MOST IMP QUESTIONS

1. Define algorithm and flowchart.
2. Explain and draw symbol of flowchart
3. Draw a flowchart and write algorithm to calculate area of circle.
4. Explain structure of c program.
5. Write advantages or features of c language
6. Explain primary data type with size and range
7. Write rules of variables
8. Define C-token, keyword and identifiers
9. Explain constant variable with example
10. Explain Type casting /Type conversion with example
11. Explain bitwise operator with example
12. Explain ternary or conditional with example
13. Explain logical operator with example
14. Explain goto statement with example
15. Explain switch statement with example
16. Explain if…ladder with example.
17. Explain for loop with example
18. Write difference between while loop and do…while loop
19. Explain break and continue.
20. Write program of different patterns
21. Define array.
22. Write characteristics of an array.
23. What is an array? How to declare and initialization of array.
24. Write a program to read 5 elements of array and display it.

1. Define algorithm and flowchart.

Ans:

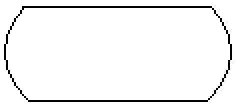
Algorithm: Algorithm is stepwise solution of any problem or program.

Flowchart: Flowchart is Graphical or Pictorial Representation of problem or algorithm.

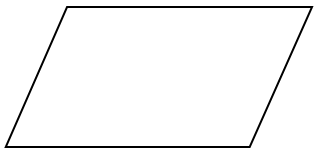
2. Explain and draw symbol of flowchart

Ans:

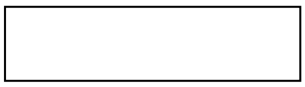
1. **Terminal**:
    - This Symbol is used for the Starting and Ending Process of Flowchart.



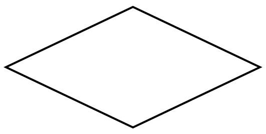
2. **Input / Output**
    - This Symbol is used for Receiving input Data From user and displaying output to the user



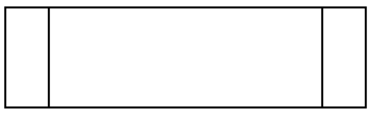
3. **Process**
    - This Symbol is used for Computing.
    - It is Use for Some Arithmetic Calculation.



4. **Decision Making**
    - This Symbol is used for making decision base on Condition.
    - This Symbol has one incoming arrow for entry and Two outgoing arrow.

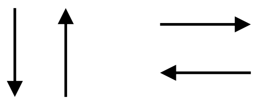


5. **Subroutine**
    - This Symbol is used for user defined function or sub program.

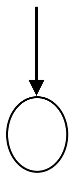


6. **Flow / Direction**
    - Flow line Symbol.

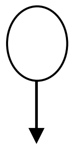
- This Symbol Indicates flow of data from one Symbol to Another Symbol.



7. **Page Break**
    - When flowchart is do not fit into single page then Page Break and Page Continue Symbol is used.
    - Page Break Symbol Indicate break in current page.



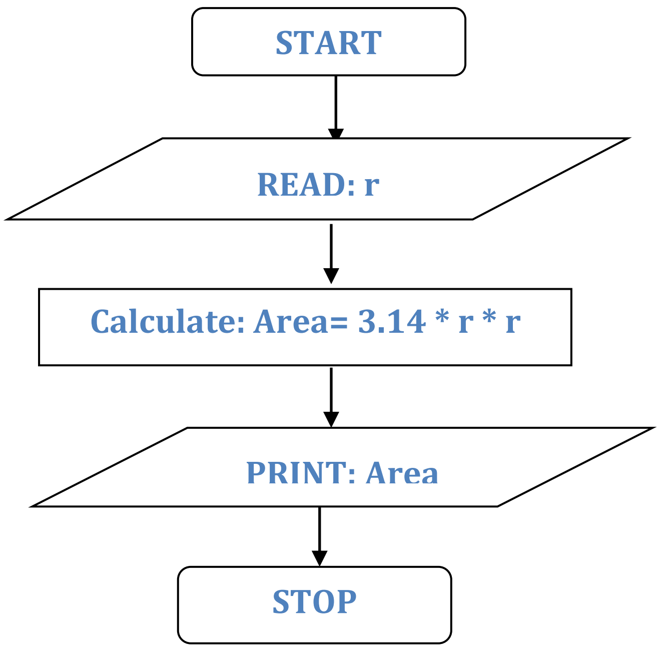
8. **Page Continue**
    - Page Continue Symbol Indicate Continue flowchart in another page.



