Introduction of C Language
What is C Language?
1.It is a basic ,general-purpose programming language.
2.It is the base of all high level programming language so it is called basic programming language.
3.It is developed by Dennis Ritchie.
4.It is developed at AT and T's Bell Laboratory(USA).
5.It was developed in 1972.
6.Basically it was developed to implement UNIX operating system.

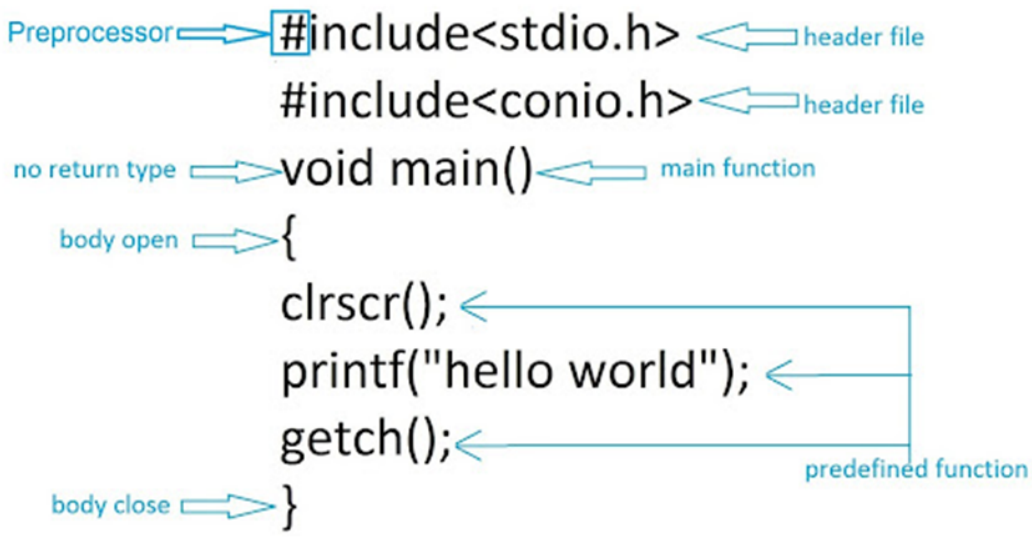
👉Features of C Language
1.Machine Independent: C is called platform independent because a java program can be run on different kind of machine for example Laptop, Desktop, Mobile etc.
2.Structure Oriented: C supports structure oriented programming so it is called structure oriented.
3.Powerful: C is a powerful programming language.

4.Flexible: An application developed in C can be modified as per user requirement so it is called flexible programming language.
5.Portable: A C program written in one system can be run in any other system. In simple a C program can be transferred from one system to another.
6.Simple: C is very simple and easy to learn.
7.High Speed: C language is designed in such a way that it takes very less time to execute as compare to other so it is called High speed programming language.
8.High Efficiency

👉Application of C Language
1.Developing Games: C language is used to develop different types of game for example Snake game,Tetris Mania etc.
2.Operating Systems: C language is primarily created for developing operating system[Unix].
3.Compilers: C language is also used to develop compilers.
4.Database Systems: C language is used to develop Database systems.
5.Network Drivers : C language is used to develop network drivers.
6.Graphics Packages: C language is also used to develop Graphics packages.
7.Interpreters: C language is also used to develop interpreters.
8.Editors:C language is also used to develop Editors.

Syntax of C Language

Syntax of C language is given below



👉stdio
1.It stands for standard input output.
2.It is a collection of predefined functions/methods.
3.It is also called library of c.
👉include
To include the header file into the program.
👉#
1.It is called preprocessor.
2.It includes the library of c into the program before the execution of program.
👉conio
1.It stands for console input output.
2.It is used to show the output on console window.
👉void
1.It is a keyword.
2.It indicates that no one value is being returned by the function.
3.If we use any other keywords like int or float or char etc in place of void then we should use return keyword.
👉main
1.It is the function which is called the entry point of any program.
2.The execution of any program starts from the main function.

3.If in a program we use only one function then it should be main function.
👉clrscr
1.It stands for clear screen.
2.It is a predefined function which is used to clear the output screen.
3.It acts like a duster on output screen.
4.clrscr() function is a pre-defined function in the conio.h header file.
👉printf
1.It is a predefined function which is use used to print data or information on to the output screen.
2.printf() function is a pre-defined function in the stdio.h header file.
👉getch
1.It is a predefined function which is used to hold the output screen.
2.It acts like a duster on output screen.
3.It is defined in the conio.h header file.

Variable and Constant in C Language

What is Variable ?
1.It is a name of storage space which is used to store data.
2.It's value may be changed.
3.It always contains last value stored to it.
4.It is always declared with data type.
👉Variable Declaration

```
int rollno;
float marks;
char grade;
```

Here rollno is a variable of type int ,marks is a variable of type float and grade is a variable of type char
👉Variable Initialization

When you assign a variable, you use the = symbol. The name of the variable goes on the left and the value you want to store in the variable goes on the right.

```
int rollno=201;
float marks=85.6;
char grade='A';
```

Here 201 is the value of rollno,85.6 is the value of marks and A is the value of grade. Character value is always written in single quotes.
Note: Make sure you don't confuse the assignment operator (=) with the equality operator (==). The individual = symbol assigns value while the == symbol checks if two things are equal.

👉Second way of Initialization

```
int rollno;
float marks;
char grade;
rollno=201;
marks=85.6;
grade='A';
```

👉Rules to declare a variable
1.The first letter of a variable should be alphabet or underscore(_).
2.The first letter of variable should not be digit.
3.After first character it may be combination of alphabets and digits.
4.Blank spaces are not allowed in variable name.
5.Variable name should not be a keyword.

👉What is Constant?

1.An element of program whose value can not be changed at the time of execution of program is called constant.
2.It is also called literals.
3.It may be int, float and character data type.
👉Rules for constructing integer constant
1.It must have at least one digit.
2.It must not have a decimal point.
3.It may be positive or negative.
4.The range of integer constant is between -32768 to +32767.
5.No comma or blank space are allowed in integer constant.
👉Rules for constructing floating point constant
1.It must have at least one digit.
2.It must have a decimal point.
3.It may be positive or negative.
4.No comma or blank space are allowed in floating point constant.
👉Rules for constructing character constant
1.It is a single alphabet, digit or special symbol.
2.The length of character constant is 1 character.
3.Character constant is enclosed within single quotes (Example char c='A';).

👉Example

```
int rollno=201;
float marks=85.6;
char grade='A';
```

Here 201 is integer constant,85.6 is float constant and A is character constant.

Data Type and Keywords in C Language

What is Data Type ?
1.It is a type of data which is used in the program.
2.There are many predefined data types in c library like int,char,float etc.

Basic Type	Integer Type(int)
	Floating Type(float)
	Character Type(char)
	Pointer
Derived Type	Array
	Structure
	Union

👉Integer Type

Data Type	Size in bytes	Range
short	2	-32768 to +32767
int	2	-32768 to +32767
unsigned int	2	0 to 65536
long	4	-2147483648 to +2147483647
unsigned long int	4	0 to 4,294,967,295

👉Float Type

Data Type	Size in bytes	Range
float	4	3.4E-38 to 3.4E+38
double	8	1.7E-308 to 1.7E+308
long double	10	3.4E-4932 to 1.1E+4932

👉Character Type

Data Type	Size in bytes	Range
char	1	-128 to +127

signed char	1	-128 to +127
unsigned char	1	0 to 255

👉What is Keyword ?
1.The word has a predefined meaning is called keywords.
2.It's functionality is also predefined.
3.It can not be used as an identifier.
4.There are 32 keywords are used in c.
👉List of Keywords
1.default
2.float
3.register
4.struct
5.volatile
6.break
7.do
8.for
9.return
10.switch
11.while
12.case
13.double
14.goto
15.short
16.typedef
17.char
18.else
19.if
20.signed
21.union
22.const
23.enum

24.int
25.sizeof
26.unsigned
27.continue
28.extern
29.long
30.static
31.void
32.auto

Operator in C Language

👉Operator
It is a special symbol which is used to perform logical or mathematical operation on data or variable.
👉Operand
It is a data or variable on which the operation is to be performed.
👉Types of Operator
⇒Arithmetic Operators
⇒Relational Operators
⇒Logical Operators
⇒Assignment Operators
⇒Bitwise Operators
⇒Increment/Decrement Operators
⇒Conditional Operators
⇒Special Operator
👉Arithmetic Operators

Symbol	Operation	Example
+	Addition	x+y
-	Subtraction	x-y
*	Multiplication	x*y

/	Division	x/y
%	Modulus	x%y

```
#include<stdio.h>
int main()
{
int x=16,y=3;//For operation
int add,sub,multi,div,mod;//To store the result
add=x+y;
sub=x-y;
multi=x*y
div=x/y;
mod=x%y;
printf("Addition=%d\n",add);
printf("Subtraction=%d\n",sub);
printf("Multiplication=%d\n",multi);
printf("Division=%d\n",div);
printf("Modulus=%d\n",mod);
}
```

Output
Addition=19
Subtraction=13
Multiplication=48
Division=5
Modulus=1 //because % stores remainder

👉Relational Operators

Symbol	Operation	Example
==	Equal to	2==3 returns 0

!=	Not equal to	2!=3 returns 1
>	Greater than	2>3 returns 0
<	Less than	2<3 returns 1
>=	Greater than or equal to	2>=3 returns 0
<=	Less than or equal to	2<=3 returns 1

👉Logical Operators

(x>y)&&(x>z)	Here this expression returns true if both conditions are true.
(x>y)||(x>z)	Here this expression returns true if any one or both conditions are true.
!(x>y)	Not operator reverses the state means if the condition is true it returns false and if the condition is false it returns true.

```
#include<stdio.h>
int main()
{
int a,b,c;
clrscr();
printf(“enter a ,b and c vaue”);
scanf(“%d%d%d”,&a,&b,&c);
if(a>b && a>c)
{
printf(“a gretest”);
}
if(b>a && b>c)
{
printf(“b gretest”);
}
if(c>a && c>b)
```

```
{
printf(“c gretest”);
}

}
```

