Chapter: 5
Control Structure

Advanced College of Engineering And Management.

Topics to be covered

Introduction

Type of Control Structure

- ❑ Branching:
  - ✓ if
  - ✓ if else
  - ✓ if elseif and
  - ✓ switch
- ❑ Looping:
  - ✓ for loop
  - ✓ while loop
  - ✓ do while loop
- ❑ Jumping:
  - ✓ goto
  - ✓ break
  - ✓ continue

Nested Control Structure

Introduction

- ✓ The statement which alter flow of execution of a program are known as control statements.
- ✓ Sometimes tasks are performed on the basis of certain logic test where output comes according to true or false tests or conditions.
- ✓ Similarly , sometimes it is necessary to perform repeated actions or skip some statements from execution. For these operations, control statements are needed.
- ✓ They control flow of program so that it is not necessary to execute statements in the same order in which they appear in the program

Types of Control statements

- Selective (Branching)

  a) Two way branching : if...else

  b) Multiple branching : if...else if...elseif and switch
- Repetitive (Looping)

  a) Entry controlled : while and for

  b) Exit controlled : do...while
- Jumping:

  a) goto

  b) break

  c) continue

Selective (Branching) Statements

- ✓ Selective structure are used when we have a number of situations where we may need to change the order of execution of statements based on certain condition.
- ✓ The decision making statements test a condition and allow to execute some statements on the basis of result of the test (i.e. either true or false).
- ✓ C provides the following statements for selective structures.

  a) Two way branching : if...else

  b) Multiple branching : if...else if...elseif and switch

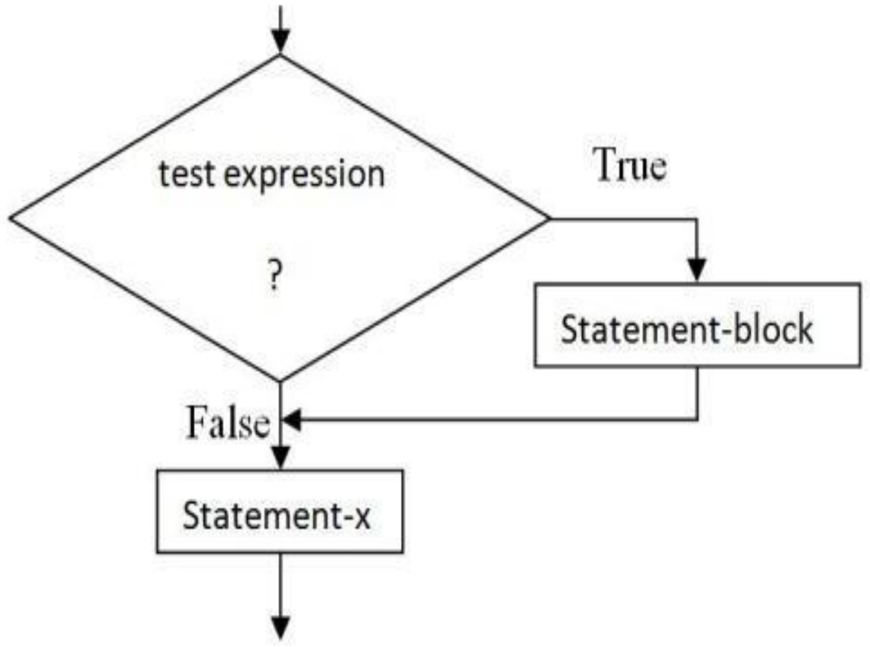
The if statement

- ✓ The **if** statement is used to execute a block of code conditionally based on whether the given condition is true or false.
- ✓ If the condition is true , the block of code will be executed, otherwise it will be skipped.

Syntax:

```
if (test expression)
        {
                statement-block;
        }
statement-x;
```

