C Programming Language

History of c:

- **1960- FORTRAN** (formula translator) and **COBOL**(common object business oriented language) was developed but for specific use and purpose.
- **1960- ALGOL**(algorithmic oriented language) was developed at Cambridge university which was the combination of all the programming language developed at that time but the problem was too big to learn and had many features.
- **1963-CPL**(combined programming language) was developed at Cambridge University which was the combination of all the programming language developed at that time but the problem was too big to learn and had many features.
- **1967- BCPL** was developed by Martin Richards at Cambridge University but was less powerful and less specific.
- **1970- B** was developed by Ken Thompson at AT & T Bells labs but has the same problem as the above programming languages.
- **1972- C** was developed by Danish Ritchie at AT & T Bells lab by combining all the good features of CPL, BCPL and B. It was machine independent, easy to learn and implement by the user of C a very powerful language called as UNIX was developed.

CHARACTERISTICS OF 'C' LANGUAGE

- **Modularity:** Ability to breakdown a large module into manageable sub modules called as modularity that is an important feature of structured programming languages.
- **Portability:** The ability to import i.e. to install the software in different platform is called portability. The software that is 100% portable is also called as platform independent software or architecture neutral software.
- **Extendibility:** Ability to extend the existing software by adding new features is called an extendibility.
- **Speed:** 'C' is also called as middle level language because programs written in 'c' language run at the speeds matching to that of the same programs written in assembly language so 'c' language has both the merits of high level and middle level language and because if this feature it is mainly used in developing system software.

- **Flexibility:** Key words are reverse words. ANSIC has 32 reverse words in 'c' language has right number of reverse words which allows the programmers to have complete control on the language. 'C' is also called as programmer's language since it allows programmers to induce creativeness into the programmers.

Advantages of programming in C:

- Easy to understand.
- Freedom of using different type of data.
- Short listed words could be use.
- Efficient and fast programming.
- It can be used as mid-level language.
- Any type of software and operating system be developed with the help of C language.

Disadvantages of programming in C:

- OOP(object oriented programming) is not possible in C.
- Multiple inheritance is not possible in C.
- System setting can be easily changed by the use of pointer.
- It is hard to learn for beginners.

Basic idea about C:

As we learn by any language, we should first learn about the basic of that language. For e.g. if we are going to learn English language then we should first learn about the alphabets of English (i.e. a,b,c………..x,y,z). After learning these alphabets we combine these alphabets to form words. Later we learn to combine these words to form sentences and these sentences combine to form a paragraph. It's same in C programming also. In c programming different character set (alphabets, digits, special symbols, spaces) combine to form constant, variable and keywords, these combine to form instructions and these instructions finally combine to form a program.

$$\begin{tikzcd}
\boxed{\text{Alphabets, digits, special symbol}} \arrow[r] & \boxed{\text{Constants, variables keywords}} \arrow[r] & \boxed{\text{Instructions}} \arrow[r] & \boxed{\text{Program}}
\end{tikzcd}$$

What do you mean by character sets in c programming?
Ans: Character sets in c programming is the combination of various elements i.e. alphabets (lower case and upper case) digits, special symbols and spaces valid characters. Various types of character sets are listed below.

a) Alphabet
   i. Uppercase (A-Z)
   ii. Lower case (a- z)
b) Digit (0-9)
c) Special symbols (+,-,/,%,*,@,;,!,^,||)
d) While space character : (Blank space, Newline, TAB line)

Basic terms used in C:

Keywords :

Keywords are reserve words in c programming that perform specific task. They have standard predefined meaning in C. keywords are always written in lowercase and should consist of alphabets only. There should not be blank space in the keywords (e.g. au to). There are 32 keywords in 'C'.

Auto	Double	If	Static	break	Else	Int	Struct
Case	Enum	Long	Switch	Char	Extern	Near	Typedef
Const	Float	Register	Union	Continue	For	Return	Default
For	Short	void	Do	Goto	signed	While	unsigned

Identifiers:

Identifiers are the fundamental requirement of any language. They are the user defined words which are used for giving names to entities like array, functions, structure, union etc and are used to give unique names to the objects in the program. An identifier name can be in lowercase letters or uppercase letters and the identifier name should start with the alphabet or underscore and should not have blank space in them. One important thing we should remember is that keywords cannot be used as identifiers. E.g. length, student, teacher etc.

Variables:

Variables are the entities whose value changes throughout program execution. Variables are simply names used to refer to some location in memory-allocation that holds a value with which we are working. Each and every variable must be declared before its value. Declaring variables is the way in which a C program shows the number of variables it needs, what they are going to be named, and how much memory they will need. E.g.: int length, int breadth, float height, float area. Variable names in C are made up of letters (upper and lower case) and digits. The underscore character ("-") is also permitted. Names must not begin with a digit. Variables name should start with alphabet or underscore but not with the digit. There should not be blank space within the variable (eg.

Le ngth).

Constants:

Constants are also called as literals are actual representation of fixed values in the program i.e. its value does not change during program execution. There are different types of constants in 'C' which are as follows:

a) Numeric constants
   i. Decimal constants (0-9)
   ii. Hexadecimal constant (0-9,A,B,C,D,E,F)
   iii. Octal constants (0-7)
   iv. Floating point constant
b) Character constant
c) String constant

a) Numeric constant

Numeric constants are fixed numeric representation that can be positive or negative. There are four types of numeric constants which are defined below:

- **Decimal Constants:**

Decimal constants does not contain the fractional part or does not contain the decimal point (.). An integer constant must have at least one digit, can be +ve or -ve and should not have blank spaces or commas. Default sign of decimal constant is +ve. The decimal numbers represented with base-10. It consists of digits 0-9 eg:123,45,65,93.

- **Hexadecimal Constants:**

  Hexadecimal numeric constant consists of digits 1,2,3,4,5,6,7,8,9,A,B,C,D,E,F i.e. 10 digits and 6 symbols. Its base is 16. E.g. $(abc)_{16}$, $(92F)_{16}$. A hexadecimal constant is declared by prefixing the constant with either "0x" or "0X".

  ```
  Int value3 = 0x1234;
  Int value3 = 0X1234;
  ```

- **Octal Constant:**

In octal number system there are 8-digit i.e. 0,1,2,3,4,5,6,7 and its base is 8. An octal constant is declared by prefixing the constant with a "0". Here is an example that declares an octal constant and assigns it to a variable. A common error is to prefix constant with a zero when the constant is intended to be a decimal number.

```
Int value2= 01234;
```

Let's look simple program to better understand all of this.

```
#include <stdio.h>
Int main()
{
Int value1=1234;      /*Decimal*/
Int value2=01234;     /*Octal*/
Int value3=0x1234;  /*Hexadecimal*/

Printf("value1= %d\n",value1);
Printf("value2= %d\n",value2);
Printf("value3= %d\n",value3);
```

```
Return0;
}
```

Output:
Valule1=1234
Value2=668
Value3=4660</output>

- **Floating Point Constants**
  Floating point constant consists of fractional part or contains the decimal point(.) and can be +ve or -ve. No commas or blank spaces are allowed with in floating point constant. If there is only 0 in fractional part than we can omit it. E.g.: 325.75.372.52.

b) Character Constants:

A "character constant" is formed by enclosing a single character from the represent able character set within single quotation marks (' '). Character constant can have integer values. Character constants are used to represent characters in the execution character set. There must be at least one character and should be enclosed with in single quotation to be a valid character constant. Eg.'a','x','c' .

c) String Constants:

A "string literal" is a sequence of characters from the source character set enclosed in double quotation marks (""). String literals are used to represent a sequence of character which taken together, form a null-terminated string. It's double quote or single quotes are part of string then it should be preceded by backslash(\).

<u>Operators in c programming:</u>

C programming language provides several operators to perform different kind of operations. There are operators for assignment, arithmetic functions, logical functions and many more. There operators generally work on many types of variables or constants, though some are restricted to work on certain types. Most operators are binary, meaning they take two operands. A few are unary and only take one operand.

E.g. **x+y=z**

Here x,y,z are the operands where as + and = are operators. These operators act upon the operand to perform any calculation.

Different type of operators are used in c ie.

1) Arithmetical operators
2) Relational operators
3) Logical operators
4) Bitwise operators
5) Assignment operators
6) Unary operators (increment/decrement operator)
7) Conditional operator(ternary operator)

<u>Arithmetic operators</u>

Arithmetic operators include the familiar addition (+), subtraction(-), multiplication(*) and division(/) operations. In addition there is the modulus operator(%) which gives the remainder left over from a division operation.

S.N	a=10, b=6			
	Symbols	**Meaning**	**Operator**	**Result**
1.	+	Addition	a+b	16
2.	-	Subtraction	a-b	4
3.	*	Multiplication	a*b	60
4.	/	Division	a/b	1
5.	%	Remainder	a%b	4

<u>Relational operators:</u>

Relational operators compare operands and return 1 for true or 0 for false. The following relational operators are available:

S.N	**Symbols**	**Meaning**	**expression**
1.	>	Greater than	a>b

2.	<	Less than	a<b
3.	>=	Greater than or equal to	a>=b
4.	<=	Less than or equal to	a<=b
5.	==	Equal to	a==b
6.	!=	Not equal to	a!=b

Logical operators

A logical operator combines one or two conditions into a single new condition. These operators return 0 for false and 1 for true. C provides three logical operators:

Operator	**Meaning**	**Example**
"&&"(two ampersands)	Logical AND	(x>y)&&(x>z)
"|| " (two vertical bars)	Logical OR	(x>y)||(x>z)
"!" (an exclamation)	Logical NOT	!(x==y)

- **AND operator:** if both the conditions to the left and right are true then only the whole compound statement is true.
- **OR operator:** Here the condition is evaluated true if any one of the given two conditions are true.
- **NOT operator:** It takes single expression and evaluates to true if the condition is false.

