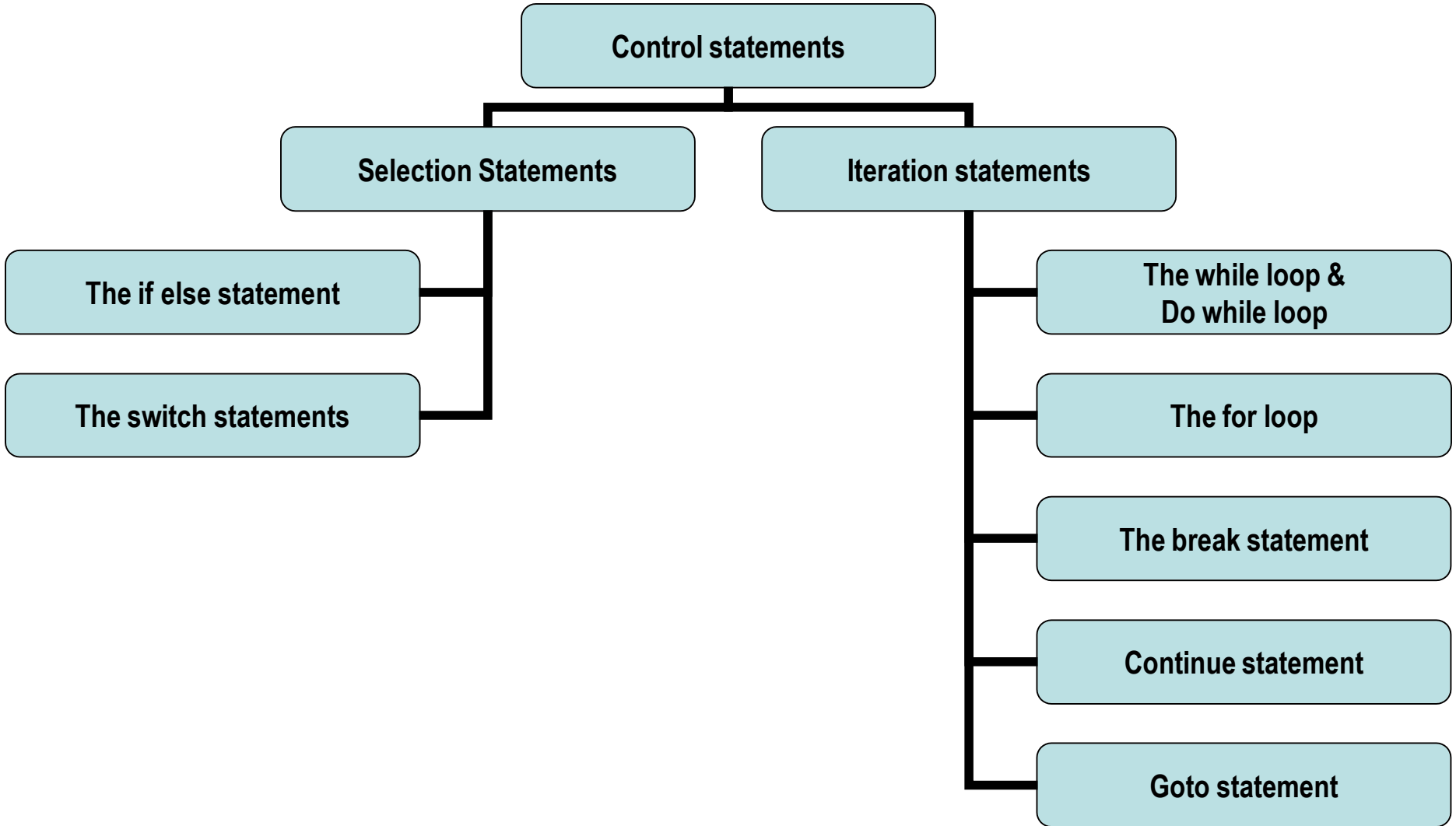
Control Statements (Decision Making)

Control Statements

Control statements in C are used to write powerful programs by;

1. Repeating important sections of the program.
2. Selecting between optional sections of a program.



SELECTION STATEMENT

Types of Selection Statement

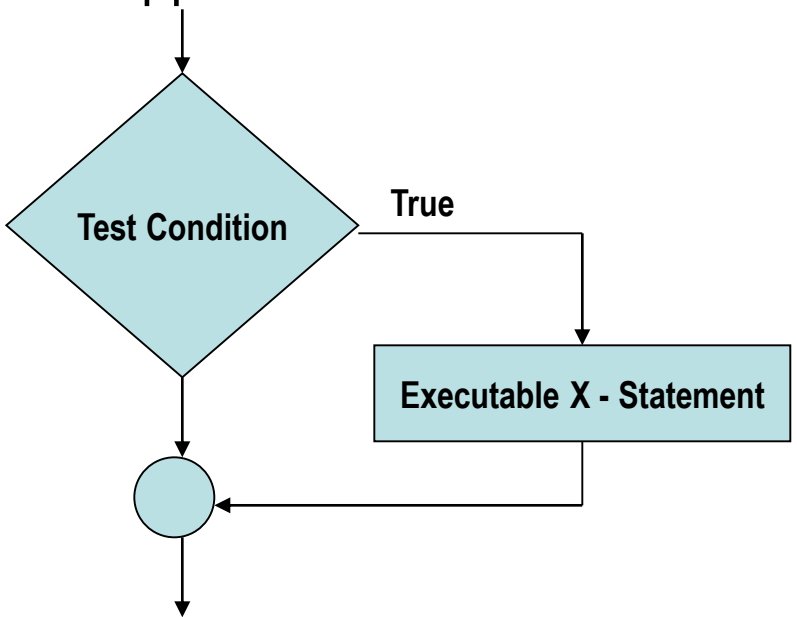
1. Simple if Selection statement
2. if else Selection statement
3. Nested if else Selection statement
4. else if ladder Selection statement

Simple if Selection statement

It is used to control the flow of execution of the statements and also to test logically whether the condition is true or false.

```
Syntax:
      if ( condition )
      {
         statement ;
      }
```

if the condition is true then the statement following the “if “ is executed if it is false then the statement is skipped.