Output

```
Enter a,b and c value
10
20
30
C gretest
```

👉Assignment Operators

Symbol	Example	Same as
=	x=y	x=y
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
%=	x%=y	x=x%y

```
#include<stdio.h>
int main()
{
int a,b;
clrscr();
printf(“enter a ,b  vaue”);
scanf(“%d%d”,&a,&b);
a+=b;
printf(“a+=”+a);
printf(“enter a ,b  vaue”);
scanf(“%d%d”,&a,&b);
```

```
a-=b;
printf(“a-=”+a);
printf(“enter a ,b  vaue”);
scanf(“%d%d”,&a,&b);
a*=b;
printf(“a*=”+a);
printf(“enter a ,b  vaue”);
scanf(“%d%d”,&a,&b);
a/=b;
printf(“a/=”+a);
printf(“enter a ,b  vaue”);
scanf(“%d%d”,&a,&b);

a%=b;
printf(“a%=”+a);

}
```

Output

```
Enter a,b and c value
5
3
a+=8
Enter a,b and c value
5
3
a-=2
Enter a,b and c value
5
3
a*=15
Enter a,b and c value
5
3
a/=1
```

Enter a,b and c value
5
3
a%=2

👉Bitwise Operators

Symbol	Operation	Example
&	Bitwise AND	x&y
|	Bitwise OR	x|y
<<	Shift Left	x<<2
>>	Shift Right	x>>2

```
#include<stdio.h>
int main()
{
int a,b,c;
clrscr();
printf(“enter a ,b  vaue”);
scanf(“%d%d”,&a,&b);
c=a&b;
printf(“a&b=%d”,c);
c=a|b;
printf(“a|b=%d”,c);
c=a^b;
printf(“a^b=%d”,c);
c=a<<b;
printf(“a&b=%d”,c);
c=a>>b;
printf(“a&b=%d”,c);
c=~a;
printf(“~a=%d”,c);

}
```

Output
Enter a,b value
10
2
a&b=2
a|b=10
a^b=8
a<<b=40
a>>b=2
~a=-11

👉Increment/Decrement Operators

Symbol	Name	Function	Example
++	Increment	It increments the value by 1	++x
--	Decrement	It decrements the value by 1	--x

```
#include<stdio.h>
int main()
{
int a;
clrscr();
printf(“enter a  value”);
scanf(“%d”,&a);
printf(“++a=%d\n”,(++a));
printf(“a=%d\n”,a);
printf(“a++=%d\n”,(a++));
printf(“a=%d\n”,a);
printf(“--a=%d\n”,(--a));
printf(“a=%d\n”,a);
printf(“a–=%d”,(a--));
printf(“a=%d\n”,a);
```

```
}
```

Output

```
Enter a,b value
10
++a=11
a=11
a++=11
a=12
- -a=11
a=11
a- -=11
a=10
```

👉Conditional Operators

```
 #include<stdio.h>
int main()
{
int no;
clrscr();
printf(“enter no  value”);
scanf(“%d”,&no);
(no%2==0)?printf(“even no.”):printf(“odd no.”);
}
```

Output

```
Enter no value
10
Even no
```

👉Special Operators

Symbol	Description	Example

&	It is used find address of a variable	int a=10; printf("%d",&a);
*	It is used to declare pointer type variable.	int *p;
sizeof()	printf(“++a=%d”,a); printf(“a=%d”,a); It returns the memory size of variable.	int a=10; printf("%d",sizeof(a));

Printf and Scanf in C Language

1.There are many predefined function in the library of c.
2.printf and scanf are the predefined function.
3.They are commonly used for input and output operation.
4.printf and scanf both are predefined in stdio.h header file.

👉What is printf function ?

1.It is a predefined function.
2.It is used to print information or data on to the output screen.
3.It is predefined in the stdio.h header file.
4.printf is case sensitive means Printf is wrong,so it must be in lowercase.

👉Syntax pf printf

printf(format_specifier,variable_name);

👉What is scanf function ?

1.It is a predefined function.
2.It is used to take user input at the time of execution of program.
3.It is also predefined in the stdio.h header file.
4.scanf is case sensitive means Scanf is wrong,so it must be in lowercase.

👉printf and scanf with integer

```
#include<stdio.h>
int main()
```

```
{
int rollno;//declaring integer variable
printf("Enter your roll no\n");
scanf("%d",&rollno);
printf("Roll no=%d",rollno);
}
```

Output
Enter your roll no
204
Roll no=204

👉printf and scanf with float

```
#include<stdio.h>
int main()
{
float marks;//declaring float variable
printf("Enter your marks\n");
scanf("%f",&marks);
printf("Marks=%f",marks);
}
```

Output
Enter your marks
82.3
Marks=82.3

👉printf and scanf with character

```
#include<stdio.h>
int main()
```

```
{
char grade;//declaring character variable
printf("Enter your grade\n");
scanf("%c",&grade);
printf("Grade=%c",grade);
}
### Output ###
Enter your grade
A
Grade=A
```

👉printf and scanf with string

```
#include<stdio.h>
int main()
{
char name[50];//declaring character variable
printf("Enter your name\n");
scanf("%s",&name);
printf("Name=%s",name);
}
### Output ###
Enter your name
Rockey
Name=Rocky
```

If Statement in C Language

Syntax

1.If the condition is true its body execute otherwise does not execute.

2.In the case of if in the place of condition always zero and non-zero value is checked in which zero means condition false and non-zero means condition true

👉Example 1

```
#include<stdio.h>
int main()
{
//Assigning value to the variable
int x=50,y=20;
//checking the condition
if(x>y)
{
printf("x is greater than y");
}
}
```

Output

x is greater than y

👉Example 2

```
#include<stdio.h>
int main()
{
//Assigning value to the variable
int x=50,y=20;
//checking the condition
if(x<y)
{
printf("x is greater than y");
}
}
/*
```

Output

condition is false so there is no output

```
*/
```

👉Example 3

```
#include<stdio.h>
```

```
int main()
{
if(8)
{
printf("Hello");
}
}
/*
### Output ###
Hello
*/
```

In case of at the place of condition always zero and non-zero value is checked in which zero means condition false otherwise true. Here in above example 8 is a non-zero value so condition will be true and output will be Hello

👉Example 4

```
#include<stdio.h>
int main()
{
if(0)
{
printf("Hello");
}
}
/*
### Output ###

*/
```

0 means condition false so there will be no output here.

👉Example 5

```
#include<stdio.h>
```

```
int main()
{
if(8,9,6,0,0,7,8)
{
printf("Hello");
}
}
/*
### Output ###
Hello
*/
```

In this situation always last value is checked and at last there is 8 so condition is true because 8 is non-zero value.

👉Program: Greatest Value three numbers

```
#include<stdio.h>
int main()
{
int no1,no2,no3;
printf("Enter number1:");
scanf("%d",&no1);
printf("Enter number2:");
scanf("%d",&no2);
printf("Enter number3:");
scanf("%d",&no3);
if(no1>no2&&no1>no3)
printf("%d is greaest value",no1);
if(no2>no1&&no2>no3)
printf("%d is greaest value",no2);
```

```
if(no3>no2&&no3>no1)
printf("%d is greaest value",no3);

}
/*
### Output ###
Enter number1:84
Enter number2:95
Enter number3:67
95 is greatest value
*/
```

If else statement in C Language

👉Syntax

1.If the condition is true if part executes and if the condition is false else part executes.
2.In the case of if in the place of condition always zero and non-zero value is checked in which zero means condition false and non-zero means condition true

👉Example 1

```
#include<stdio.h>
int main()
{
//Assigning value to the variable
int x=50,y=80;
//checking the condition
if(x>y)
{
printf("x is greater than y");
}
```

```
else
printf("y is greater than x");
}
/*
### Output ###
y is greater than x
*/
```

👉Example 2

```
#include<stdio.h>
int main()
{
//Assigning value to the variable
int x=50,y=80;
//checking the condition
if(x<y)//condition is true here
{
printf("x is greater than y");
}
else
printf("y is greater than x");
}
/*
### Output ###
x is greater than y
*/
```

👉Example 3

```
#include<stdio.h>
int main()
{
if(9)
{
printf("Hello");
}
else
printf("hi");
```

```
}
/*
### Output ###
Hello
*/
```

In case of if at the place of condition always zero and non-zero value is checked in which zero means condition false otherwise true. Here in above example 9 is a non-zero value so condition will be true and output will be Hello

👉Example 4

```
#include<stdio.h>
int main()
{
if(0)
{
printf("Hello");
}
else
printf("hi");
}
/*
### Output ###
Hi
*/
```

In case of if at the place of condition always zero and non-zero value is checked in which zero means condition false otherwise true. Here in above example there is 0 in place of condition so condition will be false and output will be Hi

👉Example 5

```
#include<stdio.h>
int main()
{
if(8,9,6,0,0,7,8)
{
printf("Hello");
```

```
}
else
{
printf("Hi");
}
}
/*
### Output ###
Hello
*/
```

👉Program: Check given number is even or odd

```
#include<stdio.h>
int main()
{
int no;
printf("Enter any number:");
scanf("%d",&no);
if(no%2==0)
printf("even");
else
printf("odd");
}
/*
### Output ###
Enter any number:54
even
*/
```

If else if Statement in C Language

Syntax

1.It is a part of conditional statement that executes only one condition at a time.
2.If all conditions are false then else part executes.
3.It executes that condition that becomes first true from the top.
4.In the case of if in the place of condition always zero and non-zero value is checked in which zero means condition false and non-zero means condition true

👉Example 1

```
#include<stdio.h>
int main()
{
//Assigning value to the variable
int x=10;
if(x>5)//checking the condition
{
printf("x is greater than 5");
}
else if(x<8)//checking the condition
{
printf("x is less than 8");
}
else if(x==10)//checking the condition
{
printf("x is equal to 10");
}
else
{
printf("No one condition is true");
}
}
```

output
x is greater than 5

As we can see from the above program there are three conditions in which first and third condition are true but it executes only one condition that becomes first true from the above so the output is x is greater than 5

