powerful, dominant
- and supple language.

Why name ‘c’ was given to this language?

- Many of the ideas of c language were derived and taken from ‘b’ language
- As many featured came from B it was named as ‘c’.

About “c”

C programming language – structured and disciplined approach to program design.

- C is structured programming language.
- C supports functions that enables easy maintainability of code, by breaking large file into smaller modules.
- Comments in C provides easy readability.
- C is a powerful language.
- C programs built from
    - Variable and type declarations.
    - Function.
    - Statement.
    - Expression.

Header files

- The files that are specified in the include section is called as Header file.
- These are precompiled files that has some function defined in them.
- We can call those function in our program by suppling parameters.
- Header files is given an extension .h.
- C source file is given an extension .c.

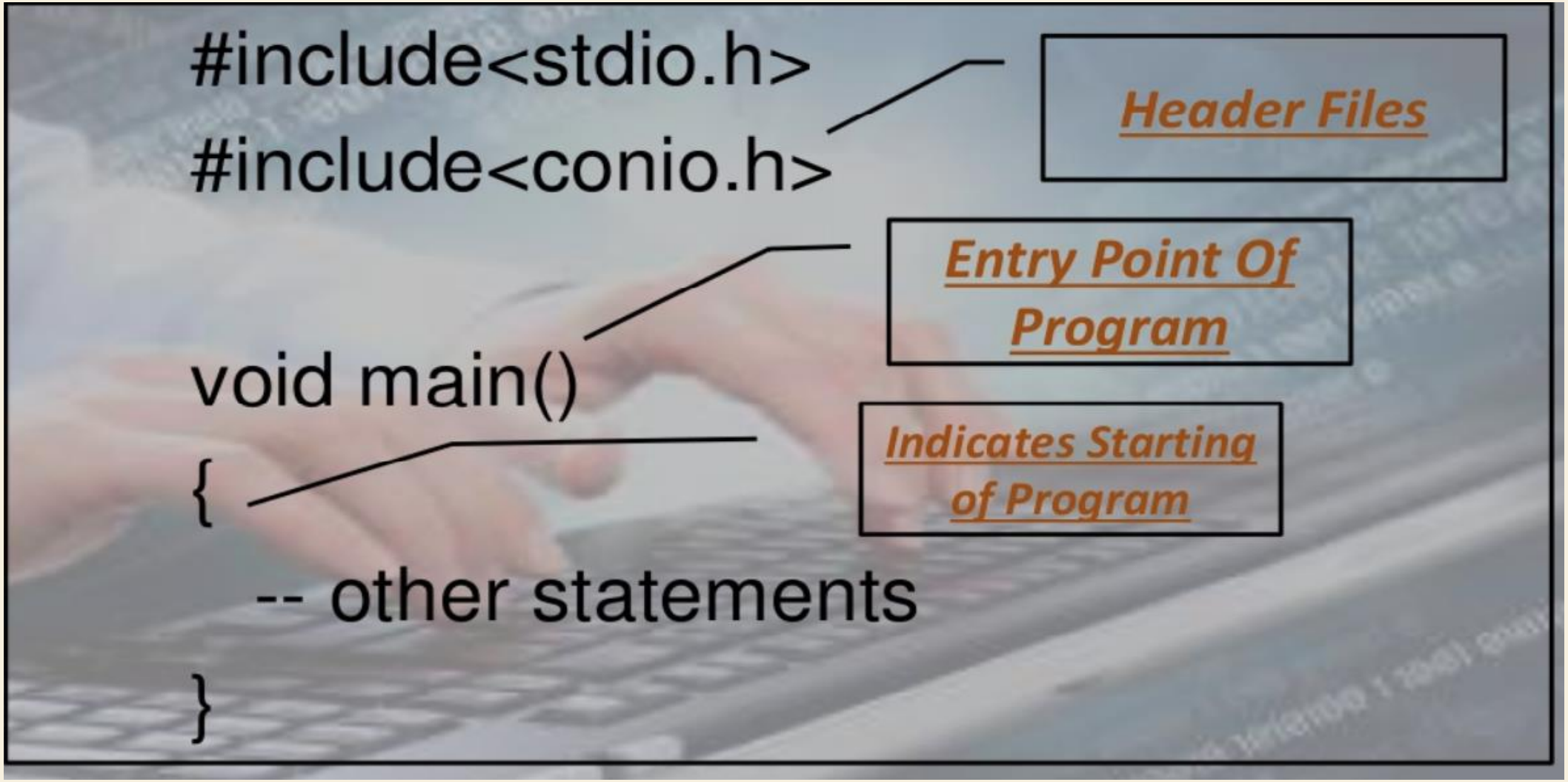
MAIN Function

- This is the "Entry point" of a program.
- When a file is executed, the start point is the main function.
- From main function the flow goes as per the programmers choice.
- There may or may not be other function written by user in a program.
- Main function is compulsory for any C program.

Running a ‘C’ Program

- Type a program.
- Save it.
- Compile the program – This will generate an .exe file (executable)
- Run the program (Actually the exe created out of compilation will run and not the .c file)
- In different compiler we have different option for compiling and running.

Basic Stucture of "C" program



C Hello World! Example:Your First Program

```
#include <stdio.h>
Void main()
{
printf("Hello, World!");
}
```

Output

```
Hello, World!
```

What Are Variables in C?

- A Variable is a data name that is used to store any data value.
- Variables are used to store values that can be changed during the program execution.
- Variables in C have the same meaning as variables in algebra. That is, they represent some unknown, or variable, value.
- $x = a + b\ z + 2 = 3(y - 5).$
- Remember that variables in algebra are represented by a single alphabetic character.

Declaring Variables:-

- Before using a variable, you must give the compiler some information about the variable; i.e., you must declare it.
- The declaration statement includes the data type of the variable.
  Examples of variable declarations:
  int length ;
  float area ;
- Variables are not automatically initialized. For example, after declaration int sum; the value of the variable sum can be anything (garbage).
- Thus, it is good practice to initialize variables when they are declared.
- Once a value has been placed in a variable it stays there until the program alters it.