3. **Draw a flowchart and write algorithm to calculate area of circle.**

Ans:

Flowchart



Algorithm:

1. **START**
2. **Read:** r
3. **Calculate: Area**= 3.14 * r *r
4. **PRINT**: **Area**
5. **STOP**

4. Explain structure of c program.

Ans:

Documentation section
Header File
Constants and Global variables Declaration
Main () function section
{
Statement(s);
}
User Defined functions and procedures with their body

Documentation section:

- ✓ Documentation consists of a set of comment line(/* */) giving the name of the program.

Header File(Link Section):

- ✓ Link section provides instruction to the compiler to link function from the system library(Header Files like #include<stdio.h>).

Constant Declaration:

- ✓ Definition section defines all symbolic constants (e.g.-#define pi 3.14).

Global declaration section:

- ✓ There are some variable that are used more than one function. Such Variables are called global variable and that is outside all the function.
- ✓ This section also declares the entire user define functions.

Main () function section:

- ✓ Every c program must have one main () function section.
- ✓ This section contains two parts. Declaration part and executable part.
- ✓ Declaration part declares all the variables used in the executable part.
- ✓ These two parts must be appearing between opening and closing braces.

User Defined functions and procedures:

- ✓ The subprogram section contains all the user defines functions that are called in the main function.

In C, all the statements must be terminated by semicolon (;).

5. Write advantages or features of c language
Ans:

a) C is portable.
'C' Program can be run on any hardware.

b) C does not support input/output statements.
'C' does not support input/output statement like input and print, but support library functions printf() and scanf().

c) C supports bitwise operators.
The C support bitwise operator like AND, OR, NOT etc. normally these operations do not supported by higher level languages.

d) C is modular language.
The program written in from of functions. A C program is group of one or more functions. Dividing the program into small functions makes it to easy develop and maintain the programs.

6. Explain primary data type with size and range
Ans: Data type is used to specify which type of value to be stored in variable.

Type	Size(Bytes)	Size(Bits)	Range	Control String
char	1	8	-128 to +128	%c
int	2	16	-32768 to +32767	%d
float	4	32	3.4 e -38 to 3.4 e +38	%f
double	8	64	1.7 e -308 to 1.7 e +308	%ld

7. Write rules of variables

Ans: A **variable** is a name which is used to store a temporary value.

- ✓ The variable name may consist of letters (A-Z, a-z), digits (0-9) and underscore symbol (_).
  Example: int no_1; / int no1; / int NO1;
- ✓ The variable name must begin with a letter. Some system permit underscore as a first character.
  Example: int 1NO; is not valid
- ✓ The length should not be more than eight characters.
- ✓ Variable name should not be a keyword.
  Example: int case is not valid
- ✓ A white space is not allowed in the name of variable.
  Example: int NO 1; is not valid.
- ✓ Variable is case sensitive. Uppercase and lowercase are different.
  Example: int MAX;int max;
  Here both MAX and max are treated as different variable.

8. Define C-token, keyword and identifiers

Ans:

C Token : The smallest individual unit in a C program is known as C token.

C has six types of token as given below:

- ✓ Keywords (float,while)
- ✓ Identifier(main,amount)
- ✓ Constants(-15.2, 10)
- ✓ String("abc", "hello")
- ✓ Special Symbols ([,] , *, ! etc)
- ✓ Operators (+,-, *,/)

Keyword: Keywords are those words whose meaning is predefined.

- ✓ The programmer cannot change the meaning of keywords.
- ✓ There are 32 keywords available in c.
- ✓ All the keywords must be written in lowercase.
- ✓ Examples:

Int	char	float	double
Long	short	signed	unsigned
If	else	do	while
For	break	continue	switch

Identifiers:

- ✓ Identifier are the words which are defined by the programmer in a program.
- ✓ Identifier refers to the name of variable, array, function etc.
- ✓ It consists of letters, digits and underscore.
- ✓ First character must be an alphabet or underscore.
- ✓ Keywords cannot be use as identifier.
- ✓ Identifier name are case sensitive.
- ✓ The name of identifier should be meaningful.

9. Explain constant variable with example

Ans: Constant means fixed value.