👉Program: Show result according to percent

```
/*
Suppose
First div  >=65
Second div between 64 and 45
Third div  between 44 and 33
Fail    <33
*/
#include<stdio.h>
int main()
{
//variable declaration
float p;
//taking user input of percentage
printf("Enter your percentage:");
scanf("%f",&p);
if(p>=65)
printf("First Division");
else if(p>=45)
printf("Second Division");
else if(p>=33)
printf("Third Division");
else
printf("Sorry! You are fail!!!");
}
/*
### Output ###
Run1
Enter your percentage:85
```

First Division
Run2
Enter your percentage:30
Sorry! You are fail!!!
*/

Nested if in C Language

Syntax

1.Nested means one inside another so one if inside another if is called nested if.
2.In the case of if in the place of condition always zero and non-zero value is checked in which zero means condition false and non-zero means condition true
👉Example 1

```
#include<stdio.h>
int main()
{
//Assigning value to the variable
int x=10;
if(x>5)//true
{
  if(x<5)//false
        {
        printf("Hello");
        }
  printf("Hi");
}
}
/*
### output ###
Hi
```

```
*/
👉Example 2
#include<stdio.h>
int main()
{
//Assigning value to the variable
int x=10;
if(x>5)//true
{
  if(x>5)//true
        {
        printf("Hello");
        }
  printf("Hi");
}
}
/*
### output ###
HelloHi
*/
👉Example 3
#include<stdio.h>
int main()
{
//Assigning value to the variable
int x=10;
if(x<5)//false
{
  if(x>5)//true
        {
        printf("Hello");
        }
  printf("Hi");
}
}
```

```
/*
### output ###
No output because outer if condition is false
*/
```

👉Example 4: Find greatest value in three number

```
#include<stdio.h>
int main()
{
 //variables declaration
 int no1,no2,no3;
 //taking user input
 printf("Enter first number\n");
 scanf("%d",&no1);
 printf("Enter second number\n");
 scanf("%d",&no2);
 printf("Enter third number\n");
 scanf("%d",&no3);
 //applying condition
 if(no1>no2)
 {
        if(no1>no3)
        {
                printf("%d is greatest",no1);
        }
 }
  if(no2>no1)
 {
        if(no2>no3)
        {
                printf("%d is greatest",no2);
        }
 }
  if(no3>no1)
 {
        if(no3>no2)
```

```
        {
                printf("%d is greatest",no3);
        }
 }
}
/*
###Output###
Enter first number
85
Enter second number
96
Enter third number
47
96 is greatest
*/
```

Switch Statement in C Language

Switch statement allows us to execute one statement from many statement and that statements are called case. Actually in switch statement, inside the body of switch a number of cases are used and a parameter are passed and from which case this parameter is matched, executed
👉Syntax

1.In the switch statement a value/number is passed in the place of parameter and that case will execute which is equal to that value/number.
2.If no case matched with parameter then default case will execute.
👉Example 1

```
#include<stdio.h>
int main()
{
//Assigning parameter's value
int p=2;
```

```
switch(p)
{
case 1:
printf("it is case 1");
break;
case 2:
printf("it is case 2");
break;
case 3:
printf("it is case 3");
break;
default:
printf("no case matched");
}
return 0;
}
```

output

it is case 2

//because p=2 so case 2 will execute

👉Example 2

```
#include<stdio.h>
int main()
{
//Assigning parameter's value
int p=5;
switch(p)
{
case 1:
printf("it is case 1");
break;
case 2:
printf("it is case 2");
break;
case 3:
printf("it is case 3");
```

```
break;
default:
printf("no case matched");
}
return 0;
}
/*
### output ###
no case matched
because value of p is 5 and it will not match with any case
so default case will execute
*/
```

👉Example 3

```
#include<stdio.h>
int main()
{
//Assigning parameter's value
int p=5;
switch(p)
{
case 1:
printf("Hello");
break;
case 2:
printf("Hi");
break;
case 3:
printf("Tata");
break;
case 2+1:
printf("Bye");
break;
default:
printf("no case matched");
}
```

```
return 0;
}
/*
### output ###
Error
*/
```

Duplicate case are not allowed in switch statement and here case 3 and case 2+1 are duplicate of each other.

👉Example 4

```
#include<stdio.h>
int main()
{
//Assigning parameter's value
int p=2;
switch(p)
{
case 1:
printf("Hello");
break;
case 2:
printf("Hi");

case 3:
printf("Tata");
break;
case 4:
printf("Bye");
break;
default:
printf("no case matched");
}
return 0;
}
/*
```

output
HiTata
*/

If we remove break from any case then the next case will execute automatically so here value of parameter is 2 and it will match with case so Hi will print but there is no break after case 2 so here case 3 will also execute and it will print Tata so the final output is HiTata.
👉Example 5

```
#include<stdio.h>
int main()
{
//Assigning parameter's value
int p=3;
switch(p)
{
default:
printf("no case matched");
case 1:
printf("Hello");
break;
case 2:
printf("Hi");
break;
case 3:
printf("Tata");
break;
case 4:
printf("Bye");
break;
}
return 0;
}
/*
### output ###
```

Tata
*/

default case can be write anywhere in the body of switch but it will execute when no case matched.

👉Program: Check given Alphabet is Vowel or Consonant

```
#include<stdio.h>
int main()
{
//variable declaration
char ch;
printf("Enter any alphabet\n");
scanf("%c",&ch);
switch(ch)
{
case 'A':
case 'E':
case 'I':
case 'O':
case 'U':
case 'a':
case 'e':
case 'i':
case 'o':
case 'u':
        printf("Vowel");
        break;
default:
        printf("Consonant");
}
return 0;
}
/*
### output ###
Enter any alphabet
```

M
Consonant
*/

For loop in C Language

To run the particular block of code continuously until a required condition is fulfill is called looping. It is used to perform looping operation. When the condition will become false the execution of loop will be stopped.
👉Syntax

1.In for loop there are three part initialization, condition and increment/decrement.
2.Initialization part executes only once.
3.All the three parts of for loop are optional
👉Example 1

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

Output
1
2
3
4
5
6
7

8
9
10

In the above program i is a variable which is initialized with 1,condition goes to 10 and it is incremented by 1 so the output will be 1 to 10.
👉Example 2

```
#include<stdio.h>
int main()
{
for( ; ; )
{

        printf("Hello ");
}
}
/*
### output ###
Hello Hello Hello .....Infinite times
*/
```

All the three parts of for loop are optional so there is no error in the above program and output will be infinite times Hello because there is no condition to stop the execution of loop.

Fibonacci Series in C
Fibonacci Series in C: In case of fibonacci series, next number is the sum of previous two numbers for example 0, 1, 1, 2, 3, 5, 8, 13, 21 etc. The first two numbers of fibonacci series are 0 and 1.

```
#include<stdio.h>
#include<conio.h>
Void main()
{
Int a=0,b=1,c=0,n;
```

```
clrscr();
printf(“enter n value”);
scanf(“%d”);
for (int i=1;i<=n;i++)
{
printf(“%d ”,c);
a=b;
b=c;
c=a+b;
}
getch();
}
```

While Loop in C Language

To run the particular block of code continuously until a required condition is fulfill is called looping. It is used to perform looping operation. When the condition will become false the execution of loop will be stopped.

👉Syntax

Its body will execute until the given condition is true.

👉Example 1

```
#include<stdio.h>
int main()
{
int i=1;
while(i<=10)
{
 printf("%d ",i);
 i++;
}
}
/*
### output ###
```

```
1 2 3 4 5 6 7 8 9 10
*/
```

In the above program i is a variable which is initialized with 1,condition goes to 10 and it is incremented by 1 so the output will be 1 to 10.

👉Program: Factorial program

The product of all positive number which is less than or equal to a given positive number is called factorial of that number.

For example if we want to find factorial of 5 then it will be product of 1,2,3,4 and 5.

To make a factorial program we will take three variables , one for number , second for running loop and third one for storing the product of numbers.

After taking user input of number we will run loop from 1 to number itself and then we will find product of number from 1 to number and store it to a variable.

After multiplying number from 1 to number we will print the product as a output of program.

```
#include<stdio.h>
int main()
{
 //variables declaration
 int no,fact=1,i=1;
 //taking user input
 printf("Enter any number\n");
 scanf("%d",&no);
 //loop for multiplying numbers
 while(i<=no)
 {
        //incrementing factor count
        fact=fact*i;
```

```
        //incrementing loop
        i++;
 }
 //printing result
 printf("Factorial of %d is %d",no,fact);

}
/*
###Output###
Enter any number
5
Factorial of 5 is 120
*/
```

Do While loop in C Language

To run the particular block of code continuously until a required condition is fulfill is called looping. It is used to perform looping operation. When the condition will become false the execution of loop will be stopped.
👉Syntax

Its body will execute until the given condition is true.
👉Example 1

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

```
/*
### output ###
1 2 3 4 5 6 7 8 9 10
*/
```

In the above program i is a variable which is initialized with 1,condition goes to 10 and it is incremented by 1 so the output will be 1 to 10.

👉Program: Factorial program
The product of all positive number which is less than or equal to a given positive number is called factorial of that number.

For example if we want to find factorial of 5 then it will be product of 1,2,3,4 and 5.

To make a factorial program we will take three variables , one for number , second for running loop and third one for storing the product of numbers.

After taking user input of number we will run loop from 1 to number itself and then we will find product of number from 1 to number and store it to a variable.

