EXPRESSIONS AND STATEMENTS

Expressions

- An expression represents data item such as variables, constants and are interconnected with operators as per the syntax of the language.
- An expression is evaluated using assignment operators.

Syntax

Variable = expression;

Example: 1

*x=a*b-c;*

- In example 1, the expression evaluated from left to right. After the evaluation of the expression the final value is assigned to the variable from right to left.

Example: 2

a++;

- In example 2, the value of variable a is incremented by 1, i.e, this expression is equivalent to a = a + 1.

Statements

- A statement is an instruction given to the computer to perform an action. There are three different types of statements in C:
  1. Expression Statements
  2. Compound Statements
  3. Control Statements

Expression Statement

- An expression statement or simple statement consists of an expression followed by a semicolon (;).

Example

```
a=100;
b=20;
c=a/b;
```

Compound Statement

- A compound statement also called a block, consists of several individual

statements enclosed within a pair of braces { }.

Example

```
{
        a=3;
        b=10;
        c=a+b;
}
```

Control Statement

- A single statement or a block of statements can be executed depending upon a condition using control statements like if, if-else, etc.

Example

```
a=10;
if (a>5)
{
        b= a+10;
}
```

CONDITIONAL STATEMENTS

- The conditional statement requires the programmer to specify one or more conditions to be evaluated or tested by the program, along with a statement or statements to be executed if the condition is determined to be true, and optionally, other statements to be executed if the condition is determined to be false.
- In a conditional statement, the flow of execution may be transferred from one part to another part based on the output of the conditional test carried out. It has been further classified into selective and loop constructs. In a selective constructs, the statements are selected for execution based on the output of the conditional test given by an expression. It supports the following constructs such as if-else, if-else-if, nested-if and switch case statement. In loop constructs, the block of statements will be executed repeatedly until the condition is true else the loop will be terminated. It supports the following constructs such as For, While and Do-while loops.

Conditional Branching Statement

Selection Statement

❖ Simple If statement

- The syntax for a simple if statement is

```
if (expression)
{
block of statements;
}
```

- In this statement, if the expression is true, the block of statements are executed otherwise false and it comes out of the if condition.

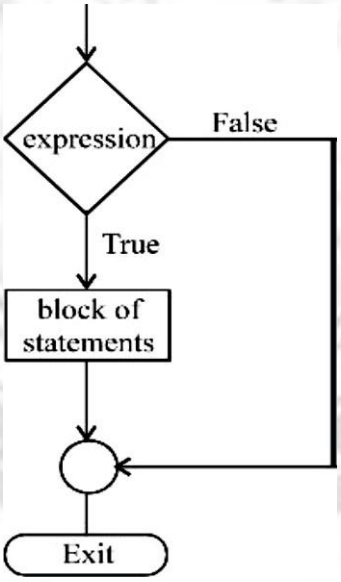

Fig. 1.7 Flowchart for an If statement

Program 1.10

/*Program to find the given number is divisible by 2 */

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

{
int n;
printf("\n Enter the number");
scanf("%d",&n);
```

```
if(n%2==0)
{
printf("\n The given number%d is divisible by 2", n);
}
getch();
}
```

Output

Enter the number : 10

The given number 10 is divisible by 2

Program 1.11

/* Program to check the given numbers are equal or not */

```
#include<stdio.h>
#include<conio.h>
void main()
{
int m,n;
clrscr();
printf("\n Enter two numbers:");
scanf("%d %d", &m,&n);
if (m==n)
printf("\n Two numbers are equal");
getch();
}
```

Output

Enter two numbers: 10 10

Two numbers are equal.

- **If –else statement**
  - The syntax for the if-else statement is