- ✓ If the value of a variable remain constant during the execution of a program, the we can declare the variable with **const** at the time of initialization.

Syntax:

```
const  DataType  ConstantName
```

Example: const float PIE = 3.14;

10.Explain Type casting /Type conversion with example

Ans:

- ✓ The type conversion are automatic as part of expression evaluation.
- ✓ But some times the user needs explicit type conversion.
- ✓ Example, int / int is always int. But, if we need the answer in float , then type casting is used.

Syntax:

```
(type _ name) expression ;
```

Example:

```
float average;

int sum=25, n=10;

average = (float) sum / n;
```

where sum / n results into 2.5 rather than 2, because the division is performed after converting sum in to float and the result of float / int is always float.

So, average =(float) sum / n;

= (float) 25 / 10;

= 25.0 / 10 = 2.5

11. Explain bitwise operator with example

Ans:

- ✓ The bitwise operators are used to manipulate data at bit level.
- ✓ These operators are used **for testing bits** or **shifting them left or right.**
- ✓ They **can not** be applied to **float and double.**

bitwise operators:

1) **Bitwise AND(&):**
2) **Bitwise OR(|):**
3) **Bitwise Exclusive OR(^):**
4) **Shift left(<<):**
5) **Shift right(>>):**
6) **One 's complement(~):**

Bit 1	Bit 2	Bit1 & Bit2	Bit1 | Bit2	Bit1 ^ Bit2
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Example: Bitwise AND(&)

```
x=5
y=7
z = x & y

    x  - -> 0000 0101
    y  - -> 0000 0111
  ____________________
    z  - -> 0000 0101
```

Example: Bitwise or(|)

```
x=5,    y=7
z = x & y

    x  - ->  0000 0101
    y  - -> 0000  0111
  ____________________
    z  - ->  0000  0111
```

Example: Bitwise Exclusive OR(^)

```
x=5,    y=7
z = x & y

    x  - ->  0000 0101
    y  - -> 0000  0111
  ____________________
    z  - ->  0000  0010
```

Example: 1's complement(~)

x=5,
z = ~ x

x - -> 0000 0101

---

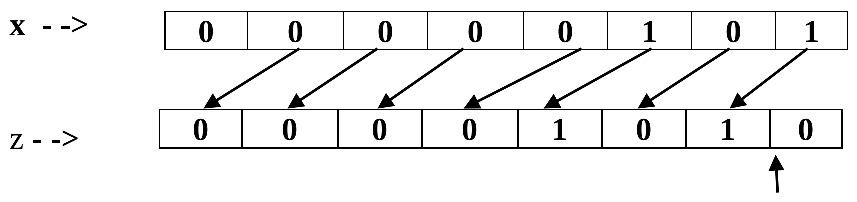
z - -> 1111 1010

Left Shift << operator shifts each bit of the operand left by one or more position.

Example:

z = x << 1;

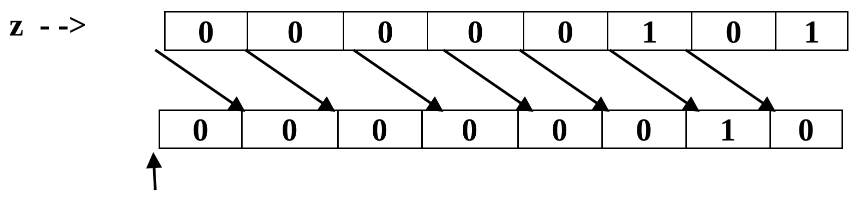
x - -> 0000 0101



Right Shift >> operator shifts each bit of the operand right by one or more position.

Example: **z = x >>1;**

x - -> 0000 0101



12.Explain ternary or conditional with example

Ans:

- ✓ The conditional operator is also known as ternary operator because it has three operands.
- ✓ The general form of ternary operator is follow:

  Syntax: e1 ? e2 : e3 ;

  where e1 is logical expression(condition) , and evaluated first.

- ✓ If the condition is true then the statement is followed the? Is executed(e2 is executed) otherwise the statement followed the : is executed(e3 is executed).

Example:

max=(x>y) ? x : y;

First (x>y) is checked, if it is true then x is assigned to max , otherwise y is assigned to max.

13.Explain logical operator with example

Ans:

- ✓ Logical operators are used when we want to combine more than one condition.