After multiplying number from 1 to number we will print the product as a output of program.

```
#include<stdio.h>
int main()
{
 //variables declaration
 int no,fact=1,i=1;
 //taking user input
 printf("Enter any number\n");
 scanf("%d",&no);
 //loop for multiplying numbers
```

```
 do
 {
        //incrementing factor count
        fact=fact*i;
        //incrementing loop
        i++;
 }
 while(i<=no);
 //printing result
 printf("Factorial of %d is %d",no,fact);

}
/*
###Output###
Enter any number
5
Factorial of 5 is 120
*/
```

Nested loop in C Language

To run the particular block of code continuously until a required condition is fulfill is called looping. It is used to perform looping operation. When the condition will become false the execution of loop will be stopped.
A loop inside another loop is called nested loop so one for loop inside another for loop is called nested for loop and so on.
👉Syntax
From the syntax we can see that there are two for loops are used in which one for loop is inside other. Here i am using only for loop but we can use any loops like for loop, while loop or do while loop inside any loops. for example we can use while loop inside for loop, for loop inside while loop , while loop inside do while loop etc.

👉Example : Pattern Printing

Pattern program is one of the most useful program which is made using nested loop. Here i am going to print a simple square pattern of 4x4 dimension. Here i am using two for loop to print this pattern. Outer for loop is using for row and inner for loop is using for column. Variable i is used for outer loop or row and variable j is used for inner loop or column.
Note: Don't forget to change line after closing brackets of inner loop other all stars will print in a single line.

```
#include<stdio.h>
int main()
{
//outer loop
for(int i=1;i<=4;i++)
{
  //inner loop
  for(int j=1;j<=4;j++)
  {
        printf("* ");
  }
  printf("\n");
}
}
/*
### output ###
* * * *
* * * *
* * * *
* * * *
*/
```

👉Example :Print all prime numbers between range

This type of program mostly asked in any Viva or interview or exam paper of computer science students that can be made by using nested loop in a very simple way.

```
#include<stdio.h>
int main()
{
 int n;
 printf("Enter number upto you want to print prime number\n");
 scanf("%d",&n);
 for(int i=2;i<=n;i++)
  {
        int no=i,m=0;
        for(int j=2;j<=no-1;j++)
        {
          if(no%j==0)
          m=1;
        }
         if(m==0)
         printf("%d ",no);
  }
}
/*
### Output ###
Enter number upto you want to print prime number
30
2 3 5 7 11 13 17 19 23 29
*/
```

Fibonacci Series in C

PRANOTI PATIL

Fibonacci Series in C: In case of fibonacci series, *next number is the sum of previous two numbers* for example 0, 1, 1, 2, 3, 5, 8, 13, 21 etc. The first two numbers of fibonacci series are 0 and 1.

There are two ways to write the fibonacci series program:

- Fibonacci Series without recursion
- Fibonacci Series using recursion

Fibonacci Series in C without recursion

Let's see the fibonacci series program in c without recursion.

```
1. #include<stdio.h>
2. int main()
3. {
4.  int n1=0,n2=1,n3,i,number;
5.  printf("Enter the number of elements:");
6.  scanf("%d",&number);
7.  printf("\n%d %d",n1,n2);//printing 0 and 1
8.  for(i=2;i<number;++i)//loop starts from 2 because 0 and 1 are already printed
9.  {
10.  n3=n1+n2;
11.  printf(" %d",n3);
12.  n1=n2;
13.  n2=n3;
14. }
15.  return 0;
16. }
```

Output:

Enter the number of elements:15

0 1 1 2 3 5 8 13 21 34 55 89 144 233 377

Fibonacci Series using recursion in C

Let's see the fibonacci series program in c using recursion.

```
1.  #include<stdio.h>
2. void printFibonacci(int n){
3.     static int n1=0,n2=1,n3;
4.     if(n>0){
5.         n3 = n1 + n2;
6.         n1 = n2;
7.         n2 = n3;
8.         printf("%d ",n3);
9.         printFibonacci(n-1);
10.    }
11.}
12.int main(){
13.    int n;
14.    printf("Enter the number of elements: ");
15.    scanf("%d",&n);
16.    printf("Fibonacci Series: ");
17.    printf("%d %d ",0,1);
18.    printFibonacci(n-2);//n-2 because 2 numbers are already printed
19.  return 0;
20. }
```

Output:
Enter the number of elements:15
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377

Prime Number program in C

Prime number in C: **Prime number** is a number that is greater than 1 and divided by 1 or itself. In other words, prime numbers can't be divided by other numbers than itself or 1. For example 2, 3, 5, 7, 11, 13, 17, 19, 23.... are the prime numbers.

> Note: Zero (0) and 1 are not considered as prime numbers. Two (2) is the only one even prime number because all the numbers can be divided by 2.

Let's see the prime number program in C. In this c program, we will take an input from the user and check whether the number is prime or not.

```
1. #include<stdio.h>
2. int main(){
3. int n,i,m=0,flag=0;
4. printf("Enter the number to check prime:");
5. scanf("%d",&n);
6. m=n/2;
7. for(i=2;i<=m;i++)
8. {
9. if(n%i==0)
10.{
11.printf("Number is not prime");
12.flag=1;
13.break;
14.}
15.}
16.if(flag==0)
17.printf("Number is prime");
18.return 0;
19. }
```

Output:
Enter the number to check prime:56
Number is not prime

Enter the number to check prime:23
Number is prime

Palindrome program in C

Palindrome number in c: A **palindrome number** is *a number that is same after reverse*. For example 121, 34543, 343, 131, 48984 are the palindrome numbers.

Palindrome number algorithm

- Get the number from user
- Hold the number in temporary variable
- Reverse the number
- Compare the temporary number with reversed number
- If both numbers are same, print palindrome number
- Else print not palindrome number

Let's see the palindrome program in C. In this c program, we will get an input from the user and check whether number is palindrome or not.

```
1. #include<stdio.h>
2. int main()
3. {
4. int n,r,sum=0,temp;
5. printf("enter the number=");
6. scanf("%d",&n);
7. temp=n;
8. while(n>0)
```

```
9.  {
10. r=n%10;
11. sum=(sum*10)+r;
12. n=n/10;
13. }
14. if(temp==sum)
15. printf("palindrome number ");
16. else
17. printf("not palindrome");
18. return 0;
19. }
```

Output:
enter the number=151
palindrome number

enter the number=5621
not palindrome number

Factorial Program in C

Factorial Program in C: Factorial of n is the *product of all positive descending integers*. Factorial of *n* is denoted by n!. For example:

1. 5! = 5*4*3*2*1 = 120
2. 3! = 3*2*1 = 6

Here, 5! is pronounced as "5 factorial", it is also called "5 bang" or "5 shriek".

The factorial is normally used in Combinations and Permutations (mathematics).

There are many ways to write the factorial program in c language. Let's see the 2 ways to write the factorial program.

- Factorial Program using loop
- Factorial Program using recursion

Factorial Program using loop

Let's see the factorial Program using loop.

```
1. #include<stdio.h>
2. int main()
3. {
4.  int i,fact=1,number;
5.  printf("Enter a number: ");
6.   scanf("%d",&number);
7.     for(i=1;i<=number;i++){
8.       fact=fact*i;
9.   }
10.  printf("Factorial of %d is: %d",number,fact);
11. return 0;
12.}
```

Output:
Enter a number: 5
Factorial of 5 is: 120

Factorial Program using recursion in C

Let's see the factorial program in c using recursion.

```
1. #include<stdio.h>
2.
```

```
3.  long factorial(int n)
4.  {
5.    if (n == 0)
6.      return 1;
7.    else
8.      return(n * factorial(n-1));
9.  }
10.
11. void main()
12. {
13.  int number;
14.  long fact;
15.  printf("Enter a number: ");
16.  scanf("%d", &number);
17.
18.  fact = factorial(number);
19.  printf("Factorial of %d is %ld\n", number, fact);
20.  return 0;
21. }
```

Output:
Enter a number: 6
Factorial of 5 is: 720

Armstrong Number in C

Before going to write the c program to check whether the number is Armstrong or not, let's understand what is Armstrong number.

Armstrong number is *a number that is equal to the sum of cubes of its digits*. For example 0, 1, 153, 370, 371 and 407 are the Armstrong numbers.

Let's try to understand why **153** is an Armstrong number.

1. 153 = (1*1*1)+(5*5*5)+(3*3*3)
2. where:
3. (1*1*1)=1
4. (5*5*5)=125
5. (3*3*3)=27
6. So:
7. 1+125+27=153

Let's try to understand why **371** is an Armstrong number.

1. 371 = (3*3*3)+(7*7*7)+(1*1*1)
2. where:
3. (3*3*3)=27
4. (7*7*7)=343
5. (1*1*1)=1
6. So:
7. 27+343+1=371

Let's see the c program to check Armstrong Number in C.

```
1. #include<stdio.h>
2.  int main()
3. {
4. int n,r,sum=0,temp;
5. printf("enter the number=");
6. scanf("%d",&n);
7. temp=n;
8. while(n>0)
9. {
10. r=n%10;
```

```
11. sum=sum+(r*r*r);
12. n=n/10;
13. }
14. if(temp==sum)
15. printf("armstrong  number ");
16. else
17. printf("not armstrong number");
18. return 0;
19. }
```

Output:
enter the number=153
armstrong number

enter the number=5
not armstrong number

Sum of digits program in C

C program to sum each digit: We can write the sum of digits program in c language by the help of loop and mathematical operation only.

Sum of digits algorithm

To get sum of each digits by c program, use the following algorithm:

- Step 1: Get number by user
- Step 2: Get the modulus/remainder of the number
- Step 3: sum the remainder of the number
- Step 4: Divide the number by 10
- Step 5: Repeat the step 2 while number is greater than 0.

