



Chapter 3

Control Statements

Learning Objectives

After completing this chapter, you will be able to:

- *Understand flowcharts.*
- *Understand control structures.*
- *Understand sequential control statements.*
- *Understand decision control statements.*
- *Understand loop control statements.*
- *Understand jump statements.*
- *Understand logical operators.*

In this chapter, you will learn about the flowcharts and the control structures.

FLOWCHART

A flowchart is a graphical representation of the steps that constitute a program. It shows how the control moves in a program. A flowchart is drawn using some special symbols, which are as follows:

Oval

The oval symbol represents the start and the end of the program. The symbol is as follows:

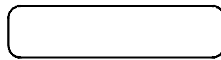


Figure 3-1 An oval symbol

Rectangle

The rectangle symbol represents the process box in which actions such as calculations are performed. The symbol is as follows:



Figure 3-2 A rectangle symbol

Diamond

The diamond symbol represents the decision box in which a condition is checked. The symbol is as follows:

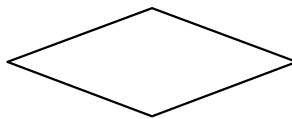


Figure 3-3 A diamond symbol

Arrow

The arrow symbol represents the path through which the control passes from one symbol to another symbol. The symbol is as follows:



Figure 3-4 An arrow symbol

Parallelogram

The parallelogram symbol represents the input or the output box. The symbol is as follows:

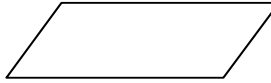


Figure 3-5 A parallelogram symbol

CONTROL STRUCTURES

The programs, during execution, may need to take decisions and repeat a particular block of code a number of times. C++ provides **control structures** or **control statements** that specify the order of the execution of the statements. The control structures control the flow of the execution in a program. The control structures are of three types:

- a. Sequential control structure
- b. Selection or Decision control structure
- c. Repetition, Iteration, or Loop control structure

Sequential Control Structure

In a sequential control structure, the program statements are executed one after the other in the same order as they appear in the program. In all programming languages, the sequential control is used as the default control. All the programs that have been described in the earlier chapters are sequential.

Selection or Decision Control Structure

The selection or decision control structure is used to alter the flow of control in a program. The flow of control depends on the result of a particular condition applied to it. C++ supports three different types of selection control structures:

- a. if statement
- b. if-else statement
- c. switch statement

The if Statement

The **if** statement is a single path statement, which means it will execute a statement or a block of statements only if the condition is true. The syntax for the **if** statement is as follows:

```
if(condition or expression)
    statement1;
```

In the above syntax, the condition or expression can be true or false. If the condition or expression is true, **statement1** will be executed. Otherwise, the control will be transferred to the next statement after the **if** block.

In case there is more than one statement that must be executed if the condition is true, then the statements should be grouped together in braces {}. The syntax for the **if** statement is as follows:

```
if(condition or expression)
{
    statement 1;
    statement 2;
    _____
    _____
    statement n;
}
```

In the above syntax, all the statements from 1 to n will be executed, if the condition or expression within the **if** statement is true. Otherwise, they will not be executed and the control will be transferred to the next statement after the **if** block.

The following example illustrates the use of the **if** statement.

Example 1

Write a program to find the greater of the two numbers.

The following program will prompt the user to enter two numbers, compare them, and display the greater of the two on the screen. The numbers on the right are not a part of the program and are for reference only.

```
//Write a program to find the greater of the two numbers      1
#include<iostream>                                           2
using namespace std;                                         3
int main()                                                    4
{                                                            5
    int a,b;                                                  6
    cout<<"Enter two numbers"<<endl;                        7
    cin>>a>>b;                                               8
    if(a>b)                                                  9
        cout<<"a is greater than b"<<endl;                 10
    cout<<"b is greater than a"<<endl;                       11
    return 0;                                                12
}                                                            13
```

Explanation

Line 6

int a,b;

In this line, **a** and **b** are declared as integer type variables.

Line 7

```
cout<<"Enter two numbers"<<endl;
```

This line will display the following on the screen:

Enter two numbers

Line 8

```
cin>>a>>b;
```

This line is used to accept the values of variables **a** and **b** from the user.

Line 9

```
if(a>b)
```

In this line, the **if** statement is used to check the condition whether it is **true** or **false**. If the condition **a>b** (a is greater than b) is true, the next line (line 10) will be executed. Otherwise, line 10 will be skipped by the compiler and the control will be transferred to line 11.

Line 10

```
cout<<"a is greater than b"<<endl;
```

This line will display the following on the screen:
a is greater than b

Line 11

```
cout<<"b is greater than a"<<endl;
```

This line will display the following on the screen:
b is greater than a

The output of the program is:

Enter two numbers

3

2

a is greater than b

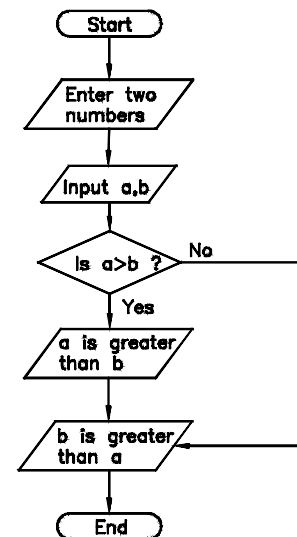


Figure 3-6 Flowchart of Example 1

The flowchart in Figure 3-6 gives a diagrammatic representation of the program described in Example 1.