Here NOT operator is an unary operator where as other two are binary operators.

Bitwise operators:

Bitwise operator's only work on a limited number of types: **int** and **char**. Bitwise operators fall into two categories: binary bitwise operators and unary bitwise operators. Binary operators take two arguments, while unary operators only take one. Bitwise operators, like arithmetic operators, do not change the value of arguments. Instead, a temporary value is created. This can them be assighned to a variable.

Name of operator	**Symbol**

Bitwise AND	&
Bitwise OR	|
Bitwise XOR	^
Bitwise NOT	~

<u>Unary operator :</u>

Unary operators are very useful in c programming and are used in for, while and do…..while loops. There are two types of unary operator that is increment and decrement operators. There two unary operators can again be classified on the basis of prefix and postfix operators.

Symbol	**Meaning**
++x	Prefix increment
x++	Postfix increment
--x	Prefix increment
x--	Postfix increment

The increment operator adds a value by 1 to the current value of the operand and decrement operator decrements the value by 1 to the current value of the operand.

Program to understand the use of prefix increment/decrement

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

Void main()

{

Int x=8

Printf("x=%d\t",x);

Printf("x=%d\t",++xprintf("x=%f",x);

Printf("x=%d"\t,--x);
```

```
 Printf("x=%d\n",x);                                    output:
 Getch();                                               x=8  x=9  x=9
 }                                                      x=8  x=8
```

Program to understand the use of postfix increment/decrement

```
 #include(stdio.h)
#include(conio.h)
void main()
 {
 Clrscr();
int x=8;
 Printf("x=%d\n",x);
 printf("x=%d\t",x++); /*postfix increment*/
 Printf("x=%d\t"x--);  /*postfix decrement*/
 Printf("x=%d\n");
getch();                                          output:
 }                                                     x=8
                                                       x=8   x=9   x=9  x=8
```

Conditional or ternary operator:

Conditional operator is a ternary operator which requires three expressions. It has two symbols question mark (?) and colon (;).

Syntax:

Condition ? exp1: exp2

If the test condition is true exp1 is evaluated if the test condition is false exp2 is evaluated.

E.g a>b ? a:b

Data types:

C supports different types of data types and is defined as the set of values that a variable can store along with the set of operations that can be performed on that variable. They are mainly used to define the type and nature of data such that compiler detects and proceed. Each and variable and constant that we are using in the program must be a declared before they are used. Mainly there are four types of data types i.e. char, int, float, and double.

S.N	**Data types**	**Memory Requirement**
1	Char	1 bytes
2	Int	2 bytes
3	float	4 bytes
4	double	8 bytes

With this four basic data types of c supports more data types which can be extended by using type qualifiers long, signed, unsigned. For example unsigned int, signed int, long int etc.

Escape Sequences:

Escape are the non printing characters that start with the back slash(\) which are used in output function. They contain two characters but represent one character only. There are various types of escape sequences used in C programming which are as follows:

Escape sequences	**Meaning**
\b	Back space
\a	Bell
\n	New line
\t	Horizontal tab

\'	Single quote
\"	Double quote
\?	Question mark
\v	Vertical tab

Format Specifier:

When the data is being inputted or outputted it must be notified with some identifiers and their format specifiers which consists of % sign and followed by a character. Format specifier specifies the type of data being processed.

Data type	Format specifier symbol
Integer	%d
Unsigned integer	%u
Octal	%o
Hexadecimal	%x
Float	%f
Float (scientific)	%e
String	%c
Character	%s

Header File:

Header file are the standard file of C programming which contain the definitions of library functions. Each and every library function that are used in the C program are defined in the header file. So for different library function there are different header files. All the header files have the same syntax i.e. enclosed with in triangle bracket (<………..>) and has the extension .h. The entire header file must be written in small case letters.

Header file	**Contains of definition of**
#include<stdio.h>	Input/output functions: scanf(), printf()
#include<conio.h>	Console input/output functions: clrscr(), getch()
#include<math.h>	Mathematical functions: Pow(), sqrt(), sin(),tan(), cos()
#include<graphics.h>	Graphic processing functions: line(), circle(), rectangle()
#include<string.h>	Strin processing function: strlen(),strcat(),strcpy()

Preprocessor directives:

It is the unique feature of c programming which makes program easy to read, modify, portable and efficient. It is the command that processes the code before it passes through the compiler and operates under the command line and directive. Preprocessor directive are placed in the source program before the main line.

Preprocessor directive	**Function**
#define	Define a macro substitution and symbolic constant
#include	Species the file to be included in the program

INPUT/OUTPUT FUNCTION

Input function:

Input data can be entered into the memory form a standard input device known as keyboard. C provides the scanf() library function for entering input data. These functions can take all types of value such as number, character, and string as input. The scanf() function can be written as:

Scanf ("control statement",address1, address2,……….);

The function should have at list two parameters first perimeter is a control string and is written with in double quote. The other parameter are address or variable in the scanf() function they must be at list one variable or add them. The address of the variable is preceded by and ampersand (&).

Eg: scanf("length=%d",&1);

Write a program to find the addition, subtraction, multiplication and division.

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

{

Float a, b, c, d, e, f ;
Clrscr();
```

Output:

```
Enter the value of a:8
Enter the value of b:6
The sum of a and b is:14.000000
The subtraction of a and b is:2.000000
The multiplication of and b is :48.00000
The division of a and b is:1.333333
```

```
Printf("enter the value of a");
Scanf("%f",&a);
Printf("enter the value of b");
Scanf("%f",&b);
c=a+b;
d=a-b;
e=a*b;
f=a/b;
printf("the sum of a and b is=%f",c);
printf("the subtraction of a and b is=%f",d);
printf("the multiplication of a and b is =%f",f);
getch();
}
```

Output Function:

Output data can be written from computer memory to standard output monitor device using printf() library functions with these function all types of values such as number, character or string can be written as output. The printf() statement can be written as:

```
Printf("control statement", address1, address2,….);
```

The control string in enclosed with in double quote. There should be at list one address in the printf() statement and is not preceded by ampersand sign(&).

```
Eg. Printf("the sum=%d",total);
```

Write a program to print five names of your friends.

```
#include<stdio.h>
#include<conio.h>
Void main ()
{
Clrscr();
Printf("\ndipika");
Printf("\ntopkala");
Printf("\nalisha");
Printf("\nniraj");
Printf("\nkapur");
```

Output:

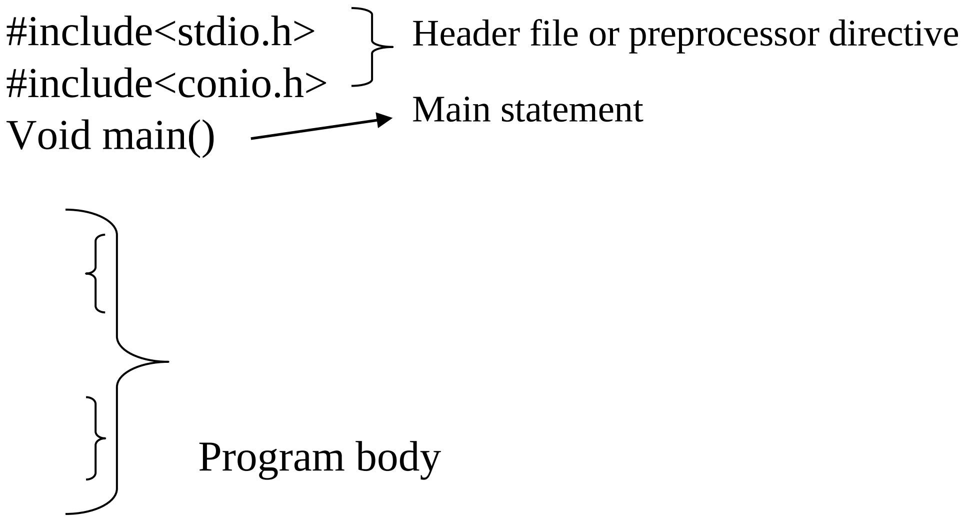
```
Dipika
Topkala
Alisha
Niraj
kapur
```

Here in the above program we have used multiple printf() statements are used, instead of multiple printf() statement we can use single printf() statement to print all those names.

```
#include<stdio.h>
#include<conio.h>
Void main()
{
clrscr();
Printf("Dipika\nTopkala\nAlisha\nNiraj\nKapur");
getch();
}
```

Here in the above program you can also try by using other escape sequences (\t,\b,\v) or by removing the applied escape sequences.

<u>Structure of C program:</u>



```
#include<stdio.h>
#include<conio.h>
Void main()
{
}
```

A general c program starts with header file which is followed by main function. After the main function opening braces is written after which program body starts, and the finally braces are closed.

Here in all these programs variable are assigned a value directly but we can also assign the values to the variables dynamically at run time. To do so scanf() input statement is used in the program.