Selection Statement

- Properties of an if statement

a) if the condition is true then the simple or compound statements are executed.
b) If the condition is false it will skip the statement.
c) The condition is given in parenthesis and must be evaluated as true or false.
d) If a compound structure is provided, it must be enclosed in opening and closing braces

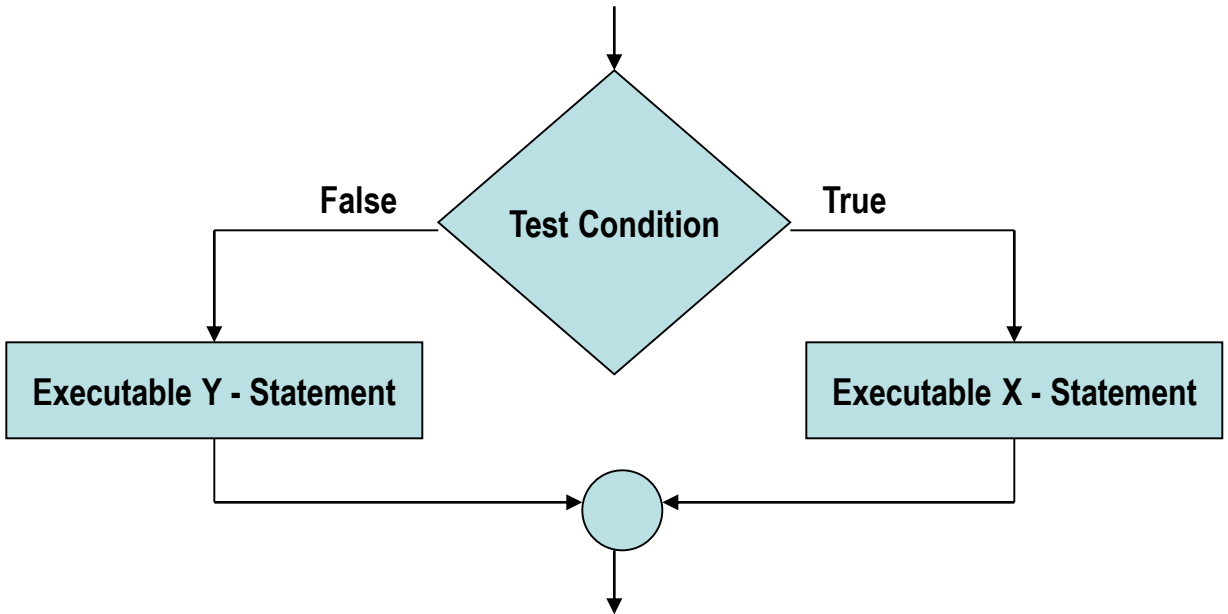
```
//Biggest of Two Numbers
#include <stdio.h>

void main()
{
    int a, b;
    clrscr();

    printf(“Enter the A and B Value:\n”);
    scanf(“%d”, &a);

    if (a > b)
    {
        printf(“A is Big”);
    }

    getch();
}
```

The if else statement

It is used to execute some statements when the condition is true and execute some other statements when the condition is false depending on the logical test.

Syntax:

```
if ( condition )
{
   statement 1 ;      (if the condition is true this statement will be executed)
}
else
{
    statement 2 ;     (if the condition is false this statement will be executed)
}
```



// Biggest of Two Numbers

```
#include <stdio.h>

void main()
{
      int a, b;
      clrscr();

      printf("Enter the A and B Value:\n");
      scanf("%d", &a);

      if (a > b)
      {
            printf("A is Big");
      }
     else
      {
            printf("B is Big");
      }

      getch();
}
```

// Given Number is ODD or EVEN Number

```
#include <stdio.h>

void main()
{
      int n;
      clrscr();

      printf("Enter the Number:\n");
      scanf("%d", &n);

      if (n % 2 == 0)
      {
            printf("Given Number is Even
      Number");
      }
      else
      {
            printf("Given Number is Odd Number");
      }

      getch();
}
```