The if-else Statement

The **if-else** statement is a dual path statement, which means if the condition given within the **if** statement is true, the statements associated with the **if** block will be executed. Otherwise, the statements associated with the **else** block will be executed. The syntax for the **if-else** statement is as follows:

```

if(condition or expression)
{
    statement 1;
    statement 2;
}
  
```

```

    }
    else
    {
        statement 3;
        statement 4;
    }

```

In the above syntax, if the given condition or expression is true, **statements 1** and **2** will be executed and the **else** block will be skipped. Otherwise, the **if** block will be skipped and the **statements 3** and **4**, which are associated with the **else** block, will be executed.

The following example illustrates the use of the **if-else** statement.

Example 2

Write a program to find the greater of the two numbers by using the **if-else** statement.

The following program will prompt the user to enter two numbers, compare them, and display the greater of the two on the screen.

```

//Write a program to find the greater of the two numbers          1
#include<iostream>                                                2
using namespace std;                                             3
int main()                                                        4
{                                                                    5
    int a,b;                                                       6
    cout<<"Enter two numbers"<<endl;                             7
    cin>>a>>b;                                                    8
    if (a>b)                                                       9
        cout<<"a is greater than b" <<endl;                     10
    else                                                           11
        cout<<"b is greater than a" <<endl;                       12
    return 0;                                                      13
}                                                                    14

```

Explanation

Line 6

int a,b;

In this line, **a** and **b** are declared as integer type variables.

Line 7

cout<<"Enter two numbers"<<endl;

This line will display the following on the screen:

Enter two numbers

Line 8

```
cin>>a>>b;
```

This line is used to accept the values of variables **a** and **b** from the user.

Line 9

```
if (a>b)
```

In this line, the **if** statement is used to verify whether the condition within the **if** statement is **true** or **false**. If the condition **a>b** (a is greater than b) is true, the next line (line 10) will be executed. Otherwise, line 10 will be skipped and the control will be transferred to line 11.

Line 10

```
cout<<"a is greater than b" <<endl;
```

This line will display the following on the screen:
a is greater than b

Line 11

```
else
```

This statement and the block of code associated with this will be executed only if the condition given in the **if** statement is false. Otherwise, the whole **else** block will be skipped.

Line 12

```
cout<<"b is greater than a" <<endl;
```

This line will display the following on the screen:
b is greater than a

The output of this program is as follows:

Enter two numbers

2

3

b is greater than a

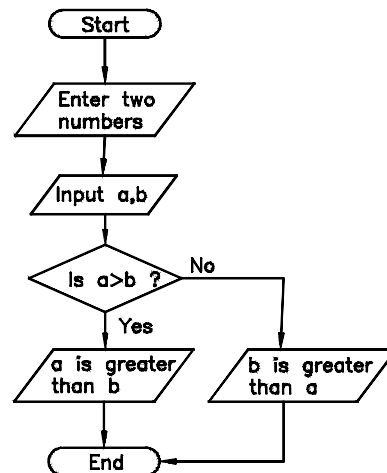


Figure 3-7 Flowchart of Example 2

The flowchart in Figure 3-7 gives a diagrammatic representation of the program described in Example 2.

Nested-if statement

When an **if** statement is used within another **if** statement, the resulting statement is known as a **nested-if** statement. In the **nested-if** structure, the last **else** statement is always associated with the **if** block that precedes it. The syntax for the **nested-if** statement is as follows:

```

if(condition)
{
    if(condition)

```

```

        {
            statements;
            if(condition)
            else
            {
                statements;
            }
        }
    }

```

The following program illustrates the use of the **nested-if** statement.

Example 3

Write a program to find the greatest of the three numbers by using the **nested-if** statement.

The program given next will prompt the user to enter three numbers, compare them, and display the greatest of the three numbers on the screen.

```

//Write a program to find the greatest of the three numbers      1
#include<iostream>                                              2
using namespace std;                                           3
int main()                                                       4
{                                                                 5
    int a,b,c;                                                  6
    cout<<"Enter three numbers"<<endl;                          7
    cin>>a>>b>>c;                                              8
    if(a>b)                                                      9
    {                                                            10
        if(a>c)                                                  11
        {                                                        12
            cout<<"a is the greatest number"<<endl;              13
        }                                                        14
        else                                                    15
        {                                                        16
            cout<<"c is the greatest number"<<endl;              17
        }                                                        18
    }                                                            19
    else                                                         20
    {                                                            21
        if(b>c)                                                  22
        {                                                        23
            cout<<"b is the greatest number"<<endl;              24
        }                                                        25
        else                                                    26
        {                                                        27
            cout<<"c is the greatest number"<<endl;              28
        }
    }
}

```



```
        }
    }
}
```

29
30
31

Explanation

Line 6

int a,b,c;

In this line, **a**, **b**, and **c** are declared as integer type variables.

Line 7

cout<<"Enter three numbers"<<endl;

This line will display the following on the screen:

Enter three numbers

Line 8

cin>>a>>b>>c;

This line is used to accept the values of variables **a**, **b**, and **c** from the user.

Line 9

if(a>b)

In this line, the **if** statement is used to check whether the value of variable **a** is greater than the value of variable **b**. If the condition is true, the control will be transferred to line 11. Otherwise, the control will be transferred to line 20.

Line 10

{

This line indicates the start of the **if** statement (line 9).

Line 11

if(a>c)

If the condition given in line 9 is true, the control will pass to line 11. Otherwise, this line will be skipped by the compiler. In this line, the **if** statement is used to check whether the value of variable **a** is greater than the value of variable **c**. If the condition is true, the control will be transferred to line 13. Otherwise, the control will be transferred to line 15.

Line 12

{

This line indicates the start of the **if** statement (line 11).