```
if(expression)
```

```
{
block of statements1;
}
else
{
block of statements2;
}
```

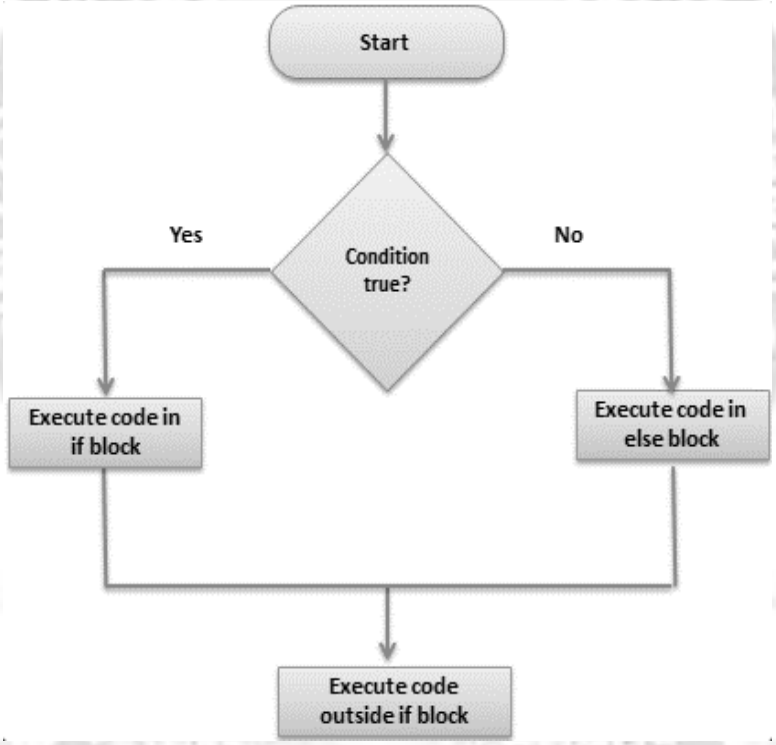

Fig. 1.8 Flowchart for the If-else statement

- In this statement, if the expression is true the block of statements1 will be executed, otherwise the block of statements2 will be executed.

Program 1.12

/* Program to find the given number is positive or negative */

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

```
int n;
printf("\n Enter the number:");
scanf("%d",&n);
if(n>0)
{
printf("\n The given number %d is positive", n);
}
else
{
printf("\n The given number %d is negative", n);
}
}
```

Output

Enter the number : 5

The given number 5 is positive.

Program 1.13

/* Program to find the given number is even or odd */

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

```
getch();
}
```

Output

```
Enter the number: 10
10 is an even number.
```

❖ **Conditional Expression**

- The *ternary operator* is used to form a conditional expression. It uses three operands and hence it is called as a *ternary operator*. The syntax for a conditional expression is:

  <expression-1> ? <expression-2> : <expression-3>;

- In this method if expression-1 is true then expression-2 is evaluated, otherwise expression-3 is evaluated.

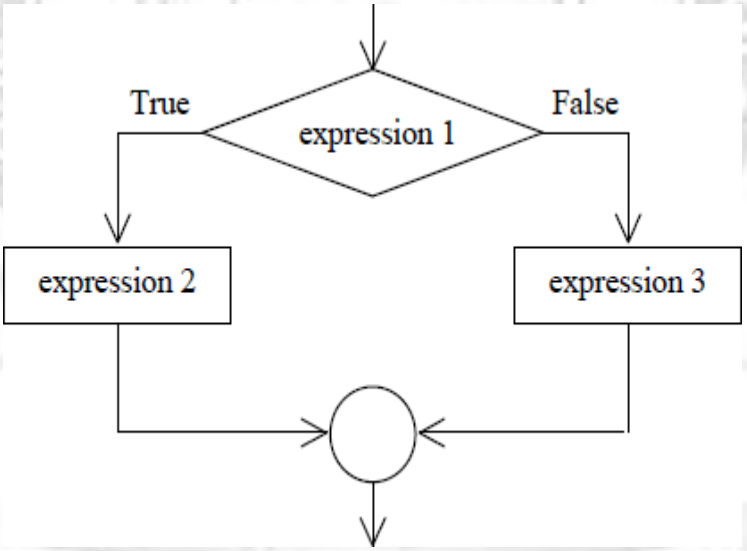

Fig. 1.9 Flowchart for a Conditional expression

Program 1.14

/* Program to find biggest of two given numbers */

