Unit – II

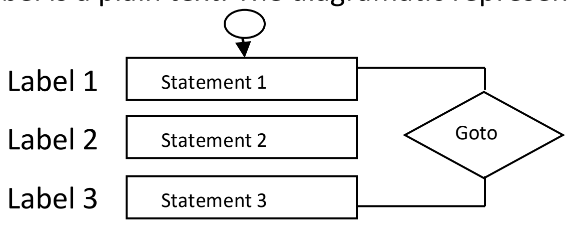
Decision Control & Looping Statements

Goto Statement: A goto statement in “C” programming provides an unconditional jump form goto to a labled statement in the same function.

Syntax: The syntax for a goto statement in C is as follows.

```
1. Forward Direction                 2.   Backward Direction
   Goto label;                            label:statement;
   --------------                         ---------------------
   ---------------                        ---------------------
   Label:statement;                       goto label;
```

In the above syntax label is a plain text. The diagramatic representation is as follows.



Lets us consider an example “C” program using goto statement.

```
#include<stdio.h>
void main()
{
        int a = 1;
loop:   while ( a<=5)
        {
        if(a==4)
        {
                a=a+1;
                goto loop;
        }
        printf(“ a value is %d”,a);
        a++;
        }
}
```

The above program will produce the following output

```
a value: 1
a value: 2
a value: 3
a value: 5
```

Nesterd loops:

C Programming allows to use one loop inside another loop is know as nested loops.

The following are some of the examples of nested loops.

Syntax:

1. The syntax for a nested for loop statement in C as follows.

```
For(initialization;condition;incr/decr);
{
        --------
        For(in initialization;condition;incr/decr);
        {
                -------
                -------
        }
}
```

2. The syntax for a nested while loop statement in C as follows

```
while (condition)                                           ----------
{                                                           ----------
        -----------                                         }
        ------------                                        -------------
        while(condition);                                   -------------
        {                                           }
```

3. The syntax for a nested “do.. while loop” statement in C as follows.

```
do
{
        -----------
        -----------
        do
        {
                -----------
                -----------
        }while(condition);
```

Let us consider a simple nested loop program as follows.

```
#include<stdio.h>
void main()
{
        int i,j;
        for(i=1;i<=5,i++);
        {
                for(j=1;j<=5;j++);
                {
                        printf(“%d%d”,i,j);
                }
        }
}
```

Decision Making and Branching Statements:

Java programs contains sequence of instructions. If we want to change the orders of execution i.e., when a program break the sequential flow and jump to another part of code is called Branching. When the branching is made based on a condition then it is known as conditional branching.

The conditional branching statements are

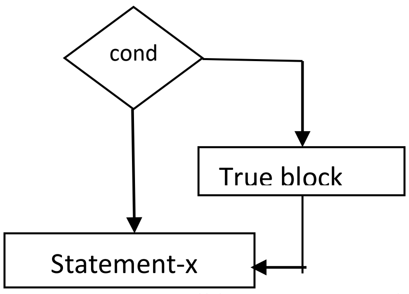
1. If statement
2. Switch statement

1. **If statement:** The if statement is implemented in different forms based on the complexity of a program.
    a. Simple if
    b. If else
    c. If else if
    d. Nested if

a. **Simple if:** the general form of if statement is as follows.

Syntax:

```
if(condition)
{
        true block
}
        statement x
```

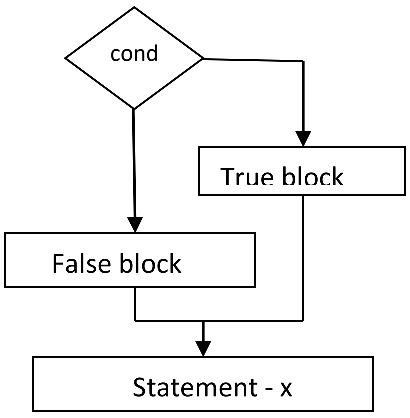


In simple if, if the test condition is true then true block will be executed and statement –x will be executed otherwise true block will be stopped and the execution will be jumped in to statement – x.

b. **If else:** if else statement is an extension of simple if statement. The general syntax of if else statement is as follows.