Line 13

cout<<"a is the greatest number"<<endl;

This line will display the following on the screen:

a is the greatest number

Line 14

}

This line indicates the end of the **if** statement (line 11).

Line 15

else

If the condition given in line 11 is true, the control will pass to line 15. Otherwise, the entire **else** block will be skipped by the compiler.

Line 16

{

This line indicates the start of the **else** statement (line 15).

Line 17

cout<<"c is the greatest number"<<endl;

This line will display the following on the screen:
c is the greatest number

Line 18

}

This line indicates the end of the **else** statement (line 15).

Line 19

}

This line indicates the end of the **if** statement (line 9).

Line 20

else

This **else** block is associated with the **if** statement given in line 9. If the condition given in line 9 is false, the control will pass to the **else** statement. Otherwise, the entire **else** block will be skipped by the compiler.

Line 22

if(b>c)

In this line, the **if** statement is used to check whether the value of variable **b** is greater than the value of variable **c**. If the condition is true, line 24 will be executed. Otherwise, it will be skipped and the control will be transferred to line 26.

Line 24

cout<<"b is the greatest number"<<endl;

This line will display the following on the screen:
b is the greatest number

Line 26

else

This **else** block is associated with the **if** statement given in line 22. If the condition given in line 22 is false, the control will pass to the **else** statement. Otherwise, the entire **else** block will be skipped by the compiler.

Line 28

```
cout<<"c is the greatest number"<<endl;
```

This line will display the following on the screen:

c is the greatest number

The output of the program is as follows:

Enter three numbers

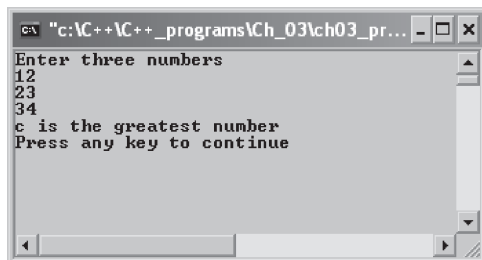
12

23

34

c is the greatest number

The output will be displayed on the screen as follows:



The flowchart in Figure 3-8 gives a diagrammatic representation of the program described in Example 3.

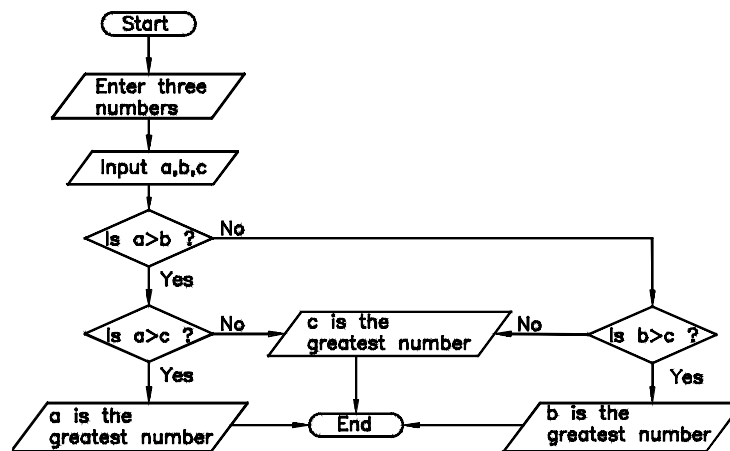


Figure 3-8 Flowchart of Example 3

The else if Statement

The **else if** statement is a conditional statement that is used when you want to verify more than one condition. The working of the **else if** statement is the same as that of the **if** statement, except that the **else if** statement must be preceded by an **if** statement. The syntax for the **else if** statement is as follows:

```
if(condition)
    statement1;
else if(condition)
    statement2;
else
    statement3;
```

In the above syntax, the condition given in the **if** statement is checked if it is true or false. If the condition is true, **statement1** will be executed. Otherwise, the condition given in the **else if** statement will be checked. If it is true, **statement2** will be executed. Otherwise, **statement3** will be executed.

The following program illustrates the use of the **else if** statement.

Example 4

Write a program to display the grades based on the points scored by the user.

The following program will prompt the user to enter the points scored, calculate the grades, and display the equivalent grade on the screen.

```
//Write a program that displays the grades according to the points scored
#include<iostream>
using namespace std;
int main()
{
    int points_scored;
    cout<<"Enter the points scored"<<endl;
    cin>>points_scored;
    if(points_scored>=90)
        cout<<"Grade A"<<endl;
    else if(points_scored>=80)
        cout<<"Grade B"<<endl;
    else if(points_scored>=70)
        cout<<"Grade C"<<endl;
    else
        cout<<"FAIL"<<endl;
    return 0;
}
```

Explanation

Line 6

```
int points_scored;
```

In this line, **points_scored** is declared as an integer type variable.

Line 7

```
cout<<"Enter the points scored"<<endl;
```

This line will display the following on the screen:

Enter the points scored

Line 8

```
cin>>points_scored;
```

This line is used to accept the value of the variable **points_scored** from the user.

Line 9

```
if(points>=90)
```

In this line, the condition (the value of the variable **points_scored** is greater than or equal to 90) within the **if** statement is checked. If the condition is true, the next statement (line 10) will be executed. Otherwise, the next statement (line 10) will be skipped and the control will be transferred to the **else-if** statement (line 11).

Line 10

```
cout<<"Grade A"<<endl;
```

This line will display the following on the screen:

Grade A

Line 11

```
else if(points_scored>=80)
```

If the condition given in the **if** statement is false, the control will pass to line 11. In this line, the condition (the value of the variable **points_scored** is greater than or equal to 80) within the **else-if** is checked. If the condition is true, **line 12** will be executed. Otherwise, the control will be transferred to **line 13**.