Example



Example: Write a program that prompts a user to input balance in his/her bank account and 10% bonus is given if his/her balance is greater than or equal to 20000.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    float bal;
    printf("Enter Balance:");
    scanf("%f", &bal);
    if(bal>=20000)
    {
        bal = bal + bal*0.1;
    }
    printf(“Balance = %.f", bal);
    getch();
}
```

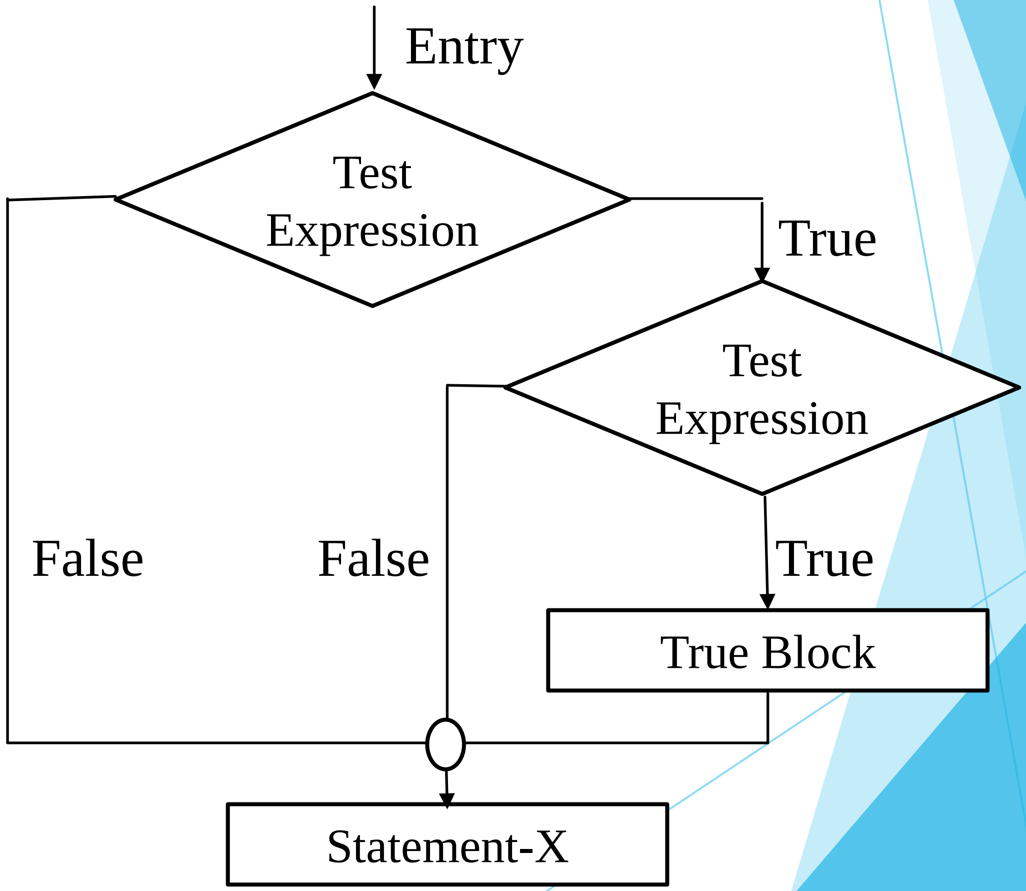
Nested if statement

- ✓ If a ***if*** statement is written within the body of another ***if*** statement then it is called Nested ***if*** statement

Syntax

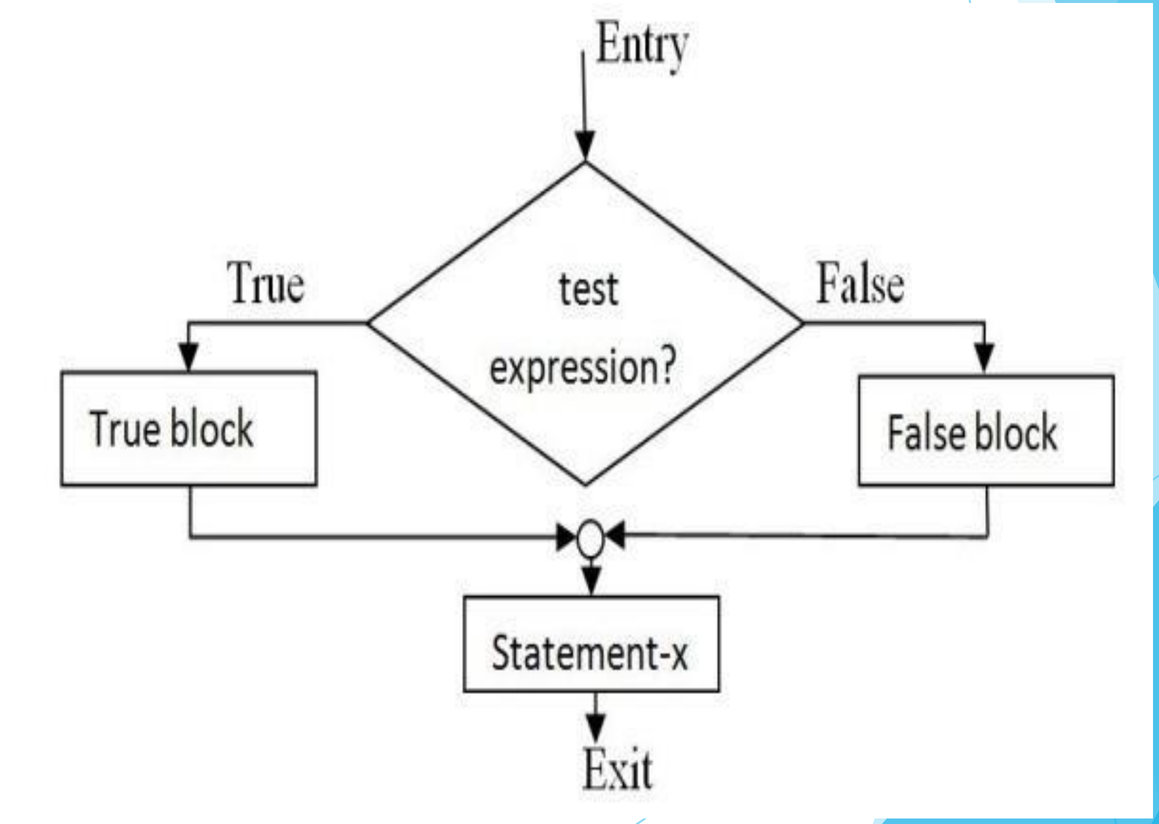
```
if(test expression)
{
      if(test expression)
      {
            true block statement(s)
      }
}
```

Flowchart



The if … else statement

- ✓ The if...else statement extends the idea of the if statement by specifying another section of code that should be executed only if the condition is false

Syntax:	Flowchart
`if(test expression) { true block statement(s) } else { false block statement(s) }`	 Entry test expression? True False True block False block Statement-x Exit

```
if(test expression)
 {
       true block statement(s)
 }
 else
{
       false block statement(s)
}
```

Example: Program to check if given number is odd or even.

```
void main()
{
        int num;
        printf("Enter a num");
        scanf("%d", &num);
        if(num%2==0)
        {
              printf("Even");
        }
        else
        {
              printf("Odd");
        }
}
```

Nested if … else Statement

- ✓ When a series of decision are involved, we may have to use more than one if...else statement in nested form.

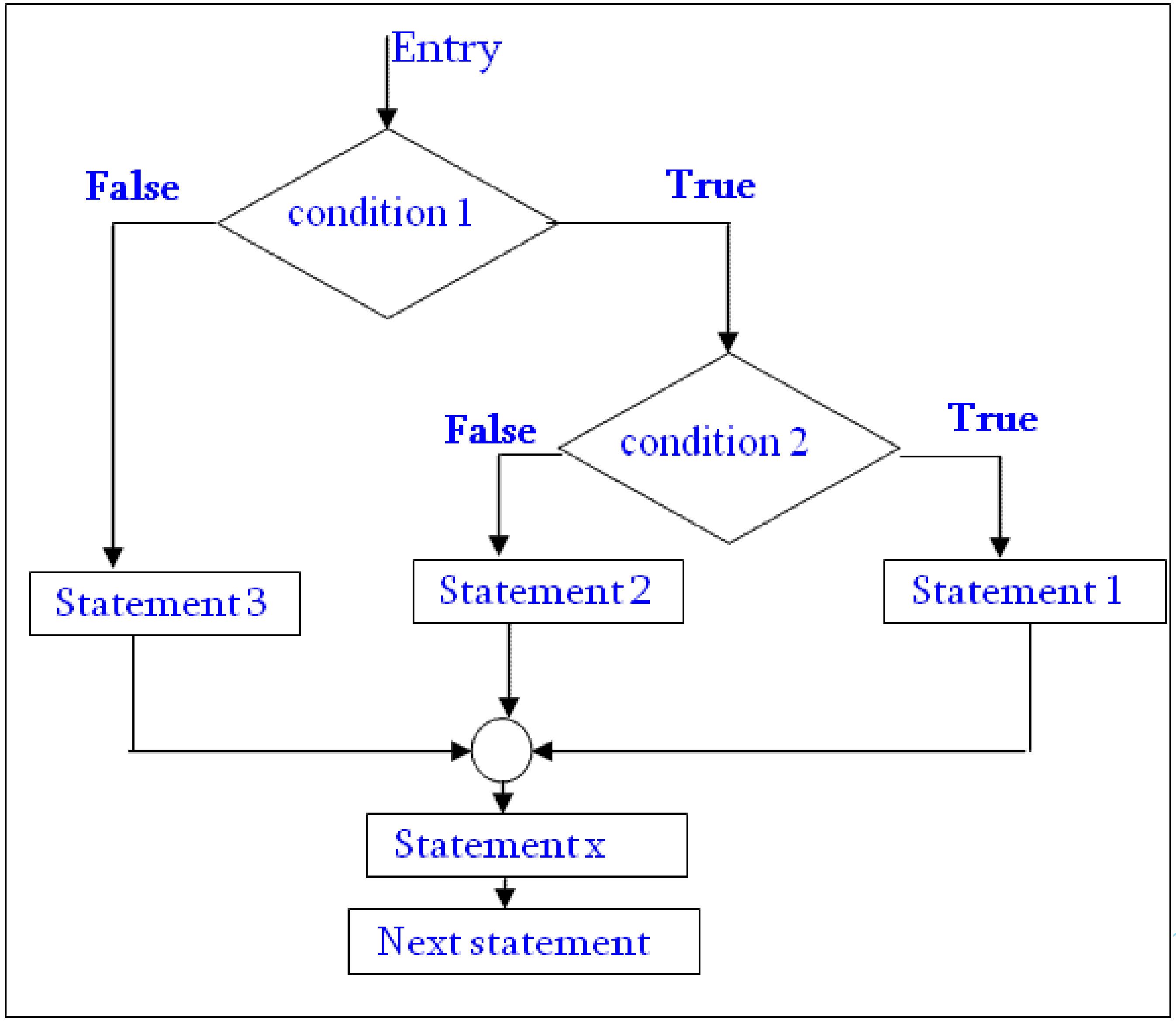
Syntax1:

```
if(test-expression1)
{
    if(test-expression2)
    {
        statementblock-1;
    }
    else
    {
        statementblock-2;
    }
}
else
{
statementblock-3;
}
```

Syntax2:

```
if(test-expression1)
{
     statementblock-1;
}
else
{
    if(test-expression2)
    {
        statementblock-2;
    }
    else
    {
        statementblock-3;
    }
}
```

Flowchart


Entry
condition 1
False
True
condition 2
False
True
Statement 3
Statement 2
Statement 1
Statement x
Next statement

Example : Q1. WAP to find largest of three different integer number just entered by user.

```
void main()
{
        int a, b, c;
        printf("Enter three numbers: ");
        scanf("%d%d%d", &a, &b, &c);

        if(a>b)
        {
              if(a>c)
              {
                     printf("\nThe largest no. is %d", a);
              }
              else
              {
                     printf("\nThe largest no. is %d", c);
              }
         }
        else
        {
                if(b>c)
                {
                          printf("\nThe largest no. is %d", b);
                }
                else
                {
                           printf("\nThe largest no. is %d", c);
                }
          }

}
```

The if ..elseif statements (elseif Ladder)

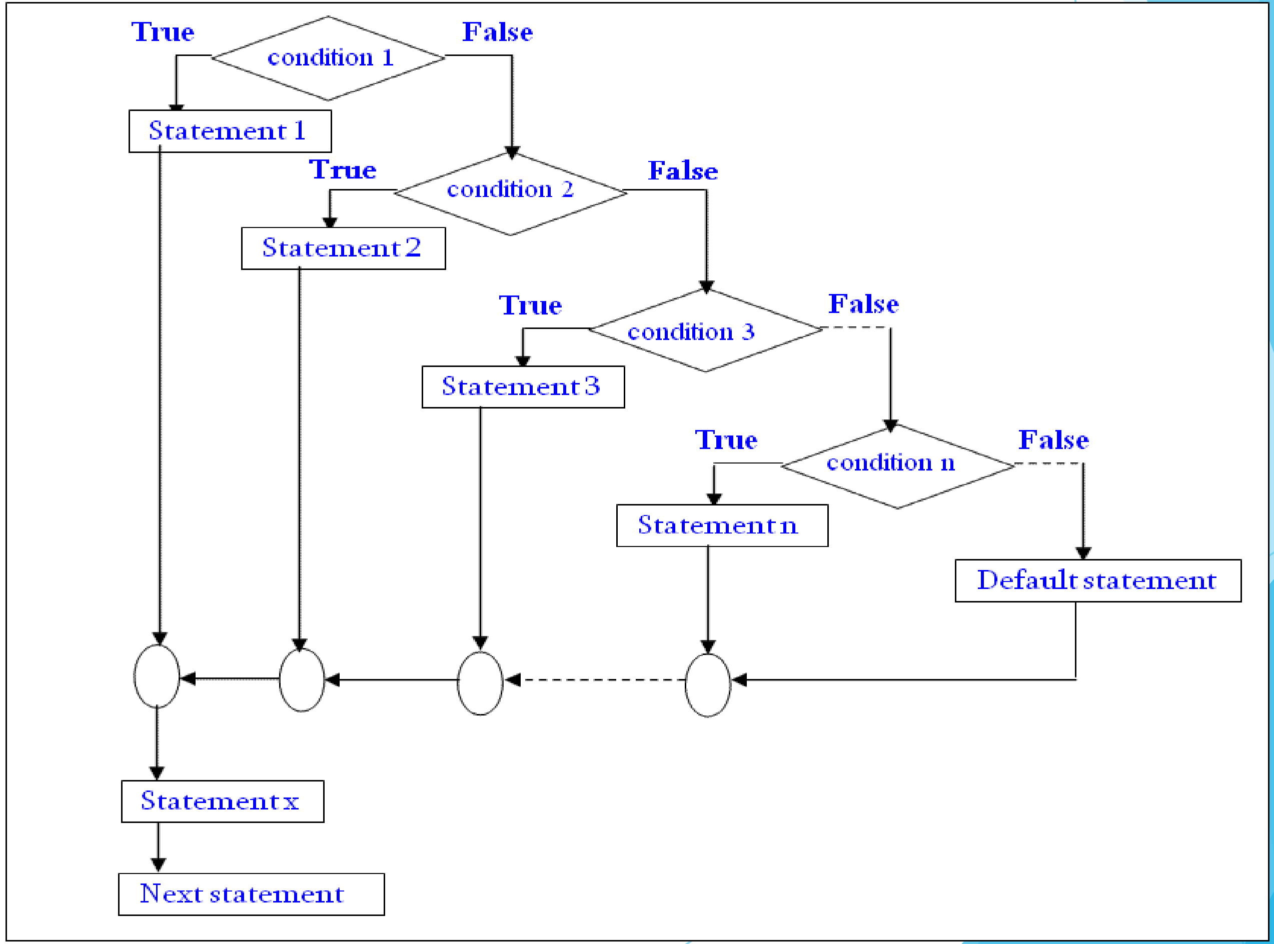
- ✓ It is another way of putting **ifs** together when multipath decisions are involved. A multipath decision is a chain of **ifs** in which the statement associated with each **else** is an **if**

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;

statement-x;
```

Flowchart:



Example:

WAP to read the marks secured by a student and display the appropriate message as follows:

- Marks greater or equal to 40 and less than 65 display PASS.
- Marks greater or equal to 65 and less than 80 then display 1st division.
- Marks greater or equal to 80 then display Distinction.
- Otherwise failed.