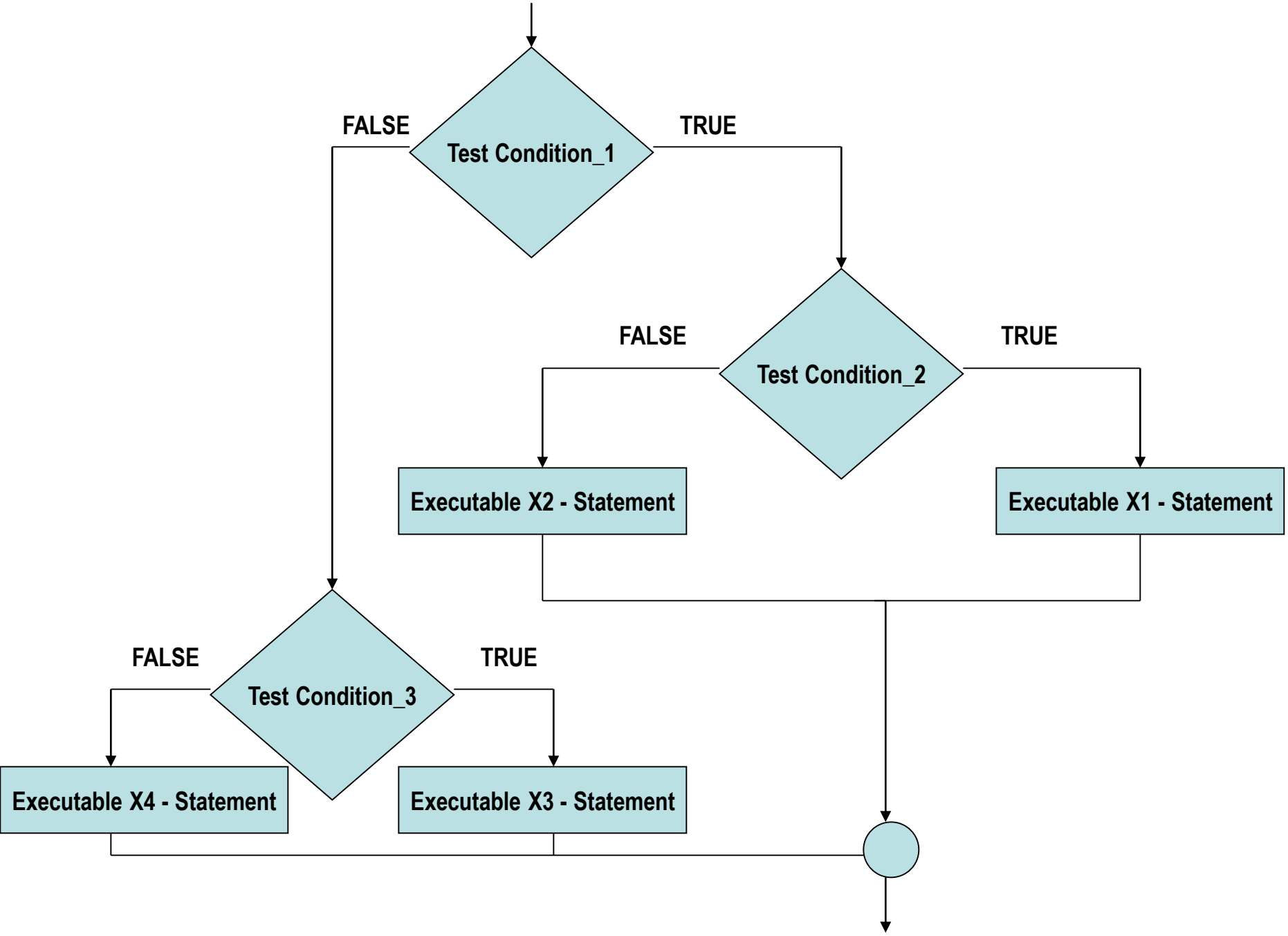
Nested if….. else statement

when a series of if…else statements are occurred in a program, we can write an entire if…else statement in another if…else statement called nesting

Syntax:

```
if ( condition 1)
{
        if ( condition 2)
                statement 1 ;
        else
                statement 2 ;
}
else
{
        if (condition 3)
                statement 3;
        else
                statement 4;
}
```

```
// Biggest of Three Numbers

#include<stdio.h>
void main()
{
    int a, b, c;
    clrscr();

    printf("Enter the Three Numbers:\n");
    scanf("%d%d%d",&a,&b,&c);

    if (a > b)
    {
        if (a > c)
                printf("A is Big");
        else
                printf("C is Big");
    }
    else
    {
        if (b > c)
                printf("B is Big");
        else
                printf("C is Big");
    }
    getch();
}
```

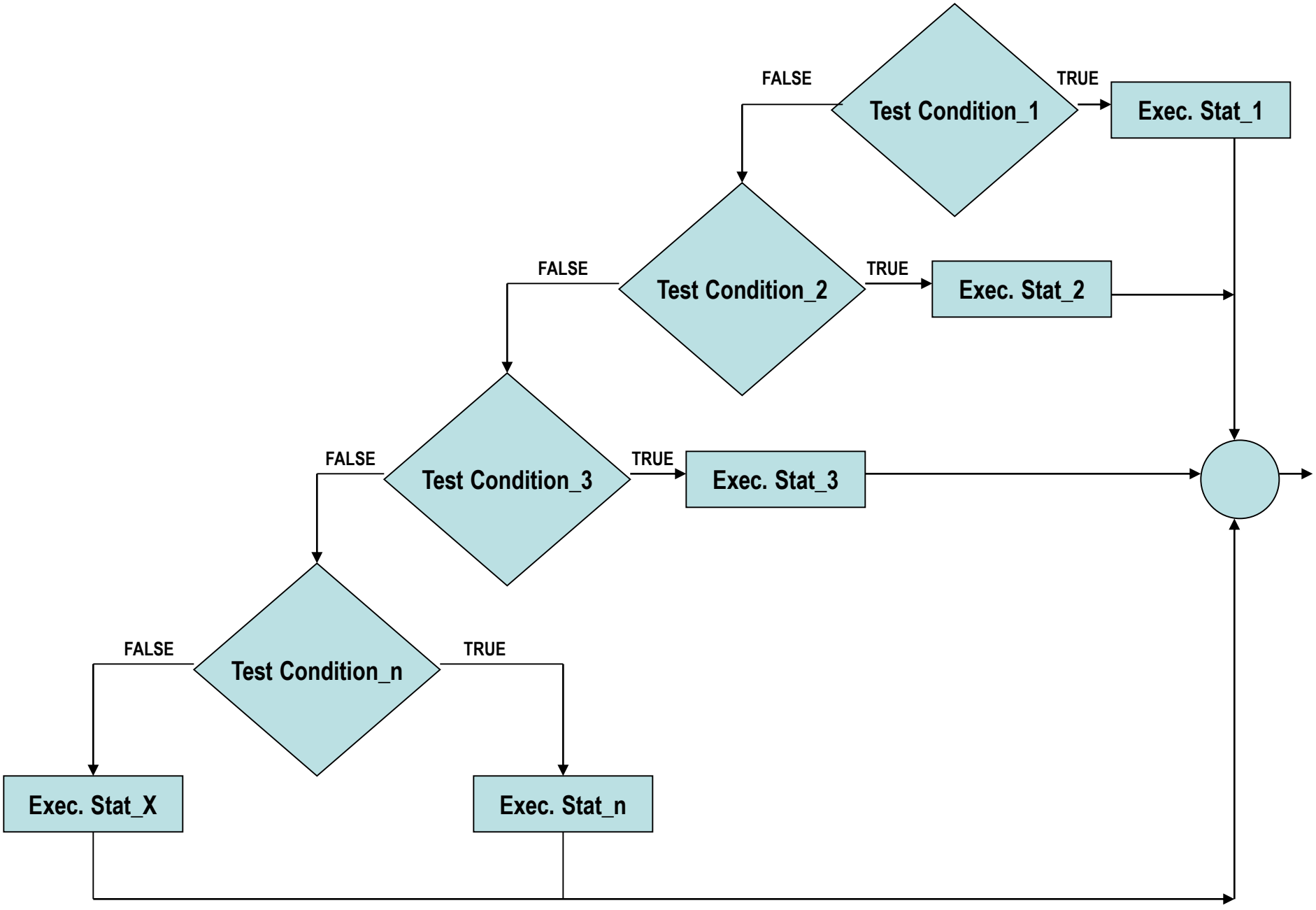

Test Condition_1
FALSE
TRUE
Test Condition_2
FALSE
TRUE
Executable X2 - Statement
Executable X1 - Statement
Test Condition_3
FALSE
TRUE
Executable X4 - Statement
Executable X3 - Statement

else if Ladder or Multiple if else Statements

When a series of decisions are involved we have to use more than one if – else statement called as multiple if's. Multiple if – else statements are much faster than a series of if – else statements, since theif structure is exited when any one of the condition is satisfied.