Line 12

```
cout<<"Grade B"<<endl;
```

This line will display the following on the screen:

Grade B

Line 13

```
else if(points_scored>=70)
```

If the condition given in **line 11** is false, the control will pass to this line (line 13). In this line, the condition (the value of the variable **points_scored** is greater than or equal to 70) within the **else if** is checked. If the condition is true, line 14 will be executed. Otherwise, the control will be transferred to line 15.

Line 14

```
cout<<"Grade C"<<endl;
```

This line will display the following on the screen:

Grade C

Line 15

```
else
```

If the compiler cannot find any conditional match in the program, the control will be transferred to line 15 and the statements associated with it will be executed.

Line 16

```
cout<<"FAIL"<<endl;
```

This line will display the following on the screen:

FAIL

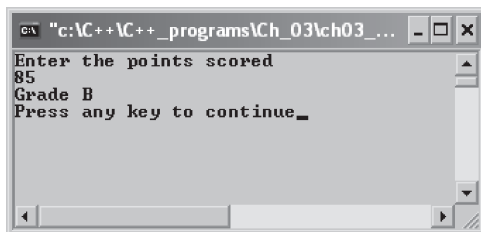
The output of the program is as follows:

Enter the points scored

85

Grade B

The output will be displayed on the screen as follows:



The switch Statement

The **switch** statement is a selection or a case control statement, which makes it possible for the compiler to transfer the control to different statements within the **switch** body depending on the value of a variable or expression. In a **switch** statement, the flow of execution is controlled by a value of the variable or expression. This variable or expression is known as the control variable. In the same **switch** body, two case constants cannot have identical values. Also, the upper and lowercase character constants are differentiated by the compiler.

The syntax for the **switch** statement is as follows:

```
switch(expression)
{
    case constant 1:
        statement;
        break;
    case constant 2:
        statement;
```

```

        break;
    case constant n:
        statement;
        break;
    default: //Optional
        statement;
}

```

In the above syntax, the value of the **switch** expression is matched with the case constants one by one. If a match is found, the control will be transferred to the statement associated with that particular case label. If no match is found, the default case will be executed in case it exists. If, on the other hand, a match is not found and also there is no default case in the **switch** body, no case will be executed.

In the above code, the **break** statement is optional. The **break** statement is used to transfer the control to the end of the **switch** body. If the **break** statement is not used in the **switch** body, then all the cases from the one matched will be executed.

The following example illustrates the use of the **switch** statement without using the **break** statement.

Example 5

Write a program to display a message according to the value entered by the user.

The following program will prompt the user to enter a number, compare it with the given cases, and display the statement associated with the particular case.

```

//Write a program that illustrates the use of the switch statement
#include<iostream>
using namespace std;
int main()
{
    int i;
    cout<<"Enter a number"<<endl;
    cin>>i;
    switch(i)
    {
        case 1:
            cout<<"GOOD MORNING"<<endl;
        case 2:
            cout<<"GOOD AFTERNOON"<<endl;
        case 3:
            cout<<"GOOD EVENING"<<endl;
        case 4:
            cout<<"GOOD NIGHT"<<endl;
    }
}

```

```

        default:                                19
        cout<<"HAVE A NICE DAY"<<endl;          20
    }                                           21
    return 0;                                  22
}                                              23

```

Explanation

Line 6

int i;

In this line, **i** is declared as an integer type variable.

Line 7

cout<<"Enter a number"<<endl;

This line will display the following on the screen:

Enter a number

Line 8

cin>>i;

This line is used to accept the value of the variable **i** from the user.

Line 9

switch(i)

In this line, the **switch** statement is used to match the value of the variable **i** with all the case constants one by one. If a match is found, the control will be transferred to the statement associated with that particular case.

Line 10

{

This line is used to indicate the start of the **switch** body.

Line 11

case 1:

This line begins with the **case** keyword, which is followed by an integer value (**1**). This value is matched with the value of the control variable (**i**). If a match is found, line 12 will be executed. Otherwise, the control will be transferred to **case 2**.

Line 12

cout<<"GOOD MORNING"<<endl;

This line will display the following on the screen:

GOOD MORNING

The functionality of lines 13 to 18 is the same as that of lines 11 and 12.

Line 19

default:

In this line, the **default** keyword is used. If no match is found, the code or statement associated with the **default** case will be executed. This statement is optional.

Line 20

```
cout<<"HAVE A NICE DAY"<<endl;
```

This line will display the following on the screen:

HAVE A NICE DAY

The output of the program is:

Enter a number

2

GOOD AFTERNOON

GOOD EVENING

GOOD NIGHT

HAVE A NICE DAY

**Note**

In this program, no break statement has been used. Therefore, all the cases from the one matched, including the default, are executed, as shown in the output of the program.

The following example illustrates the use of the **switch** statement with the **break** statement.

Example 6

Write a program to display a message according to the value entered by the user.

The following program will prompt the user to enter a number, compare it with the given cases, and display the statement associated with the particular case.

```
//Write a program that illustrates the use of the switch statement
//with a break statement
#include<iostream>
using namespace std;
int main()
{
    int i;
    cout<<"Enter a number"<<endl;
    cin>>i;
    switch(i)
    {
        case 1:
            cout<<"GOOD MORNING"<<endl;
            break;
        case 2:
            cout<<"GOOD AFTERNOON"<<endl;
            break;
        case 3:
            cout<<"GOOD EVENING"<<endl;
            break;
    }
```

```

        case 4:                                20
        cout<<"GOOD NIGHT"<<endl;            21
        break;                                22
        default:                                23
        cout<<"HAVE A NICE DAY"<<endl;        24
    }                                           25
    return 0;                                  26
}                                              27

```

The functioning of Example 6 is the same as Example 5. In Example 6, the **break** statement has been used. As a result, the program executes only the matched case and its associated statements. For example, if the user enters 3, only **case 3** will be executed and all the other cases will be skipped by the compiler.