Write a program to find the sum of two integer number.

```
#include<stdio.h>
#include<conio.h>
Void main()
{
Int a=5;
Int b=4;
Int c;
Clrscr();
C=a+b;
Printf(" the sum of a and b is:%d",c);
getch();
}                              output
```

output

```
The sum of a and b is: 9
```

Write a program to find the differences of two integer number.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a=10;
int b=2;
int c;
clrscr();
c=a-b;
printf("the subtraction of a and b is:%d",c);
getch();
}
```

output

```
The subtraction of a and b is: 8
```

Write a program to find the product of two integer number.

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

```
{
int a=8;
int b=8;
int c;
clrscr()
c=a*b;
printf("the product of a and b is:%d",c);
getch()
}
```

<u>output</u>

The product of a and b is: 64

Write a program to find the quotient of two integer number.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a=8;
int b=2;
int c;
clrscr();
c=a/b;
printf("the quotient of a and b is:%d",c);
getch();
}
```

<u>Output</u>

The quotient of a and b is:4

Write a program to find the sum of two floating point number.

```
#include<stdio.h>
#include<conio.h>
void main()
{
float a=8.5;
float b=9.6;
float c;
clrscr();
c=a+b;
printf("the sum of a and b is:%f",c);
getch();
}
```

output

The sum of a and b is:18.1000000

Write a program to find the difference of two floating point number.

```
#include<stdio.h>
#include<conio.h>
void main();
{
float a=8.3;
float b=1.6;
float c;
clrscr();
c=a-b;
printf("the difference of a and b is:%f ",c);
getch();
}
```

Write a program to find the product of floating point number.

```
#include<stdio.h>
#include<conio.h>
void main()
{
float a=9.6;
float b=8.7;
float c;
clrscr();
c=a+b;
printf("the sum of a and b is:%f",c);
getch();
}
```

output

The product of a and b is:83.520004

Write a program to find the division of two float number.

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

```
void main()
{
float a=15.8;
float b=8.7;
float c;
clrscr();
c=a/b;
printf("the division of a and b is:%f",c);
getch();
}                                          output
                                           The division of a and b is:1.816092
```

Note: Now in the above program all the program variables are assigned a value directly but now we will try to input the values to the variable at the run time by using scanf() statement.

Write a program to input two numbers and find the sum, difference, product and quotient of two integer numbers and print them.

```
#include<stdio.h>
#include<conio.h>
Void main()
{
int a, b, c, d, e, f;
clrscr();
pintf("please enter the value of a:");
scanf("%d",&a);
printf("please enter the value of b:");
scanf("%d",&b);
c=a+b;
d=a-b;
e=a*b;
f=a/b;
printf("the sum of a and b is:%d14",c);                    output
printf("the difference of a and b is:%d",d);        enter the value of a:5
printf("the product of a and b is:%d",e);           enter the value of b:9
printf("the quotient of a and b is:%d");            the sum of a and b is:14
getch();                                            the difference of a and b is: -4
}                                            The quotient of a and b is:1
```

Write a program to find the addition, difference, product and quotient of floating point numbers.

```
#include<stdio.h>
#include<conio.h>
void main()
{
float a, b, c, d, e ,f;
clrscr;
printf("enter the value of a:") ;
scanf("%f",&a);
printf("enter the value of b");
scanf("%f",&);                          Output
c=a+b;                                  Enter the value of a:8
d=a-b;                                  Enter the value of b: 6
e=a*b;                                  The sum of a and b is:14.00000
f=a/b;                                  The difference of a and b is:2.000000
printf("the sum of a and b is:%f",c);   The product of a and b is:48.00000
printf("the difference of a and b is:%f",d); The quotient of a and b is:1.333333
printf("the product a and b is:%f",e);
printf("the quotient of a and b is:%f",f);
getch();
}
```

Write a program to find the area of rectangle.

```
#include<stdio.h>
#include<conio.h>
void main()
{
float a, b, l;                  Output
clrscr();                       Enter the length:8
printf("enter the length");     Enter the breadth:7
scanf("%f",&l);                 The area of  rectangle is:56.00000
printf("enter the breadth");
scanf("%f",&b);
```

```
a=l*b;
printf("\nthe area of rectangle is:%f",a);
getch();
```

write a program to find the perimeter of rectangle.

```
#include<stdio.h>
#include<conio.h>
void main()
{
float p,l,b;
clrscr();
printf("enter the length l:");
scanf("%f",&l)
printf("enter the breadth b:");
scanf("%f",&b);
p=2*(l+b);
printf("the perimeter of rectangle is:%f",p);
getch();
}
```

Output

```
Enter the length:8
Enter the breadth:5
The perimeter of rectangle
is:26.0000
```

write a program to find the area of sqare.

```
#include<stdio.h>
#include<conio.h>
void main()
{
float a,l;
clrscr();
printf("enter the length of square l:",)
scanf("%f",&l);
a=l*l;
printf("the area of square is:%f",a);
getch();
}
```

output

```
Enter the length of square:9
The are of square is:81.00000
```

Write a program to find the perimeter of square.

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

```
void main()
{
float p,l;
clrscr();
printf("enter the length of the square l:");
scanf("%f",&l);
p=4*l;
printf("the perimeter of the square is:%f",p);
getch();
}
```

Output

```
Enter the length of the square l:8
The perimeter of the square is:32.0000
```

Write a program to find the area of circle.

```
#include<stdio.h>
#include<conio.h >
void main ()
{
float a,r;
clrscr();
printf("enter the value of radius r:");
scanf("%f",&r);
a=3.142*r*r;
printf("\nthe area of circle is:%f",a);
getch();
}
```

Output

```
Enter the value of radius:8
The area of circle is:32.000000
```

Write a program to find the volume of cuboids.

```
#include<stdio.h>
#include<conio.h>
void main()
{
float l,b,h,v;
clrscr();
printf(" enter the value of length l:",);
scanf("%f",&l);
printf(" enter the value of breadth b:",);
scanf("%f",&b);
printf(" enter the value of height v:",);
scanf("%f",&h);
```

Output

```
Enter the value of length:8
Enter the value of breadth:5
Enter the value of height:2
The volume of cuboid is:80.000000
```

```
v=l*b*h;
printf(" \nthe volume of cuboid is:%f",v);
getch();
}
```

Write a program to find the simple interest.

```
#include<stdio.h>
#include<conio.h>
void main()
{
float i,p,t,r;
clrscr();
printf("enter the vlaue of principal p:");
scanf("%f",&p);
printf("enter the value of time t:");
scanf("%f",&t);
printf("enter the value of rate r:");
scanf("%f",&r);
i=(p*t*r)/100;
printf("the simple interest is:%f",i);
getch();
}
```

Output

```
Enter the value of principal p:50000
Enter the value of time t:2
Enter the value rate r:4
The simple interest is:4000.000000
```

Write a program to find the length of a rectangle.

```
#include<stdio.h>
#include<conio.h>
void main()
{
float l,a,b;
clrscr();
printf(" enter the value of a:")
scanf("%f",&a);
printf("enter the value of b:");
scanf("%f",&b);
l=a/b;
```

Output

```
Enter the value of a:80
Enter the value of b:7
The length of rectangle is:11.428572
```

```
printf(" \n the length of rectangle is:%f",l);
getch();
}
```

Write a program to convert the rupee into paisa and paisa into rupee.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int r,p,a,b;
clrscr();
printf("enter the value of rupee r:");
scanf("%d",&r);
printf("enter the value of paisa a:");
scanf("%d",&a);
p=r*100;
b=a/100;
printf("\n the value of paisa is:%d",p);
printf("\n the value of rupee is:%d",b);
getch();
}
```

Output

```
Enter the value of rupee r:5
Enter the value of paisa a:500
The value of paisa is:500
The value of rupee is:5
```

Control Statement

In 'C' program statement are executed sequentially i.e. in the order in which they appear in the program. But sometimes we may want to use a condition for executing only a part of program. Control statement enables us to specify the order in which various instruction in the program are to be executed. This determined the flow of control. Control statements define how the control is transferred to the other part of the program.

There are three types of control structures which are listed below:

- Selective control structure (branching statement)
- Repetitive control structure(looping statement)
- Jumping structure(unconditional statement)

Selective control structure (branching statement)

In selective control structure selection id made on the basis of condition. Here the statement is executed on basis of condition. The flow of program statement execution is totally directed by the result obtained from checking condition. Hence selective control statement is also called as the conditional statement. Selective statements are categorized into two types:

- Conditional statement
- Switch-case statement

Control statement:

This is one of the most common decision making control structure which controls the flow of program statement execution based on condition checked.

a) If statement
b) If-else statement
c) If-else-if statement (if-else ladder)
d) Nested if-else statement

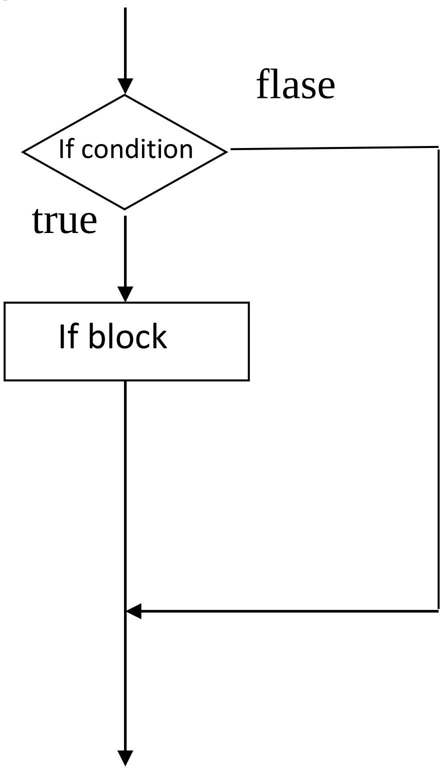
If statement:

This is the simplest form of conditional statement in which statements are executed if the test condition id true. When the condition is false there is no option to go, then control gets out of the statement.

Entry

Syntax:

```
If (condition)
{
  Statement;
}
```



Write a program to input two numbers and find weather they are equal, less than, greater than etc.