Syntax:

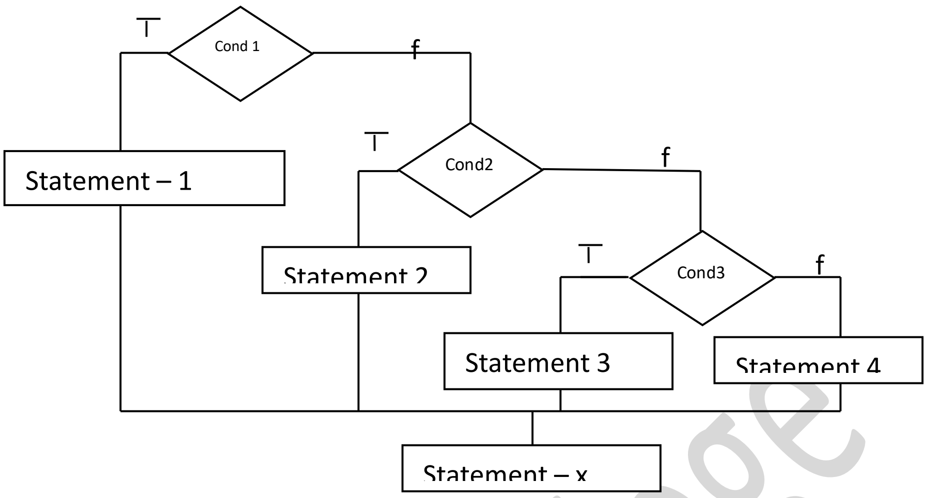
```
if (condition)
 {
        true block
}else
{
        false block
}
        statement - x
```



In if else statement, if the test condition is true, then true block will be executed and the statement – x will be executed. Note two blocks will not executed only one block will executed.

c. **If else if:** There is another way of putting if else statements to gether when multiple decisions are involved. If else if statement is a chain of if else statements. The general form of if else if statement as follows.

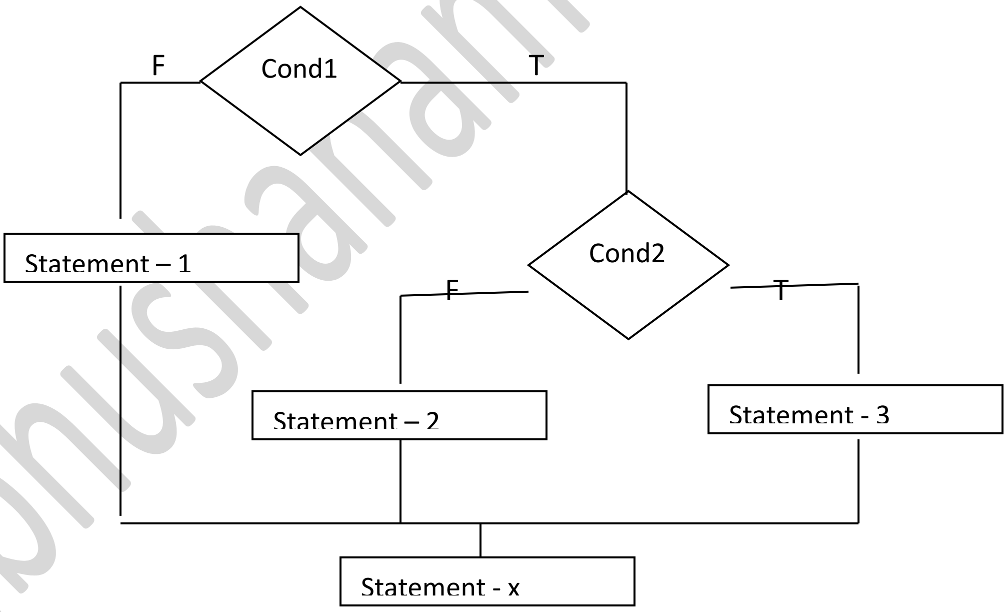
```
Syntax: if (condition- 1)
                statement 1
        else if(condition – 2)
                statement 2
        else if(condition – 3)
                statement – 3
        else if(condition – n)
                statement – n
        statement – x
```



In if else if statement, it the test condition is true statement 1 will be executed and statement – x will be executed. If it is false, condition – 2 will be checked. If condition – 2 is false, true statement – 2 and statement – x will be executed. If condition 2 false, condition 3 will be executed. If it is false statement 4 and statement x will be execute.

d. **Nested if:** when a series of decisions are involved we may have to use more than one if else statements in nested form as follows.