```
#include<stdio.h>
void main()
{
int x,y,z;
printf("\n Enter the value of x and y:");
```

```
scanf("%d%d",&x,&y);
z = ((x>y)?x:y);
printf("The biggest value is %d",z);
getch();
}
```

Output

Enter the value of x and y: 5 10

The biggest value is 10

- **If-else-if statement**
  - The syntax for the if-else-if statement is

```
if(expression1)
{
statements1;
}
else if(expression2)
{
statements2;
}
else if(expression3)
{
statements3;
}
else
{
statements4;
}
```

- In this statement, if the expression1 is true, statements1 will be executed, otherwise the expression2 is evaluated, if it is true then statements2 is executed, otherwise the expression3 is evaluated, if it is true then statements3 is executed, otherwise statements4 is executed.

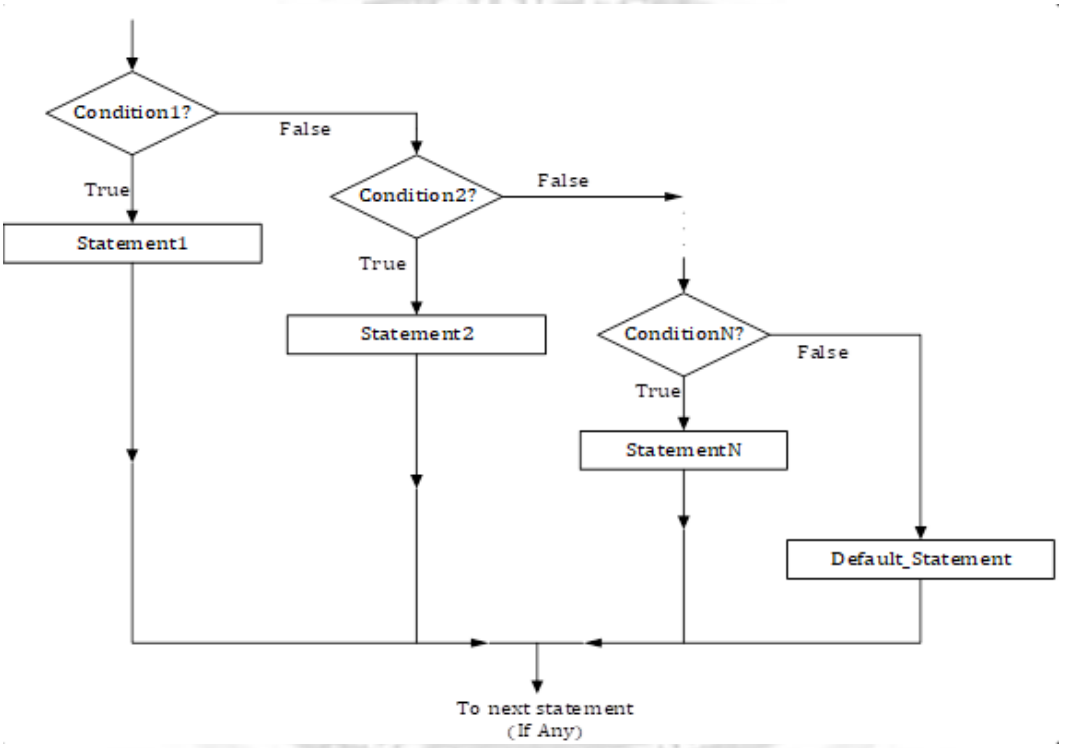

Fig. 1.10 Flowchart for the If-else-if statement

Program 1.15

/* Program to find the student's class for the given average marks using if-elseif*/

```
#include<stdio.h>
void main()
{
int Avg_Mark;
printf ("Enter the Average mark:")
scanf("%d",&Avg_Mark);
if(Avg_Mark>=75)
{
```