Let's see the sum of digits program in C.

```
1. #include<stdio.h>
2.  int main()
3. {
4. int n,sum=0,m;
5. printf("Enter a number:");
6. scanf("%d",&n);
7. while(n>0)
8. {
9. m=n%10;
10. sum=sum+m;
11. n=n/10;
12. }
13. printf("Sum is=%d",sum);
14. return 0;
15. }
```

Output:

```
Enter a number:654
Sum is=15

Enter a number:123
Sum is=6
```

C Program to reverse number

We can reverse a number in c using loop and arithmetic operators. In this program, we are getting number as input from the user and reversing that number. Let's see a simple c example to reverse a given number.

```
1. #include<stdio.h>
```

```
2.  int main()
3.  {
4.  int n, reverse=0, rem;
5.  printf("Enter a number: ");
6.    scanf("%d", &n);
7.    while(n!=0)
8.    {
9.       rem=n%10;
10.     reverse=reverse*10+rem;
11.     n/=10;
12.   }
13.  printf("Reversed Number: %d",reverse);
14. return 0;
15.}
```

Output:
Enter a number: 123
Reversed Number: 321

C Program to swap two numbers without third variable

We can swap two numbers without using third variable. There are two common ways to swap two numbers without using third variable:

1. By + and -
2. By * and /

Program 1: Using + and -

Let's see a simple c example to swap two numbers without using third variable.

```
1. #include<stdio.h>
2.  int main()
3. {
4. int a=10, b=20;
5. printf("Before swap a=%d b=%d",a,b);
6. a=a+b;//a=30 (10+20)
7. b=a-b;//b=10 (30-20)
8. a=a-b;//a=20 (30-10)
9. printf("\nAfter swap a=%d b=%d",a,b);
10. return 0;
11. }
```

Output:
Before swap a=10 b=20
After swap a=20 b=10

Program 2: Using * and /

Let's see another example to swap two numbers using * and /.

```
1. #include<stdio.h>
2. #include<stdlib.h>
3.  int main()
4. {
5. int a=10, b=20;
6. printf("Before swap a=%d b=%d",a,b);
7. a=a*b;//a=200 (10*20)
8. b=a/b;//b=10 (200/20)
9. a=a/b;//a=20 (200/10)
10. system("cls");
```

```
11. printf("\nAfter swap a=%d b=%d",a,b);
12. return 0;
13. }
```

Output:
Before swap a=10 b=20
After swap a=20 b=10

Pointer in C Language

1.It is a special type of variable which is used to store the address of another variable.
2.It can store the address of same data type means an integer pointer can store the address of integer variable, character pointer can store the address of character variable and so on.
3.If we add asterisk(*) symbol with any variable at the time of declaring variable then this variable is called pointer variable.
4.we use ampersand symbol to get the address of variable.
5.* symbol is used to get the value at address which is hold by pointer
👉Syntax

1.Here a is a normal variable.
2.p is a pointer variable because it is associated with * symbol.

👉Example 1

```
#include<stdio.h>
int main()
{
int a=10;//initializing normal variable
int *p;//declaring pointer variable
p=&a;//address of variable a is assigned to p
```

```
printf("value of a=%d\n",a);
printf("address of a=%d\n",&a);
printf("value of p=%d\n",p);
printf("address of p=%d\n",&p);
printf("value of *p=%d\n",*p);
}
/*
### Output ###
value of a=10
address of a=6618652
value of p=6618652
address of p=6618640
value of *p=10
*/
```

👉Example 2:Addition Program using pointer

```
#include<stdio.h>
int main()
{
int a,b=10,c=20;//initializing normal variable
int *p,*q,*r;//declaring pointer variable
p=&a;//address of variable a is assigned to p
q=&b;//address of variable b is assigned to q
r=&c;//address of variable c is assigned to r
*p=*q+*r;//Performing operation
printf("Add=%d",*p);
}
/*
### Output ###
Add=30
*/
```

👉Example 2:User input in pointer

```
#include<stdio.h>
int main()
{
int a,b,c;//initializing normal variable
int *p,*q,*r;//declaring pointer variable
p=&a;//address of variable a is assigned to p
q=&b;//address of variable b is assigned to q
r=&c;//address of variable c is assigned to r
//Taking user input
printf("Enter first number\n");
scanf("%d",q);
printf("Enter second number\n");
scanf("%d",r);

*p=*q+*r;//Performing operation
printf("Add=%d",*p);
}
/*
### Output ###
Enter first number
50
Enter second number
40
Add=90
*/
```

👉Example 3:Pointer to Pointer

```
#include<stdio.h>
```

```
int main()
{
int a=10;//initializing normal variable
int *p,**q,***r,****s;//declaring pointer variable
p=&a;//address of variable a is assigned to p
q=&p;//address of pointer p is assigned to q
r=&q;//address of pointer q is assigned to r
s=&r;//address of pointer r is assigned to s
printf("a=%d\n",a);
printf("*p=%d\n",*p);
printf("**q=%d\n",**q);
printf("***r=%d\n",***r);
printf("****s=%d\n",****s);
}
/*
### Output ###
a=10
*p=10
**q=10
***r=10
****s=10
*/
```

👉Character Pointer

```
#include<stdio.h>
int main()
{
char a='Z';//initializing normal variable
char *p;//declaring pointer variable
p=&a;//address of variable a is assigned to p
printf("value of a=%c\n",a);
printf("Value  of *p=%c\n",*p);
```

```
}
/*
### Output ###
value of a=Z
Value  of *p=Z
*/
```

👉Float Pointer

```
#include<stdio.h>
int main()
{
float a=85.4;//initializing normal variable
float *p;//declaring pointer variable
p=&a;//address of variable a is assigned to p
printf("value of a=%f\n",a);
printf("Value  of *p=%f\n",*p);
}
/*
### Output ###
value of a=85.400002
Value  of *p=85.400002
*/
```

Array in C Language

1.It is a collection of data of same data type.
2.It is used to store group of data simultaneously.
3.It can store data of same data type means an integer array can store only integer value ,character array can store only character value and so on.

4.We can not fetch data from array directly therefore we use index point.
5.The indexing of array always start with 0.
6.Index value is always an integer number.
7.Array may be of any data type like int, char, float etc.
👉Syntax

1.Here a is the name of array.
2.int is the data type of array.
3.Size of array is 5 means we can store maximum 5 values in this array.

👉Initialization of array method 1

```
int a[5]={ 45,23,89,12,78 };
```

👉Initialization of array method 2

```
int a[5];
a[0]=45;
a[1]=23;
a[2]=89;
a[3]=12;
a[4]=78;
```

👉Printing of array element method 1

```
#include<stdio.h>
int main()
{
int a[5]={ 45,23,89,12,78 };
printf("value at a[0]=%d\n",a[0]);
printf("value at a[1]=%d\n",a[1]);
printf("value at a[2]=%d\n",a[2]);
printf("value at a[3]=%d\n",a[3]);
```

```
printf("value at a[4]=%d\n",a[4]);
}
```

Output
value at a[0]=45
value at a[1]=23
value at a[2]=89
value at a[3]=12
value at a[4]=78

Here in the above program if the array elements increases then number of printf statement will also increase. Here we are using only 5 array elements so you will find that it is very simple to print all elements of array but this method of printing array elements is very time consuming and is for beginners, if you want to print all array elements using very few line then go for second method which is below.

👉Printing of array element method 2

Here i am going to use for loop to print all the elements of array and this is the best method to access all the elements of array. Here in this case if the array elements increases then only we have to increase loop count and remain code will be same.

```
#include<stdio.h>
int main()
{
int a[5]={ 45,23,89,12,78 };
for(int i=0;i<=4;i++)
{
printf("value at a[%d]=%d\n",i,a[i]);
```

```
}
}
/*
### Output ###
value at a[0]=45
value at a[1]=23
value at a[2]=89
value at a[3]=12
value at a[4]=78
*/
```

👉User input in array

This method of taking user input in array is very time consuming but if you are beginner then you can use it otherwise go for user input using loop which is discussed after this topic.

```
#include<stdio.h>
int main()
{
int a[3];//declaring array
printf("Enter element 1=");
scanf("%d",&a[0]);//taking input at index 0
printf("Enter element 2=");
scanf("%d",&a[1]);//taking input at index 1
printf("Enter element 3=");
scanf("%d",&a[2]);//taking input at index 2
for(int i=0;i<=2;i++)
{
printf("value at a[%d]=%d\n",i,a[i]);
```

```
}
}
/*
### Output ###
Enter element 1=12
Enter element 2=45
Enter element 3=17
value at a[0]=12
value at a[1]=45
value at a[2]=17
*/
```

👉User input using loop

To assign elements in array by taking user input is also very simple in array because using loop we can simply store many elements in array using very few line of code.

```
#include<stdio.h>
int main()
{
int a[3];//declaring array
for(int j=0;j<=2;j++)
{
printf("Enter element %d=",(j+1));
scanf("%d",&a[j]);//taking user input
}
for(int i=0;i<=2;i++)
{
printf("value at a[%d]=%d\n",i,a[i]);
}
}
```

```
/*
### Output ###
Enter element 1=12
Enter element 2=45
Enter element 3=17
value at a[0]=12
value at a[1]=45
value at a[2]=17
*/
```

Double Dimensional Array in C Language

1.Double dimensional array is called array of array.
2.It is organized as matrix that can be represented by the collection of rows and columns.
3.It is a collection of data of same data type.
4.It is used to store group of data simultaneously.
5.It can store data of same data type means an integer array can store only integer value ,character array can store only character value and so on.
6.We can not fetch data from array directly therefore we use index point.
7.The indexing of array always start with 0.
8.Index value is always an integer number.
9.Array may be of any data type like int, char, float etc.
👉Syntax

1.Here a is the name of array.
2.int is the data type of array.
3.Size of array is 3x3 means we can store maximum 9 values in this array.