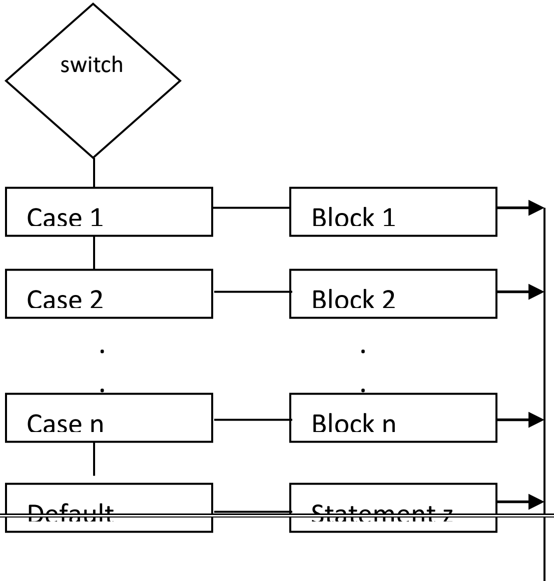
```
Syntax:
if(condition – 1)
if(condition – 2)
     statement – 1
     else
     statement – 2
     else
     statement – 3
statement – x
```



In nested if statement, if the left condition 1 is true, condition 2 will be checked, if condition 2 is true statement 3 and then statement x will be executed. If condition 1 is false, statement 1 will be executed and then statement x will be executed. If condition 2 is false statement 2 will be executed and statement x will be executed.

2. **Switch Statement:** In java language there is another way to make multipath decisions are involved or character value and test against a list of case values, when a match is found a block of statements will be executed. Otherwise, default block will be executed. The general syntax of switch statement is as follow.

```
switch(expression)
{
        case1:     block;
                   break;
        case2:     block;
                   break;
        .          .
        .          .
        case n:    block;
                   break;
```



```
default:statement z;
}
statement – x
```

4. **Looping statements or iterative statements:**

In looping a block of statements are repeated again and again until the test condition becomes false. A loop consist of two segments. The first is a condition and second is body of loop.

In C language loop construct as follows

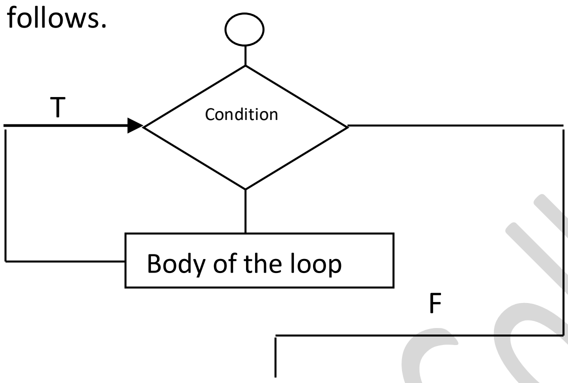
- Entry control loop – while , for
- Exit control loop -- do while

While loop:

The simplest of all the looping structures in the is while loop statement. The while loop is entry controlled loop. The general syntax is as follows.

Syntax:

```
while(condition)
{
        ………
        ………
        Body of the loop
        ……….
        ……….
}
```



If the test condition is true then body of the loop is executed. After execution of body, the test condition is checked again and if it is true the body of the loop is executed again. This process continuous until the test condition becomes false.

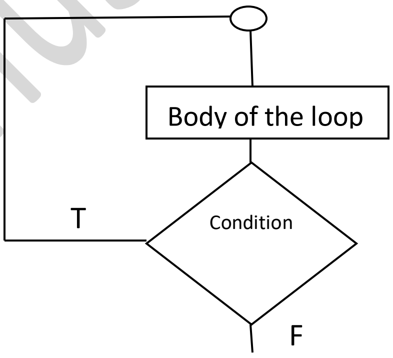
```
Ex:     i=1;
        while(i<=5)
        {
            printf("%d",i);
            i++;
        }
```

Do while loop statement: the while loop checks the condition first, if it is true body of the loop is executed. Therefore the body of the loop may not executed if the condition is false.

In same situations, it is necessary to execute boy of the loop atlease once. This type of situations is handle by do while loop statement. The general syntax is as follows.