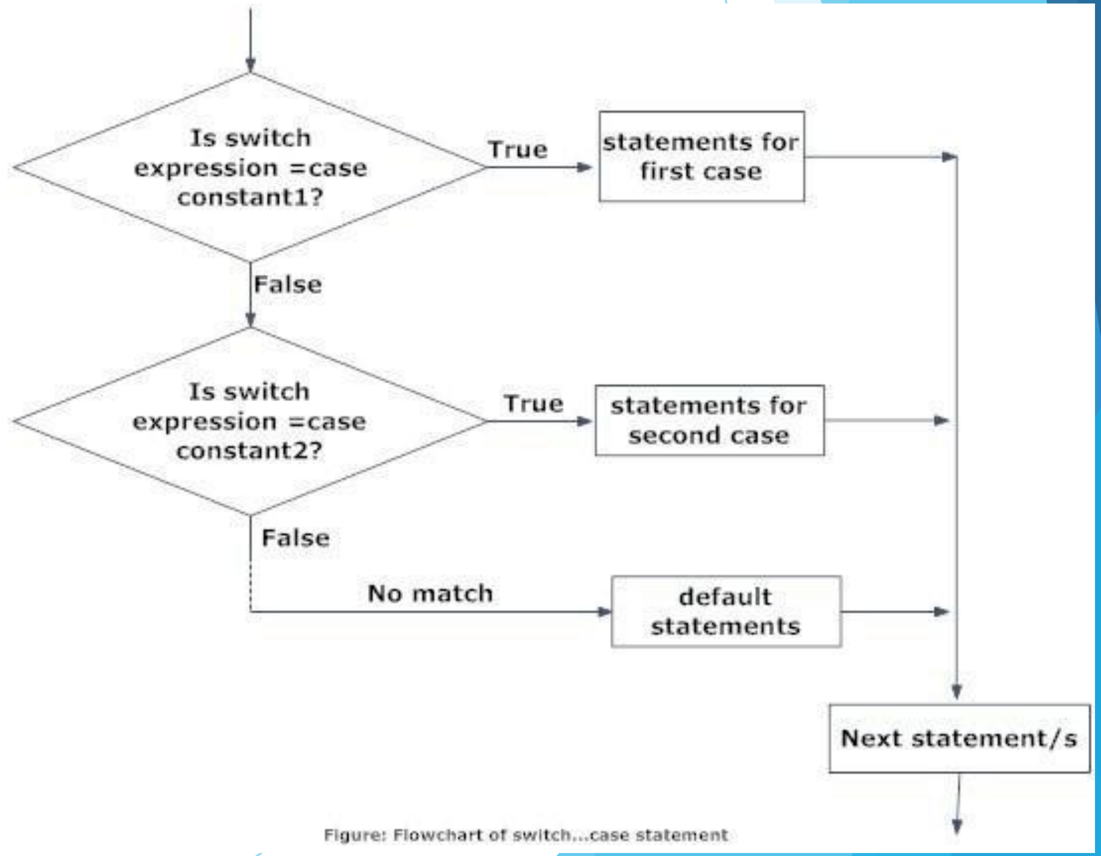
The switch Statement

- ✓ C has built a multi way decision statements known as switched, that tests the value of an expression against a list of case values (integer or character constants).
- ✓ When a match is found, the statements associated with that case is executed.

Syntax

```
switch (expression)
{
        case constant1:
            block of case constant1;
            break;
        case constant2:
            block of case constant2;
            break;
        case constant3:
            block of case constant3;
            break;
        ................................................
        ................................................
        default:
 }
```

Flowchart:



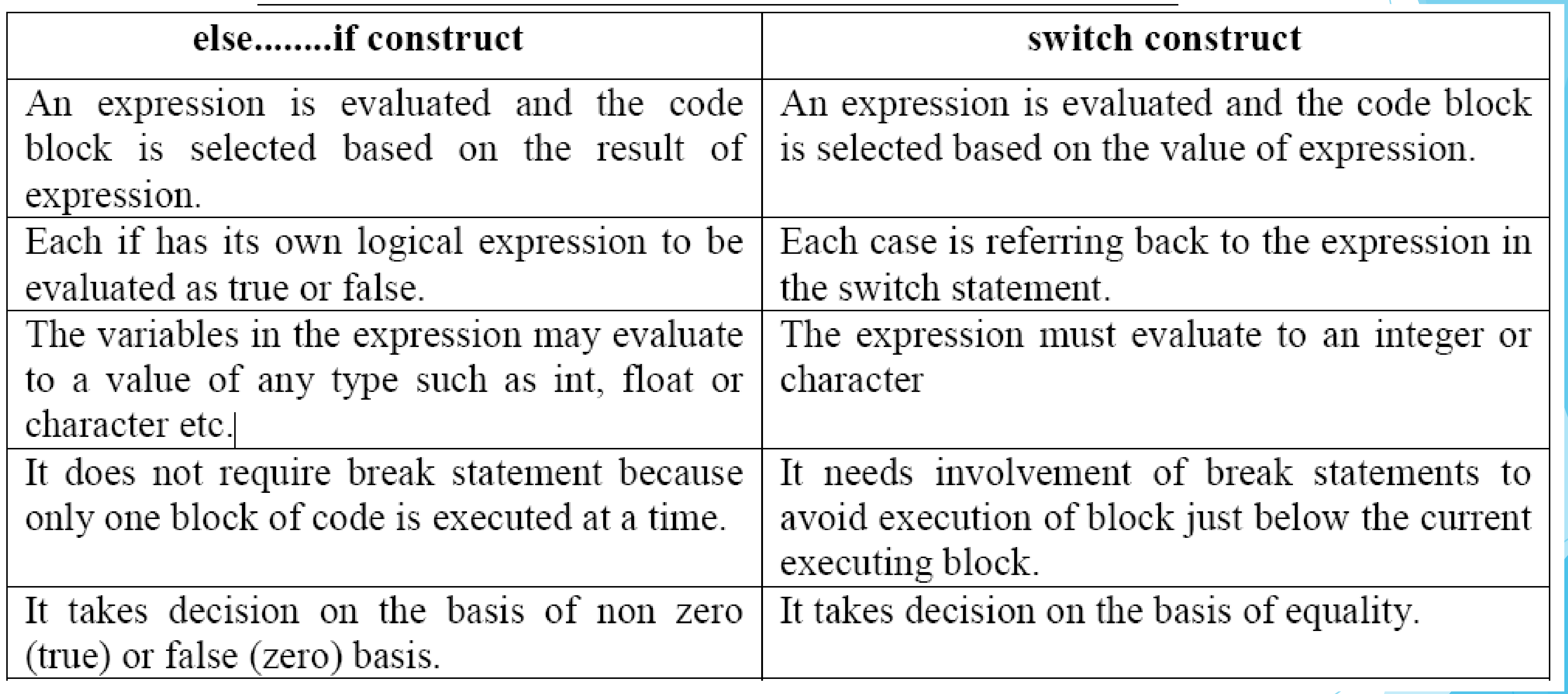
Is switch expression =case constant1?
True
statements for first case
False
Is switch expression =case constant2?
True
statements for second case
False
No match
default statements
Next statement/s

Figure: Flowchart of switch...case statement

Examples:

Q1. WAP that asks an arithmetic operator (‘+’, ‘-’, ‘*’, ‘/’, ‘%’) and two operands and performs the corresponding operation according to operator entered by user on the operands using switch case.