```
printf("Distinction");
}
elseif((Avg_Mark>=60) && (Avg_Mark<75))
{
printf("First Class");
}
elseif((Avg_Mark>=50) && (Avg_Mark<60))
{
printf("Second Class");
}
elseif((Avg_Mark>=45) && (Avg_Mark<50))
{
printf("Third Class");
}
else
{
printf("Fail");
}
}
```

Output

Enter the Average Mark : 65

First Class

Program 1.16

Write a program to calculate the gross salary for the conditions given below:

Basic Salary (Rs)	DA (Rs)	HRA (Rs)	Conveyance (Rs)
Basic >= 5000	110% of basic	20% of basic	500
Basic >= 3000 && basic < 5000	100% of basic	15% of basic	400
Basic < 3000	90% of basic	10% of basic	300

/* Program to Calculate the gross salary */

```
#include<stdio.h>
#include<conio.h>
void main()
{
float bs, hra, da, cv, ts;
clrscr();
printf("\n Enter Basic salary:");
scanf("%f",&bs);
if(bs>=5000)
{
hra = bs*20/100;
da = bs*110/100;
cv = 500;
}
else if(bs>=3000 && bs<5000)
{
hra = bs*15/100;
da = bs*100/100;
cv = 400;
}
else if(bs<3000)
{
```

```
hra = bs*10/100;
da = bs*90/100;
cv = 300;
}
ts = bs+hra+da+cv;
printf("\nBasic salary: %5.2f",bs);
printf("\nHRA: %5.2f",hra);
printf("\nDA: %5.2f",da);
printf("\nConveyance: %5.2f",cv);
printf("\nGross Salary: %5.2f",ts);
getch();
}
```

Output

Enter basic salary: 5400

Basic salary: 5400

HRA: 1080

DA: 5940

Conveyance: 500

Gross salary: 12920

- **Nested if statement**
  - The syntax for the nested if statement is

```
if(expression1)
{
statements1;
}
else
{
if(expression2)
```

{

statements2;

}

else

{

statements3;

}

}

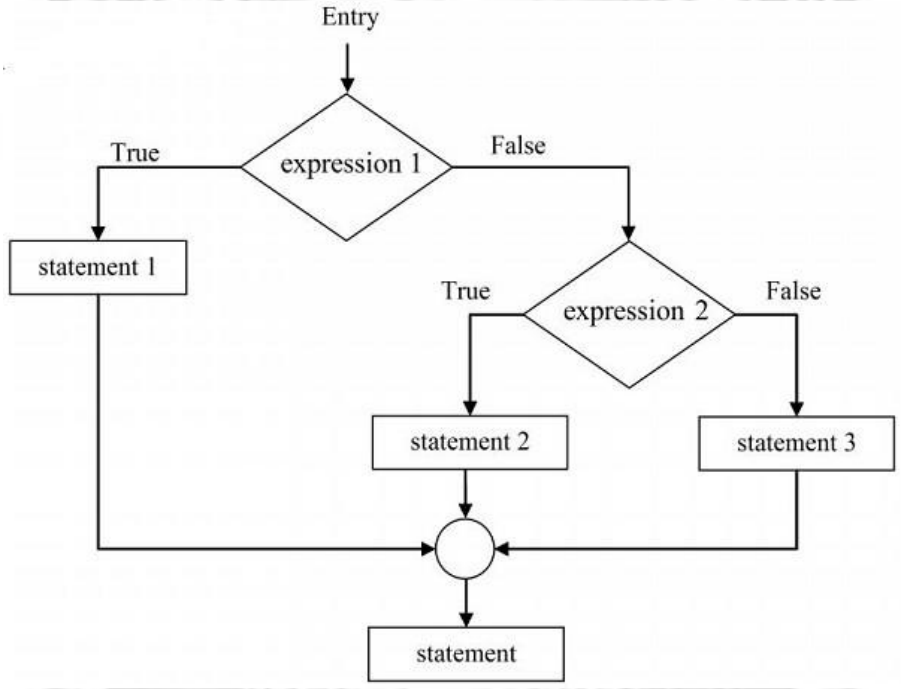

Fig. 1.11 Flowchart for Nested if statement

- In this statement, if expression1 is true, then statement1 is evaluated, otherwise the inner if expression2 is true then statements2 will be executed otherwise inner else statements3 will be executed.

Program 1.17

/* Program to find the biggest of given three numbers */

#include<stdio.h>

void main()