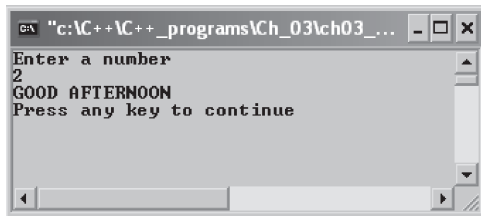
The output of the program is as follows:

Enter a number

2

GOOD AFTERNOON

The output will be displayed on the screen as follows:



The flowchart in Figure 3-9 gives a diagrammatic representation of the program described in Example 6.

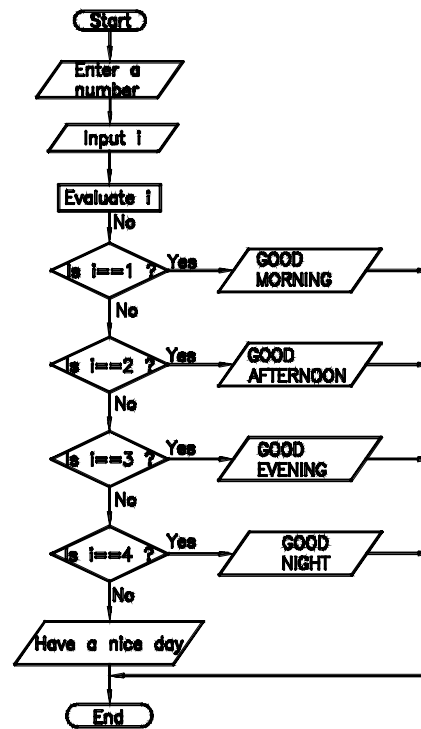


Figure 3-9 Flowchart of Example 6

Repetition, Iteration, or Loop Control Structure

Loops are used to repeat a particular block of code for a certain number of times. The block of code is repeated while the condition is true. If the condition is false, the loop ends and the control is transferred to the next statement immediately after the loop. In C++, there are the following three kinds of loops:

1. while loop
2. for loop
3. do-while loop

The while Loop

The **while** loop is a flow control statement that is used to execute a particular block of code as long as a certain condition is true. The **while** loop is mostly used in cases when you do not know how many times the loop will be repeated. The syntax for the **while** loop is given next.

```
while(condition)
{
    statement or block of code;
}
```

In the above syntax, the condition within the **while** statement is evaluated first. If the condition is true, the next statement or the block of code is executed. This code will be repeated until the condition evaluates to false.

The following example illustrates the use of the **while** loop.

Example 7

Write a program to illustrate the working of the **while** loop.

The following program will prompt the user to enter a number, compare it with a particular condition, and display the resultant value on the screen.

```
//Write a program that will prompt the user to enter a number
//and then repeat the block of code until it is not equal to zero.
#include<iostream>
using namespace std;
int main()
{
    int n=1;
    while(n!=0)
    {
        cout<<"Enter a value"<<endl;
        cin>>n;
        cout<<"The value entered by you is: "<<n<<endl;
    }
    cout<<"Exit from the while loop"<<endl;
    return 0;
}
```

Explanation

Line 7

int n=1;

In this line, **n** is declared as an integer type variable and the value one is assigned to it.

Line 8

while(n!=0)

From this line, the **while** loop begins. The code associated with the **while** statement will be executed as long as the condition **n!=0** (**n** is not equal to zero) is true. In this program, the variable **n** is initialized to one, as mentioned in line 7. So the code associated with the while statement must be executed at least once and then the control will be transferred to line 9.

Line 9

```
cout<<"Enter a value"<<endl;
```

This line will display the following on the screen:

Enter a value

Line 10

```
cin>>n;
```

This line is used to accept the value of variable **n** from the user.

Line 11

```
cout<<"The value entered by you is: "<<n<<endl;
```

This line will display the following on the screen:

The value entered by you is: value of the variable **n**

Line 13

```
cout<<"Exit from the while loop"<<endl;
```

When the **while** condition is false, the control is directly transferred to this line. This line will display the following on the screen:

Exit from the while loop

The output of the program is as follows:

Enter a value

4

The value entered by you is:

4

Enter a value

0

The value entered by you is:

0

Exit from the while loop

The flowchart in Figure 3-10 gives a diagrammatic representation of the program described in Example 7.

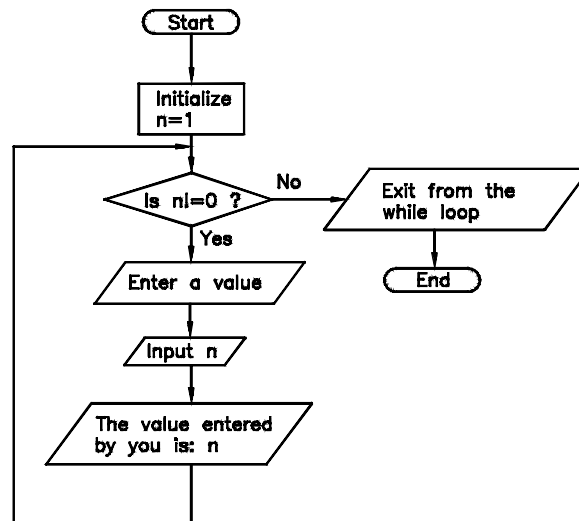


Figure 3-10 Flowchart of Example 7

The for Loop

The **for** loop is also a flow control statement that is used to execute a particular block of code for a specific number of times. The **for** loop is mostly used when you know the number of times the body of the loop will be executed. The **for** loop is easy to understand because all the control elements (initialization, condition, and increment or decrement) are placed together at one place. The syntax for the **for** loop is as follows:

```
for(initialization; condition; increment or decrement)
```

In the above syntax, the **for** loop consists of three expressions: initialization, condition, and increment or decrement. These three expressions must be separated by using a semicolon (;).