Logical AND	**&&**
Logical OR	**||**
Logical NOT	**!**

Logical AND(&&)

- ✓ The truth table logical AND (&&) operator is given below:

Condition 1	**Condition 2**	**Answer**
False	**False**	**False**
False	**True**	**False**
True	**False**	**False**
True	**True**	**True**

Logical OR(||)

- ✓ The truth table logical OR (||) operator is given below:

Condition 1	**Condition 2**	**Answer**
False	**False**	**False**
False	**True**	**True**
True	**False**	**True**
True	**True**	**True**

Logical NOT (!):

- ✓ The truth table logical NOT (!) operator is given below:

Condition	**Answer**
False	**True**

True	False

✓ **Examples :**
Suppose a=5, b=3, c=6;

1) (a>b) && (a>c) - false
2) (a>b) || (a>c) - true
3)!(a>b) - false

14.Explain goto statement with example
Ans:

✓ **Use:** the go to statement in the c programming is used to transfer the control unconditionally from one part of the program to other part of the program.
✓ 'c' language provide a **unconditional branching mechanism** called as go to statement.

Syntax:

```
Goto label ;
```

✓ Here , label is the label to the statement to which go to transfer control.
✓ Label must be a valid identifier.

✓ There are two possible use of goto statement.
1. **Forward Reference**
2. **Backward Reference**

Forward reference	**Backward reference**
goto label; …………. …………. Label: Statement;	Label: Statement; ……… ……….. Goto label;
Target statement comes after the goto.	Target statement comes before the goto.

15.Explain switch statement with example

Ans:

- ✓ The switch statement is also known as **multi-choice** or **multi decision statement**.

Syntax:

```
switch(variable name or expression)
{
Case  label1:
          statement(s) 1;
          Break;
Case  label2:
          statement(s) 2;
          Break;
                          .
                          .
Case  label N:
          statement(s) N;
          Break;
Default:
          Default statement(s);
}
```

- ✓ The expression or variable name is an integer or characters.
- ✓ Each case label value must be unique with a switch statement.
- ✓ Each case label must end with (:) .

The switch..case statement is executed in the following order:

- ✓ The value of the expression is compared against the value of case label.
- ✓ If the case is found ,then the statement associated with the case is executed. there is a single statement or multiple statement. Break statement send the control to the next statement after switch statement.
- ✓ If the case not found then the statement associated with the default case is executed.

Example:

```
#include<stdio.h>
#include<conio.h>
 void main()
 {
 int day;
 clracr();
 printf("Enter day number between (1-7)\n");
 scanf("%d",&day);

 switch(day)
 {

 case 1:
             printf("Sunday \n");
             break;
case 2:
             printf("Monday \n");
             break;
 case 3:
             printf("Tuesday \n");
             break;
 case 4:
             printf("Wednesday \n");
             break;
 case 5:
             printf("Thursday \n");
             break;
 case 6:
             printf("Friday \n");
             break;
 case 7:
             printf("Saturday \n");
             break;
 default:
         printf("Wrong input\n");

 getch();
 }
```

16.Explain else if…ladder with example
Ans:

- ✓ It is used **for multiple choice**.