```
{
int x,y,z;
printf("\n Enter the three numbers");
scanf("%d%d%d",&x,&y,&z);
if ((x>y) && (x>z))
{
printf("The Biggest number = %d",x);
}
else
{
if(y>z)
{
printf("The Biggest number =%d",y);
}
else
{
printf("The Biggest number =%d",z);
}
}
getch();
}
```

Output

Enter the three numbers: 5 2 8

The Biggest number = 8

- **Switch () Case Statement**
  - The switch ( ) case statement is like if statement that allows us to make a decision from a number of choices. The switch statement requires only one argument of any data type, which is checked with a number of case options.
  - The switch statement evaluates the expression and then looks for its value among

the case constants.

- If the value matches with a case constant, this particular case statement is executed. If not, the default is executed. The general syntax for the switch - case statement is:

```
switch<exprn>
{
case constant_1:
{
statements1;
break;
}
case constant_2:
{
statements2;
break;
}
case constant_3:
{
statements3;
break;
}
case constant_n:
{
statementsn;
break;
}
default:
{
default statements;
}
```

}

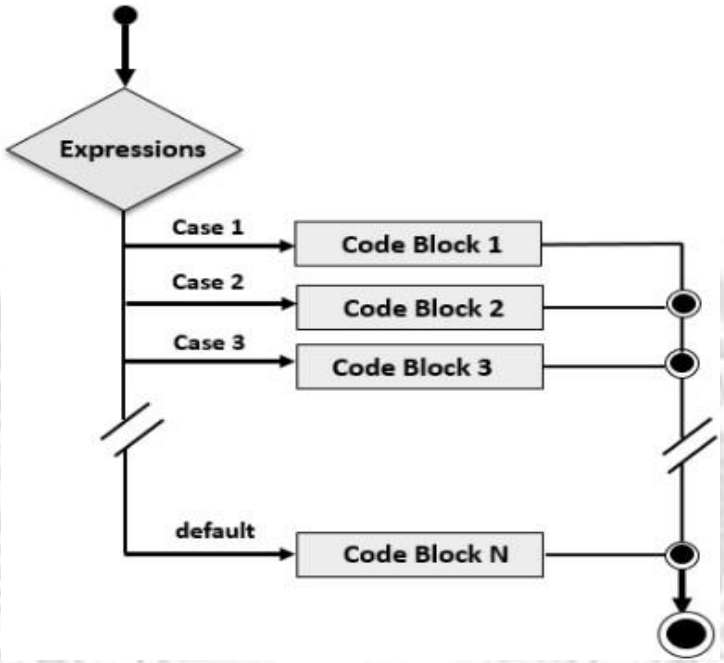

Fig. 1.12 Flowchart for Switch - Case statement

Program 1.18

/* Program to provide multiple functions such as 1. Addition 2. Subtraction 3. Multiplication 4. Division by using switch statements. */

```
#include<stdio.h>
#include<conio.h>
void main()
{
float c;
int a,b,n;
printf("\n MENU");
printf("\n 1.Addition");
printf("\n 2.Subtraction");
printf("\n 3.Multiplication");
printf("\n 4.Division");
```