Syntax:

```
if (condition_1)
        executed statement_1;
    else if (condition_2)
            executed statement_2;
        else if (condition_3)
                executed statement_3;
                    ----------------------
                    ----------------------
                        else if (condition_n)
                                executed statement_n;
                        else
                                executed statement_x;
```


FALSE
Test Condition_1
TRUE
Exec. Stat_1
FALSE
Test Condition_2
TRUE
Exec. Stat_2
FALSE
Test Condition_3
TRUE
Exec. Stat_3
FALSE
Test Condition_n
TRUE
Exec. Stat_X
Exec. Stat_n

```
/*This program reads in a simple expression with a very restricted format and
   prints out its value. */
main()
 {
    int n1,n2;
    int val;
    char op;
    printf("Enter a simple expression ");
    scanf("%d%c%d",&n1,&op,&n2);
    if(op == '+')
       val = n1 + n2;
       else if(op == '-')
                val = n1 - n2;
                else if(op == '/')
                         val = n1 / n2;
                         else if(op == '*')
                                  val = n1 * n2;
else
   {
    printf("?? operator %c\n",op);
    exit(1);
   }
   printf("%d%c%d = %d\n",n1,op,n2);
   }
```

Sample Program

- **Write a program to calculate the sales commission for the data given below:**

Sales value (Rs)	Commission(%)
Less than 1000	No commission
Above 1000 but below 2000	5% of sales
Above 2000 but below 5000	8% of sales
Above 5000	10% of sales

```
#include<stdio.h>
#include<conio.h>
Void main()
{
   float sales, com;
   printf("Enter the sales value :");
   scanf("%f", &sales);
   if(sales<=1000)
     com = 0;
       else if(sales>1000 && sales <=2000)
            com = sales*5/100;
              else if(sales>2000 && sales
     <=5000)
                    com = sales*5/100;
                      else
                          com = sales * 10/100;
   printf("The commission for the sales
     value %f is %f", sales, com); }
```

THE SWITCH STATEMENT

- The control statements which allow us to make a decision from the number of choices is called switch (or) Switch-case statement.
- It is a multi way decision statement, it test the given variable (or) expression against a list of case value.

```
switch (expression)
{
  case constant 1:
                simple statement (or)
                compound statement;
  case constant 2:
                simple statement (or)
                compound statement;
  case constant 3:
                simple statement (or)
                compound statement;
 }
```

```
switch (expression)
{
  case constant 1:
                simple statement (or)
                compound statement;
  case constant 2:
                simple statement (or)
                compound statement;
  default :
                simple statement (or)
                compound statement;
 }
```

Example Without Break Statement

```
#include<stdio.h>
void main ()
{
	int num1,num2,choice;
	printf(“Enter the Two Numbers:\n”);
	scanf(“%d%d”,&num1,&num2);
	printf(“1 -> Addition\n””);
	printf(“2->Subtraction\n”);
	printf(“3->Multiplication\n”);
	printf(“4->Division\n”);
	printf(“Enter your Choice:\n”);
	scanf(“%d”,&choice);

	switch(choice)
	{
		case 1:
			Printf(“Sum is %d\n”, num1+num2);

		case 2:
			Printf(“Diif. is %d\n”, num1-num2);

		case 3:
			Printf(“Product is %d\n”, num1*num2);

		case 4:
			Printf(“Division is %d\n”, num1/num2);

		default:
			printf (“Invalid Choice…..\n”);
	}

	getch();
}
```

Example With Break Statement

```
#include<stdio.h>
void main ()
{
	int num1,num2,choice;
	printf(“Enter the Two Numbers:\n”);
	scanf(“%d%d”,&num1,&num2);
	printf(“1 -> Addition\n””);
	printf(“2->Subtraction\n”);
	printf(“3->Multiplication\n”);
	printf(“4->Division\n”);
	printf(“Enter your Choice:\n”);
	scanf(“%d”,&choice);

	switch(choice)
	{
		case 1:
			printf(“Sum is %d\n”, num1+num2);
			break;
		case 2:
			printf(“Diif. is %d\n”, num1-num2);
			break;
		case 3:
			printf(“Product is %d\n”, num1*num2);
			break;
		case 4:
			printf(“Division is %d\n”, num1/num2);
			break;
		default:
			printf (“Invalid Choice…..\n”);
	}

	getch();
}
```

Rules for Switch

- The expression in the switch statement must be an integer or character constant.
- No real numbers are used in an expression.
- The default is optional and can be placed anywhere, but usually placed at end.
- The case keyword must be terminated with colon (:);
- No two case constant are identical.
- The values of switch expression is compared with case constant in the order specified i.e from top to bottom.
- The compound statements are no need to enclose within pair of braces.
- Integer Expression used in different case statements can be specified in any order.
- A switch may occur within another switch, but it is rarely done. Such statements are called as nested switch statements.
- The switch statement is very useful while writing menu driven programs.

Limitations of using a switch statement

- **Only One variable can be tested with the available case statements with the values stored in them (i.e., you cannot use relational operators and combine two or more conditions as in the case of if or if – else statements).**
- **Floating – point, double, and long type variables cannot be used as cases in the switch statement.**
- **Multiple statements can be executed in each case without the use of pair of braces as in the case of if or if – else statement.**

Iteration Statements

1. **Iteration statements is also known as Looping statement.**
2. **A segment of the program that is executed repeatedly is called as a loop.**
3. **Some portion of the program has to be specified several number of times or until a particular condition is satisfied.**
4. **Such repetitive operation is done through a loop structure.**
5. **The Three methods by which you can repeat a part of a program are,**

1. while Loops **2. do….while loops** **3. for Loop**

Loops generally consist of two parts :

Control expressions: *O*ne or more *control expressions* which control the execution of the loop,

Body : which is the statement or set of statements which is executed over and over

Any looping statement , would include the following steps:

a) **Initialization of a condition variable**

b) **Test the control statement.**

c) **Executing the body of the loop depending on the condition.**

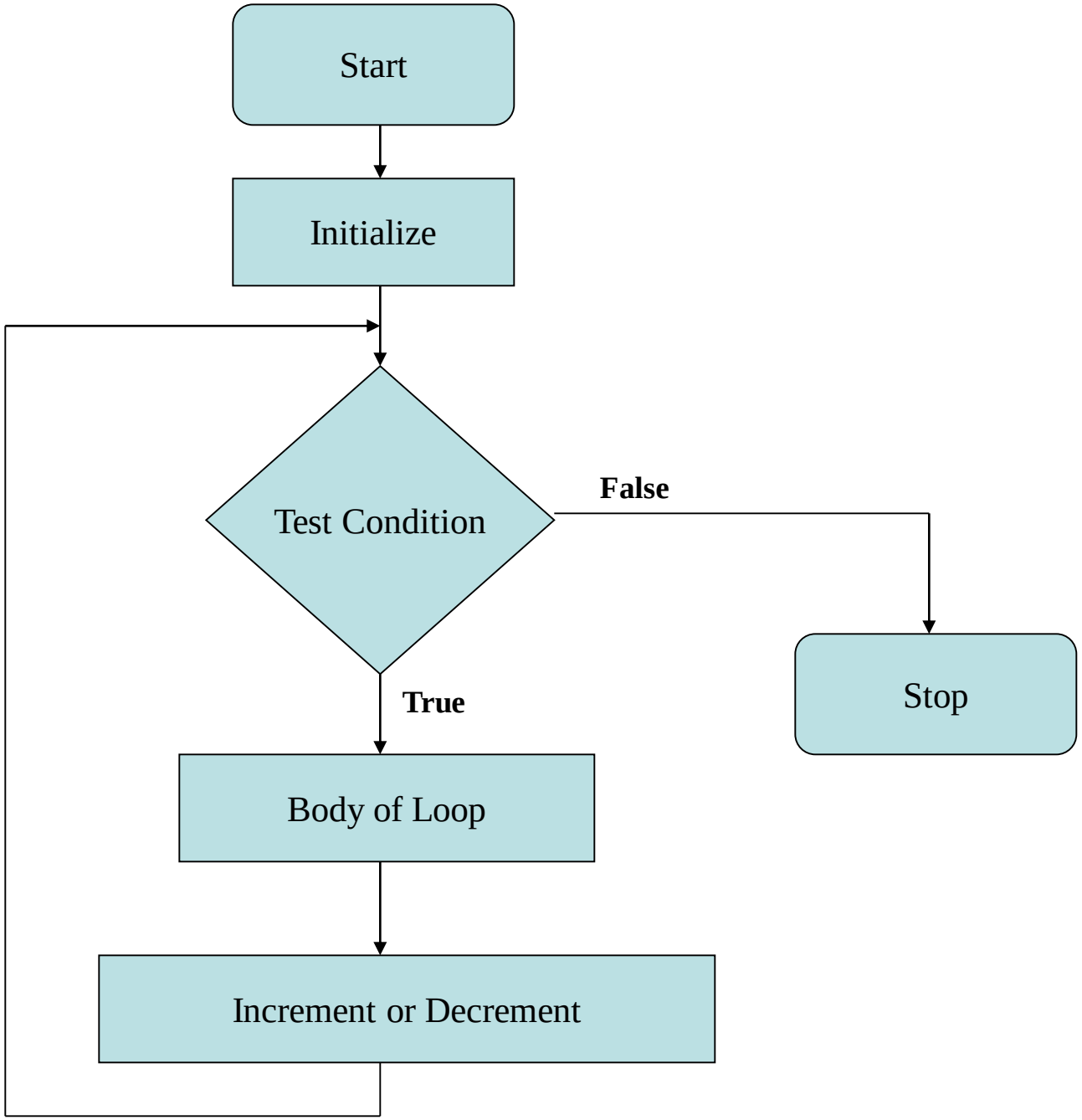
d) **Updating the condition variable.**

While Loop

A **while loop** has one control expression, and executes as long as that expression is true. The general syntax of a while loop is

```
initialize loop counter;
while (condition)
{
        statement (s);
        increment or decrement loop counter
}
```

A while loop is an entry controlled loop statement.


Start
Initialize
Test Condition
False
Stop
True
Body of Loop
Increment or Decrement

Example:

```
// Print the I Values
#include <stdio.h>
void main()
{
        int i;
        clrscr();
        i = 0;
        while(i<=10)
        {
            printf("The I Value is :%d\n",i);
            ++I;
        }
        getch();
}
```

```
// Summation of the series 1 + 2 + 3 + 4 + …….
#include <stdio.h>
void main()
{
        int i, sum;
        clrscr();
        i = 1;
        sum = 0;
        while(i<=10)
        {
            sum = sum + i
            printf("The Sum Value is:%d\n",i);
            ++I;
        }
        getch();
}
```

Example:

```
//Summation of the series 1^2 + 2^2 + 3^2 + …..
#include <stdio.h>
#include<math.h>
void main()
{
        int i, sum;
        clrscr();
        i = 1;
        sum = 0;
        while(i<=10)
        {
            sum = sum + i*i; //or I ^2 or pow(i, 2)
            printf("The Sum Value is:%d\n",i);
            ++I;
        }
        getch();    }
```

```
//Summation of the series 1^1 + 2^2 + 3^3 + …..
#include <stdio.h>
#include<math.h>
void main()
{
      int i, sum;
      clrscr();
      i = 1;
     sum = 0;
      while(i<=10)
      {
          sum = sum + pow(i,i)
          printf("The Sum Value is:%d\n",i);
           ++I;
      }
      getch();            }
```

Wap to print the summation of digits of any given number.

```
#include<stdio.h>
void main()
{
    int number=0, rem=0, sum=0;

    clrscr();

    printf(“Enter the value for number”);
    scanf(“%d”,&n);

    while(number > 0)
    {
        rem = number % 10;
        sum = sum + rem;
        number = number / 10;
    }

    printf(“the summation value of  the given number %d is =
    %d”,number,sum);
}
```