Difference between else if construct and switch



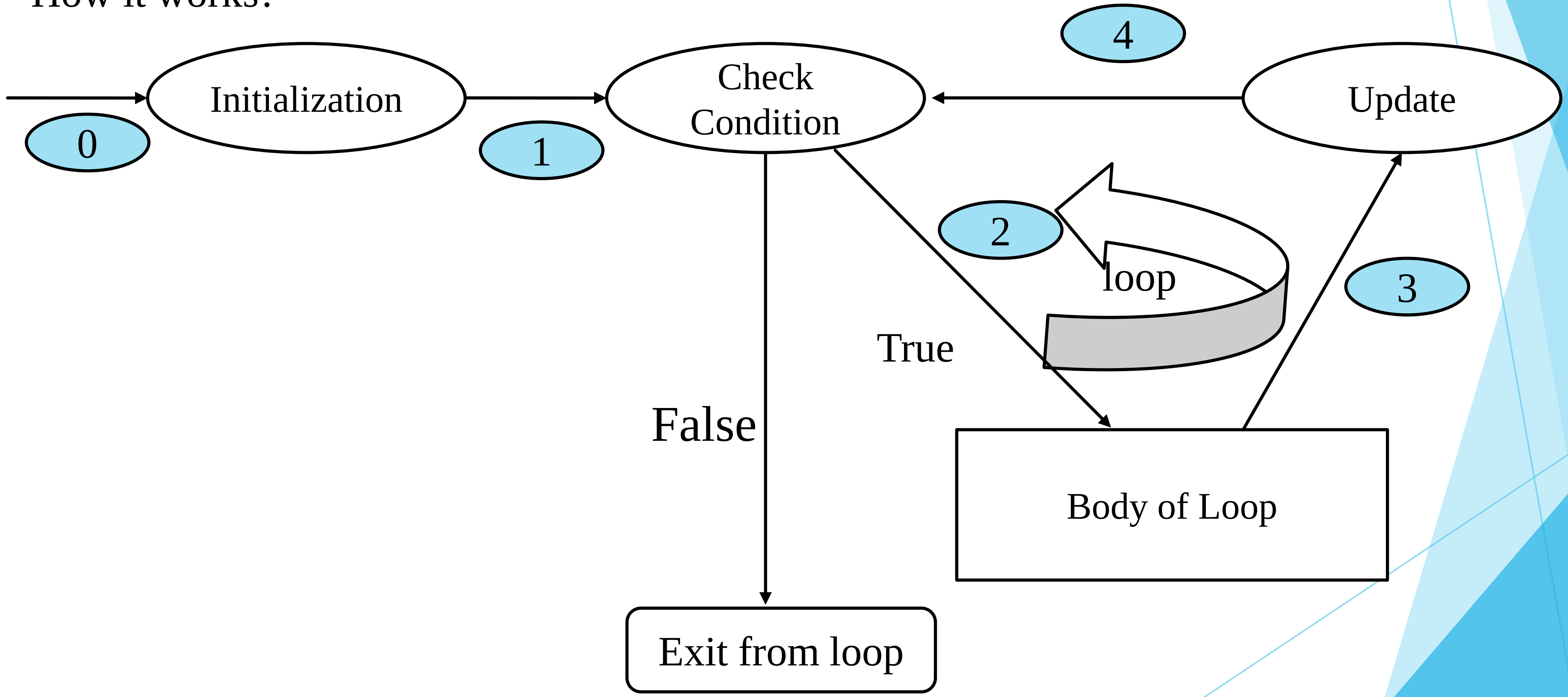
else……..if construct	**switch construct**
An expression is evaluated and the code block is selected based on the result of expression.	An expression is evaluated and the code block is selected based on the value of expression.
Each if has its own logical expression to be evaluated as true or false.	Each case is referring back to the expression in the switch statement.
The variables in the expression may evaluate to a value of any type such as int, float or character etc.	The expression must evaluate to an integer or character
It does not require break statement because only one block of code is executed at a time.	It needs involvement of break statements to avoid execution of block just below the current executing block.
It takes decision on the basis of non zero (true) or false (zero) basis.	It takes decision on the basis of equality.

Repetitive Structure (Looping)

- ✓ Repetition means executing the same section of code more than once.
- ✓ A looping process, in general, would include the following 4 steps.
  1) Setting and initialization of a counter.
  2) Execution of statements in the loop.
  3) Test for a specified condition for execution for the loop.
  4) Updating the counter.
- ✓ C provides 3 loop constructs for performing loop generations.
  - ❑ For loop
  - ❑ while loop
  - ❑ do while loop

The for loop

- ✓ **for** statement is used to execute a block of code for a fixed number of repetition.
- ✓ It is entry controlled loop
- ✓ How it works?



Syntax

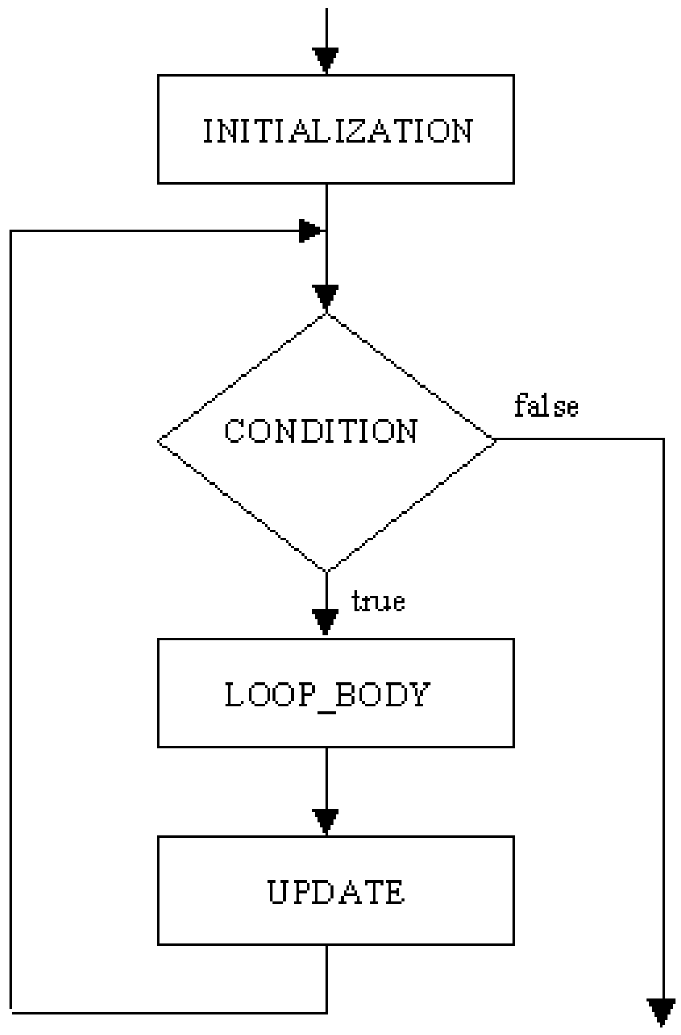
```
for (initialization ; test_expression; update_expression)
{
        body of loop.
}
```

Example:

```
void main()
{
    int i, n = 10, count = 0;
    for (i = 0; i < n ; i++)
     {
            count++;
     }

}
```

Flowchart



Example: WAP to read a non-negative integer n and display its factorial.

```
#include<stdio.h>
#include<conio.h>
     void main()
     {     int n,i;
           long int fact=1;
           printf("Enter a non-negative integer : ");
           scanf("%d", &n);
           for(i=1;i<=n;i++)
           {
                 fact=fact*i;
           }
           printf("Factorial of given number is %ld", fact);
           getch();
      }
```

The while loop

- ✓ It specifies that a section of code should executed while a certain condition holds true.
- ✓ It is entry controlled loop.