Data types :-

There are three classes of data types here:

- **Primitive data types** –
  int, float, double, char
- **Aggregate OR derived data types –**
  Arrays come under this category –
  Arrays can contain collection of int or float or char or double data
- **User defined data types –**
  Structures and Enum fall under this category.

Data Types- different attributes:-

Type	Size	Representation	Minimum range	Maximum range
char, signed char	8 bits	ASCII	-128	127
unsigned char	bool 8 bits	ASCII	0	255
Short, signed char	16 bits	2's complement	-32768	32767
Insigned short	16 bits	Binary	0	65535
Int, signed int	16 bits	2's complement	-32768	32767
Unsigned int	16 bits	Binary	0	65535
Long, signed long	32 bits	2's complement	-2,147,483,648	65535
Unsigned long	32 bits	Binary	0	4,294,967,295
Float	32 bits	IEEE 32 bit	1.175495e-38	3.4028235e+38
Double	32 bits	IEEE 32 bit	1.175495e-38	3.4028235e+38
Long double	32 bits	IEEE 32 bit	1.175495e-38	3.4028235e+38

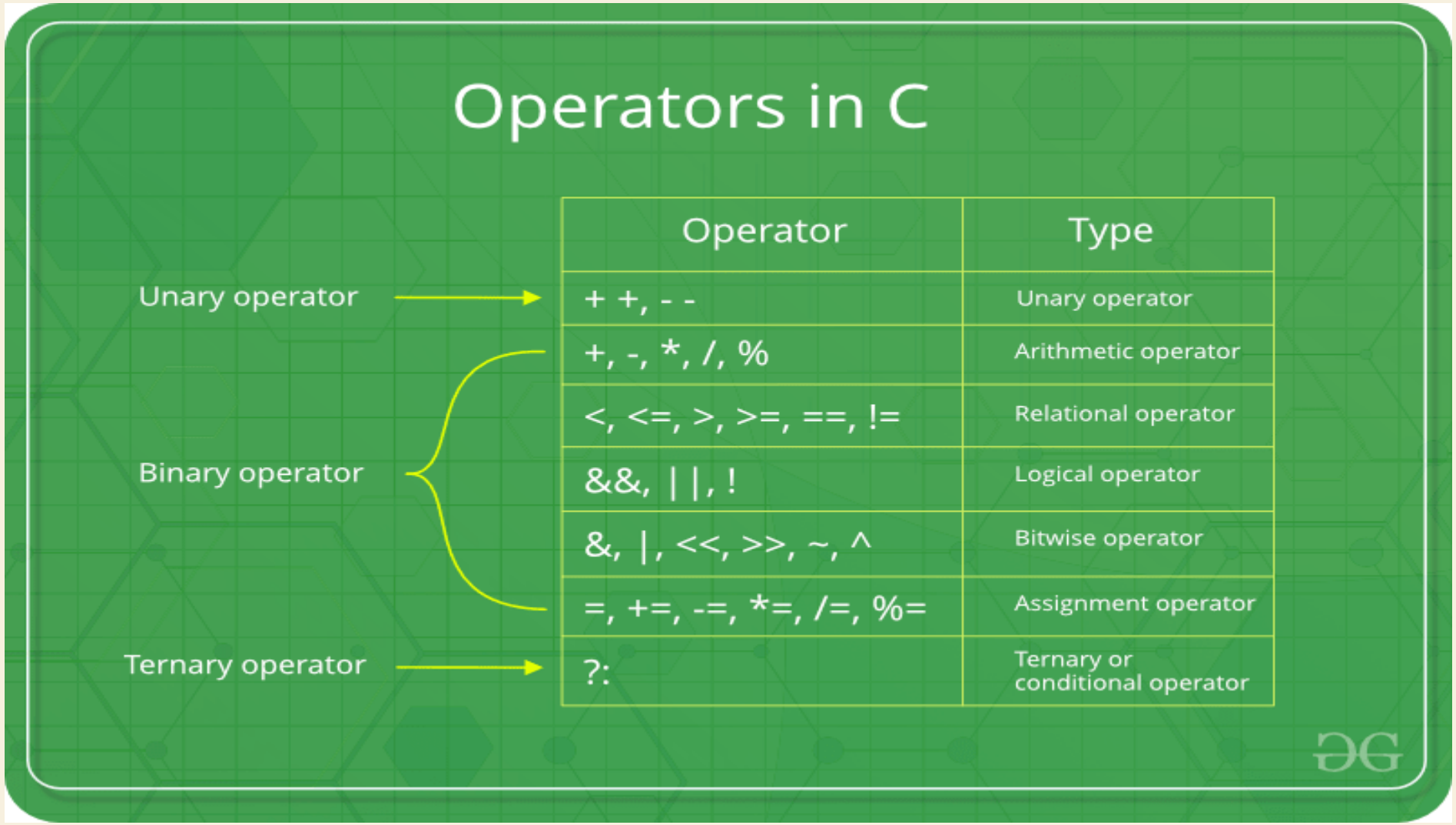
Printf() function

- In C programming language, printf() function is used to print the "character, string, float, integer, octal and hexadecimal values" onto the output screen.
- We use printf() function with %d format specifier to display the value of an integer variable.
- Similarly %c is used to display character, %f for float variable, %s for string variable, %lf for double and %x for hexadecimal variable.
- To generate a newline,we use "\n" in C printf() statement

scanf() function

In C programming language, scanf() function is used to read character, string, numeric data from keyboard

OPERATORS:


Operators in C
Operator
Type
Unary operator
+ +, - -
Unary operator
Binary operator
+, -, *, /, %
Arithmetic operator
<, <=, >, >=, ==, !=
Relational operator
&&, ||, !
Logical operator
&, |, <<, >>, ~, ^
Bitwise operator
=, +=, -=, *=, /=, %=
Assignment operator
Ternary operator
?:
Ternary or conditional operator

Type of Operator	Symbolic representation
Arithmetic operators	+, -, *, /, %
Relational operators	>, <, ==, >=, <=, !=
Logical operators	&&, ||, !=
Increment and decrement operator	++ and --
Assignment operator	=
Bitwise operator	&, |, ^, >>, <<, ~
Comma operator	,
Conditional operator	?:

Arithmetic operator

ARITHMETIC OPERATIOR:-

Basic **arithmetic operations** include addition(+), subtraction(-), multiplication(*) and division(/). They are performed on numerical data types like int , float and double .