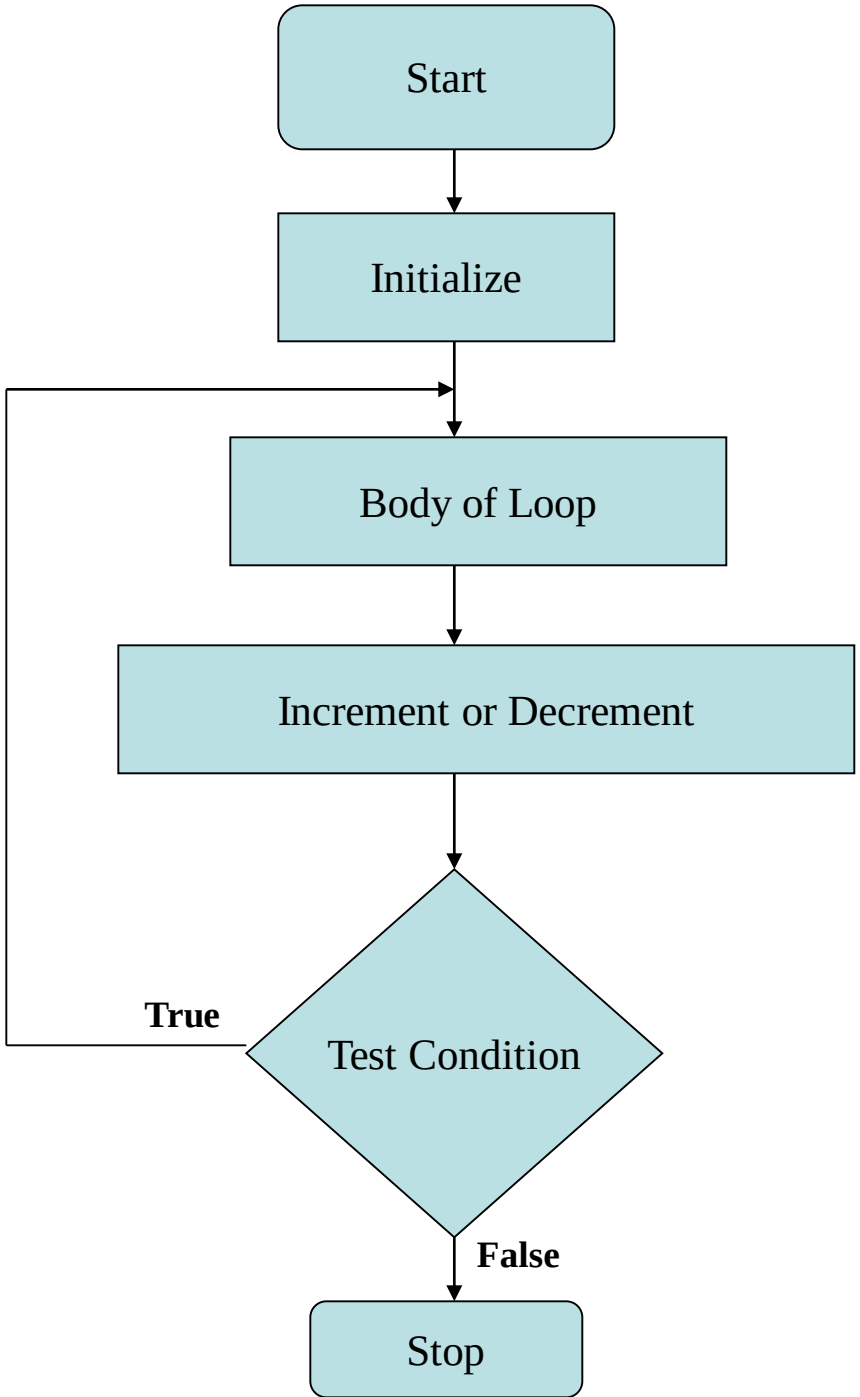
THE do-while LOOP

- The body of the loop may not be executed if the condition is not satisfied in while loop.
- Since the test is done at the end of the loop, the statements in the braces will always be executed at least once.
- The statements in the braces are executed repeatedly as long as the expression in the parentheses is true.

```
initialize loop counter;
do
{
        statement (s);
        increment or decrement loop counter
}
while (condition);
```

Make a note that do while ends in a ; (semicolon)

Note that Do… While Looping statement is Exit Controlled Looping statement


Start
Initialize
Body of Loop
Increment or Decrement
Test Condition
True
False
Stop

Difference Between While Loop and Do – While Loop

Sl.No.	while loop	do-while loop
1.	The while loop tests the condition before each iteration.	The do – while loop tests the condition after the first iteration.
2.	If the condition fails initially the loop is Skipped entirely even in the first iteration.	Even if the condition fails initially the loop is executed once.

Example:

```
// Print the I Values
#include <stdio.h>
void main()
{
        int i;
        clrscr();
        i = 0;
        do
        {
            printf("The I Value is :%d\n",i);
            ++I;
        }
         while(i<=10);
        getch();
}
```

```
// Print the I Values
#include <stdio.h>
void main()
{
        int i;
        clrscr();
        i = 11;
        do
        {
            printf("The I Value is :%d\n",i);
            ++I;
        }
         while(i<=10);
        getch();
}
```

Wap to print the Fibonacci series for any given number Using Do….While Loop

```
#include <stdio.h>
void main()
{
    int i, f1,f2,f3;
    clrscr();
    f1 = 0;
    f2 = 1;
    printf(“The Fibonacci Series is:\n”)
    printf(“%d\n”,f1);
    printf(“%d\n”,f2);
    do
    {
            f3 = f1 + f2;
            printf(%d\n”,f3);
            f1 = f2;
            f2 = f3;

            ++i;
    }
    while(i <= 10);

    getch();
}
```

for Loop

- The for loop is another repetitive control structure, and is used to execute set of instruction repeatedly until the condition becomes false.
- To set up an initial condition and then modify some value to perform each succeeding loop as long as some condition is true.

The syntax of a for loop is

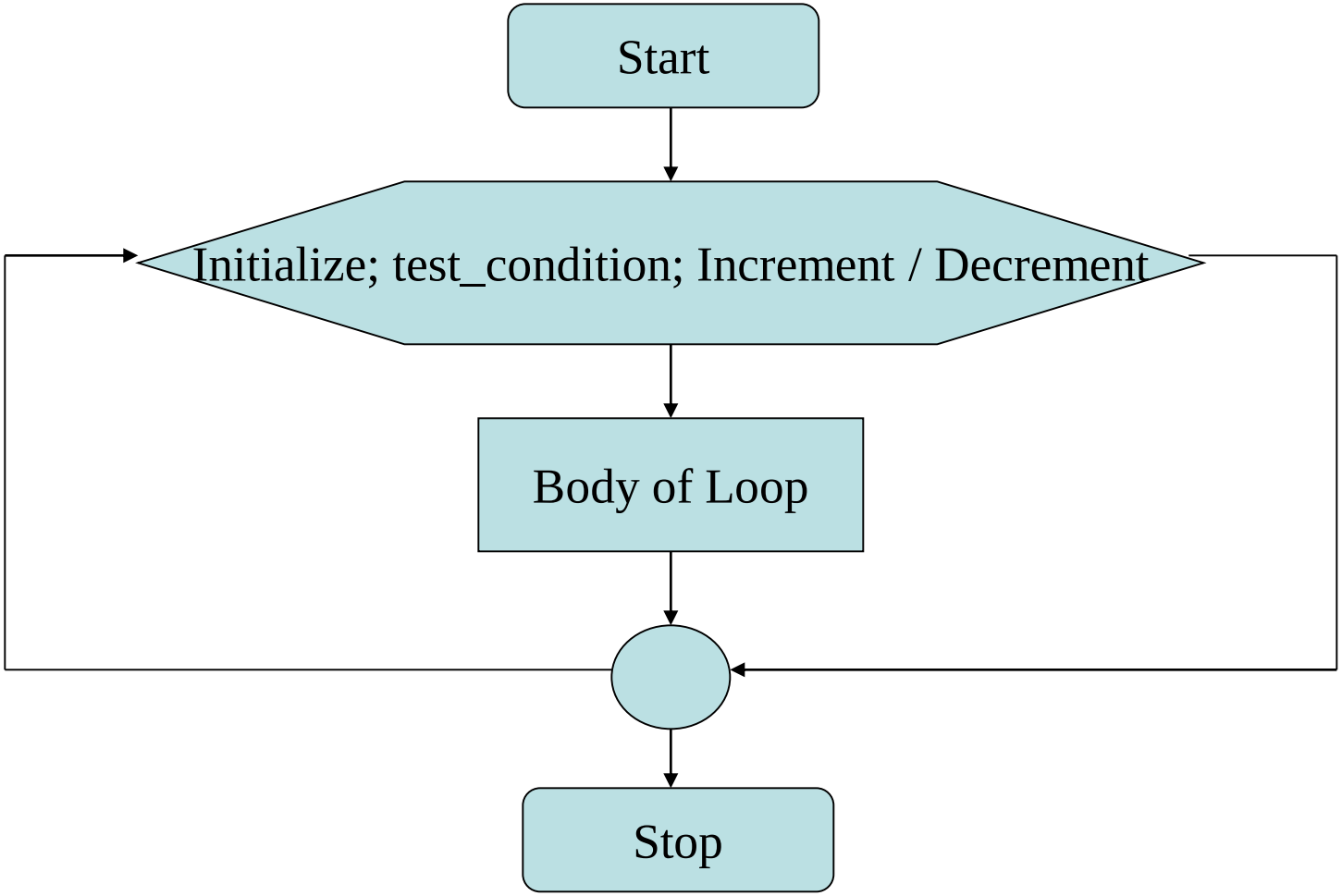
```
for( expr1; expr2 ;expr3)
{
 Body of the loop;
}
```