Syntax:

```
while(test_expression)
{
      //body of loop
}
```

Example:

```
void main()
{
       int n=1, count=0;
       while ( n <= 10)
       {
             count++;
       n++;
       }
}
```

Flowchart:

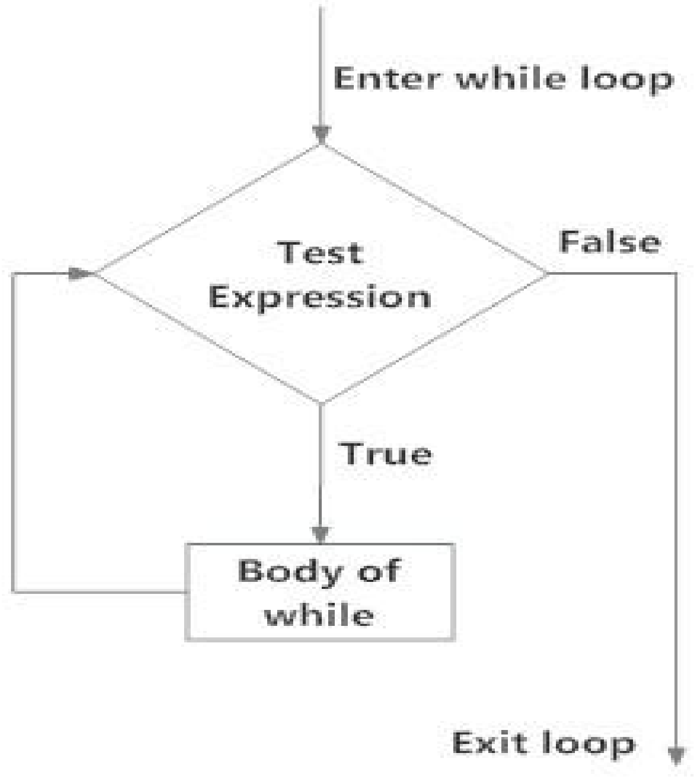


Fig: operation of while loop

Example: WAP to ask a integer number n to user and find the sum of first n natural numbers.

```
void main()
{
        int n, sum = 0;
        printf("Enter a num");
        scanf("%d", &n);
        while(n > 0 )
        {
                sum = sum + n;
                n = n - 1;

        }
        printf("The sum is: %d", sum);
}
```

The do while loop

- ✓ It also specifies that a section of code should be executed while a certain condition holds true.
- ✓ It is exit controlled loop.

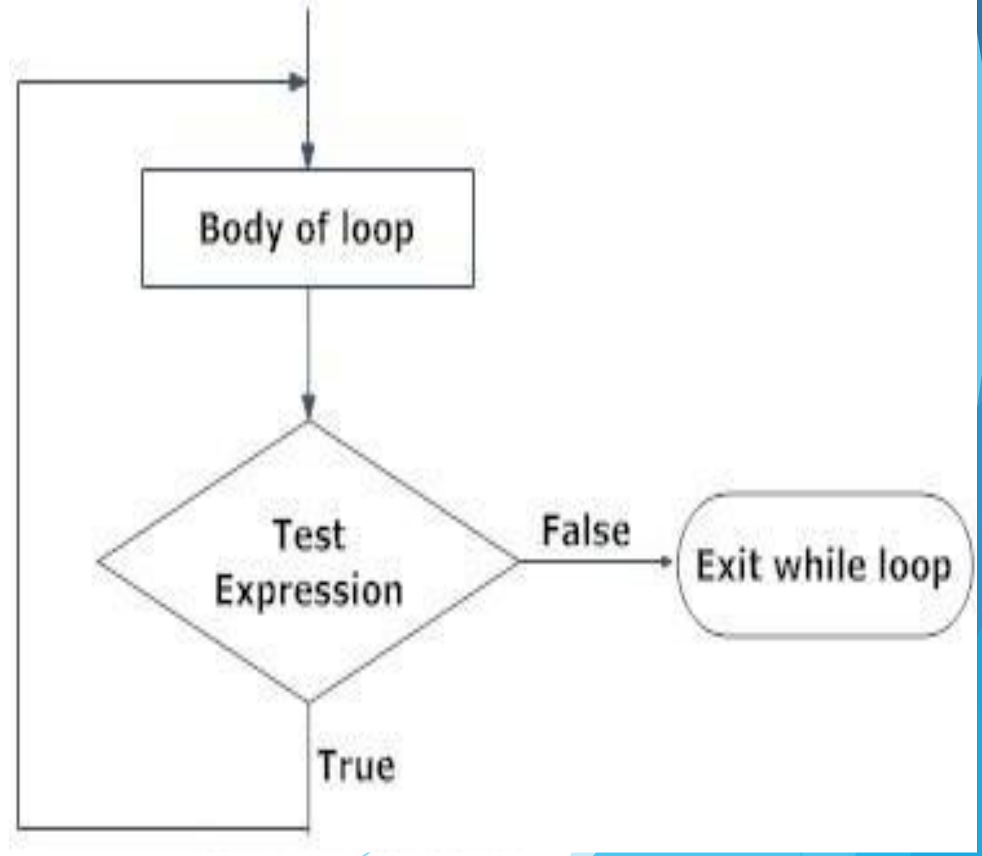
Syntax

```
do
{
        //body of loop.
}while(test_expression);
```

Example:

```
void main()
{
        int n = 1, count = 0;
        do{
                count++;
                 n++;
            } while ( n <= 10);
}
```