EXAMPILE OF OPERATORS:-

ARITHMETIC OPERATIOR:-

```
#include<conio.h>
#include<stdio.h>
Void main()
{
Int a=45,b=21,c;
c=a+b;
 Printf("sum=%d",c);
}
```

Header files

Main function

Initialization variables and declared their values

Output: sum=66

RELATIONAL OPERATORS:-

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. **Relational operators** are used in decision making and loops.

Symbols

<	**less than**
<=	**less than equals to**
>	**Greater than**
>=	**Greater than equals to**

RELATIONAL OPERATOR OPERATIOR:-

```
#include<conio.h>
#include<stdio.h>
Void main()
{
Int a=45,b=21;
If(a>b)
    {
    printf("A is greater ");
    }
    else(
    printf("B is greater");
    )
}
```

Output: A is greater.

Logical operator:-

```
#include <stdio.h>
Void main(){
int a = 5, b = 5, c = 10;
printf("%d == %d is %d \n", a, b, a == b);
                }
```

Output:-

```
5 == 5 is 1
```

INCREMENT AND DECREMENT OPERATOR

```
#include <stdio.h>
                int main()
               {
                int x = 10,y = 20;
               printf("----INCREMENT OPERATOR EXAMPLE---- \n");
               printf("Value of x : %d \n", x); //Original Value printf("Value of x
               %d \n", x++); // Using increment Operator printf("Value of x :
               %d \n", x); //Incremented value
```

ASSIGNMENT OPERATOR:-

```
# include<stdio.h>
int main()
{
int Total=0,i;
for(i=0;i<10;i++)
{
Total+=i; // This is same as Total = Toatal+i
}
printf("Total = %d", Total);
}
```

Conditional Statement:

If family:--

There are 4 types:-

1) If statement.(**when you want only true statement**.)
2) If else statement.(**when you want both statement true or false**.)
3) Else if statement.(**when you have multiple possibilities**.)
4) Nested if statement.(**when you have nested conditions**.)

Conditional Statement

1. If statement :-

It is one of the powerful conditional statement. If statement is responsible for modifying the flow of execution of a program. If statement is always used with a condition.

Syntax :-

```
if (condition) {
instruction; }
```

Example :-

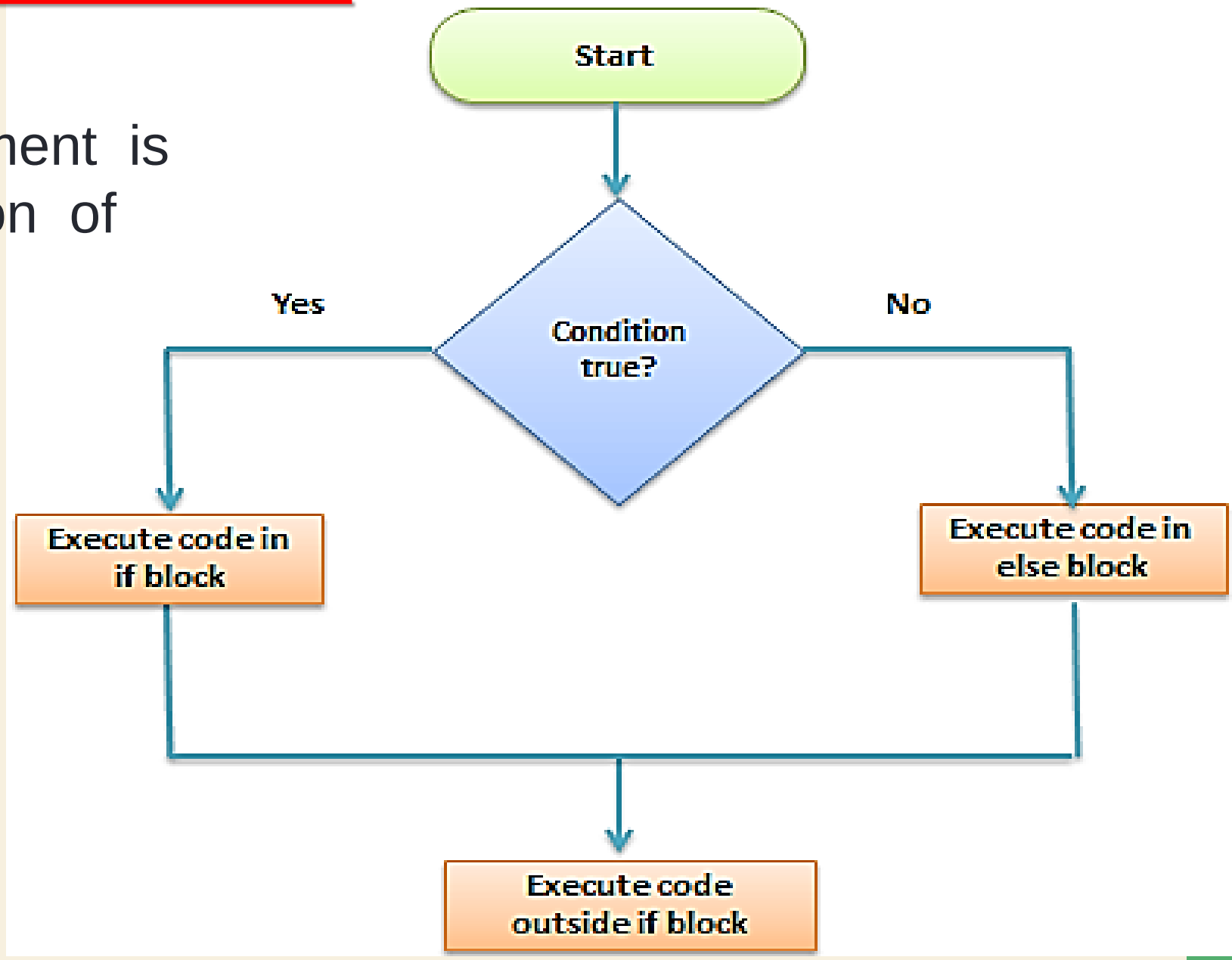
```
#include<stdio.h>
 void main()
 {
 int num1=1;
 int num2=2;
 if(num1<num2)//test-condition
 {
 printf("num1 is smaller than num2");
}
 }
```