Syntax:

```
do
{
        ……….
        ……….
        body of loop
        ………
        ………
}while(condition);
```



In do while loop first body of the loop will be executed and then the test condition is checked if it is true, boy of the loop is executed once again. This process continuous until the test condition becomes false.

```
Ex:     i=1;
        do
        {
            printf("%d",i);
            i++;
        }while(i<=5);
```

For loop statement: The for loop is another entry controlled loop. It provides a more concise looping structure. The general syntax is as follows.

```
Syntax:   for(initialization; condition; increment/decrement)
          {
                  …………
                  ………
                  body of the loop
                  …………
          }
```

```
ex:     for(i=1;i<=5;i++)
        {
          count<<i
        }
```

In for loop initialization of the variable is done first, then the variable is tested using a condition. If it is true, body of the loop is executed, otherwise loop is terminated. If the body of the loop is executed the variables value is incremented or decremented.

Nested looping structure:

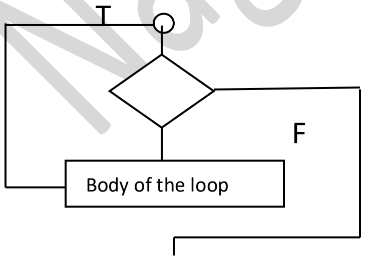
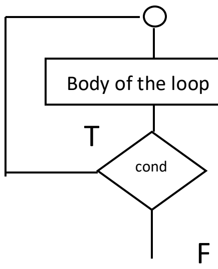
Nesting of loops i.e., one loop statement defined inside another loop statement.

```
Ex:     while(condition)
        {
          while(condition)
          {
                ……..
                ………..
                body of the loop
                ………
          }
        }
Ex:     for(initialization; condition; increment/decrement)
        {
          for(initialization; condition; increment/decrement)
          {
                ……………..
                ……………..
                Body of the loop
                ……………..
          }
        }
```

Differences between while and do while loops:

While	**Do while**
1.While loop is an entry controlled loop	1.Do while loop is an exit controlled loop
2. In while loop, the test condition is placed on the top of the loop.	2. In do while loop, the test condition is placed on the bottom of the loop.
3. The test condition will be checked at first.	3. The test condition will be checked at last.
4. Body of the loop will be executed only if test condition is true.	4. Body of the loop will be executed at least once even test condition is false.
5. it is also called as pre-testing loop.	5. It is also called as post-testing loop.
6. The while loop doesn't have semicolon to teminate the loop.	6. The do while loop condition have semicolon to terminate the loop.

7. Syntax: while(condition)
{
Body of the loop
}
	7. Syntax: do
	{
	Body of the loop
	}while(condition);

8. 	8. 

9. Ex: i=1
while(condition)
{
count <<i;
i++;
}
	9. Ex: do
	{
	printf("%d",i);
	i++;
	}while(condition);

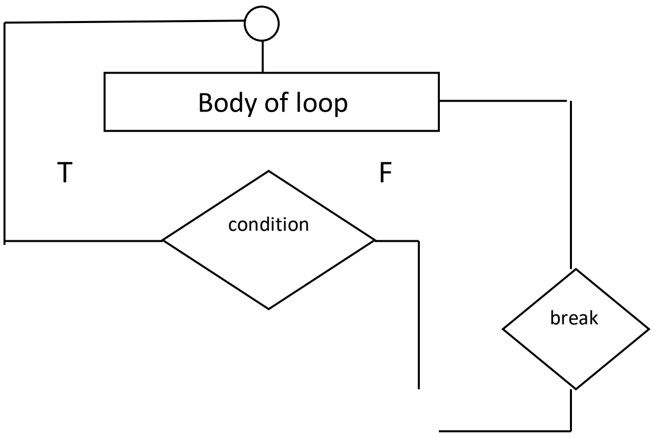
Break & Continuous key words:

Introduction: There are two statements built in C++ programming language. There are two keywords to alter or modify the normal flow of a program loops perform repeating a block of statements until the test condition becomes false. But in some situations it is need to skip some statements inside the loop or terminate the loop immediately without checking any condition. In such cases, break and continue statements are used.

Break: In C++ programming, break is used to terminate the loop immediately without checking any condition. The break statement is also used to switch statement to exit case statement.

Syntax: Break;

The break statement is used in all 3 looping statements. The diagram representation of break statements is as follows.



The diagram explains the working of break statement in all 3 loops.