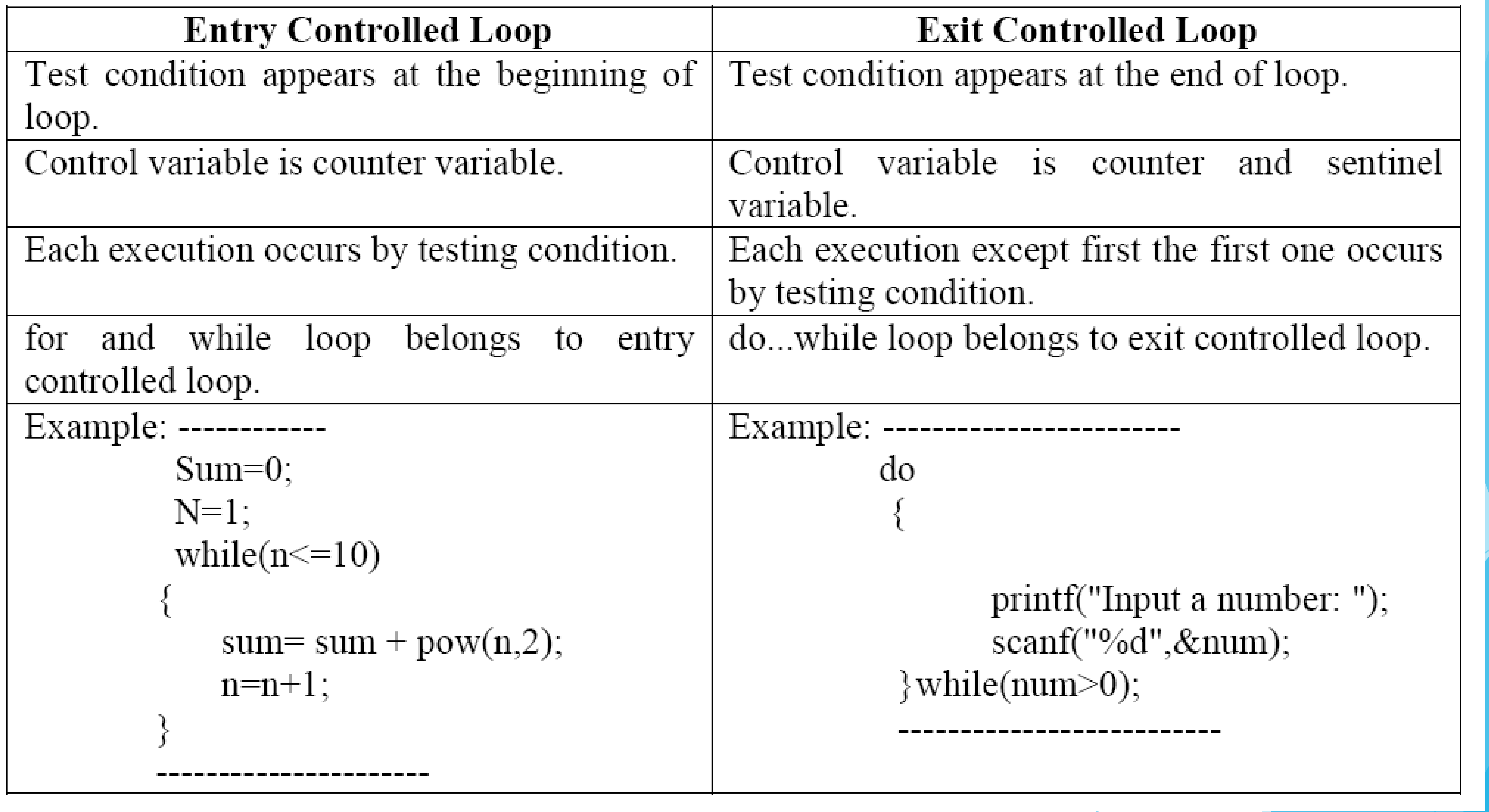
Flowchart



Example: WAP to ask a integer number n to a user and find the sum of all the first positive n even numbers.

```
void main()
{
 int n, i = 1, sum = 0, count = 0;
 printf("Enter a num");
 scanf("%d", &n);
 do{
         if( i % 2 == 0)
         {
                 sum = sum + i;
                 count++;
         }
         i++;
 }while ( count < n);
 printf("The sum is: %d", sum);
 }
```

Differentiate between entry controlled and exit controlled loops



Entry Controlled Loop	**Exit Controlled Loop**
Test condition appears at the beginning of loop.	Test condition appears at the end of loop.
Control variable is counter variable.	Control variable is counter and sentinel variable.
Each execution occurs by testing condition.	Each execution except first the first one occurs by testing condition.
for and while loop belongs to entry controlled loop.	do...while loop belongs to exit controlled loop.

Example: ------------
Sum=0;
N=1;
while(n<=10)
{
sum= sum + pow(n,2);
n=n+1;
}
---------------------- 	
	Example: ------------------------
	do
	{
	printf("Input a number: ");
	scanf("%d",&num);
	}while(num>0);
	-------------------------- 	

Jumps in the program

- ✓ C permits a jump from one statement to another within the programs.
- ✓ The jumping statements used in C programming are:
  - ❑ break,
  - ❑ Continue
  - ❑ goto and
  - ❑ return

The goto statement

✓ It is used to alter the normal sequence of program execution by transferring control to some other part of the program.

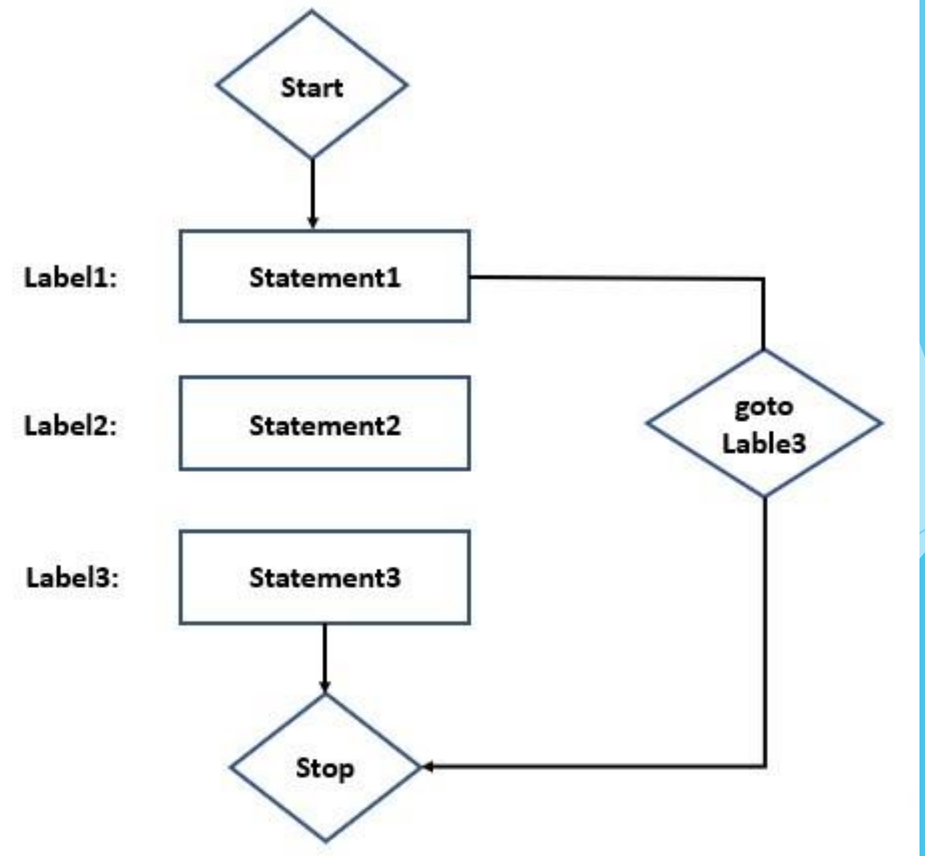
Syntax:

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

Example:

```
    void main()
    {
        int i = 0, sum = 0;
        label 222:
            sum = sum + i;
            i++;
            goto 222;
    }
```

Flowchart



The break statement

- ✓ It is used to jump out of a loop.
- ✓ It terminates the execution of the nearest enclosing loop.
- ✓ It is used with conditional statements and with the while, do..while and for loop statements.