Output :-

num1 is smaller than num2

The If-Else statement :-

The if-else is statement is an extended version of If.



Syntax:-

```
if (test-expression)
{
 True block of statements
 }
 Else
 {
 False block of statements
 }
 Statements;
```

Example :-

```
#include<stdio.h>
 void main()
 {
 int num=19;
 if(num<10)
 { printf("The value is less than 10");
}
Else
 {
 printf("The value is greater than 10"); }
}
```

Output :-

The value is greater than 10

Nested If-else Statements :-

```
#include<stdio.h>
 void main()
 {
 int num=1;
 if(num<10)
{
if(num==1)
{ printf("The value is:%d\n",num); }
 else
 { printf("The value is greater than 1"); }
}
else
 { printf("The value is greater than 10"); }
 }
```

Output :-

The value is:1

Switch Statement :-

The switch statement in C is an alternate to if-else-if ladder statement which allows us to execute multiple operations for the different possibles values of a single variable called switch variable. Here, We can define various statements in the multiple cases for the different values of a single variable.

Syntax :-

```
switch(expression){
case value1:
 //code to be executed;
 break;  //optional
case value2:
 //code to be executed;
 break;  //optional
.....

default:
 code to be executed if all cases are not matched;
}
```

Example :-

```
#include<stdio.h>
int main(){
int number=0;
printf("enter a number:");
scanf("%d",&number);
switch(number){
case 10:
printf("number is equals to 10");
break;
case 50:
printf("number is equal to 50");
break;
default:
printf("number is not equal to 10, 50 or 100");    }   }
```

Output

enter a number:
4 number is not equal to 10, 50 or 100
enter a number:
50 number is equal to 50

GOTO STATEMENT:-

- **goto** is a jumping statement in **c language**, which transfer the **program's** control from one statement to another statement (where label is defined).
- **goto** can transfer the **program's** within the same block and there must a label, where you want to transfer **program's** control.

Syntax1

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

Syntax2

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

Loops :-

The looping can be defined as repeating the same process multiple times until a specific condition satisfies. There are three types of loops used in the C language. In this part of the tutorial, we are going to learn all the aspects of C loops.

There are two type of Loop :-

1. Entry Control Loop: -
2. Exit Control Loop:-

Entry Control Loop:

In entry control loop there are two types of loop:-

i. For Loop:-

Syntax:-

```
for ( initlization; condition; increment/decrement )
{
 statement(s);
 }
```

Ex:-

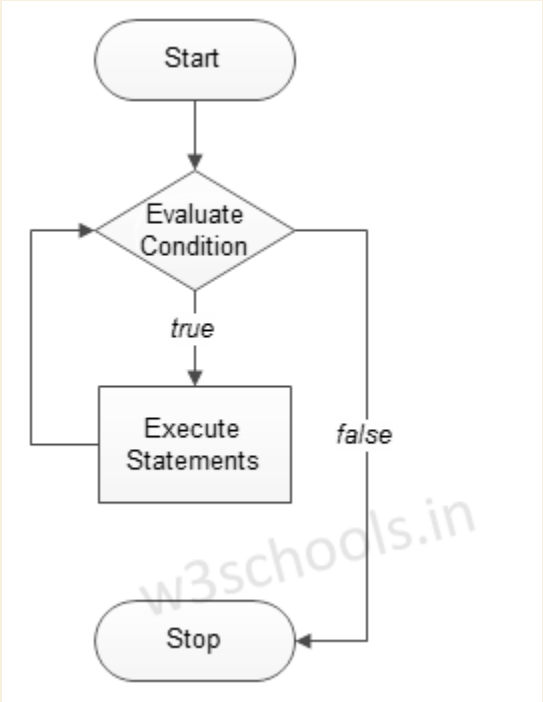
```
#include<stdio.h>
int main ()
{
int a; /* for loop execution */
for( a = 10; a < 20; a = a + 1 )
{ printf("value of a: %d\n", a);
}}
```

```
Output:-
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

ii. While Loop:-

Syntax:-

```
While (condition)
 { statement(s);
Incrementation; }
```



Ex:-

```
#include<stdio.h>
Void main ()
{int n = 1,times=5;
 while( n <= times )
 { printf("C while loops: %d\n", n);
 n++;
 }
 }
```

Output:-