👉Initialization of array method 1

```
int a[3][3]={ {40,50,60},{10,20,30},{70,80,90} };
```

👉Initialization of array method 2

```
int a[5];
a[0][0]=40;
a[0][1]=50;
a[0][2]=60;
a[1][0]=10;
a[1][1]=20;
a[1][2]=30;
a[2][0]=70;
a[2][1]=80;
a[2][2]=90;
```

👉Printing of array element method 1

```
#include<stdio.h>
int main()
{
int a[3][3]={{40,50,60},{10,20,30},{70,80,90}};//Initializing array
printf("value at a[0][0]=%d\n",a[0][0]);
printf("value at a[0][1]=%d\n",a[0][1]);
printf("value at a[0][2]=%d\n",a[0][2]);
printf("value at a[1][0]=%d\n",a[1][0]);
printf("value at a[1][1]=%d\n",a[1][1]);
printf("value at a[1][2]=%d\n",a[1][2]);
printf("value at a[2][0]=%d\n",a[2][0]);
printf("value at a[2][1]=%d\n",a[2][1]);
printf("value at a[2][2]=%d\n",a[2][2]);
}
/*
```

```
### Output ###
value at a[0][0]=40
value at a[0][1]=50
value at a[0][2]=60
value at a[1][0]=10
value at a[1][1]=20
value at a[1][2]=30
value at a[2][0]=70
value at a[2][1]=80
value at a[2][2]=90
*/
```

Here in the above program if the array elements increases then number of printf statement will also increase. Here we are using only 9 array elements so you will find that it is very simple to print all elements of array but this method of printing array elements is very time consuming and is for beginners, if you want to print all array elements using very few line then go for second method which is below.

👉Printing of array element method 2

Here i am going to use for loop to print all the elements of array and this is the best method to access all the elements of array. Here in this case if the array elements increases then only we have to increase loop count and remain code will be same.

```
#include<stdio.h>
int main()
{
int a[3][3]={{40,50,60},{10,20,30},{70,80,90}};//Initializing array
for(int i=0;i<=2;i++)
```

```
{
  for(int j=0;j<=2;j++)
  {
   printf("%d ",a[i][j]);
  }
   printf("\n");
}
}
/*
### Output ###
40  50  60
10  20  30
70  80  90
*/
```

👉User input using loop

```
#include<stdio.h>
int main()
{
int a[3][3];//Declaring array
int i,j;
//Nested loop for user input
for(i=0;i<=2;i++)
{
  for(j=0;j<=2;j++)
  {
  printf("Enter element at a[%d][%d]=",i,j);
  scanf("%d",&a[i][j]);
  }
}
//Nested loop for printing output
for(i=0;i<=2;i++)
```

```
{
  for(j=0;j<=2;j++)
  {
   printf("%d ",a[i][j]);
  }
   printf("\n");
}
}
/*
### Output ###
Enter value at a[0][0]=4
Enter value at a[0][1]=5
Enter value at a[0][2]=6
Enter value at a[1][0]=1
Enter value at a[1][1]=2
Enter value at a[1][2]=3
Enter value at a[2][0]=7
Enter value at a[2][1]=8
Enter value at a[2][2]=9
4  5  6
1  2  3
7  8  9
*/
```

Function in C Language

👉Function

1.It is a collection of statements that performs an specific task.
2.It executes when it is called by its name.
3.A large program is divided into a number of small building block for simplicity and this building block is called function.
4.We can call a function again and again.
5.The most important features of function is code reusability.
6.The C library provides many pre-defined functions.

👉Syntax of Function

	Description
return type	1.It is a keyword which indicates that which type of value is being returning by the function. 2.If we do not to return any value then we use void keyword in place of return_type.
function name	1.It is the actual name of the function. 2.It is also used at the time of calling the function.
parameter list	1.It is the place where we can pass a number of parameter/variable. 2.These variables may be used in the program. 3.The value of parameter can be initialized or we can pass it from the calling of function. 4.It is optional part.
body	It is the place where the actual code is written to perform the specific task.

Key point about the function.

Function Declaration: At this stage the function is declared .For example :

void add();

This is a function declaration in which void(no return type) indicates there is no value returning by this function and add is the name of function.

Note-It is optional.

Function Definition: This is the place where actual code is written to perform the task. For example :

```
void add()
{
        int x,y=20,z=30;
        x=y+z;
        printf("Add=%d",x);
}
```

This is a function definition and here we can see that code is written to perform the addition task.

Function Calling: At this stage the function is called .For example :

add();
To call a function just write function name and put semi-colon(;) after it.
👉Complete Example

```
#include<stdio.h>
void add();//function declaration
void add()//function definition
{
        int x,y=20,z=30;
        x=y+z;
        printf("Add=%d",x);
}
int main()
{
        add();//function calling
}
/*
###Output###
Add=50
*/
```

Types of Function
There are two types of function
1.Pre-defined Function
2.User-defined Function

Predefined Function
The function which is predefined in the library is called predefined function. for example:-
printf(),scanf(),clrscr(),getch() etc.
User-defined Function
The function which is made by the user is called user-defined function. for example:-
add(),sub(),multi(),div()[Note: These are user-defined name.it may different.] etc.
Category of User-defined Function

There are four category of user-defined function
1. Function with no return type and no parameter.
2. Function with no return type and with parameter.
3. Function with return type and no parameter.
4. Function with return type and with parameter.

Function with no return type and no parameter

The function in which there is no parameter and there is no value returning by that function is called Function with no return type and no parameter.

```
#include<stdio.h>
void add();//function declaration
void add()//function definition
{
	int x,y=20,z=30;
	x=y+z;
	printf("Add=%d",x);
}
int main()
{
	add();//function calling
}
/*
###Output###
Add=50
*/
```

Function with no return type and with parameter

The function in which there is some parameter and there is no value returning by that function is called Function with no return type and with parameter.

```
#include<stdio.h>
void add(int y,int z);//function declaration
void add(int y,int z)//function definition
{
	int x;
	x=y+z;
	printf("Add=%d",x);
```

```
}
int main()
{
	//here 20 will assign to y and 30 to z
	add(20,30);//function calling
}
/*
###Output###
Add=50
*/
```

In the above example there are two parameter of integer type namely y and z there at the time of calling two integer value will be passed in which first will assign to y and second will assign to z.

Function with return type and no parameter

The function in which there is no parameter and there is some value returning by that function is called Function with return type and no parameter.

```
#include<stdio.h>
int add();//function declaration
int add()//function definition
{
	int x,y=20,z=30;
	x=y+z;
	//return value of x
	return x;
}
int main()
{
	//assigning return value to variable rs
	int rs=add();//function calling
	printf("Add=%d",rs);
}
/*
###Output###
Add=50
```

```
*/
```

In the above example there is no parameter but the function will return integer value because there is int keyword in the place of return type and returned value will assign to variable rs.

Function with return type and with parameter

The function in which there is some parameter and there is some value returning by that function is called Function with return type and with parameter.

```
#include<stdio.h>
int add(int y,int z);//function declaration
int add(int y,int z)//function definition
{
	int x;
	x=y+z;
	//return value of x
	return x;
}
int main()
{
	//assigning return value to variable rs
	int rs=add(20,30);//function calling
	printf("Add=%d",rs);
}
/*
###Output###
Add=50
*/
```

In the above example there is two parameter and the function will return integer value because there is int keyword in the place of return type and returned value will assign to variable rs.

Calling of Function

There are two way of calling a function

1. Call By Value.
2. Call By Reference.

Call by Value

In this type of calling a function direct value is passed at the time of calling. In call by value the changes made in formal parameters don't reflect in actual parameters.

```
#include<stdio.h>
 void add(int m)//function definition
{
        //adding 10 to m
        m=m+10;
}
int main()
{
        int x=10;
        printf("Before Calling x=%d\n",x);
        add(x);
        printf("After Calling x=%d",x);
}
/*
###Output###
Before Calling x=10
After Calling x=10
*/
```

In the above example we can see that direct value is passed at the time of calling.
Here x is actual parameter and m is formal parameter.

Call by Reference

1.In this type of calling a function ,the reference of the value is passed at the time of calling.
2.In call by value the changes made in formal parameters reflect in actual parameters.
3.Reference is also called address.
4.When the address of data is passed at the time of calling so it is necessary to use pointer in the place of parameter.
For better understanding see the example below-

```
#include<stdio.h>
 void add(int *m)//function definition
```

```
{
        //adding 10 to m
        *m=*m+10;
}
int main()
{
        int x=10;
        printf("Before Calling x=%d\n",x);
        add(&x);
        printf("After Calling x=%d",x);
}
/*
###Output###
Before Calling x=10
After Calling x=20
*/
```

To know the difference between call by value and call by reference focus on the output of the program.

Function with default value

In this type of function ,the function contains a number of parameter with some initial value[for example: void sum(int x=10,int y=20)].At the time of calling if there is no value is passed [for example: sum();] then the default value will be x=10 and y=20,but if value is passed [for example: sum(5,6);] then the value will be x=5 and y=6.

For better understanding see the example below-

```
#include<stdio.h>
void sum(int x=10,int y=20)//function definition
{
        int result=x + y;
        printf("Add=%d\n",result);
}
int main()
{
        int x=10,y=20;
        printf("Without value\n");
```

```
        sum();//function calling without value
        printf("With value\n");
        sum(5,6);//function calling with value
}
/*
### Output ###
Without value
Add=30
With value
Add=11
*/
```

Passing Array to Function

In this type of function ,there is an array in the place of parameter [for example:void sum(int ar[5])] and its value is passed at the time of calling.

```
#include<stdio.h>
void sum(int ar[5])//function definition
{
        int s=0;
        for(int i=0;i<5;i++)
        //finding the sum
        s=s+ar[i];
        printf("Total sum of element=%d",s);
}
int main()
{
        int x[5]={10,20,50,40,60};
        sum(x);//function calling with array
}
/*
### Output ###
Total sum of element=180
*/
```

In the above example we can see that there is an array ar[5] in place of parameter and there is an other array x[5]={10,20,50,40,60} and it is

passed at the time of calling therefor the value of array x will be copied into array ar.

Recursion

The process of calling a function by itself is called Recursion and the function that calls itself is called Recursive Function.