```
printf("\n 0.Exit");
printf("\n Enter your choice:");
scanf("%d",&n);
printf("Enter two numbers:");
scanf("%d%d",&a,&b);
switch(n)
{
case 1:
c = a + b;
printf("\n Addition: %d",c);
break;
case 2:
c = a - b;
printf("\n Subtraction: %d",c);
break;
case 3:
c = a * b;
printf("\n Multiplication: %d",c);
break;
case 4:
c = a / b;
printf("\n Division: %d",c);
break;
case 0:
exit();
break;
default:
printf("Invalid choice");
```

break;

}

getch();

}

Output

Menu

1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exit

Enter your choice: 2

Enter two numbers: 40 20

Subtraction: 20

Looping Statements

- A loop is defined as a block of statements which are repeatedly executed for certain number of times. The `C' language supports three types of loop statements.
  1. *for* statement
  2. *while* statement
  3. *do-while* statement

❖ For Loop Statement

- The *for* loop allows to execute a set of instructions until a certain condition is satisfied. Condition may be predefined or open-ended.
- The syntax *for loop* this is follows:

for<initial value>;<condition>;<incrementation/decrementation>)

{

block of statements;

}

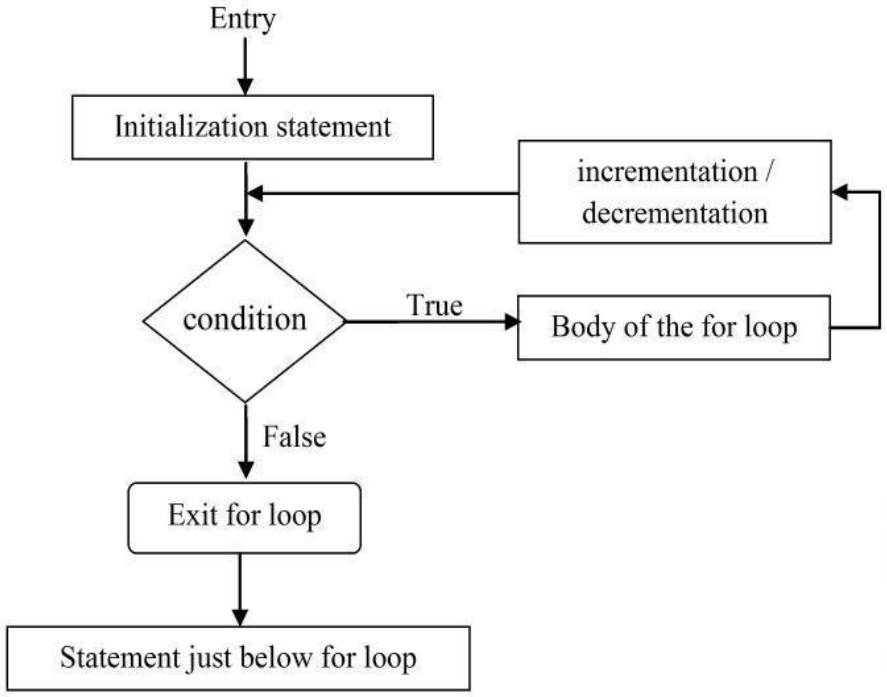

Fig. 1.13 Flowchart of the For loop statement

- Here the initial value means the starting value assigned to the variable and condition in the loop counter to determine whether the loop should continue or not. Incremention / decrementation is to increment/decrement the loop counter value each time the program segment has been executed.

Program 1.19

/* Program to Generate numbers from 1 to 10 */

```
#include<stdio.h>
void main()
{
int i,n;
printf("\n Enter the limit");
scanf("%d",&n);
for(i=1;i <=n;i++)
{
printf("%d\n",i);
```

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

Output

Enter the limit: 10

1

2

3

4

5

6

7

8

9

10

❖ **While Loop Statement**

- The syntax for the *while loop* statement is