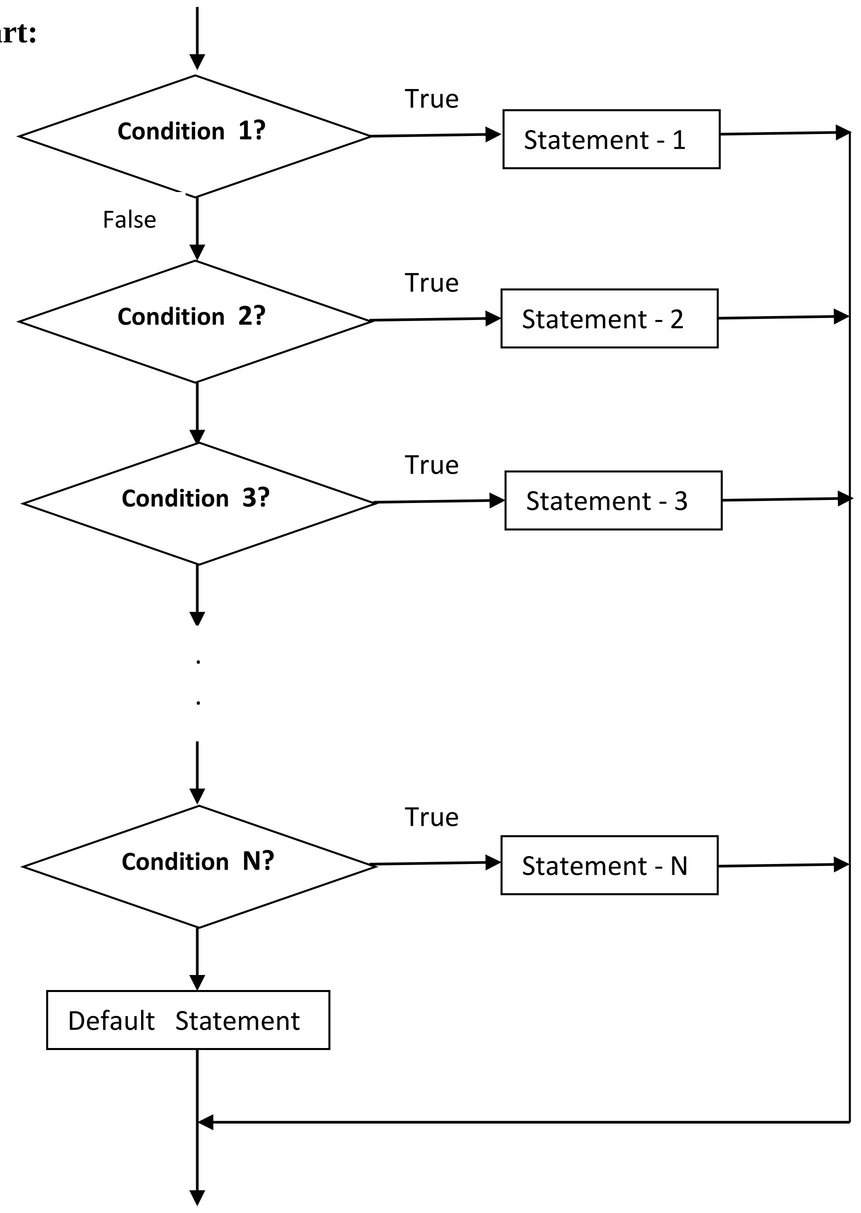
Syntax:

```
        if(condition-1)
        {
                  Statement - 1;
        }
        else if(condition-2)
        {
                  Statement - 2;
        }
        else if(condition-3)
        {
                  Statement - 3;
        }
                .
                .
        else if(condition-N)
        {
                Statement -N;
        }
        else
        {
                Default statement;
        }
```

if-else-if ladder is executed in the following order:

1. First condition-1 is executed , if the condition-1 is true then the statement-1 is executed.
2. if the condition-1 is false then condion-2 is checked. If condition-2 is true then statement-2 is executed.
3. This procedure repeated until all the condition is checked. if all the condition became false then the default statement is executed.

Flowchart:



Example:

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int no;
    printf("enter number\n");
    scanf("%d",&no);
    if( no > 0 )
    {
    printf("Number is Positive");
    }
    else if (no  < 0)
    {
    printf( "Number is Negative" );
    }
    else
    {
    printf( "Number is Zero" );
    }
    getch();
}
```

17.Explain for loop with example.

Ans:

Syntax:

```
For  (Initialization ; Condition ; Increment or decrement)
{
Body of the loop;
}
```

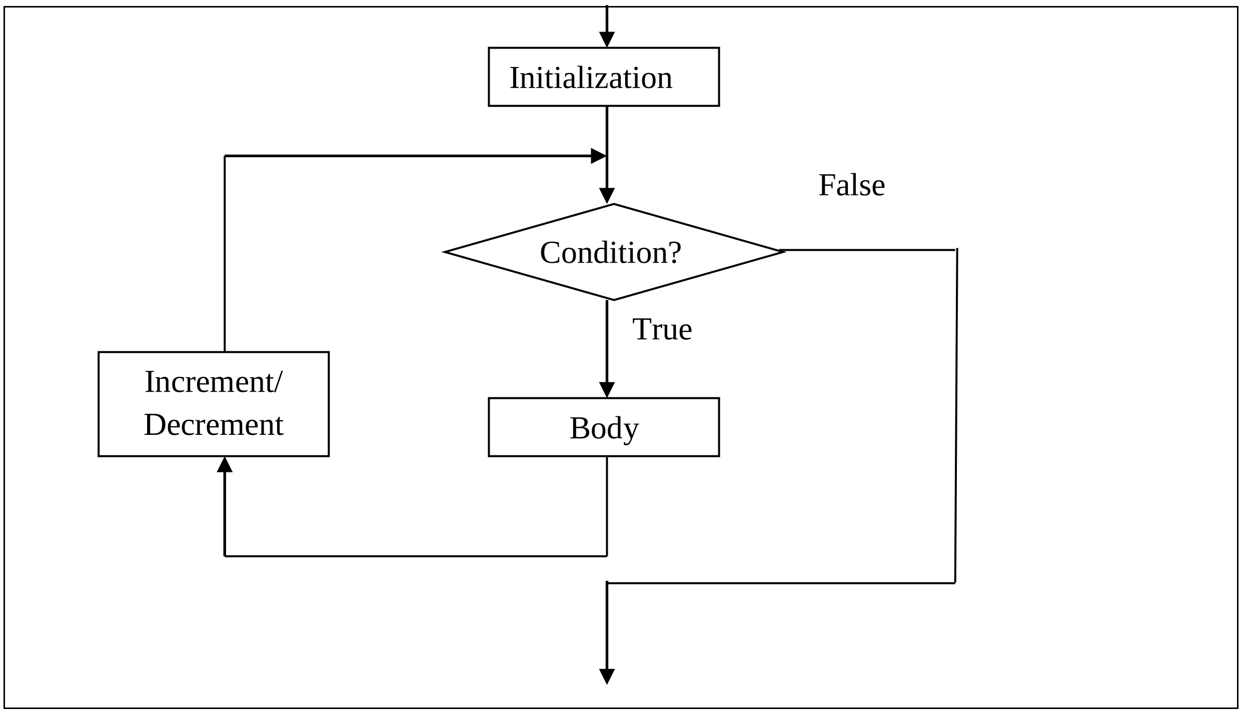
- ✓ **Initialization:** The control variable is initialized in the initialization expression.
  We require one variable which is used to control the loop, which is called as control variable.

It executes only once.

- ✓ **Condition:** The condition statement checks the value of the control variable. If the condition statement is true, the body of the loop is executed.
- ✓ **Increment/Decrement:** The increment/decrement of the control variable is done in this part. After the incrementing/decrementing the value of control variable, it is tested using condition if condition is true than again the body of loop is executed and this process is repeated until the condition become false.

If the body of the loop contains only one statement, then { } is not required.

Flowchart:



Example: Print 1 to 5 numbers using for loop.