The three expressions :

expr1 - sets up the initial condition,

expr2 - tests whether another trip through the loop should be taken,

expr3 - increments or updates things after each trip.


Start
Initialize; test_condition; Increment / Decrement
Body of Loop
Stop

Example

Given example will print the values from 1 to 10.

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

There is no need of { } Braces for single line statement and for Multiple line it is essential else it will consider only next line of for statement.

Example

Given example of Multiplication Table

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int mul,limit,c,i;
    clrscr();

    printf("Enter the Multiplication Number:\n");
    scanf("%d",&mul);
    printf("Enter the Limits:\n");
    scanf("%d",&limit);

    for(i=1;i<=limit;i++)
    {
        c = i * mul;
        printf("%d * %d: %d\n",i,mul,c);
    }

    getch();
}
```

Additional Features of for Loop

Case 1:

The statement

```
p = 1;
for (n = 0; n < 17; ++ n)
```

can be rewritten as

```
for (p = 1, n = 0; n < 17;++n)
```

Case 2:

The second feature is that the test – condition may have any compound relation and

the testing need not be limited only to the loop control variable.

```
sum = 0;
for (i = 1; i < 20 && sum < 100; ++ i)
{
        sum = sum + i;
        printf("%d %d\n", i, sum);
}
```

Additional Features of for Loop Conti…

Case 3:

It also permissible to use expressions in the assignment statements of initialization and increments sections.

For Example:

```
for (x = (m + n) / 2; x > 0; x = x / 2)
```

Case 4:

Another unique aspect of for loop is that one or more sections can be omitted, if necessary.

For Example:

```
m = 5;
for ( ; m ! = 100 ;)
{
        printf("%d\n",m);
        m = m + 5;
}
```

Both the initialization and increment sections are omitted in the for statement. The initialization has been done before the for statement and the control variable is incremented inside the loop. In such cases, the sections are left 'blank'. However, the semicolons separating the sections must remain. If the test – condition is not present, the for statement sets up an 'infinite' loop. Such loops can be broken using break or goto statements in the loop.

Additional Features of for Loop Conti…

Case 5:

We can set up time delay loops using the null statement as follows:

```
for ( j = 1000; j > 0; j = j – 1)
```

1. This is loop is executed 1000 times without producing any output; it simply causes a time delay.
2. Notice that the body of the loop contains only a semicolon, known as a null statement.

Case 6:

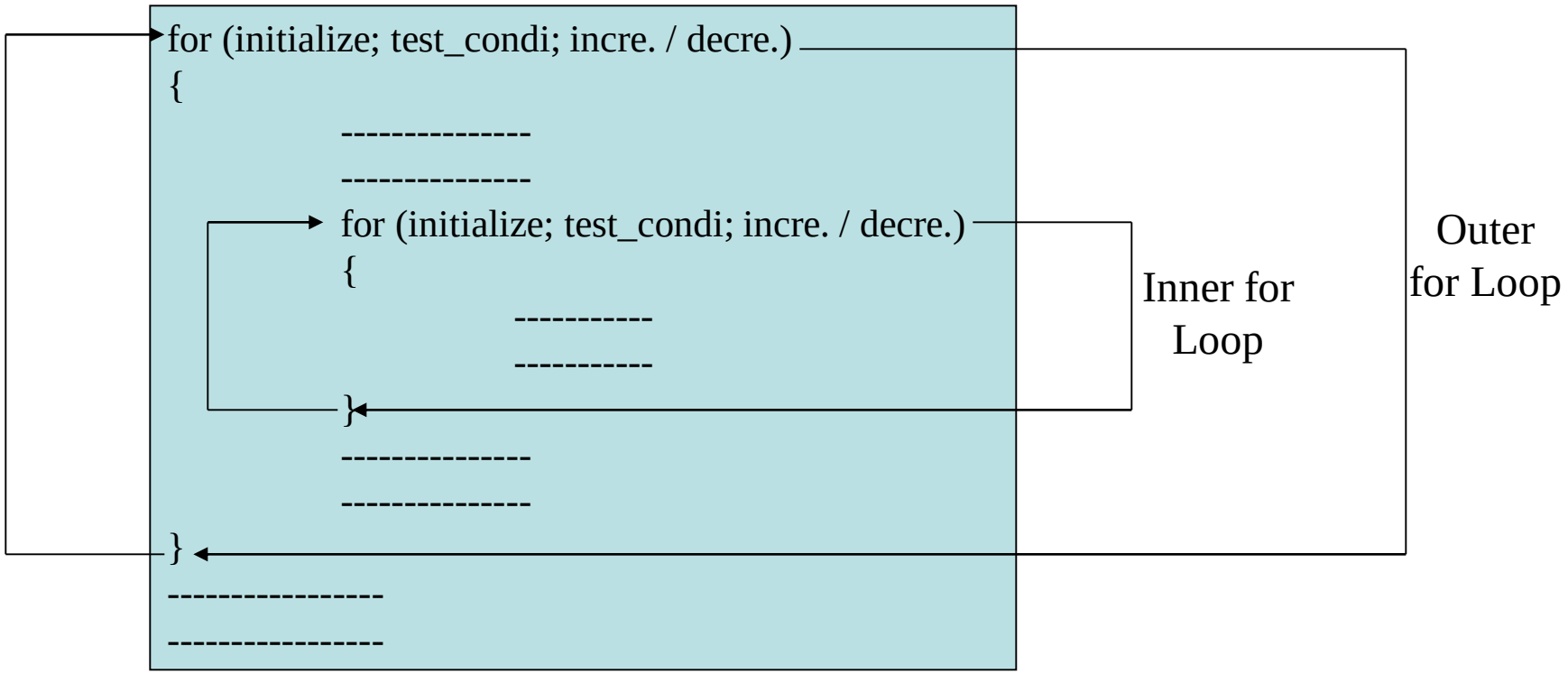
```
for ( j = 1000; j > 0; j = j – 1)
```

This implies that the C compiler will not give an error message if we place a semicolon by mistake at the end of a for statement. The semicolon will be considered as a null statement and the program may produce some nonsense.

Nesting of for Loop

The One for statement within another for statement is called Nesting for Loop.

Syntax:



Example

// Print the I and J Value

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

      for (i = 1; I < = 10 ; I ++)
      {
            printf (“The I Value is %d \n", i);

            for (j = 1; j < = 10; j ++)
            {
                printf (“The J Value is %d \n", j);
            }
      }
getch();
}
```