Initialization Expression

The initialization expression executes only once at the start of the loop. It is used to set the initial value of the loop control variable such as **int x=1**.

Condition Expression

The condition expression is evaluated each time before the execution of the body of the loop. The execution of the body of the loop depends on the condition whether it is true or false. If the condition is true, the body of the loop will be executed. Otherwise, it will be skipped and the control will be transferred to the next instruction after the **for** loop.

Increment or Decrement Expression

The increment or decrement expression is used to increase or decrease the value of the loop control variable by one or some other value, which is specified by the programmer. This expression is always executed after the body of the loop has been executed.

The following example illustrates the use of the **for** loop.

Example 8

Write a program to display the multiplication table of a number.

The following program will prompt the user to enter a number, calculate its multiples, and display the table on the screen.

```
//Write a program that displays the multiplication table of a number
#include<iostream>
using namespace std;
int main()
{
    int num,i,c;
    cout<<"Enter a number"<<endl;
    cin>>num;
    cout<<"The multiplication table is as follows: "<<endl;
    for(i=1;i<=10;i++)
```

1
2
3
4
5
6
7
8
9
10

```
        {                                     11
            c=num*i;                          12
            cout<<c<<endl;                    13
        }                                     14
    return 0;                                15
}
```

Explanation

Line 6

int num,i,c;

In this line, **num**, **i**, and **c** are declared as integer type variables.

Line 7

cout<<"Enter a number"<<endl;

This line will display the following on the screen:

Enter a number

Line 8

cin>>num;

This line is used to accept the value of variable **num** from the user.

Line 9

cout<<"The multiplication table is as follows:"<<endl;

This line will display the following on the screen:

The multiplication table is as follows:

Line 10

for(i=1;i<=10;i++)

This line shows the working of the **for** loop. The variable **i** (loop control variable) is initialized to one and the condition **i<=10** is checked. If the condition is true, the body of the loop will be executed and the control will be transferred to line 12. Otherwise, the body of the loop will be skipped and the control will be transferred to line 15.

Line 12

c=num*i;

In this line, the value of the variable **num** is multiplied by the value of variable **i** and the resultant value is assigned to the variable **c**.

Line 13

cout<<c<<endl;

This line will display the following on the screen:

Value of the variable **c**

The output of the program is as follows:

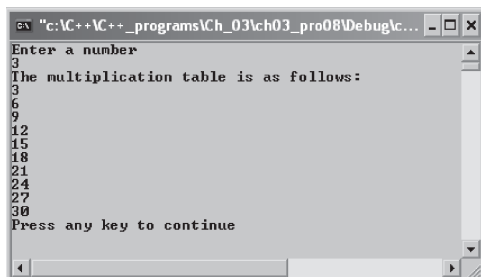
Enter a number

3

The multiplication table is as follows:

3
6
9
12
15
18
21
24
27
30

The output will be displayed on the screen as follows:



The flowchart in Figure 3-11 gives a diagrammatic representation of the program described in Example 8.

Multiple Initialization

You can specify more than one expression in the initialization part and as well as the increment/decrement part by separating them using the comma operator (.). Also, you can have only one condition expression in the **for** loop.

For example:

```
for(x=0,y=50;x!=y;x++,y--)
{
    body of the loop;
}
```

In this example, the variable **x** is initialized to zero and **y** is initialized to fifty. In the increment/decrement part, the variable **x** is incremented by one and **y** is

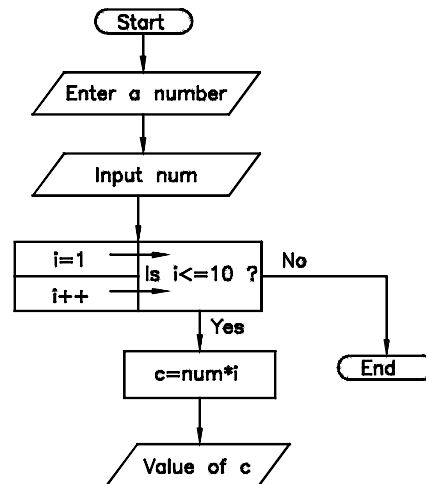


Figure 3-11 Flowchart of Example 8

decremented by one. There is only one condition expression **x!=y** (x is not equal to y). The body of the loop will be executed until the given condition becomes false.

You can also skip some or all the expressions of the **for** loop. The syntax is as follows:

```
for( ; ; )
```

If you skip the condition expression, it is assumed to be true. Now, the loop works similar to the infinite loop.

For example:

```
for(i=1; ;i++)
{
    body of the loop;
}
```

In the above example, the variable **i** is initialized to one and the condition part is skipped. So, the condition will always be assumed as true and the body of the loop will be executed indefinitely. The value of the variable **i** is incremented by one after every execution of the loop body.