```
#include<stdio.h>
#incude<conio.h>
void main()
{
int a,b;
clrscr();
printf("enter the value of a and b:");
scanf("%d%d",&a,&b);
if(a<b)
printf("\n%d is less than %d",a,b);
if(a<=b)
printf("\n%d is less than equal to %d",a,b);
if(a==b)
printf("\n%d is equal to %d ",a,b);
i f(a!=b)
printf("\n %d is not equal to %d",a,b);
if(a>b)
printf("%d is greater than %d",a,b);
if(a>=b)
printf("\n %d is greater than equal to %d",a,b);
getch();
}
```

Output

```
Enter the value of a and b:12
7
12 is not equal to 7
12 is greater than 7
12 is greater than or equal  to 7
```

Write a program to find the greatest among two number using if conditions.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a,b;
clrscr();
printf("enter the value of a and b :");
scanf("%d%d",&a,&b);
if(a>b)
printf("a is greatest");
if(a<b)
printf("b is greatest");
if(a==b)
printf("a and b are equal");
getch();
}
```

<u>Output</u>

```
Enter the value of a and b :8
2
a is greatest
```

write a program asking total sales made by sales man he get commission to 5% only if he sales exceeds Rs. 5000 find the commission.

```
#include<stdio.h>
#include<conio.h>
void main()
{
float s,c;
clrscr();
printf("enter the total sales made:");
scanf("%f",&s);
if(s>=5000)
{
c=s*0.05;
printf("the commission is:%f",c);
}
getch();
}
```

<u>Output</u>

```
Enter the total sales made:70000
The commission is:3500.000000
```

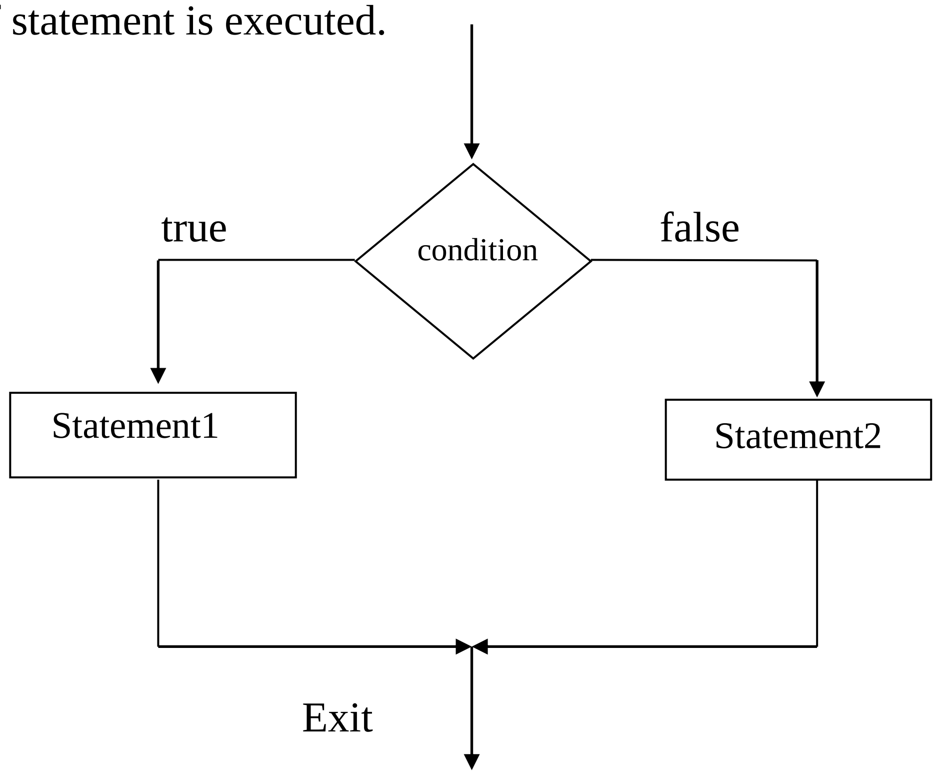
if-else statement:

This is a bidirectional conditional control statement this statement is used to test a condition and take one of the two possible actions. If the condition is true than a single statement block of statement is executed otherwise another single statement or block of statement is executed.

Syntax:

Entry

```
    If(condition)
{
Statement1;
}
Else
{
Statement2;
}
```



Write a program to enter any two numbers and find the greatest number among these two numbers by using if-else statement.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int x,y;
clrscr();
```

<u>Output</u>

```
Enter any two number:7
9
The greatest number is:9
```

```
printf("enter any two number");
scanf("%d%d",&x,&y);
if(x>y)
{
printf("the greatest number is:%d",x);
}
else
{
printf("the greatest number is:%d",y);
}
getch();
}
```

Write a program to find if the given number is even or odd using if…else condition.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a;
clrscr();
printf("enter the value of a:");
scanf("%d",&a);
if(a%2==0)
{
printf(" the number is even");
}
else
{
Printf("the number is odd");
}
Getch();
}
```

Output

```
Enter the value of a:8
The number is even
```

Write a program to find any year and check whether the year is leap year or not.

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

```
#include<conio.h>
void main()
{
int x;
clrscr();
printf(" enter any year:");
scanf("%d",&x);
if(x%4==0)
{
printf("the year is leap year");
}
else
{
printf("the year is not leap year");
}
getch();
}
```

Output

```
Enter any year:8
This is leap year
```

If-else-if statement (if-else ladder):

In this statement, condition are evaluated from the top of the ladder to downwards. As soon as the true condition is found, the statements associated with it are executed and the control is transferred to outside of the ladder skipping the remaining part of the ladder.

Syntax:

```
if(condition 1)
{
statement 1;
}
else if(condition 2)
{
statement 2;
}
else if(condition n)
{
statement n;
}
else
```

```
{
default statement;
}
statement x;
```

Write a program to find out the grade of a student when the marks of five subjects are given, the ranges of grade are assign below:

Per>=85	**distinction**
Per<85 and per>=70	**first division**
Per<70and per>=55	**second division**
Per<55and per>=40	**third division**
Per<40	**fail**

```
#include<stdio.h>
#include<conio.h>
void main()
{
float m1,m2,m3,m4,m5,total,per;
char grade;
printf("enter the marks of subject");
scanf("%f%f%f%f%f",&m1,&m2,&m3,&m4,&m5);
total=m1+m2+m3+m4+m5;
printf("total marks=%f",total);
per=total/5;
printf("percentage=%f",per);
if(m1>=40&&m2>=40&&m3>=40&&m4>40&&m5>=40)
{
if(per>=85)
printf("distinction");
else if(per>=70)
printf("first division");
else if(">=55")
printf("second division");
else if(>=40)
printf("third division");
else
printf("you are fail");
}
```

Output

```
Enter the marks of the subject:
50
80
70
40
90
Total marks=320
Percentage=66
First division
```

```
getch();
}
```

Switch case statement:

Switch statement is a multipath decision making statement that tests whether a variable or expression matches one of a number of constants integer value, and branches accordingly. It allows selection of a particular block of statements from several block of statements based on the value of an expression which is included in the switch statement.

Syntax:

```
        Switch(expression);
        {
        Case constant 1;
        Statement 1;
        Break;
        Case constant 2:
        Statement2;
        Break;
        Case constant n:
        Statement n;
        Break;
        Default Statement;
        }
```

Q. Write a program to print the day of the week according to the number entered 1 for Sunday…..7 for Saturday.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int day;
clrscr();
printf("enter the number:");
scanf("%d",&day);
switch(day)
{
case 1: printf("\n sunday");
break;
case 2: printf("\n monday");
break;
case 3: printf("\n tuesday");
break;
case 4: printf("\n wednesday");
break;
case 5: printf("\n thursday");
break;
case 6: printf("\n friday");
break;
case 7: printf("\n saturday");
break;
default: printf("\n invalid number");
}
getch();

}
```

<u>Output:</u>

```
enter the number:8
invalid number
```

Q. write a program to perform the operation according to the number entered 1 for sum, 2 for difference……5 for remainder.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a,b,c,ch;
clrscr();
printf(" enter the value of a and b:");
scanf("%d%d",&a,&b);
printf("enter your choice");
scanf("%d,&ch");
switch(ch)
{
case 1:
c=a+b;
printf("sum is=%d",c);
break;
case 2:
c=a-b;
printf("difference is=%d",c);
break;
case 3:
c=a*b;
printf("product is=%d",c);
break;
case 4:
c=a/b;
printf("quotient is=%d",c);
break;
case 5:
c=a%b;
printf("remainder is=%d",c);
break;
default:
printf("enter valid operator\n");
}
getch();
}
```

output:

```
enter the value of a and b:6
5
enter your choice:2
difference =1
```

Q. Write a program to print the month of the year according to the number entered 1 for January , 2 for february……12 for December.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int month;
clrscr();
printf("enter the number");
scanf("%d",&month);
switch(month)
{
case 1:printf("\n january");
break;
case 2:printf("\n february");
break;
case 3:printf("\n march");
break;
case 4:printf("\n april");
break;
case 5:printf("\n may");
break;
case 6:printf("\n jun");
break;
case 7:printf("\n july");
break;
case 8:printf("\n agust");
break;
case 9:printf("\n september");
break;
case 10:printf("\n octuber");
break;
case 11:printf("\n november");
break;
case 1:printf("\n december");
break;
dafault:printf("\n invalid number");
}
getch();
```

}

Looping:

Repetitive control structure is used to execute the same program statement or block of program statements repeatedly for specified number of times or till a given is satisfied. With repetitive control structure we do not need to type the same program statements again and again. Repetitive control structure is also known as loop control structure or iterative control structure.

For loop:

It is the most common type of loop which is used to execute program statements of block of program statements repeatedly for specified number of times. It is definite loop. Mainly it consists of three expressions, initialization, condition and counter. The initialization defines the loop starting point, condition stopping point and counter helps to increment and decrement the value of counter variable.

Syntax:

```
For (initialization; test condition; increment or decrement)
{
Statement;
}
```

Q. write a program to print name up to twenty time (using for condition)*/

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i;
clrscr();
for(i=1;i<=20;i++)
{
printf("\n suraj");
}
getch();
}
```

}

Output:

```
Suraj
Suraj
Suraj
Suraj
Suraj
Suraj
Suraj
Suraj
Suraj
suraj
Suraj
Suraj
Suraj
Suraj
suraj
Suraj
Suraj
Suraj
Suraj
suraj
```

Write a program to print number from 1 to 10 using for condition.