```
C while loops:1
C while loops:2
C while loops:3
C while loops:4
C while loops:5
```

Exit control loop:-

In exit control loop there are only one loop that is:-

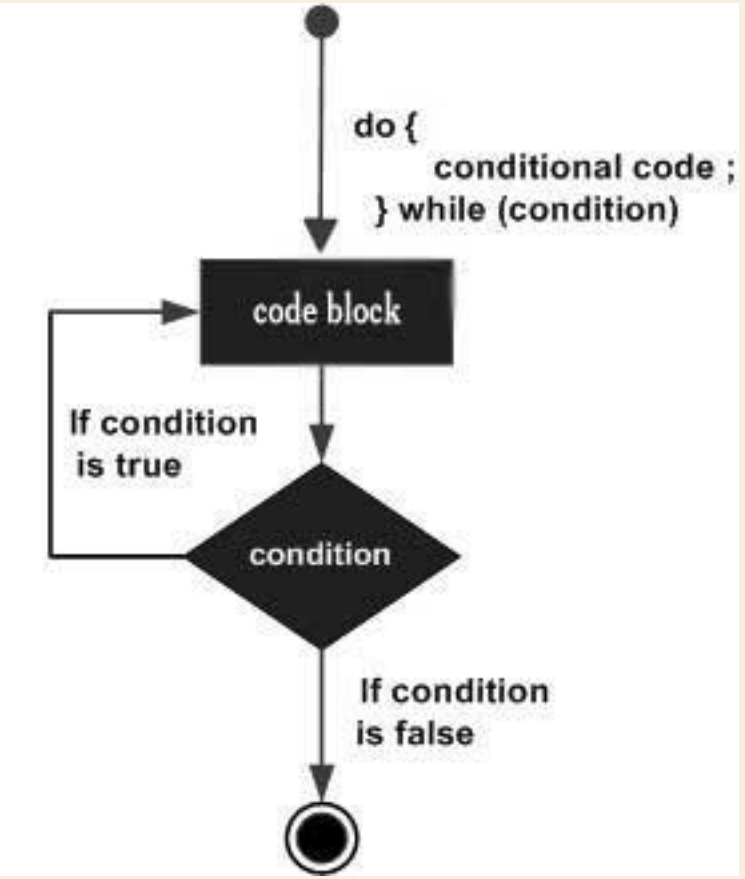
Do while loops:-

Syntax:-

```
do{statement(s);}
while(condition);
```

Ex:-

```
#include <stdio.h>
int main ()
{
a=10;
Do{printf("value of a: %d\n",a);
a = a + 1; }
while( a < 20 );
}
```



```c
#include <stdio.h>
void main()
{
  int row, c, n;
  printf("Enter the number of rows in pyramid of stars to print\n");
  scanf("%d", &n);
  for (row = 1; row <= n; row++)  // Loop to print rows
  {
    for (c = 1; c <= n-row; c++)  // Loop to print spaces in a row
      printf(" ");
    for (c = 1; c <= 2*row - 1; c++) // Loop to print stars in a row
      printf("*");
    printf("\n");
  }
  getch();
}
```

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

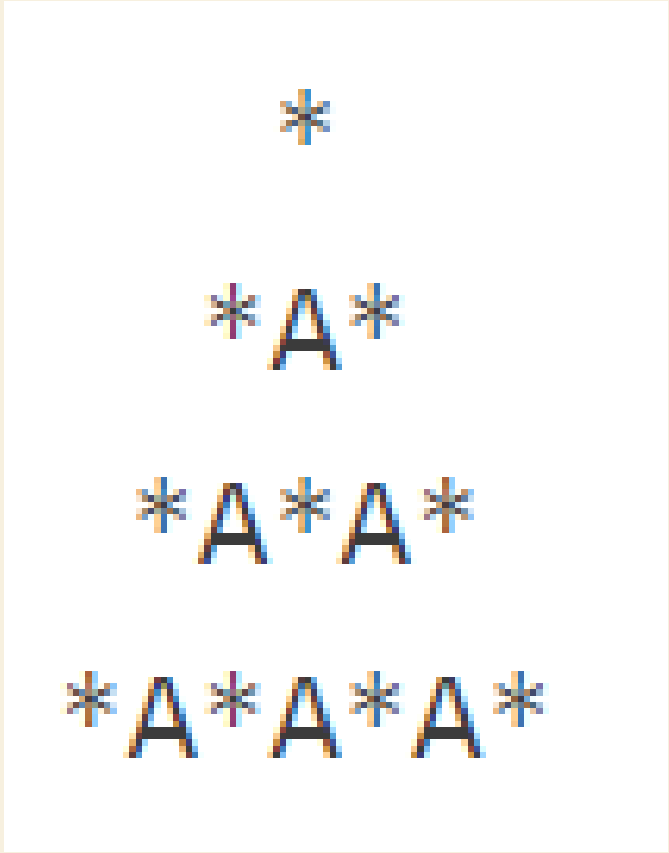
```
#include<conio.h>
#include<stdio.h>
Void main()
{
Int I,j;
Char p;
Printf(“enter symbol for patter=”);
Scanf(“%c”,&p);

For(i=0;i<=5;i++)
        {
        for(j=0;j<=I;j++)
                {
                printf(“%c”,p);
                }
        printf(“\n”);
        }
Getch();
}
```

```c
#include <stdio.h>
void main()
{
  int n, c, row, t = 1;
  scanf("%d", &n);
  for (row = 1; row <= n; row++) {
    for (c = 1; c <= n - row; c++)
      printf(" ");
    t = row;
    for (c = 1; c <= row; c++) {
      printf("%d", t);
      t++;
    }
    t = t - 2;
    for (c = 1 ; c < row; c++) {
      printf("%d", t);
      t--;
    }
    printf("\n");
  }
  getch();
}
```