Factorial of any Number using recursion

Factorial of 5=5*4*3*2*1

```
#include<stdio.h>
void factorial(int no,int f)//function definition
{
        if(no>=1)
        {
         f=f*no;
         no--;
         factorial(no,f);
        }
        else
        printf("Factorial =%d",f);
}
int main()
{
        int n;
        printf("Enter any number to find factorial\n");
        scanf("%d",&n);
        factorial(n,1);//function calling with array
}
/*
### Output ###
Enter any number to find factorial
6
Factorial =720
*/
```

String function in C Language

1.String is a collection of character.
2.C does not support string data type therefore char data type is used to make string .
3.String in c is stored in single dimension character array.
4.There are many predefined string function in C library.
5.All the string functions are predefined in string.h header file.
👉String Function

Function name	Description
strlen(s);	It is used to get the length of string.
strcpy(s1,s2);	It is used to copy the string here string s2 is copied into string s1.
strcat(s1,s2);	It is used to add two string here string s2 concatenates at the end of string s1.
strcmp(s1,s2);	1.It is used to compare two string. 2.This function returns 0 if string s1 and string s2 are same. 3.This function returns negative value if s1<s2. 4.This function returns positive value if s1 >s2 .
strrev(s);	It is used to reverse the string.
strupr(s);	It is used to convert all the character into upper alphabet.
strlwr(s);	It is used to convert all the character into lower alphabet.
strcmpi(s1,s2);	It is also used to compare two string but it ignores the case sensitivity means 'a' and 'A' are treated as same.

👉strlen()
It is used to find the length of the Specified string.

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
```

```
int main()
{
char name[200]="Easy";
printf("%d",strlen(name));
return 0;
}
```

Output
4
//because there is 4 character in Easy

👉strcpy()
It is used to copy one string into another.
It has two parameters.

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
int main()
{
char name1[200]="Radha";
char name2[200]="Rani";
strcpy(name1,name2);
printf("%s",name1);
return 0;
}
```

Output
Rani

👉strcmp()
It is used to compare one string with another.
It returns zero if both strings are same otherwise returns non-zero.

```
#include<stdio.h>
```

```
#include<conio.h>
#include<string.h>
int main()
{
char s1[200]="Easy";
char s2[200]="Easy";
if(strcmp(s1,s2)==0)
  {
  printf("string s1 and string s2 are same.");
  }
  else
  {
  printf("string s1 and string s2 are not same.");
  }
return 0;
}
```

Output
string s1 and string s2 are same

👉strcat()
It is used to concatenate one string with another.

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
int main()
{
char s1[200]="Easy";
char s2[200]=" Softwares";
printf("%s",strcat(s1,s2));
return 0;
}
```

Output

Easy Softwares

👉strrev()
It is used to reverse the string .

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
int main()
{
  char s[200]="ABCD";
  printf("%s",strrev(s));
  return 0;
}
```

Output
DCBA

👉strupr()
It is used to convert only lowercase alphabet into uppercase.

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
int main()
{
  char s[200]="Easy";
  printf("%s",strupr(s));
  return 0;
}
```

Output
EASY

👉strlwr()
It is used to convert only uppercase alphabet into lowercase.

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
int main()
{
  char s[200]="Easy";
  printf("%s",strlwr(s));
  return 0;
}
```

Output
easy

Math function in C Language

1.It is used to perform the mathematical related operation.
2.There are many predefined math function in C library.
3.All the math functions are predefined in math.h header file.

👉Math Function

Function name	Description
sin()	It is used to calculate the sine value.
sinh()	It is used to calculate hyperbolic sine value.
cos()	It is used to calculate the cosine value.
cosh()	It is used to calculate hyperbolic cosine value.
tan()	It is used to calculate the tangent value.
tanh()	It is also used to calculate the hyperbolic tangent value.
sqrt()	It is used to calculate square root of value passed to the function.

pow()	It is used to calculate the power of a number pow(base,power).
exp()	It is used to calculate the exponential of the value passed to the function.
log()	It is used to calculate natural logrithm.
log10()	It is used to calculate base-10 logrithm.
floor()	It always returns minimum round off value.
ceil()	It always returns maximum round off value.
round()	It returns maximum round off value if the input value is greater than or equal to center value otherwise minimum round off value.
abs()	It always returns positive value.
fmod()	It is used to convert all the character into lower alphabet.

👉Example

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
int main()
{
float a=2;
printf("sin(2)=%f\n",sin(a));
printf("cos(2)=%f\n",cos(a));
printf("tan(2)=%f\n",tan(a));
printf("exp(2)=%f\n",exp(a));//exponential
printf("log(2)=%f\n",log(a));//natural log
printf("log10(2)=%f\n",log10(a));//log10
printf("sqrt(4)=%f\n",sqrt(4));//square root
printf("cbrt(27)=%f\n",cbrt(27));//cube root
return 0;
```

```
}
### Output ###
sin(2)=0.909
cos(2)=-0.416
tan(2)=-2.185
exp(2)=7.389
log(2)=0.693
log10(2)=0.301
sqrt(4)=2
cbrt(27)=3
```

👉floor() function

It returns the minimum round off value.

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
int main()
{
printf("%f\n",floor(2.3));
printf("%f\n",floor(2.5));
printf("%f\n",floor(2.8));
return 0;
}
### Output ###
2.0
2.0
2.0
```

👉ceil() function

It returns the maximum round off value.

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
```

```
int main()
{
printf("%f\n",ceil(2.3));
printf("%f\n",ceil(2.5));
printf("%f\n",ceil(2.8));
return 0;
}
### Output ###
3.0
3.0
3.0
```

👉round() function

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
int main()
{
printf("%f\n",round(2.3));
printf("%f\n",round(2.5));
printf("%f\n",round(2.8));
return 0;
}
### Output ###
2.0
3.0
3.0
```

Structure in C Language

1.It is a collection of data of different data type.
2.It is a user defined data type.
3.Data can of int, char, float, double etc. data type.
4.We can access the member of structure by making the variable of structure.
5.struct keyword is used to create a structure.
👉Syntax
👉Example 1

```
struct student
{
 char name[200];
 int rollno;
 float marks;
};
```

1.Here student is the name of structure .
2.struct is a keyword.

👉Declaration of structure variable method 1

```
struct student
{
 char name[200];
 int rollno;
 float marks;
};
int main()
{
struct student student1;
return 0;
}
```

Here student1 is the variable of structure.

👉Declaration of structure variable method 2

```
struct student
{
 char name[200];
 int rollno;
 float marks;
}student1;
int main()
{
return 0;
}
```

Here student1 is the variable of structure.

👉Accessing the data members of structure The data member of structure can be accessed as structure_variable.data_member For example if we want to access the rollno of student then we can write as student1.rollno

👉Example: Write a program to store and display student information

```
#include<stdio.h>
#include<string.h>
struct student
{
 char name[200];
 int rollno;
 float marks;
};
int main()
```

```
{
struct student student1;//Declaring structure variable
strcpy(student1.name,"Lucky");
student1.rollno=201;
student1.marks=85.9;
printf("Student Name =%s\n",student1.name);
printf("Student Rollno=%d\n",student1.rollno);
printf("Student Marks=%f\n",student1.marks);
}
/*
### Output ###
Student Name=Lucky
Student Rollno=201
Student Marks=85.9
*/
```

👉Example: Method 2

```
#include<stdio.h>
#include<string.h>
struct student
{
 char name[200];
 int rollno;
 float marks;
};
int main()
{
struct student student1={"Lucky",201,85.9};
printf("Student Name =%s\n",student1.name);
printf("Student Rollno=%d\n",student1.rollno);
printf("Student Marks=%f\n",student1.marks);
}
```

```
/*
### Output ###
Student Name =Lucky
Student Rollno=201
Student Marks=85.900002
*/
```

👉User input with structure

```
#include<stdio.h>
#include<string.h>
//structure declaration
struct student
{
 char name[200];
 int rollno;
 float marks;
};
int main()
{
//creating variable of student
struct student student1;
printf("Enter name of student:");
scanf("%s",&student1.name);
printf("Enter rollno of student:");
scanf("%d",&student1.rollno);
printf("Enter marks of student:");
scanf("%f",&student1.marks);
printf("Student Name =%s\n",student1.name);
printf("Student Rollno=%d\n",student1.rollno);
printf("Student Marks=%f\n",student1.marks);
}
/*
```

```
### Output ###
Enter name of student:Rocky
Enter rollno of student:204
Enter marks of student:85.6
Student Name =Rocky
Student Rollno=204
Student Marks=85.599998
*/
```

👉Pointer with structure

```
#include<stdio.h>
#include<string.h>
//structure declaration
struct student
{
 int rollno;
};
int main()
{
//creating normal variable of student
struct student student1;
student1.rollno=204;
//creating pointer variable of student
struct student *ptr;
//passing normal variable address into pointer variable
ptr=&student1;
printf("Student Rollno=%d\n",*ptr);
}
/*
### Output ###
Student Rollno=204
*/
```

Union in C Language

1.It is a collection of data of different data type.
2.It is a user defined data type.
3.Data can be of int, char, float, double etc. data type.
4.We can access the member of union by making the variable of union.
5.union keyword is used to create a union.
6.The difference between structure and union is that structure can store multiple value simultaneously but union can store only one value at a time.
7.The region is that in structure each member of structure has its own memory space but in union a single memory space is shared by all members of union.
👉Syntax
👉Example

```
union student
{
 char name[200];
 int rollno;
 float marks;
};
```

1.Here student is the name of union.
2.union is a keyword.

👉Declaration of union variable method 1

```
union student
{
 char name[200];
 int rollno;
```

```
 float marks;
};
int main()
{
union student student1;
return 0;
}
```

Here student1 is the variable of union.

👉Declaration of structure variable method 2

```
union student
{
 char name[200];
 int rollno;
 float marks;
}student1;
int main()
{
return 0;
}
```

Here student1 is the variable of union.