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

Output:

```
0      1      2      3      4      5      6      7      8      9      10
```

Q. Wap to display 5,10,15,20,25 using for loop.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i;
clrscr();
for(i=5; i<=25;i+5)
{
printf("\t%d",i);
}
getch();
}
```

Output:

```
5      10     15     20     25
```

Q. write a program to find the factorial of any given number by using for loop.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i,n,fact=1;
clrscr();
printf("enter the number:");
scanf("%d",&n);
if(n<0)
printf(" factorial of negative number is not possible");
else
{
for(i=1;i<=n;i++)
fact=fact*i;
printf("the factorial of %d is=%d",fact,n);
```

Output:

```
Enter the number:5
The factorial of 5 is =120
```

```
}
getch();
}
```

Q. Write a program to print your school name 10 times.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i;
clrscr();
for(i=0;i<=10;i++)
printf("\t janaprakash higher secondary school");
getch();
}
```

Output:

```
Janaprakash higher secondary school
Janaprakash higher secondary school
Janaprakash higher secondary school
Janaprakash higher secondary school
Janaprakash higher secondary school
Janaprakash higher secondary school
Janaprakash higher secondary school
Janaprakash higher secondary school
Janaprakash higher secondary school
Janaprakash higher secondary school
```

Q. write a program to find the Fibonacci series.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i,f1=0,f2=1,f=1,n;
clrscr();
printf("enter the term of fibo series:");
scanf("%d",&n);
for(i=1;i<n;i++)
{
printf("the fibonacci series is=%d\t",f);
f=f1+f2;
f1=f2;
f2=f;
}
getch();
}
```

Output:

```
Enter the term of fibo series:5
The Fibonacci series =1 1 2 3 5
```

Q. write a program to find the sum of natural number using for loop.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i,n,sum=0;
clrscr();
printf("enter natural number limit :");
scanf("%d",&n);
for(i=0;i<n;i+1)
{
sum=sum+i;
}
printf("the sum of natural numbers is:%d",sum);
getch();
}
```

output:

```
enter natural number limit:50
the sum of natural numbers is:1225
```

Q. write a program to find the sum of even numbers up to the given limit.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i,n,sum=0;
clrscr();
printf("enter the even numbers limit:");
scanf("%d",&n);
for(i=0;i<n;i+2)
{
sum=sum+i;
}
printf("the sum of even number is:%d",sum);
getch();
}
```

output:

```
enter the even numbers limit:40
the sum of even number is:380
```

While loop:

It executes the program statement repeatedly until the given condition is true. It checks the condition first , if it is found true then it executes the statement written in the body part otherwise it just gets out of the loop structure. It is also known as pre-test loop or entry control loop. It may be definite or indefinite loop.

Syntax:

```
Initialization;
While (condition)
{
Statements;
……………
……………
Counter;
}
```

Q. Wap to display a number from 30 to 1.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i;
clrscr();
i=30;
while(i>1)
{
```

```
printf("%d\t",i);
i--;
}
getch();
}
```

Output:

```
30 29 28 27 26 25 24 23 22 21
20 19 18 17 16 15 14 13 12 11
 10 9 8 7 6 5 4 3 2
```

Q. Wap to print 100 to 1 using while loop.

```
#include<stdio.h>
#include<conio.h>
void main ()
{
int i;
clrscr();
i=100;
while(i>=1)
{
printf("%d\t",i);
i--;
}
getch();
}
```

Output:

```
100 99 98 97 96 95 94 93 92 91
90 89 88 87 86 85 84 83 82 81
80 79 78 77 76 75 74 73 72 71
70 69 68 67 66 65 64 63 62 61
60 59 58 57 56 55 54 53 52 51
50 49 48 47 46 45 44 43 42 41
40 39 38 37 36 35 34 33 32 31
30 29 28 27 26 25 24 23 22 21
20 19 18 17 16 15 14 13 12 11
10  9  8   7   6   5   4   3   2   1
```

Write a program to reverse the given string.

```
#include<stdio.h>
#include<conio.h>
void main ()
{
int n,r,rev=0;
clrscr();
printf("enter an integer value=");
scanf("%d",&n);
while (n!=0)
{
r=n%10;
rev=rev*10+r;
n=n/10;
}
printf("reversed number=%d",rev);
getch();
}
```

Output:

```
Enter an integer value=123
Reversed number=321
```

Q. Wap a program to find sum of first natural number.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i,n,sum=0;
clrscr();
printf("enter the number:");
scanf("%d",&n);
i=1;
while(i<=n)
{
sum=sum+1;
i++;
}
printf("sum of %d natural number is:%d",n,sum);
getch();
}
```

Output:

```
Enter the number:20
Sum of first 20 natural number is:210
```

Do while loop:

It executes the program statement repeatedly until the given condition is true. It executes the program statement once at first then the condition is checked. If the condition is found true then it executes the program statements again otherwise it gets out from the loop structure. As it checks the condition at the last it is also known as post-test-loop or exit control loop.

Syntax:

```
Do
{
Statements;
…………
…………
Counter;
}
While (condition);
```

Q. Wap to enter an integer value and find the sum of its digits.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int n,r,sum=0;
clrscr();
printf("enter an integer value=");
scanf("%d",&n);
do
{
r=n%10;
sum=sum+r;
}
while (n!=0);
printf("sum of digits=%d",sum);
getch();
}
```

Output:

```
Enter an integer value=123
Sum of digits=6
```

Q. Wap to reverse the given string and check whether the given string is palindrome or not.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int n,r,temp,rev=0;
clrscr();
printf("enter an integer value=");
scanf("%d",&n);
temp=n;
do
{
r=n%10;
rev=rev*10+r;
n=n/10;
}
while(n!=0)
if (rev==temp)
printf("not palindrome");
getch();
}
```

Output:

```
Enter an integer value=123
Not palindrome
```

Q. Wap to display number from 1 to 100 using do while loop.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i;
clrscr();
i=1;
do
{
printf("%d\t",i);
i++;
}
while(i<=100);
getch();
}
```

Output:

```
1  2  3  4  5  6  7  8  9  10  11 12 13 14 15 16 17 18 19 20
21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38
39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56
57 58 59 60 61 62 63 64 65 67 68 69 70 71 72 73 74 75
76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93
94 95 96 97 98 99 100
```

Difference between for loop and while loop.

For loop	**While loop**
It uses keyword 'for '.	It uses keyword 'while '.
It is a definite loop.	It can be both definite and indefinite loop.

Syntax:
For(initialization; test condition; increment or decrement)
{
Statement;
}
	Syntax:
	Initialization;
	While (condition)
	{
	……….
	……….
	Counter,
	}

Write a program to print your school name 10 times.	**Write a program to print your school name 10 times.**

#include<stdio.h>
#include<conio.h>
void main()
{
int i,;
clrscr();
for(i=0;i<=10;i++)
printf("\t janaprakash secondary school");
getch();
}
	#include<stdio.h>
	#include<conio.h>
	void main()
	{
	int i=1;
	clrscr();
	while(i<=10)
	printf("\t janaprakash secondary school")
	getch();
	}

Difference between while and do while loop.

While loop	**Do while loop**
In which loop the condition is checked in the beginning.	In do while loop condition is checked at the end.
It is also known as pre-test loop or entry control.	It is also known as post-test loop or exit control loop.
It is not terminated with semicolon.	It is terminated with semicolon.
In while loop; statements are not executed if the condition is false.	In do while loop, statements are executed once even if the condition is false.
It uses key word ' while '.	It uses keyword ' do- while '.

Syntax:
Initialization;
While (condition)
{
Statements;
………..
………..
Counter;
}
	Syntax:
	Initialization;
	Do
	{
	Statements;
	………….
	………….
	Counter;
	}

Write a program to print your school name 10 times.	**write a program to print your school name 10 times.**

#include<stdio.h>
#include<conio.h>
void main()
{
int i=1;
clrscr();
while(i<=10)
print(" \t janaprakashs secondary school");
getch();
}
	#include<conio.h>
	#include<conio.h>
	void main()
	{
	int i=1;
	clrscr();
	do
	{
	printf("janaprakash secondary school");
	}
	while(i<=10);
	getch();
	}

Jumping statements:

Jumping statements are particularly used to jump execution of program statements from one place to another inside the program. These statements may execute same program statement repeatedly or skip some program statement. Following are the jumping statements defined in c programming language.

- Break
- Continue
- Goto

Break statement:

It is used to break the normal flow of program statement execution in loop and switch case statement. It allows us to exit from the innermost enclosing loop or switch statement as soon as condition Is satisfied.

Example:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i;
for(i=1;i<=5;i++)
{
if(i=4)
break;
printf("%d",i);
}
getch();
}
```

Output:

```
1  2  3
```

Continue statement:

It is used to continue the normal flow of program statement execution in the loop; skipping particular iteration in the loop as soon as the certain condition is satisfied.