Nested for Loop

A nested **for** loop is a loop within the body of another loop. In the nested **for** loop, the outer loop takes control over the inner loop. The syntax is as follows:

```
for(i=0;i<5;i++) //Outer loop
{
    for(j=0;j<5;j++) //Inner loop
    {
        body of the loop;
    }
}
```

In the above syntax, when the execution begins, the compiler first encounters the outer loop. If the condition in the outer loop is true, the control will be transferred to the inner loop. When the execution of the inner loop is complete, the control will be transferred back to the increment/decrement part of the outer loop. This process is repeated until the outer loop finishes or the condition in the outer loop becomes false.

The following example illustrates the use of the nested **for** loop.

Example 9

Write a program to display a right-angled triangle made of star (*) characters.

The following program will display the right-angled triangle of the star characters on the screen.

```
//Write a program to display a right angled triangle of stars (*)      1
#include<iostream>                                                    2
using namespace std;                                                  3
int main()                                                            4
{                                                                      5
    int i,j;                                                          6
    for(i=0;i<5;i++)//outer loop                                     7
    {                                                                  8
        for(j=0;j<=i;j++)//inner loop                               9
        {                                                            10
            cout<<" * ";                                           11
        }//End of inner loop                                         12
        cout<<endl;                                                13
    }//End of outer loop                                             14
    return 0;                                                        15
}                                                                      16
```

Explanation

Line 7

for(i=0;i<5;i++)

From this line, the outer **for** loop begins. The variable **i** is initialized to zero. Next, the condition **i<5** is checked whether it is true. If the condition is true, the control will be transferred to the inner loop (line 9). After the completion of the inner loop, the control will be transferred back to the increment/decrement part of the outer loop. If the condition is false, the body of the outer loop will be skipped and the control will be directly transferred to line 15.

Line 8

{

This line indicates the start of the body of the outer **for** loop.

Line 9

for(j=0;j<=i;j++)

From this line, the inner **for** loop begins. This loop is controlled by the outer loop. If the condition in the outer loop is true, the control is transferred to this loop. Otherwise, this loop will be skipped. If the condition in the inner loop is true, the body of the loop will be executed and the control will be directly transferred to line 11. Otherwise, the body of the inner loop will be skipped and the control will be transferred back to the increment/decrement part of the outer loop.

Line 10

{

This line indicates the start of the body of the inner **for** loop.

Line 11

```
cout<<" * ";
```

This line will display the following **star character** on the screen:

```
*
```

Line 12

```
}
```

This line indicates the end of the body of the inner loop.

Line 13

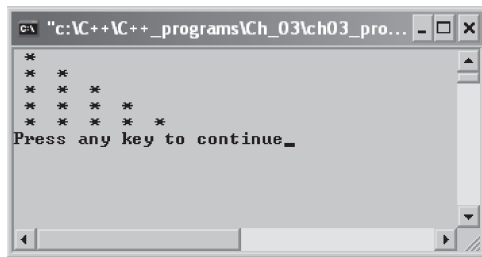
```
cout<<endl;
```

This line is used to display the next statement from the new line.

The output of the program is as follows:

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

The output will be displayed on the screen as follows:



The flowchart in Figure 3-12 gives a diagrammatic representation of the program described in Example 9.

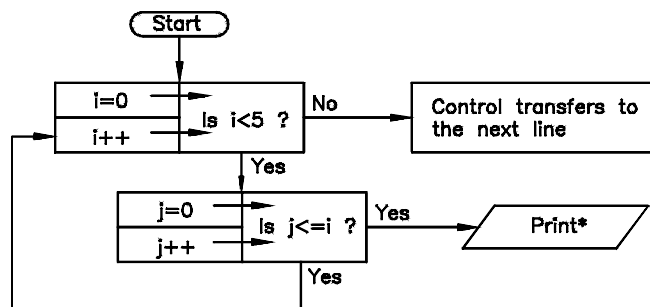


Figure 3-12 Flowchart of Example 9

The do-while Loop

The **do-while** loop is similar to the **while** loop, except that in the **do-while** loop, the body of the loop is executed first and the condition is checked at the end. Therefore, the body of the loop must be executed at least once. The **while** statement should be terminated with a semicolon (;). The syntax is as follows:

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

In the above syntax, the body of the loop is executed first and then the condition is checked. If the condition is true, the body of the loop will be executed again. This process is repeated until the condition becomes false.

The following example illustrates the use of the **do-while** loop.

Example 10

Write a program to illustrate the working of the **do-while** loop.

The following program will prompt the user to enter a number, compare it with a particular condition, and display the resultant value on the screen.

```
//Write a program that will prompt the user to enter a number
//and then repeat the block of code until it is not equal to zero.
#include<iostream>
using namespace std;
int main()
{
    int n;
    do
    {
        cout<<"Enter a value"<<endl;
        cin>>n;
        cout<<"The value entered by you is: "<<n<<endl;
    }
    while(n!=0);
    cout<<"Exit from the loop"<<endl;
    return 0;
}
```

Explanation

Line 6

int n;

In this line, **n** is declared as an integer type variable.

Line 7

do

This line is the start point of the **do-while** loop. The statements associated with the **do** statement will be executed first and then the condition within the **while** expression will be checked.

Line 8

{

This line indicates the start of the **do-while** body.

Line 9

cout<<"Enter a value"<<endl;

This line will display the following on the screen.

Enter a value

Line 10

cin>>n;

This line is used to accept the value of a variable **n** from the user.

Line 11

cout<<"The value entered by you is: "<<n<<endl;

This line will display the following on the screen.

The value entered by you is:

This is followed by the value of variable **n**

Line 12

}

This line indicates the end of the **do-while** body.