```
for(i = 1; i < =5 ; i++)
{
        printf(“ %d  \n “, i ) ;
}
```

18.Write difference between while loop and do…while loop
Ans:

While	**Do--while**
It is entry control loop, means first condition is checked, and if it is true then body of the loop executes, otherwise next statement after the loop will be executed.	It is exit control loop, first body is executed and then condition is checked and if the condition is true, then again body of the loop will be executed, otherwise next statement after the loop will be executed
At first time, condition is false, then body is not executed.	At first time, condition is false, at least once the body is executed.

Syntax:
while (condition)
{
Body of the loop;
}
	Syntax:
	Do
	{
	Body of the loop
	}while(condition);

Example:
i =1;
do
{
printf("% d \n " , i);
i = i+1;
} while (i < = 10);
	Example:
	i =1;
	do
	{
	printf("% d \n " , i);
	i = i+1;
	} while (i < = 10);

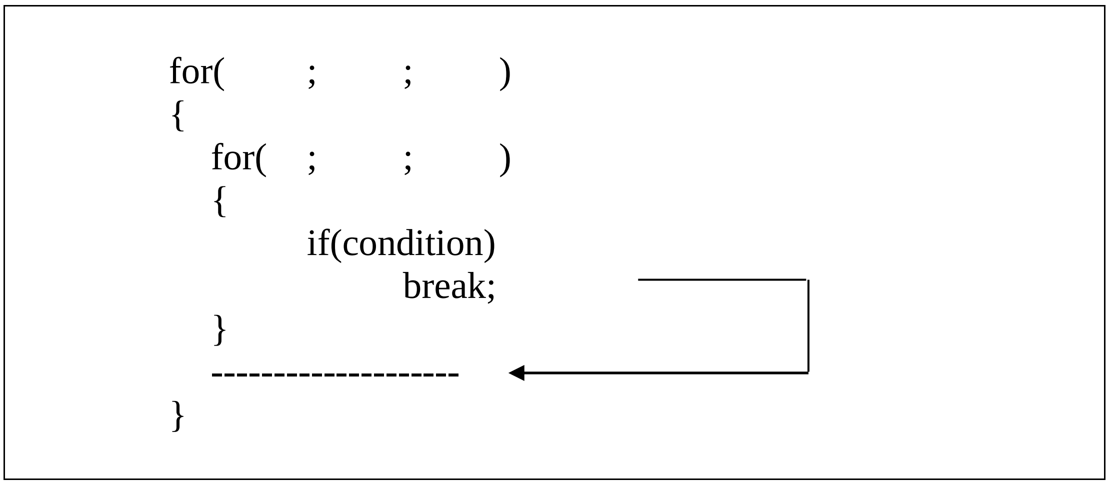
19.Explain Break and Continue

Ans:

Break: When infinite loop executes, the program doesn't terminates.

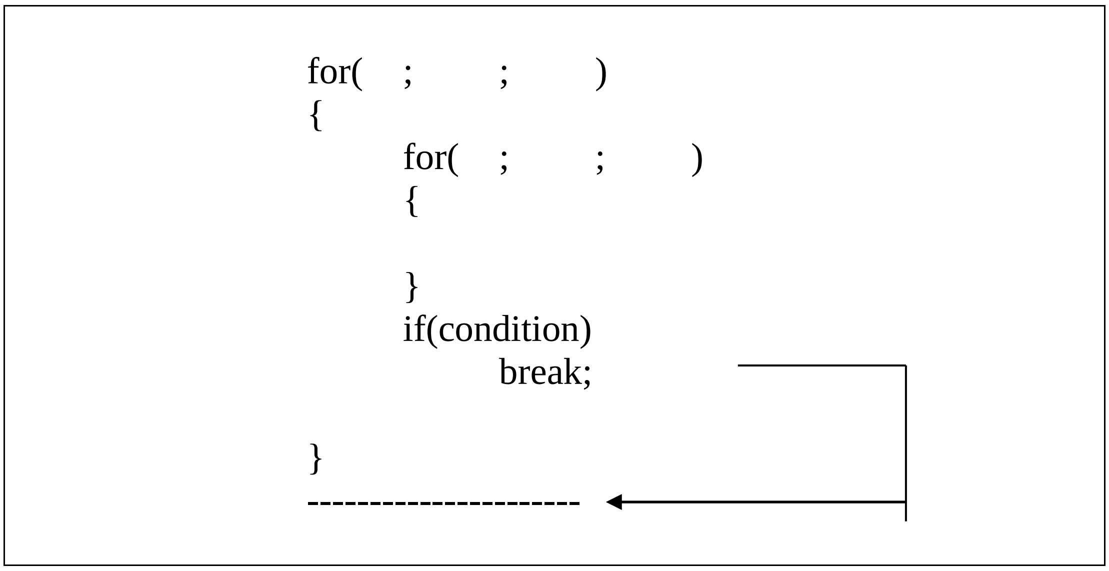
- With the use of break statement, we can terminate from the loop.
- The break statement breaks the loop, in which it appears.

- **Break in inner loop**:



```
for(     ;     ;     )
{
   for(  ;     ;     )
   {
         if(condition)
               break;
   }
   --------------------
}
```

- **Break in outer loop**:



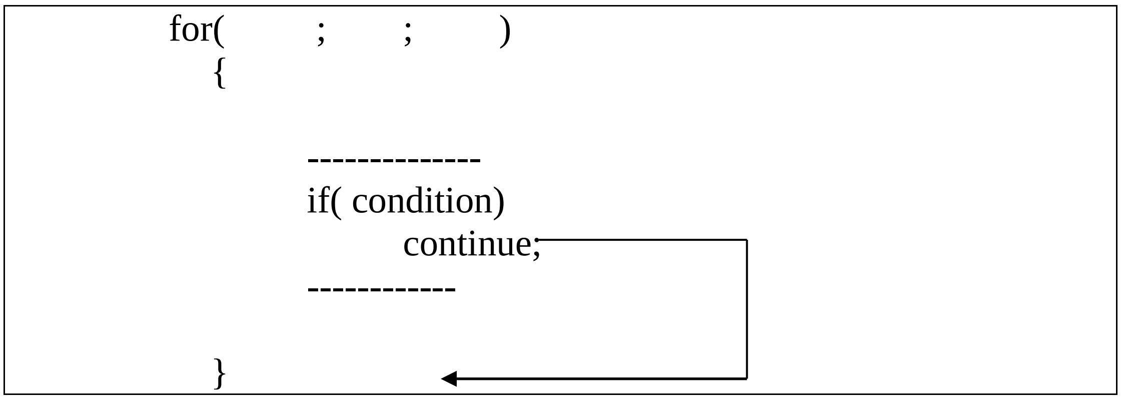
```
for(  ;     ;     )
{
      for(  ;     ;     )
      {

      }
      if(condition)
            break;

}
----------------------
```

Continue: When continue executes, skips the remaining part of the loop for current iterations and start the next iteration.

- When continue is placed with if statement in a loop, as soon as condition is true, continue executes and sends control after the last statement of the body of the loop, means completion of current iteration and control goes to beginning of the loop for next iteration.



```
for(       ;     ;     )
   {


         --------------
         if( condition)
               continue;
         ------------

   }
```

Example: Print 1 to 10 number except 5.

```
for( i=1 ; i <= 10 ; i++)
{
        if ( i ==5)
            continue;
        printf("%d  \n" ,i);
}
```

20.Write program of different patterns
Ans:

(1) W.A.P. to print

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

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

(2) W.A.P. to print

```
1
1 2
1 2 3
1 2 3 4
```

```
1 2 3 4  5
```

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

(3) W.A.P. to print

```
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
```

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

(4) W.A.P. to print

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

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

(5) W.A.P. to print

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

```
#include<stdio.h>
#include<conio.h>
void main()
 {
int i,j,k=1;
clrscr();
for(i=1;i<=5;i++)
{

                        for(j=1;j<=i;j++)
                        {
```

```
                    printf("%d ",k);
                    k++;
                    }
printf("\n");
}
getch();
}
```

21.Define array.

Ans: An array is a collection of variables of same data type known by a same name.

- ✓ Array can be divided into two types.
  1. One dimensional array.
  2. two dimensional array

22.Write characteristics of an array.

Ans:

- ✓ Array store elements that have same data type.
- ✓ Array store elements in subsequent memory location.
- ✓ Array size should be mention in the declaration.
- ✓ Array name represent the address of starting elements.

23.What is an array? and how to declare and initialization of array.

Ans:

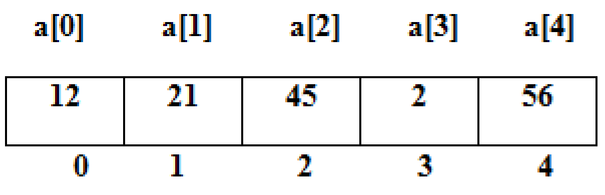
An array is a collection of variables of same data type known by a same name.

Declaration of array:

```
Syntax:    Data type  arrayname [size] ;
```

Data type: it defines the type of the array element, like int,float,char,double etc.
Array name: it is the name of array.
Size: It represent the size of the array.

```
        Example:   int a[5];
```



Initialization of array:

1) **At Compile Time:**

Syntax:

Data Type arrayname[size]={list of value};

Example:

int A[4]={1,2,3,4};

Initialize array A as follow.

A[0]=1
A[1]=2
A[2]=3
A[3]=4

2) **At Run Time:**

Example: To read 4 element from keyboard.

```
for( i = 0; i<4; i++)

{

        scanf(“ %d” ,& a[i] ) ;

}
```

24.Write a program to read 5 elements of array and display it.
Ans:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[5],i;
clrscr();
printf("Enter any 5 elements of array");
for(i=0;i<5;i++)
        scanf("%d",&a[i]);

printf("The array elements are:\n");

for(i=0;i<5;i++)
        printf("%d\n",a[i]);
getch();
}
```

Output

```
Enter any 5 elements of array
11
22
33
```

```
44
55
The array elements are:
11
22
33
44
55
```