```
    1
   232
  34543
 4567654
567898765
```

```c
#include <stdio.h>
void main()
{
  int n, c, k;
  printf("Enter number of rows\n");
  scanf("%d", &n);
  for (c = 1; c <= n; c++)
  {
    for (k = 1; k <= n-c; k++)
      printf(" ");
    for (k = 1; k < c; k++)
      printf("*A");
    printf("*\n");
  }
  getch();
}
```



C Array :-

An array is defined as the collection of similar type of data items stored at contiguous memory locations.

Arrays are the derived data type in C programming language which can store the primitive type of data such as int, char, double, float, etc.

It also has the capability to store the collection of derived data types, such as pointers, structure, etc.

Types of C Array :-

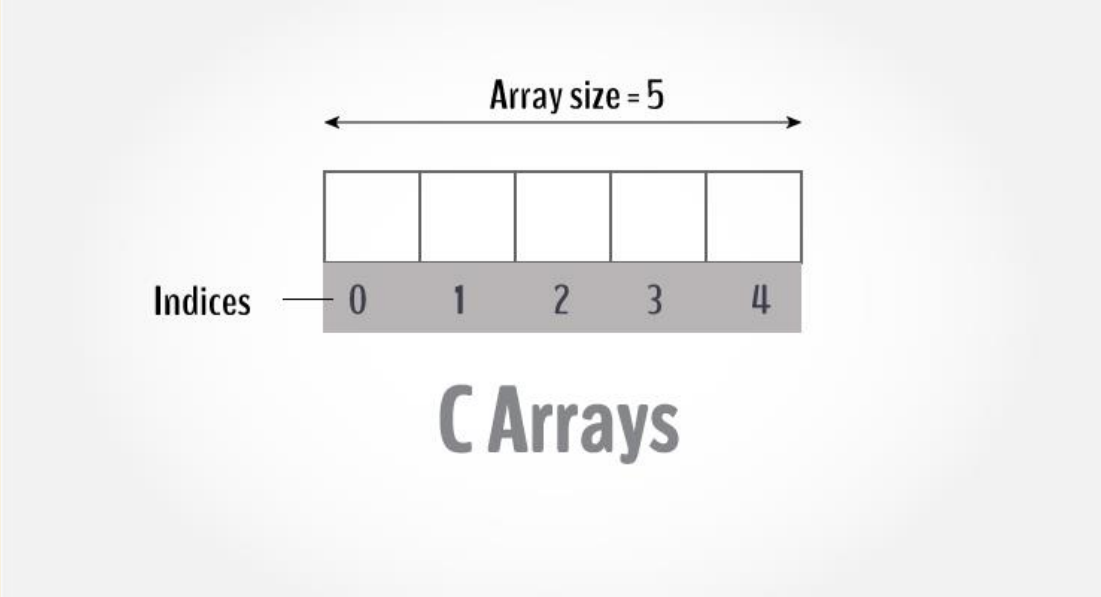
1. One Dimensional Array.
2. Two Dimensional Array.
3. Multidimensional Array.

One D- Array:-

```
#include<stdio.h>
int main(){
int i=0;
int marks[5];//declaration of array
marks[0]=80;//initialization of array
marks[1]=60;
marks[2]=70;
marks[3]=85;
marks[4]=75; //traversal of array
for(i=0;i<5;i++){
printf("%d \n",marks[i]);
}//end of for loop
}
```

Output :-

```
80
60
70
85
75
```



2-D Array:-

```
#include<stdio.h>
void main(){
int disp[2][3];
int i, j;
for(i=0; i<2; i++)
 { for(j=0;j<3;j++)
{ printf("Enter value for disp[%d][%d]:", i, j);
scanf("%d", &disp[i][j]); } }
printf("Two Dimensional array elements:\n"); for(i=0;
i<2; i++)
{ for(j=0;j<3;j++)
{ printf("%d ", disp[i][j]);
if(j==2)
{ printf("\n");
}
}

}
```

Multidemensional array

```
#include <stdio.h>
int main()
{
int test[2][3][2];
printf("Enter 12 values: \n");
for (int i = 0; i < 2; ++i)
{
for (int j = 0; j < 3; ++j)
{
for (int k = 0; k < 2; ++k)
{
scanf("%d", &test[i][j][k]);
}
```

```
// Printing values with proper index.
printf("\nDisplaying values:\n");
for (int i = 0; i < 2; ++i)
{
for (int j = 0; j < 3; ++j)
{
for (int k = 0; k < 2; ++k)
{
printf("test[%d][%d][%d] = %d\n", i, j, k,
test[i][j][k]);
}
}
}
return 0;
}
```

```
}
}
```

STRING:

- A string is nothing but a collection of characters in a linear sequence. 'C' always treats a string a single data even though it contains whitespaces. A single character is defined using single quote representation. A string is represented using double quote marks.

FEW FUNCTIONS OF STRING:

1.STRLEN: string length
2.STRCPY: string copy
3.STRNCPY:string n copy
4.STRCAT: string concatnate
5.STRNCAT: string n concatenate
6.STRCMP: string compare

Functions:-

A function is a set of statements that take inputs, do some specific computation and produces output.
The idea is to put some commonly or repeatedly done task together and make a function so that instead of writing the same code again and again for different inputs, we can call the function.

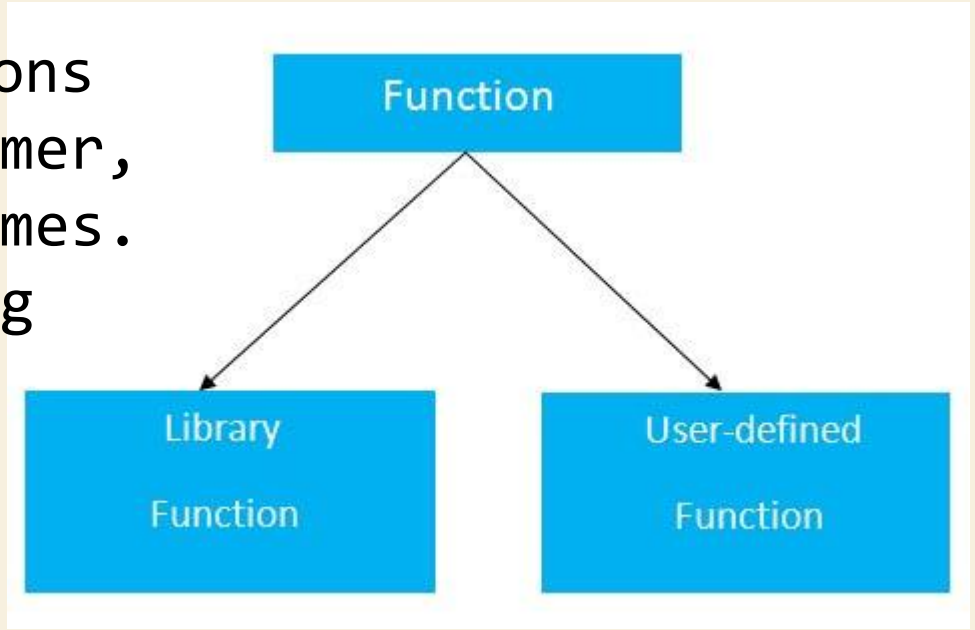
Reusability of codes

Types of Functions

There are two types of functions in C programming:

Library Functions: are the functions which are declared in the C header files such as scanf(), printf(), gets(), puts(), ceil(), floor() etc.

User-defined functions: are the functions which are created by the C programmer, so that he/she can use it many times. It reduces the complexity of a big program and optimizes the code.



TYPE OF USER DEFINE FUNTION

1. function without arguments and without return value.
2. function without arguments and with return value.
3. function with arguments and without return value.
4. function with arguments and with return value.

Example for Function without argument and return value:-