Syntax:

```
a) for ( initialization; test_condition; update)
      {
              if( condition)
                {
                        break;
                }
      }
```

```
b) while( test_condition)
      {
          if( condition)
            {
                  break;
            }
      }
```

```
c) do
      {
          if( condition)
            {
                  break;
            }
      } while( test_condition);
```

Example: Sum between num1 and num 2.

```
void main()
{
        int n1, n2, i, sum = 0;
        printf("Enter two num");
        scanf("%d  %d", &n1, &n2);
        i = n1;
        do{
                if(n1>n2)
                {
                        printf( "Enter n2 > n1");
                        break;
                }
                else
                        sum = sum + i;
                i++;

        }while (i <= n2 );
        printf("Sum = %d", sum);
}
```

Flowchart:

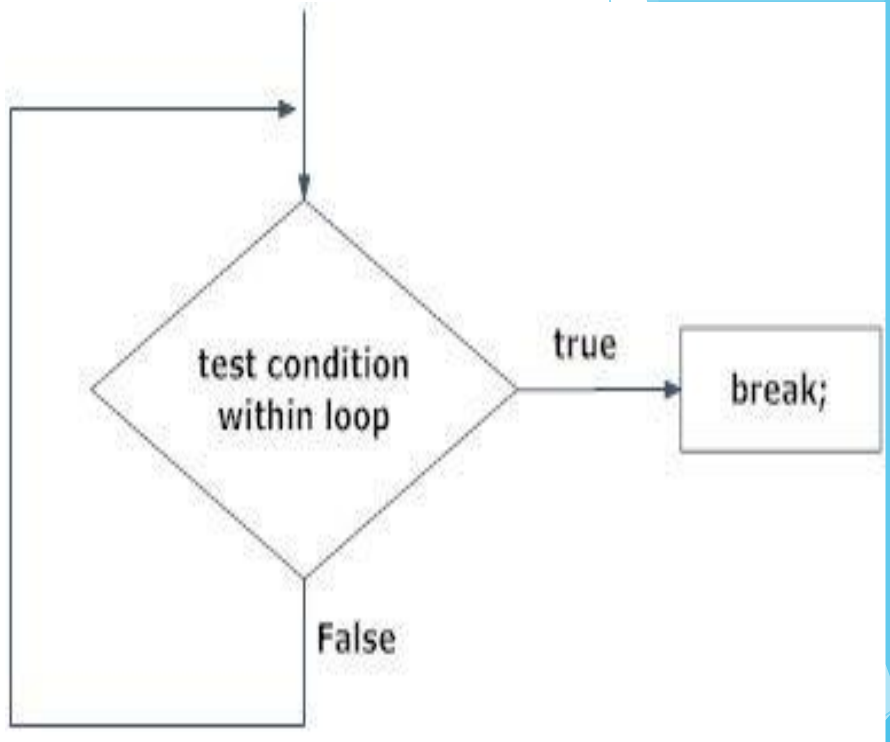


Figure: Flowchart of break statement

The continue statement

- ✓ It is used to bypass the remaining part of current pass of a loop.
- ✓ The loop will not be terminated when a continue statement is encountered.
- ✓ The remaining loop statements are skipped and the computation proceeds directly to next pass through the loop.

Syntax:

```
a) for ( initialization; test_condition; update)
     {
            if( condition)
              {
                        continue;
               }
      }
```

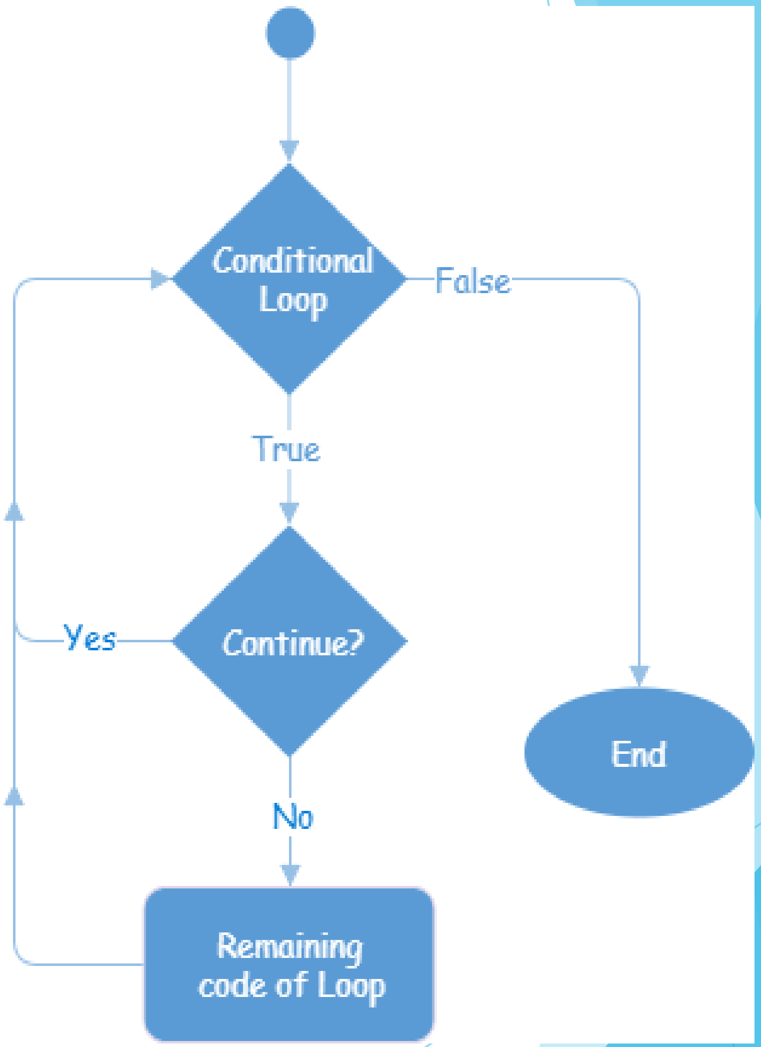
```
b) while( test_condition)
     {
        if( condition)
          {
                  continue;
           }
      }
```

```
c) do
     {
        if( condition)
          {
                  continue;
           }
     } while( test_condition);
```

Example: Odd numbers up to n.

```
void main()
{
    int n, i;
    printf("Enter a num");
    scanf("%d", &n);
    for( i = 0; i <= n; i++)
    {
        if(i%2==0)
            continue;
        else
            printf("%d\t",i);
    }
}
```

Flowchart:



Nested loops

- ✓ Putting one loop statement within another loop statement is called nesting of loop.
- ✓ Nested loops can be :
  - ❑ Nested for loops
  - ❑ Nested while loops
  - ❑ Nested do while loops

Nested for loops

- ✓ If one *for* statement is within the another *for* statement is called nesting of *for* loops.

Syntax:	**Example:**

for (initialization ; test_expression ; update_expression)
{
for (initialization ; test_expression ; update_expression)
{
// body;
}
}
	void main()
	{
	int i, j, n;
	for(i = 0 ; i < n ; i++)
	{
	for(j = 0 ; j < n ; j++)
	{
	// Body of for Loop
	}
	}
	}

Nested while loops

- ✓ If one *while* statement is within the another *while* statement is called nesting of *while* loops.

Syntax:	**Example:**

```
while ( test_ condition)
  {
          while ( test_ condition)
           {
               //body
           }
  }
```
	```
	int i, j, n;
	while ( i <= n)
	  {
	          while ( j <= n)
	           {
	               //body;
	           }
	  }
	```

Nested do while loops

- ✓ If one *do while* statement is within the another *do while* statement is called nesting of *do while* loops.

Syntax:

```
do{
          do{
                    // body;

               }while ( test_ condition);

     }while ( test_ condition);
```

Example:

```
     int i, j, n;
      do{
              do{
                        // body;

                   }while ( i <= n);

           }while ( j <= n);
```

END…