👉Accessing the data members of structure The data member of structure can be accessed as union_variable.data_member For example if we want to access the rollno of student then we can write as student1.rollno
👉Example: Write a program to store and display student information

```
#include<stdio.h>
#include<string.h>
union student
```

```
{
 char name[200];
 int rollno;
 float marks;
};
int main()
{
union student student1;//creating union variable
strcpy(student1.name,"Lucky");
student1.rollno=201;
student1.marks=85.9;
printf("Student Name =%s\n",student1.name);
printf("Student Rollno=%d\n",student1.rollno);
printf("Student Marks=%f\n",student1.marks);
}
/*
### Output ###
Student Name=garbage value
Student Rollno=garbage value
Student Marks=85.900002
*/
```

Enumeration in C Language

1.It is a collection of named integer constant.
2.It is a user defined data type.
3.enum keyword is used to create a enumeration.
4.Default value of enum's elements are starts with 0 for example suppose that there are four elements in enum like sunday , monday , tuesday and wednesday then here value of sunday is 0, monday is 1, tuesday is 2 and wednesday is 3.
5.But we can also change the default value of enum's element which we discuss later in this post.

👉Syntax

👉Example

enum week {sunday,monday,tuesday,wednesday,thrusday,friday,saturday};
1.enum is a keyword.
2.week is the name of enum and it is a user defined data type.
3.sunday,monday,tuesday,wednesday,thrusday,friday,saturday are the values of enum.
Default value of enum's member
Default value of sunday is 0
Default value of monday is 1
Default value of tuesday is 2
Default value of wednesday is 3
Default value of thrusday is 4
Default value of friday is 5
Default value of saturday is 6
For better understanding see the below example.

```
#include<stdio.h>
enum week {sunday,monday,tuesday,wednesday,thrusday,friday,saturday};
int main()
{
enum week obj;
obj=wednesday;
printf("Value of wednesday=%d",obj);
}
/*
### Output ###
Value of wednesday=3
*/
```

👉We can also change the default value of member of enum. See the example below

File Handling in C Language

In this type of programming we can perform different types of operation on file and operation result is stored in the file permanently.
👉Mainly two types of on which we perform operation
1.Text File
2.Binary File

👉Text File
1.This type of file can be created using Notepad or other simple editors.
2.The extension of text file is (.txt).
3.This file contains simple text.

👉Binary File
1.The extension of binary file is (.bin).
2.This file contains information in the form of binary code (1 or 0) and it is not readable.

👉Operation on File
1.Opening of file
2.Closing of file
3.Reading from file
4.Writing into file
5.Appending data into file

👉Mode of Operation

File Mode	Description
r	It opens an existing text file for reading purpose. If the file does not exist, fopen() returns NULL.
r+	It opens an existing text file for reading and writing purpose. If the file does not exist, fopen() returns NULL.
w	It opens a text file for writing purpose. If it does not exist, then a new file is created. If the file exists, its contents are overwritten.
w+	It opens a text file for writing and reading purpose. If it does not exist, then a new file is created. If the file exists, its contents are overwritten.
a	It is used to open a text file in writing appending mode. If the file does not exist, then a new file is created.
a+	It is used to open a text file in writing and reading appending mode. If the file does not exist, then a new file is created.

👉Some important function

Function Name	Description
fopen	1.It is used to open a text file on given path. 2.It has two parameter path(where file is created) and mode of operation(Ex- r for reading or w for writing).
fscanf	1.It is used to scan the text file. 2.It acts like an scanner ,it reads a set of data from file.
fprintf	1.printf is used to printf information or data on output screen. 2.fprintf is used to print information or data into the file. 3.It has also two parameter file and message which is to be printed.
fclose	It is a predefined function which is used to close the file.

getc	It is used to get a character from file.
putc	It is used to write a character to a file.

👉File Writing
Note->Before the execution of this program go to C Drive and create a temp folder if it does not exist.

File writing is the process of writing something into a file than can stored permanently.

```
#include<stdio.h>
int main()
{
FILE *fp;
fp=fopen("/temp/demo.txt","w");
fprintf(fp,"Hello friends, how are you???.");
fclose(fp);
}
/*
### Output ###
After compiling and executing
this program, go to c drive ,
open temp folder you will
see a text file(demo.txt)
now open it ,the content
of this file is
Hello friends, how are you???.
*/
```

👉File Reading

File reading is the process in which we can read the content of a file.

Graphics in C Language

If you want to create a colorful and attractive project in C language then you can choose graphics because graphics in c allows us to create different types of geometrical shapes like circle, rectangle, line, square, arc etc and we can also apply color on these shapes.

In this type of programming we can draw different kinds of geometrical shape for example circle, rectangle, square, line etc in different color.

👉Initialization of graph

void initgraph(int *Graphics_driver, int *Graphics_mode, char *Driver_directory_path);

1.initgraph is a predefined function which is used to initialize the graph. It has three parameters. It loads the passed graphics driver then changes the system into graphics mode.

2.Graphics_driver specifies the graphics driver. It is set to DETECT.

3.Graphics_mode specifies the graphics mode.

4.Driver_directory_path specifies the directory where all the graphics driver files are located.

👉Draw a Circle

Syntax->circle(from-left,from-top,radius)

Here from-left is the distance in pixel from the left of screen similarly from-top is the distance in pixel from the top of the screen.

```
#include<stdio.h>
#include<graphics.h>
#include<conio.h>
int main() {
        int gd = DETECT, gm;
        initgraph(&gd, &gm, "C:\\tc\\bgi"); //initialization of graphic mode
        circle(150,200,50);
```

```
        getch();
        closegraph();//closing of graphic mode
        return 0;
}
```

👉Draw a Line

Syntax->line(min-x,min-y,max-x,max-y)

```
#include<stdio.h>
#include<graphics.h>
#include<conio.h>
int main() {
   int gd = DETECT, gm;
        initgraph(&gd, &gm, "C:\\tc\\bgi"); //initialization of graphic mode
        line(150,150,250,250);
        getch();
        closegraph();//closing of graphic mode
        return 0;
}
```

if you want to create a horizontal line then min-y and max-y should be of same length.
for example line(150,100,250,100); you can try this you will get horizontal line as an output.

if you want to create a vertical line then min-x and max-x should be of same length.

for example line(100,250100,400); you can try this you will get vertical line as an output.

👉Draw a Rectangle

Syntax->rectangle(min-x,min-y,max-x,max-y)

```
#include<stdio.h>
#include<graphics.h>
#include<conio.h>
int main() {
	int gd = DETECT, gm;
	initgraph(&gd, &gm, "C:\\tc\\bgi"); //initialization of graphic mode
	rectangle(150,150,350,250);
	getch();
	closegraph();//closing of graphic mode
	return 0;
}
```

Dynamic Memory Allocation in C Language

Dynamic Memory Allocation

- It is used to allow the allocation of memory at runtime.
- There are four function are used to achieve dynamic memory allocation.
- malloc()
- calloc()
- realloc()
- free()

malloc()

- It stands for Memory Allocation.
- It allocates/reserves a single block of memory of specified size.
- It returns NULL if memory is insufficient.
- malloc function returns the address of first byte in the allocated memory if memory is sufficient.
- Syntax

ptr=(cast-type*)malloc(byte-size)

- Here ptr is the pointer, cast-type is the data type in which we want to cast the returning pointer,byte-size is the allocated memory space.
- Example

p = (int*) malloc(50 * sizeof(int));

- Here in the above example the given statements allocates 100 or 200 bytes of memory.If the size of int is 2 then 50*2=100 bytes will be allocated,if the size of int is 4 then 50*4=200 bytes will be allocated.Pointer p holds the address of first byte of allocated memory.

calloc()

- It stands for contiguous allocation.
- It allocates/reserves a multiple block of memory of same size.
- The allocated memory space is initialized to zero.
- It returns NULL if memory is insuffient.
- Syntax

ptr=(cast-type*)calloc(number of block,size of each block)

- Here ptr is the pointer,cast-type is the data type in which we want to cast the returning pointer.
- Example

p = (float*) calloc(50 , sizeof(float));

- Here in the above example the given statements allocates 50 block of memory space with size of 4 byte because size of float is 4 byte.

realloc()

- If the allocated memory space is insufficient then it is possible to modify the allocated space.
- realloc function is used to modify the size of previously allocated memory.
- Syntax

ptr = realloc(ptr, new size);

free()

· This function is used to free the allocated memory space.

· Syntax

free(ptr);

Example:malloc

Find sum of given value

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
        //declaring variable
        float sum=0,*p;
        int no;
        printf("How many elements you want to add\n");
        scanf("%d", &no);//taking input
        //allocating memory space
        p= (float*) malloc(no * sizeof(float));
        if(p == NULL)
        {
        printf("Insufficient space,memory not allocated.");
        exit(0);
        }
        //taking inputs of element
        for(int i = 0; i < no; ++i)
        {
        printf("Enter element %d\n",i+1);
        scanf("%f", p + i);
          //finding the sum of elements
        sum =sum+ *(p + i);
        }
        //Printing the sum of element
        printf("Total Sum of element= %f", sum);
        //free the allocated space
        free(p);
```

```
        return 0;
}
/*
### Output ###
How many elements you want to add
4
Enter element 1
3.4
Enter element 2
9.3
Enter element 3
9.2
Enter element 4
4.3
Total Sum of element= 26.200001
*/
Example:calloc
same program with calloc

#include <stdio.h>
#include <stdlib.h>
int main()
{
        //declaring variable
        float sum=0,*p;
        int no;
        printf("How many elements you want to add\n");
        scanf("%d", &no);//taking input
        //allocating memory space
        p= (float*) calloc(no,sizeof(float));
        if(p == NULL)
        {
        printf("Insufficient space,memory not allocated.");
        exit(0);
        }
```

```
        //taking inputs of element
        for(int i = 0; i < no; ++i)
        {
        printf("Enter element %d\n",i+1);
        scanf("%f", p + i);
        //finding the sum of elements
        sum =sum+ *(p + i);
        }
        //Printing the sum of element
        printf("Total Sum of element= %f", sum);
        //free the allocated space
        free(p);
        return 0;
}
/*
### Output ###
How many elements you want to add
4
Enter element 1
3.4
Enter element 2
9.3
Enter element 3
9.2
Enter element 4
4.3
Total Sum of element= 26.200001
*/
```