Example:

```
#include<stdio.h>
#include<conio.h>
void main()
{
if (i==4)
continue;
printf("%d",i);
}
getch();
}
```

Output:

```
1   2   3   4
```

Difference between break and continue statement;

Break statement	**Continue statement**
When break statement is encountered the entire loop or switch statement is terminated.	When continue statement is encountered the entire loop terminated only that particular iteration is skipped.
It is used with loop and switch case statement.	It is used only with loop structure.
It uses keyword 'break'	It uses keyword continue.

Example:
#include<stdio.h>
#include<conio.h>
void main()
{
int i;
for(i=1;i<=5;i++)
{
if (i==4)
break;
printf("%d",i);
}
getch();
}
<u>Output:</u>
1 2 3
	Example:
	#include<stdio.h>
	#include<conio.h>
	void main()
	{
	int i;
	for(i=1;i<=5;i++)
	{
	it (i==4)
	continue;
	printf("%d",i);
	}
	getch();
	}
	<u>Output:</u>
	1 2 3

Array:

An array is a collection of similar type of data items under the common name. It acts to store related data under the same name with an index also known as subscript which helps to access individual array elements. Array data type may be int, float, char etc depending on the nature of the problem.

Characteristics of array:

- All the array elements share the common name.

- The elements of array are stored in contiguous memory locations.
- By using array programs become short and simple.
- We can put array size of fixed length as required.
- We can randomly access to every element using numeric index.

Types of array:

There are two types of array which are as follows:

1. **One dimensional array:** an array which has only one subscript is named as on dimensional array.
   Data_type_array_name [array_size]
   Eg. Int a[5];
   Float mark [8];
   Char temp [10][5]
2. **Multi dimensional array:** multi dimensional array are difined same as one multi dimensional array, except that consists more than one pair of square bracket or subscript. Thus two dimensional arrays require two square brackets i.e. one for row and one for column.
   Syntax:
   Data_type_array_name [exepression 1 [exepression 2]....[expression n];
   Eg. Int a[5][5];
   Float mark[8][2];
   Char temp[10][5];

Q. Write a program to input 5 numbers in an array and display them.

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

Q. Write a program to input 5 numbers in an array and display their sum.

```
#include<conio.h>
#include<stdio.h>
void main()
{
int i, num[5],max[5];
clrscr();
printf("enter the 5 elements of array=");
for(i=0;i<5;i++)
scanf("%d",&num[i]);
printf("elements of array are")
for(i=0;i<5;i++)
{
printf("%d",num[i]);
sum=sum+num[i];
}
printf("sum of five numbers=%d",sum);
getch();
}
```

Q. Wap to find the largest number among n numbers.

```
#include <stdio.h>
#include<conio.h>
void main()
{
int i, n, num[100],max;
clrscr();
printf("enter the size of an array not more than 100");
scanf("%d",&n);;
printf("enter the numbers\n");
for(i=0;i<n;i++)
{
scanf("%d", &num[i]);
}
max=num[0];
for(i=1;i<n;i++)
{
```

```
if (num[i]>max)
max=num[i];
}
printf("the greatest number is =%d",max);
getch();
}
```

Output:

```
Enter the numbers:
41
121
45
63
52
84
95
102
111
101
The greatest number is= 121
```

Q. Wap to find the smallest number among n numbers.

```
#include <stdio.h>
#include<conio.h>
void main()
{
int i, n, num[100],min;
clrscr();
printf("enter the size of an array not more than 100");
scanf("%d",&n);;
printf("enter the numbers\n");
for(i=0;i<n;i++)
{
scanf("%d", &num[i]);
}
min=num[0];
for(i=1;i<n;i++)
{
if (num[i]>min)
min=num[i];
}
printf("the smallest number is =%d",max);
getch();
}
```

Output:

```
Enter the numbers:
41
121
45
63
52
84
95
102
111
101
The smallest number is= 41
```

Q. Write a program to input n numbers and sort them in ascending order.

```
#include <stdio.h>
#include<conio.h>
void main()
{
int i,j, n, num[100],t;
clrscr();
printf("enter the size of an array not more than 100");
scanf("%d",&n);;
printf("enter the numbers\n");
for(i=0;i<n;i++)
{
scanf("%d", &num[i]);
}
 for(i=0;i<n;i++)
{
for (j=i+1;j<n;j++)
{
if (num[i]>num[j])
{
t=num[i];
num[i]=num[j];
num[j]=t;
}
}
}
printf("the numbers in ascending order etc:");
for(i=0;i<n;i++)
printf("%d",num[i]);
getch();
}
```

Q. Write a program using c language to read the age of 10 persons and count the number of persons in the age between 50 and 60. Use 'for' and 'continue' statements.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int age [10],i,count=0;
clrscr();
printf("enter age of 10 persons\n");
for(i=1;i<10;i++)
{
scanf("%d",&age[i]);
}
for(i=1;i<10;i++)
{
if(age[i]<50||age[i]60)
continue;
count=count+1;
}
printf("there are %d persons of age between 50 and 60");
getch();
}
```

Output:

```
Enter age of 10 persons
40
50
55
60
5258
59
70
65
57
There are 7 persons of age between 50 and 60
```

Q. write a program find out the total sum of array element, sum of even number in array element and sum of odd number in array element.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int arr[10],i,sum=0,even=0,odd=0;
clrscr();
for(i=0;i<10;i++)
{
printf("enter the value for%d element of array:",i);
scanf("%d",&arr[i]);
sum+=arr[i];
if(arr[i]%2==0)
even+=arr[i];
else
odd+=arr[i];
}
printf("the total sum of array element=%d and sum of even =%d and sum of
odd=%d" ,sum,even,odd);
getch();
}
```

Output:

```
Enter the value for0 element of array:7
Enter the value for1 element of array:11
Enter the value for2 element of array:10
Enter the value for3 element of array:8
Enter the value for4 element of array:34
Enter the value for5 element of array:2
Enter the value for6 element of array:3
Enter the value for7 element of array:12
Enter the value for8 element of array:15
Enter the value for9 element of array:18
The total sum of array element =120 and sum of even=84 and sum of odd=36
```

Q. program to find out the maximum and minimum number in an array.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i, j,arr[10]={2,10,23,24,45,67,99,8,2,5,11};
int max,min;
clrscr();
min=max=2;
for(i=1;1<10;i++)
{
if(arr[i]<min)
min=arr[i];
if(arr[i]>max)
max=arr[i];
}
printf("minimum=%d\n",min,max);
getch();
}
```

Output:

Minimum=2, maximum=9

***Q.* program to search the particular item in the array.**

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i,arr[10]={4,6,99,56,34,90,56,12,13,50};
int item;
clrscr();
printf("enter the item to be searched:");
scanf("%d",&item);
for(i=0;i<10;i++);
{
if(item==arr[i])
{
printf("%d is found at position %d\n",item,i+1);
break;
}
}
if(i==10);
printf("item %d not found in array\n",item);
getch();
}
```

Q. Write a program to read salaries of 200 employee and count the number of employee getting salary between 5000-10000.

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

```
#include<conio.h>
void main()
{
int s[200],i,c=0;
clrscr();
printf("enter the salary\n:");
for(i=0;i<200;i++)
{
 scanf("%d",&s[i]);
}
 for(i=0;i<200;i++)
{
if(s[i]>=5000&&s[i]<=10000)
c=c+1;
}
printf("the total number of employee getting between 5000+10000 is =%d",c);
getch();
}
```
Output:

```
Enter the salary: 10000  6000  4000  7000  8000  9000
The total number of employee getting between 5000 and 10000 is=4
```

Q. Write a program to sort integer value in descending order.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[50],i,j,temp,n;
clrscr();
printf("how many number?\n:");
scanf("%d,&n");
printf("enter number\n:");
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
 for(i=0;i<n;i++)
{
for(j=i+1;j<n;j++)
{
if (a[i]<a[j])
{
temp=a[i];
a[i]=a[j];
a[i]=temp;
}
}
}
printf("the number in descending order are\n:");
for(i=0;i<n;i++)
{
printf("%d\t",a[i]);
}
getch();
}
```

Output:

```
How many numbers?
:4
Enter the number: 1  2  3  4
The number in descending order are
: 4  3  2  1
```

Q.Wap to input elements in matrix and display to matrix.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int x[3][3],i,j;
clrscr();
printf("enter the element of matrix\n:");
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
{
```

```
scanf("%d",&x[i][j]);
}
}
for(i=0;i<3;i++)
{
printf("\n");
for(j=0;j<3;j++)
{
printf(""\t");
printf("the entered matrix is=%d"x[i][j]);
}
}
getch();
}
```

Output:

```
Enter the elements of matrix
:4
6
5
8
5
2
3
7
9

The entered matrix is: 4   6   5
                       8   5   2
                       3    7   9
```

Q. Program to input two matrix and find out the sum of two matrix.

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

Output:

```
                                                        Sum of two matrix are:
Enter the elements of matrix a   enter the elements of matrix b   10
7                                3                                11
5                                6                                17
5                                9                                13
4                                8                                11
9                                7
6                                4
8                                5
8                                8
7                                9
```

String:

String is the set of collection of characters, digits and symbols enclosed in quotation marks. We can also define string as the array of characters. The end of string is marked with special character '\0' means null character. The size in a character string represents the maximum numbers of characters that the string can hold.