While loop:

```
while(condition)
{
        statement
        if(condition)
        {
                break;
        }
}
```

do while loop:

```
do
{
        statements
        if(condition)
        {
                break;
        }
        statements
}while(condition);
```

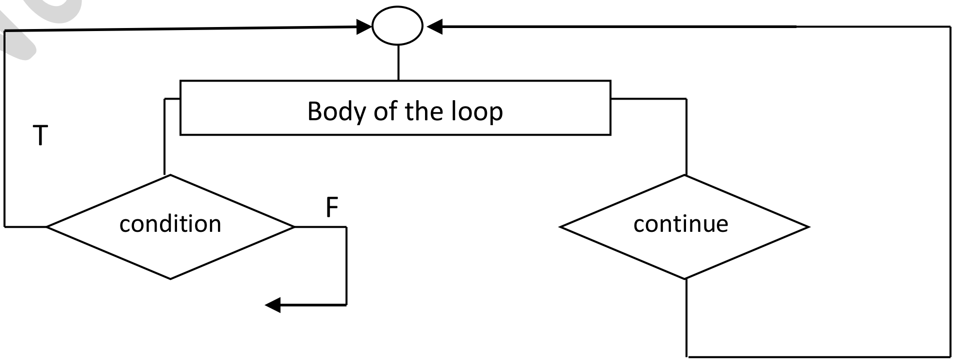
for loop:

```
for(initialization;condition;increment/decrement)
{
        statements;
        if(condition)
        {
                break;
        }
        statements;
}
```

Continuous Statement: In some times it is needed to expect or skip some statements inside the loop. In such situations continuous statement is used.

Syntax: continue

Just like break statement, continue is also used in looping statements. The diagrammatic representation of continue statement is as follows.



For the better understanding of how continue statement works analyze the following diagrams using continue statements.

While loop:

```
while(condition)
{
        statements
```

```
if(condition)
{
        continue;
        }
        statements
}
```

do while loop:

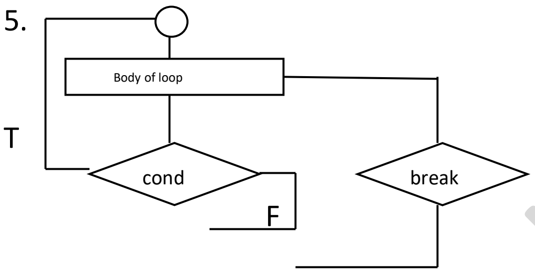
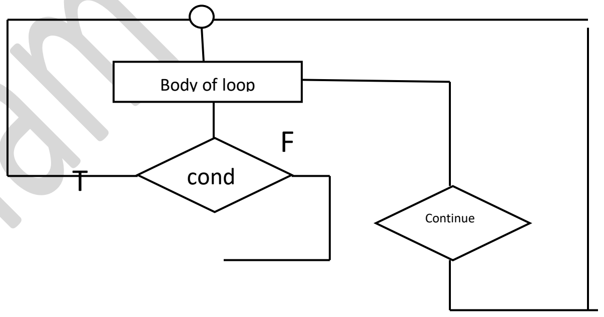
```
do
{
```

for loop:

```
for(initialization;condition;increment/decrement);
{
        statements;
        if(condition)
        {
                continue;
        }
        statements
}
```

```
        statements
        if(condition)
        {
                continue;
        }
        statements
}while(condtion);
```

Explain the difference between break & continue?

Break	Continue
1.Break statement is used to stop execution of a loop without checking any condition. 2. Break statement is used in loops switch & if statements. 3. break statement is used to exit from a loop or switch case statement. 4. syntax: break; 5.  	1. Continue statement is used to skip execution of a loop without checking any condition 2. continue statement is used loop, if statement 3. continue statement is used to return to the beginning of loop. 4. syntax: continue; 5. 

Examples:

```
while(condition)
{
        statements
        if(codition)
        {
                break;
        }
        statements
}
do
{
        statements
        if(condition)
        {
                break;
        }
        statements
}while(condition);
for(initialization;condition;inc/dec)
{
        statements
        if(condition)
        {
                break;
        }
        statements
}
for(i=1;i<=10;i++)
{
```

```
        if(i%5==0)
        {
                break;
        else
        printf("%d",i);
        }
}


while(condition)
{
        statements
        if(condition)
        {
                continue;
        }
        statements
}
do
{
        statements
        if(condition)
        {
                continue;
        }
        statements
}while(condition);
for(initialization;condition;inc/decr)
{
        statements
        if(condition)
        {
                continue;
        }
        statements
}
for(i=1;i<=10;i++)
{
        if(i%==5)
        {
                continue;
        else
                printf("%d",i);
        }
}
```