```
#include<stdio.h>
void sum();
void main()
{
    printf("\nGoing to calculate the sum of two numbers:");
    sum();
}
void sum()
{
    int a,b;
    printf("\nEnter two numbers");
    scanf("%d %d",&a,&b);
    printf("The sum is %d",a+b);
}
```

Example for Function without argument and with return value:-

```
#include<stdio.h>
int sum();
void main()
{
    int result;
    printf("\nGoing to calculate the sum of two numbers:");
    result = sum();
    printf("%d",result);
}
int sum()
{
    int a,b;
    printf("\nEnter two numbers");
    scanf("%d %d",&a,&b);
    return a+b;
}
```

Example for Function with argument and without return value:-

```
#include<stdio.h>
void sum(int, int);
void main()
{
    int a,b,result;
    printf("\nGoing to calculate the sum of two numbers:");
    printf("\nEnter two numbers:");
    scanf("%d %d",&a,&b);
    sum(a,b);
}
void sum(int a, int b)
{
    printf("\nThe sum is %d",a+b);
}
```

Example for Function with argument and with return value:-

```
#include<stdio.h>
int sum(int, int);
void main()
{
    int a,b,result;
    printf("\nGoing to calculate the sum of two numbers:");
    printf("\nEnter two numbers:");
    scanf("%d %d",&a,&b);
    result = sum(a,b);
    printf("\nThe sum is : %d",result);
}
int sum(int a, int b)
{
    return a+b;
}
```

Structure

A structure is a user-defined data type available in C that allows to combining data items of different kinds. Structures are used to represent a record.

```
struct [structure name]
{ member definition;
 member definition;
 ... member definition; };
```

```
#include <stdio.h>
struct Distance
{
int feet;
float inch;
} dist1, dist2, sum;
int main()
{
printf("1st distance\n");
printf("Enter feet: ");
scanf("%d", &dist1.feet);
printf("Enter inch: ");
scanf("%f", &dist1.inch);
printf("2nd distance\n");
printf("Enter feet: ");
scanf("%d", &dist2.feet);
printf("Enter inch: ");
scanf("%f", &dist2.inch);
sum.feet = dist1.feet +
dist2.feet;
sum.inch = dist1.inch +
dist2.inch;
while (sum.inch >= 12)
{
++sum.feet;
sum.inch = sum.inch - 12;

}
printf("Sum of distances =
%d\'-%.1f\"", sum.feet,
sum.inch);
return 0;
}
```

union

A union is a special data type available in C that allows storing different data types in the same memory location. You can define a union with many members, but only one member can contain a value at any given time. Unions provide an efficient way of using the same memory location for multiple purposes.

```
union [union name]
 {
      member definition;
      member definition;
      ...
      member definition;
};
```

```c
union test {
    int x, y;
};

int main()
{
    // A union variable t
    union test t;

    t.x = 2; // t.y also gets value 2
    printf("After making x = 2:\n x = %d, y = %d\n\n",
           t.x, t.y);

    t.y = 10; // t.x is also updated to 10
    printf("After making y = 10:\n x = %d, y = %d\n\n",
           t.x, t.y);
    return 0;
}
```

Pointers:-

Pointers in C language is a variable that stores/points the address of another variable. A Pointer in C is used to allocate memory dynamically i.e. at run time. The pointer variable might be belonging to any of the data type such as int, float, char, double, short etc.

Pointer Syntax : data_type *var_name; Example : int *p; char *p;

Where, * is used to denote that "p" is pointer variable and not a normal variable.

KEY POINTS TO REMEMBER ABOUT POINTERS IN C:-

- Normal variable stores the value where as pointer variable stores the address of the variable.
- The content of the C pointer always be a whole number i.e. address.
- Always C pointer is initialized to null, i.e. int *p = null.
- The value of null pointer is 0.
- & symbol is used to get the address of the variable.

POINTER USE TWO THINGS.

- Referencing
- Derefrencing.

EXAMPLE:-