Syntax:

Char_string_name(string size)

Q.Wap to input character wise in array and display them.

```
#include<stdio.h>
#include<stdio.h>
void main()
{
char name[10]={'c','p','r','o','g','r','a','m'};
clrscr();
printf("the given string is=%s",name);
getch();
}
```

Output:

```
The given string is =c program
```

String handling functions:

1. Strlen function:

The strlen function returns the length of a string which takes the string name as the argument.

Syntax:

1=strlen(str);

Q. Write a c program to find the length of the string.

```
#include<stdio.h>
#include<conio.h>
void main()
{
char str[50];
clrscr();
printf("enter the string:");
gets(str):
strlen(str);
printf("the length of the string=%d",1);
getch();
}
```

2. Strrev function:

The strrev function is used to reverse the given string.

Syntax:

Strrev(str);

Q. Write a program to reverse the given string.

```
#include<stdio.h>
#include<conio.h>
void main()
{
char str[50];
clrscr();
printf("enter the string:");
gets(str);
strrev(str);
printf("the rev of string =%s",str);
getch();
}
```

3. Strcpy function:

The strcpy function is used to copy one string into another string.

Syntax: strcpy(str1,str2);

Q. Write a c program to copy one string into another string.

```
#include<stdio.h>
#include<conio.h>
void main()
{
char str1[50],str2[50];
clrscr();
printf("enter the string:");
gets(str1);
strcpy(str1,str2);
printf("after copying the string=%s",str2);
getch();
}
```

4. Strcat function:

This strcat function is used to join or concatenate one string into another string.

Syntax: strcat(str1,str2);

Q. Write a program to concatenate two string into one string.

```
#include<stdio.h>
#include<conio.h>
void main()
{
char str1[50],str2[50];
clrscr();
printf("enter the first string:");
gets(str1);
printf("enter the second string:");
gets(str2);
strcat(str1,str2);
printf("after concatenating the string=%s",str1);
getch();
}
```

5. Strcmp function:

The strcmp function is used to compare two strings character by character and stops comparison when there is difference in ASCII value.

Syntax: strcmp(str1,str2);

Q. Write a c program to compare two strings.

```
#include<stdio.h>
#include<conio.h>
void main()
{
char str1[50],str2[50];
int v;
clrscr();
printf("enter the first string:");
gets(str1);
printf("enter the second string:");
gets(str2);
v=strcmp(str1,str2);
if(v==0)
printf("two strings of equal number of characters");
else if(v>0)
printf("first string has more number of characters then second string");
else
printf("second string has more num of character than first string");
getch();
}
```

6. Strlwr function:

Strlwr function is used to convert uppercase string into lowercase string.

Syntax:strlwr(str);

Q. Write c program to be converted into lower case.

```
#include<stdio.h>
#include<conio.h>
void main()
{
char str[50];
printf("enter string:");
gets(str);
printf("the given string lower case is:%s",strlwr(str));
getch();
}
```

7. Strupr function:

The strupr function is used to convert lower case characters of string into upper case characters.

Syntax: strupr(str);

Where str is string to be converted into upper case characters.

Q. Write a program to convert the given string upper case.

```
#include<stdio.h>
#include<conio.h>
void main()
{
char str[50];
clrscr();
printf("enter the string:");
gets(str);
printf("the given string lower case is:%s",strlwr(str));
getch();
}
```

Q. Write a c program to input a strings and sort them.

```
#include<stdio.h>
#include<conio.h>
void main()
{
char str [50][30], temp[30];
int i,j,n;
printf("\n how many strings:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("\n enter a string:");
scanf("%s",&str[i]);
}
for(i=0;j<n-1;j++)
gets(str);
{
for(i=0;j<n-1;j++)
{
if (strcmp(str[i],str[j])>0)
{
strcpy(temp,str[i]);
strcpy(str[i],str[j]);
strcpy(str[j],[i]);
}
}
}
printf("the sorted string are:");
for(i=0;i<n;i++)
printf("\n%s",str[i]);
}
```

Concept of function, function definition, function prototype

A function is a self contained sub program, which means to do some specific, well defined task. A c program consists of one or more functions. Execution of every c program always begins with main(). Additional functions will be subordinate to main and perhaps to one another. After each functions has done its operation, control returns back to the main(). Then remaining statements of main () are executed.

Advantages of function

- Function increases code reusability.
- The length of the source program can be reduced by using functions at appropriate palces.

- Program develop will be faster.
- Program debugging will be easier.
- Large number of programmer can involved.
- Program can be developed in short period of time.
- Program can developed in different places.
- The program can be tasted and compiled independently by different member of a programming day.

Components of Function

- Function prototype
- Function definition
- Call function

Function prototype

Function prototype provides the following information to the computer.

- The type of value returned
- The name of the function
- The number and the type of the agreements that must be supplied in a function call
- Function definition
- A function has two principle declaratory
- Body of the function

  The first line of function definition is known as **function declaration** and is followed by the **function body.** Function declaratory contains the value returned by the function followed by the function name and optionally a set of arguments, separated by commas and enclosed in parentheses.

Q. Write a program to print square of a number using function.

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

Q. Write a program to print greatest number between two numbers using function.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a,b,c;
clrscr();
printf("enter the two number:");
scanf("%d%d",&a,&b);
c=great(a,b);
printf("greatest number is=%d",c);
getch();
}
int great (int a, int b)
{
if (a>b)
{
```

```
return(a);
}
else
{
return(b);
}
}
```

Q. Write a program to print smallest number between three numbers using function.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a,b,c,d;
clrscr();
printf("enter the three number:");
scanf("%d%d%d",&a,&b,&c);
d=small(a,b,c);
printf("smallest number is=%d",d);
getch();
}
int small(int a, int b, int c)
{
if(a<b&&a<c)
{
return(a);
}
else if(b<a&&b<c)
{
return(b);
}
else
{
return(c);
}
}
```

Q. Wap to check whether the entered number is even or odd.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a;
clrscr();
printf("enter a number:");
scanf("%d",&a);
evenodd(a);
}
void evenodd(int a)
{
if(a%2==0)
{
printf("the number is even");
}
else
{
printf("the number is odd");
}
}
}
```

Q. Wap to check entered number is positive, negative or zero using function.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a;
clrscr();
printf("enter a number:");
scanf("%d",&a);
posit(a);
}
```

```
void posit(int a)
{
if(a>0)
{
printf("the number is positibe");
}
else if(a<0)
{
printf("the number is negative");
}
else
{
printf("the number is zero");
}
}
```

Q. Wap to print sum of two numbers using function.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a,b,c;
clrscr();
printf("enter the two numbers:");
scanf("%d%d",&a,&b);
c=sum(a,b);
printf("sum of two number is=%d",c);
getch();
}
int sum(int a, int b);
{
int d;
d=a+b;
return(d);
}
```

Q.Wap to print area of circle using function.

```
#include<stdio.h>
#include<conio.h>
void main()
{
flost r,s;
clrscr();
printf("enter the radius of circle=:");
scanf("%f",&r);
s=area(r);
printf("area of circle=%f",s);
getch();
}
int sum (float r)
{
float a;
a=3.141*r*r;
return(a);
}
```

Q. Wap to printf multiplication table of n numbers using function.

```
#include<stdio.h>
#include<conio.h>
Void mtable(int);
void main()
{
int n;
clrscr();
printf("enter a number=");
scanf("%d",&n);
mtable(n);
getch();
}
void mtable(int n)
```

```
{
int j;
for(j=1;j<=10;j++)
        {
        printf("%d*%d=%d/n,n,j,n*j");
}
                }
```

Recursive function:

It is process by which a function calls itself repeatedly, until the given condition has been satisfied. For the problem to solve recursively two conditions must be satisfied.

- The function must call itself.
- The problem statement must include a stopping condition.

Q. Wap to calculate factorial using recursive function.

```
#include<stdio.h>
#include<conio.h>
Int fact(int);
void main()
{
int n;
clrscr();
printf("enter a number:");
scanf("%d",&n);
f=fact(n);
printf("factorial of given number =%d",f);
}
int fact(int n)
{
if (n<1)
{
return 1;
}
else
{
```

```
return (n*fact(n-1));
}
}
```

Types of function:

There are two types of function which are as follows:

- Library (built-in function)
- User defined functions

Library functions:

Library functions are in-built functions which have predefined meaning in c program. C has mathematical functions and string handling library functions but they have special header files i.e. #include<math.h> for mathematical functions and #include<string.h> for string handling functions.

User defined functions:

User also can create their own functions for doing specific task of the programs. Such functions are called user defined function.

Categories of user defined functions:

- Function returning value and passing argument
- Function returning no value but passing argument
- Function returning value and passing no argument
- Function returning no value and passing no argument

s.n	**Library function**	**User define function**
1.	Library function are predefine function.	User define function are function which are created by the programmer according to the nill.
2.	Library function required header file to use it.	User define function require functions prototype to use it.
3.	Program develop time is faster.	Program development time is slower the library function.
4.	It is called at run time.	It is called at compile time.
5.	The name of library function not be change.	The name of use define functions can be changed any time.
6.	The program using librabry function are very short.	The program using user define function are very long.

7.	Printf(), scanf(), strlen(), strupr() etc.	Sum(), difference() etc.

Structure and union

Structure:

As we know that an array is the collection of homogeneous data items that means similar data types such as int or float. It is not possible to hold different data items with different data type in and array, so structure is the best solution to hold dissimilar data items as single unit.

Features of structure:

- it can be treated as record i.e. collection of interrelated data fields having different data type.
- It is possible to copy one structure variable to another by simply assignment operator(=).
- It allows nested structure i.e. structure inside another structure.
- It is possible to pass structure variable parameter to any function.
- It also possible to define structure pointer.

Syntax of structure declaration:

```
Struct tag_name
{
Data_type member 1;
Data_type member 2;
…………………………;
Data _type member N;
};
Struct tag_name var1, var2,……..varM;
```

Example:

```
Struct student
{
Int roll _no;
Char fname[30];
Char lname[30];
};
Struct student s1,s2,s3;
```

Q. Wap to display name, age and mark and student by using structure variable.