```
while(condition)
{
block of statements;
incr/decr;
}
```

- The while loop is often used when the number of times the loop is to be executed is not known in advance. A sequence of statements are executed until some condition is satisfied.
- When the condition specified inside the parenthesis the while loop is satisfied, the control is transferred to the statements inside the loop and executes the body of the loop. The loop continues until the condition is violated. The while tests the condition before each iteration.
- If the condition initially fails the loop is skipped entirely even in the first iteration

itself. It is otherwise called as entry controlled loop.

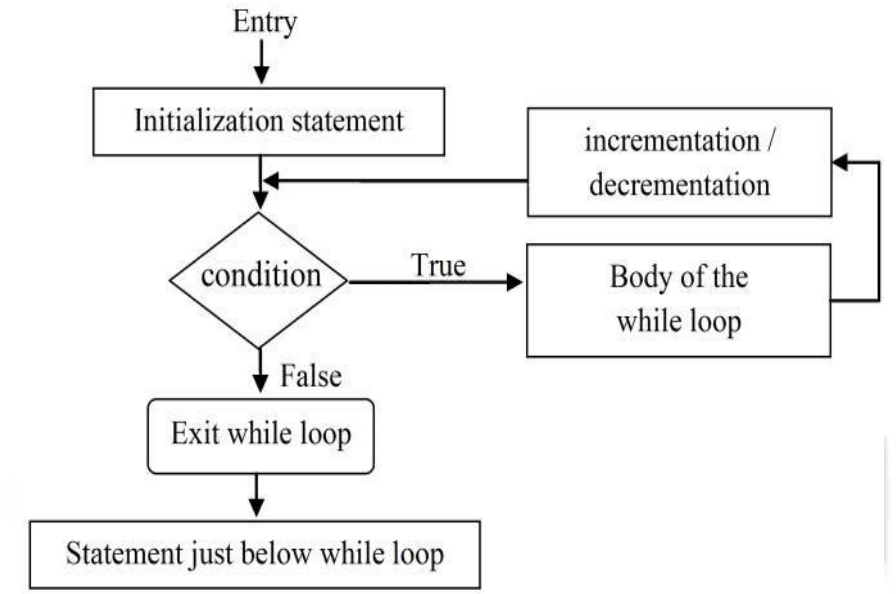

Fig. 1.14 Flowchart of the While loop

Program 1.20

/* Program to Generate the Even numbers to a given limit*/

```
#include<stdio.h>
#include<conio.h>
void main()
{
int n,i;
printf("\n Enter the limit:");
scanf("%d",&n);
i=1;
while(i<=n)
{
if(i%2==0)
printf("%d\t",i);
```

```
i++;
}
getch( );
}
```

Output

```
Enter the limit: 10
2  4 6 8 10
```

❖ **Do While Statement**

- The *do while loop* varies from the while loop in the checking condition. The condition of the loop is not tested until the body of the loop has been executed once. If the condition is false, after the first loop iteration the loop terminates. The statements are executed atleast once even if the condition fails for the first time itself. It is otherwise called as exit control loop.

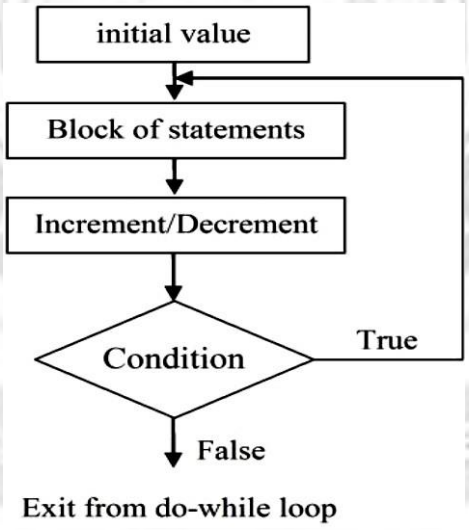

Fig. 1.15 Flowchart of the Do while loop

- The syntax for the do while loop is

```
do
{
statements;
```

Program 1.21

```
        while(condition);
}
```

/* Program to check the given number is palindrome or not */

```
#include<stdio.h>
#include<conio.h>
void main()
{
int n,t,s,r;
clrscr();
printf("\n Enter the number");
scanf("%d",&n);
t=n;
s=0;
do
{
r=n%10;
s=(s*10)+r;
n=n/10;
} while(n>0);
if(t==s)
printf("%d is a palindrome",t);
else
printf ("%d is not a palindrome", t);
getch( );
}
```

Output

Enter the number 242

242 is a palindrome