```
#include <stdio.h>
void main()
{
   int *ptr, q;
   q = 50;
   /* address of q is assigned to
ptr */
   ptr = &q;
   /* display q's value using ptr
variable */
   printf("%d", *ptr);
  }
```

Colours:-

Textcolor function:

syntax:

textcolor(COLOR OR COLOR INT VALUE);

Textbackground function:

syntax:

textbackground(COLOR OR COLOR INT VALUE);

Note:-If you want to set whole background color just use this function before clrscr(); function.

Constant	Value	grnd?	grnd?
BLACK	0	Yes	Yes
BLUE	1	Yes	Yes
GREEN	2	Yes	Yes
CYAN	3	Yes	Yes
RED	4	Yes	Yes
MAGENTA	5	Yes	Yes
BROWN	6	Yes	Yes
LIGHTGRAY	7	Yes	Yes
DARKGRAY	8	No	Yes
LIGHTBLUE	9	No	Yes
LIGHTGREEN	10	No	Yes
LIGHTCYAN	11	No	Yes
LIGHTRED	12	No	Yes
LIGHTMAGENTA	13	No	Yes
YELLOW	14	No	Yes
WHITE	15	No	Yes
BLINK	128	No	***

CPRINTF FUNCTION:

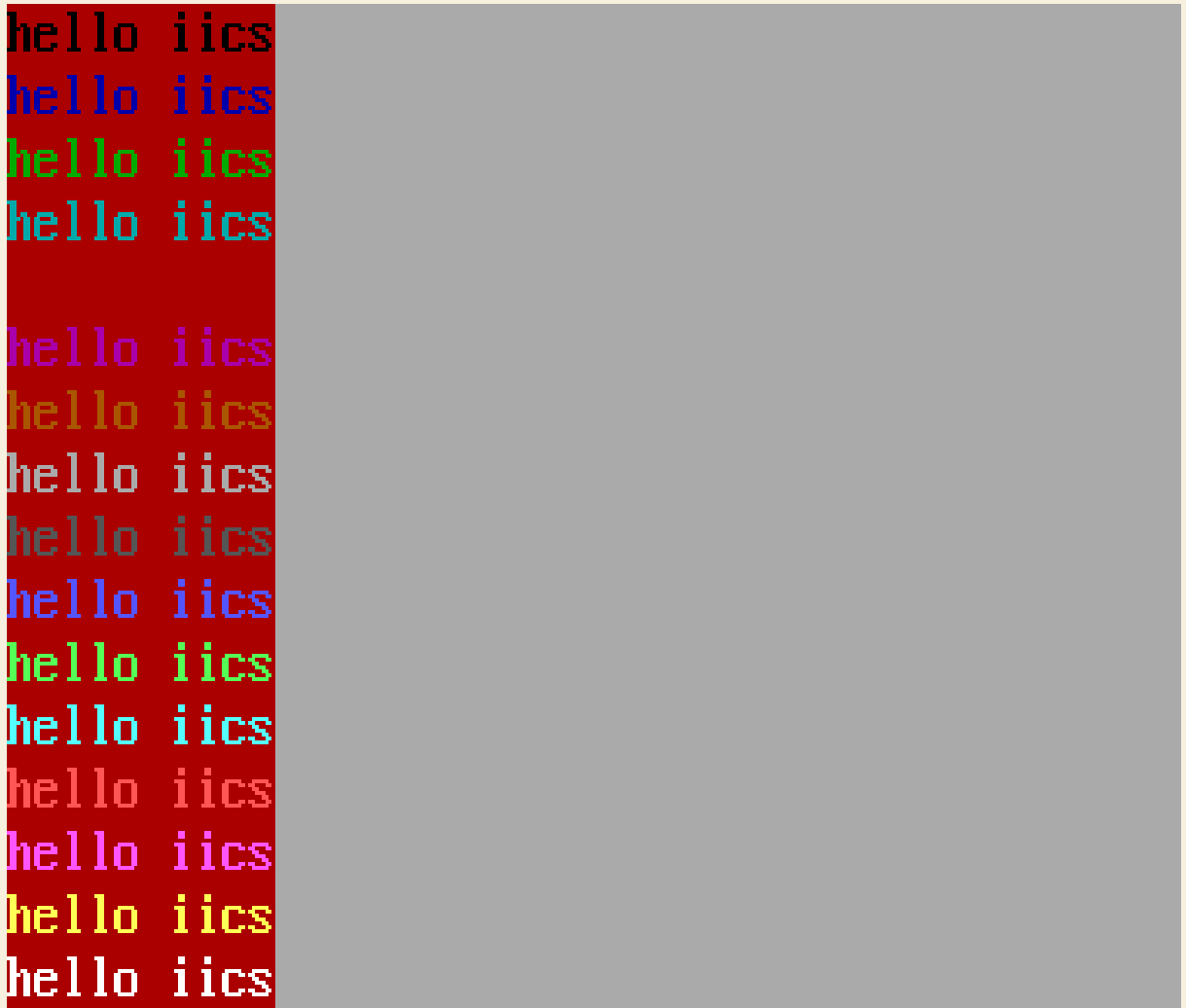
When you use color and text both then you can use cprintf function.

INPUT

```
#include<dos.h>
#include<conio.h>
#include<stdio.h>
void main()
{
int i;
textbackground(WHITE);
clrscr();
for(i=0;i<=15;i++)
{
delay(100);
textbackground(RED);
textcolor(i);
cprintf("hello iics");
printf("\n");

}
getch();
}
```

OUTPUT



TO MAKE A CIRCLE USING GRAPHICS

CIRCLE FUNCTION

SYNTAX

circle(x-axis,y-axis,radius);

REY.C

```
#include<graphics.h>
#include<stdio.h>
#include<conio.h>
void main()
{
int gdriver=DETECT,gm;
initgraph(&gdriver,&gm,"//turboc3/bgi");
setcolor(BLUE);
circle(120,150,100);
getch();
closegraph();
}
```

SET GRAPHICS DRIVER

TO CHANGE OUTLINE COLOR

X AXIS Y AXIS RADIUS

TO MAKE A LINE USING GRAPHICS

LINE FUNCTION

SYNTAX

line(x-axis1,y-axis1, x-axis2,y-axis2);

```
#include<conio.h>
#include<dos.h>
#include<stdio.h>
#include<graphics.h>
void main()
{
int gdriver=DETECT,gm;
initgraph(&gdriver,&gm,"c://turboc3/bgi");
line(1,15,1,500); //CORDINATE POINT FIRST AND SECOND
getch();
closegraph();
}
```

TO MAKE RECTANGLE USING GRAPHICS

RECTANGLE FUNCTION

SYNTAX

rectangle(left,top,right,bottom);

```
#include<graphics.h>
#include<stdio.h>
#include<conio.h>
void main()
{
int gdriver=DETECT,gm;//driver
initgraph(&gdriver,&gm,"//turboc3/bgi");     //graph
rectangle(20,20,80,80); //points(l,t,r,b);
getch();
closegraph();//closegraph_
}
```