Example

// Multiplication Table

```
#include<stdio.h>
#include<conio.h>
void main()
{
      int sum = 1,a,b;
      clrscr();

      for (a=1;a<=5;a++)
      {
          printf ("the multiplication table for %d\n",a);

              for (b=1;b<=12;b++)
              {
                        sum=a*b;
                        printf("%d*%d=",a,b);
                        printf("%d\n",sum);
              }
              sum = 0;
      }
getch();
}
```

Exercise

1) Write a program that will read in N numbers and print out their average.

2) Wap to print the following series for N given number

1+(1+2)+(1+2+3)+(1+2+3+4)……

3) Wap to print the following series for N given number

```
+
++
+++
++++
+++++
++++++   ……..
```

4) Wap to print the following series for N given number

```
1
111
11111
1111111
```

JUMPS IN LOOPS

1. Loops perform a set of operations repeatedly until the control variable fails to satisfy the test – condition.
2. The number of times a loop is repeated is decided in advance and the test condition is written to achieve this.
3. Sometimes, when executing a loop it becomes desirable to skip a part of the loop or to leave the loop as soon as a certain condition occurs.
4. Jumps out of a Loop is Classified into three types

   1. break; 2. continue; 3. goto;

The break Statement

1. A break statement is used to terminate of to exit a for, switch, while or do – while statements and the execution continues following the break statement.
2. The general form of the break statement is

```
break;
```

3. The break statement does not have any embedded expression or arguments.
4. The break statement is usually used at the end of each case and before the start of the next case statement.
5. The break statement causes the control to transfer out of the entire switch statement.

```
#include <stdio.h>
void main()
{
        int i;
        clrscr();
        i = 1;
        while (i < = 10)
        {
          printf (“The I Value is: %d \n”, i);
          if (i = = 6)
          {
            printf (“The I value is Reached 6,
            So break of the programs\n”);
            break;
          }
         ++ i
       }
}
```

```
#include<stdio.h>
void main ()
{
      int num1,num2,choice;
      printf (“Enter the Two Numbers:\n”);
      scanf(“%d%d”,&num1,&num2);
      printf(“1 -> Addition\n””);
      printf(“2->Subtraction\n”);
      printf(“3->Multiplication\n”);
      printf(“4->Division\n”);
      printf (“Enter your Choice:\n”);
      scanf (“%d”, &choice);

      switch (choice)
      {
        case 1:
                printf (“Sum is %d \n”, num1+num2);
                break;
        case 2:
                printf (“Diif. is %d \n”, num1-num2);
                break;
        case 3:
                printf (“Product is %d \n”, num1*num2);
                break;
        case 4:
                printf (“Division is %d \n”, num1/num2);
                break;
        default:
                printf (“Invalid Choice…..\n”);
      }
      getch();
}
```

The continue Statement

- The continue statement is used to transfer the control to the beginning of the loop, there by terminating the current iteration of the loop and starting again from the next iteration of the same loop.
- The continue statement can be used within a while or a do – while or a for loop.
- The general form or the syntax of the continue statement is

```
continue;
```

- The continue statement does not have any expressions or arguments.
- Unlike break, the loop does not terminate when a continue statement is encountered, but it terminates the current iteration of the loop by skipping the remaining part of the loop and resumes the control tot the start of the loop for the next iteration.

```
#include <stdio.h>
void main()
{
    int i;
    clrscr();
    i = 1;
    while (i < = 10)
    {
        printf (“The I Value is: %d \n”, i);
        if (i = = 6)
        {
            printf (“The I value is Reached 6, But Continue this Programs\n”);
            continue;
        }
        ++ i
    }
```

Differences Between Break and Continue Statement

Sl.No.	break	continue
1.	Used to terminate the loops or to exit loop from a switch.	Used to transfer the control to the start of loop.
2.	The break statement when executed causes immediate termination of loop containing it.	Continue statement when executed causes Immediate termination of the current iteration of the loop.

The goto Statement

- The goto statement is used to transfer the control in a loop or a function from one point to any other portion in that program.
- If misused the goto statement can make a program impossible to understand.
- The general form or the syntax of goto statement is

```
goto label;
            Statement (s);
            …….
label:
            statement (s);
```

- The goto statement is classified into two types

    a. Unconditional goto

    b. Conditional goto

Unconditional Goto

The Unconditional goto means the control transfer from one block to another block without checking the test condition.

Example:

```
#include <stdio.h>
void main()
{
      clrscr();
     Start:
           printf("Welcome\n");
           goto Start;
      getch();
}
```

Conditional Goto

The Conditional goto means the control transfer from one block to another block with checking the test condition.

```
#include <stdio.h>
void main()
{
        int a, b;
        clrscr();
        printf ("Enter the Two Value:\n");
        scanf ("%d", &a, &b);
        if (a >  b)
                goto output_1;
        else
                goto output_2;
        output_1:
                printf ("A is Biggest Number");
                goto Stop;
        output_2:
                printf ("B is Biggest Number");
                goto Stop;

        Stop:
        getch();
}
```