```
#include<stdio.h>
#include<conio.h>
struct student
{
char name[20];
int age;
float mark;
};
struct student s1;
void main()
{
struct student s1;
clrscr();
printf("enter the age and marks of students \n");
scanf("%s%d%f",&s1.age,&s1.mark);
printf("name\tage\tmark\n");
printf("%s\t%d\t%f",&s1.age,&s1.mark);
getch();
}
```

<u>Output:</u>

```
Enter the age marks of students
Ram
18
80
Name  age  mark
Ran     18    80.0000000
```

Q. Wap to display name and age of two students using structure variable

```
#include<stdio.h>
#include<conio.h>
struct student
{
char name[20];
int age;
float marks;
};
struct student s1,s2;
void main()
{
struct student s1,s2;
clrscr();
printf("enter the name, age, marks\n");
scanf("%s%d%f",&s1.name, &s1.age,&s1.marks);
scanf("%s%d%f",&s2.name, &s2.age,&s2.marks);

printf(" name\tage\tmarks\n");
printf("%s%d%f",s1.name, s1.age,s1.marks);
printf("%s%d%f",s2.name, s2.age,s2.marks);
getch();
}
```

Q. Wap that takes roll_no, fname and lname of five students and print the same record on the screen.

```
#include<stdio.h>
#inclue<conio.h>
struct student
{
int roll_no;
char fname[20];
char lname[20];
}s[5];
void main()
{
int i;
clrscr();
for(i=0;i<5;i++)
{
printf("enter the roll no=");
scanf("%d",&s[i].roll_no);
printf("enter the fname =");
scanf("%d",&s[i].fname);
printf("enter the lname=");
scanf("%d",&s[i].lname);
}
for(i=0;i<5;i++)
{
printf("%d%s%s",s[i].roll_no,s[i].fname,s[i].lname);
}
getch();
}
```

Q. Wap that takes roll_no, fname and lname of five students and print the same record n ascending order on the basis of roll_no.

```
#include<stdio.h>
#inclue<conio.h>
struct student
{
int roll_no;
char fname[20];
char lname[20];
}s[5];
void main()
{
int i,j;
clrscr();
for(i=0;i<5;i++)
{
printf("enter the roll no=");
scanf("%d",&s[i].roll_no);
printf("enter the fname =");
scanf("%d",&s[i].fname);
printf("enter the lname=");
scanf("%d",&s[i].lname);
}
for(i=0;i<5;i++)
{
for(j=i+1;j<5;j++)
{
if(s[i].roll_no>s[j].roll_no)
{
temp=s[i];
s[i]=s[j];
s[j]=temp;
}
}
}
for(i=0;i<5;i++)
printf("%d%s%s",s[i].roll_no,s[i].fname,s[i].lname);
}
getch();
}
```

Union:

Union is similar to structure but is differs in its storage location. The union keyword is used to define union. In structure, each member has its own memory block whereas all members of union can share the same memory location. Therefore, union can take less amount of memory then structure and can access only one member at a time.

Syntax of union declaration:

```
Union tag_name
{
Data_type member1;
Data_type member2;
……………………….;
Data_type member N;
};
Union tag_name var1, var2,……..varN;
```

Example:

```
Union student
{
int roll_no;
char fname[30];
char lname[30];
};
union student s1,s2,s3;
```

Difference between structure and array:

Structure	Array
It is collection of data item having dissimilar data types.	it is the collection of data item having similar data type.
Each data item is called member.	Each data item is called element.
It is user defined data type.	it is built in data type.
It is possible to define array of structure.	It is not possible to define array of array or structure of array.

Syntax:
Data_type member1;
Data_type member2;
………………………..;
data_type member;
};
Struct tag_name var1,var2….varM;
	Syntax:
	Data_type array_name[size];

Example:
Struct temp
{
Int p;
Float q;
}x;
	Example:
	Int x[4];

Difference between structure and unio.

Structure	Union
It is the collection of data item having dissimilar data types	Same as structure but differs in the storage class.
Memory size is determined by the sum of memories allocated to all the members	Memory size is determined by the memory allocated to the largest member.
Multiple member can be accessed simultaneously at a time.	Only one member can be accessed at a time.

Syntax:
{
Data_type member1;
Data_type member 2;
……………………….;
Data_type member;
};
Struct tag_name var1, var2,…..varM;
	Syntax:
	Union tag_name
	{
	Data_type member1;
	Data_type member 2;
	……………………….;
	Data_type member;
	};
	union tag_name var1, var2,…..varM;

Example:
Struct data
{
char x;
int y;
float z;
}var;
	Example:
	Union data
	{
	char x;
	int y;
	float z;
	}var;

Pointer:

A pointer is a variable that points to another variable which means it contains the memory address of another variable and is declared as pointer type.

Features of pointer:

- Pointer saves memory space.
- Pointer assigns the memory space and also releases it when not in use.
- The execution time is faster because data manipulation is done directly inside the memory.
- Pointer are very close to array so it is efficient for solving array and string related problems.
- Pointers are very close to hardware so it is very efficient for solving hardware related problems.

The address (&) and indirection (*) operator:

The '&' is the address operator, it represents the address of the variable and '*' operator is the value at address operator. It represents the value at the specified address.

Q. Wap to add any two number using pointer.

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

```
void main()
{
int a, b, c, *p,*q;
clrscr();
printf("enter any two number\n");
scanf("%d%d",&a,&b);
p=&a;
q=&b;
c=*p+*q;
printf("the sum is %d",c);
getch();
}
```

Output:

```
Enter any two number
8
5
The sum is 13
```

Q. Wap to substract any two number using pointer.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a, b, c, *p,*q;
clrscr();
printf("enter any two number\n");
scanf("%d%d",&a,&b);
p=&a;
q=&b;
c=*p-*q;
printf("the difference  is %d",c);
getch();
}
```

Output:

```
Enter any two number
8
5
```

```
The difference is 3
```

Q. Wap to print the employee name, salary and age .

```
#include<stdio.h>
#include<conio.h>
void main()
{
int name ,age;
float salary;
clrscr();
printf("how many employees are there\n");
scanf("%d",&n);
/* writing data into file*/
fp=fopen("employee.data"."w+");
printf("enter employee name salary and age\n");
for(i=0;i<n;i++)
{
scanf("%f %d%d",name[i],&salary[i],age[i]);
fprintf(fp,"%d\t%d\t%d",&name[i],salary[i],age[i]);
 }
for(i=0;i<n;i++)
{
fscanf(fp,"%d\t%d\t%d",&name[i],&salary[i],&age[i]);
printf("%d\t%d\t%d",name[i],salary[i],age[i]);
 }
fclose(fp);
getch();
 }
```

Output:

```
How many  employees are there
1
Enter employee name salary and age
Ram
10000
20
Ram 10000  20
```

Object-Oriented language

Object oriented language (OOP) is a programming paradigm based on the concepts of "objects" which can contain data in the form of fields, and code that is in the form of procedures.

OOP are diverse, the most popular OOP language are class-based, that means objects are nothing but instances of classes. Object-oriented programming organizes program as are nothing but instances of classes.

Features of OOP: class, object, polymorphism and inheritance

Class:

A class is a collection or of group of similar object that have same properties, commons behavior and relationship. In OOP, objects are instance of classes. For example, a data type int that refers to integer is predefined in C++. Any number of variables can be created of type int as It is needed in the program. In a similar way, many objects of the same class can be defined. An object can be called an instance or an instantiation of a class because the object is an instance of the specification provided by the class.

Object:

Object is a concept or thing with define boundaries that is relevant to the problem we are dealing with. When approaching a programming problem in object-oriented programming, the problem is divided into objects and not functions as in structured programming. Thinking in terms of objects has a helpful effect on how easily the programs can be defined. Things that become objects in object-oriented programs are limited only to imaginations.

Polymorphism:

An abstraction denotes the essential characteristics of an object oriented programming which hides the internal detail of object. It focuses on the outside view of an object, and so serves separate an object's essential behavior from its implement and essential characteristics of some object, relative to the perspective of the viewer. Lets take an example of watch. It is constructed from many different parts. The abstraction allows the user to see the time without having detail knowledge of the complexity of the parts.

Inheritance:

Inheritance as the name suggests is the concept of inheriting or deriving properties of an existing class to get new class or classes. In other words we may have common features or characteristics that may be needed by number of classes. The main advantages of using this concept of inheritance in object-oriented programming is it helps in reducing the code size since the common characteristics is placed separately called as base class and it is just referred in the derived class.