Line 13

while(n!=0);

The statements associated with the **do** loop must be executed at least once and then the condition given within the **while** statement will be checked. If the condition (**n** is not equal to zero) is true, the body of the loop will be executed again. Otherwise, the control will be transferred to the next statement following the **while** statement.

The output of the program is as follows:

Enter a value

3

The value entered by you is: 3

Enter a value

0

The value entered by you is: 0

Exit from the loop

The flowchart in Figure 3-13 gives a diagrammatic representation of the program described in Example 10.

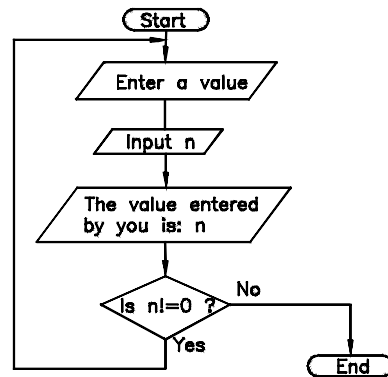


Figure 3-13 Flowchart of Example 10

Jump Statements

The **jump statements** transfer the control unconditionally to some other part in a program. In C++, there are four types of jump statements:

1. break
2. continue
3. goto
4. return

The break Statement

The **break** statement is used to exit from any kind of loop such as **for**, **while**, and **do-while** or the **switch** statement. Alternatively, it is used to transfer the control to the next statement immediately after the loop or the **switch** statement.

The following example illustrates the use of the **break** statement.

Example 11

Write a program to illustrate the working of the **break** statement.

The following program will terminate the execution of the inner loop when the values of the control variables of the outer and the inner loop are equal. This program will also display the values of the variables **i** and **j** on the screen.

```

//Write a program that terminates the execution of the inner loop when the
//values of the control variables of the outer and the inner loop are equal.
#include<iostream>
using namespace std;

```

1
2
3

```

int main()                                4
{                                          5
    int i,j;                              6
    for(i=0;i<2;i++)//outer loop          7
    {                                      8
        for(j=2;j>=0;j--)//inner loop     9
        {                                  10
            if(i==j)//Innermost statement 11
            {                              12
                cout<<"The values of i and j are equal"<<endl; 13
                break;                     14
            }                              15
            cout<<"The value of i is: "<<i<<endl; 16
            cout<<"The value of j is: "<<j<<endl; 17
        }                                  18
        cout<<"Exit the inner loop"<<endl; 19
    }                                      20
    return 0;                             21
}                                          22

```

Explanation

Line 6

int i,j;

In this line, **i** and **j** are declared as integer type variables.

Line 7

for(i=0;i<=2;i++)

From this line, the outer loop begins. The variable **i** is initialized to zero. Next, the condition **i<=2** is checked whether it is true or false. If the condition is true, the control will be transferred to the inner loop (line 10). After the completion of the inner loop, the control will be transferred back to the increment/decrement part of the outer loop. If the condition is false, the body of the outer loop will be skipped and the control will be directly transferred to line 20.

Line 8

{

This line indicates the start of the body of the outer loop.

Line 9

for(j=2;j>=0;j--)

From this line, the inner loop begins. This loop is controlled by the outer loop. If the condition in the outer loop is true, the compiler encounters this loop. Otherwise, this loop will be skipped. If the condition in the inner loop is true, the body of the loop will be executed and the control will be directly transferred to line 12. Otherwise, the body of the inner loop will be skipped and the control will be transferred back to the increment/decrement part of the outer loop.

Line 10

```
{
```

This line indicates the start of the body of the inner loop.

Line 11

```
if(i==j)
```

In this line, the **if** statement is used to check whether the value of the variable **i** is equal to the value of the variable **j**. If the condition is true, the statements associated with the **if** statement will be executed. Otherwise, these statements will be skipped.

Line 12

```
{
```

This line indicates the start of the body of the **if** statement.

Line 13

```
cout<<"The values of i and j are equal " <<endl;
```

This line will display the following on the screen:

The values of i and j are equal

Line 14

```
break;
```

The **break** statement will transfer the control outside the innermost statement.

Line 15

```
}
```

This line indicates the end of the body of the **if** statement.

Line 16

```
cout<<"The value of i is: "<<i<<endl;
```

This line will display the following on the screen:

The value of i is:

This is followed by the value of the variable **i**

Line 17

```
cout<<"The value of j is: "<<j<<endl;
```

This line will display the following on the screen:

The value of j is:

This is followed by the value of variable **j**

Line 18

```
}
```

This line indicates the end of the body of the inner loop.

Line 19

```
cout<<"Exit the inner loop"<<endl;
```

This line will display the following on the screen:

Exit the inner loop

Line 20

}

This line indicates the end of the body of the outer loop.

The output of the program is as follows:

The value of i is: 0

The value of j is: 2

The value of i is: 0

The value of j is: 1

The values of i and j are equal

Exit from the inner loop

The value of i is: 1

The value of j is: 2

The values of i and j are equal

Exit from the inner loop

The continue Statement

The **continue** statement is similar to the **break** statement except that instead of exiting from the loop, the continue statement transfers the control back to the top of the loop for the next iteration. It will skip the remaining part of the body of the loop.

For example:

```
for(i=0;i<=10;i++)
{
    statement 1;
    if(i==5)
        continue;
    statement 2;
}
```

In the above example, when the value of the variable **i** is equal to five, the **continue** statement will be executed. The **statement 2** will be skipped and the control will be transferred back to the increment part of the **for** statement for the next iteration.

The following example illustrates the use of the **continue** statement.

Example 12

Write a program to terminate the execution of the inner loop using the **continue** statement.

The following program will terminate the execution of the inner loop when the values of the control variables of the outer and the inner loop are equal. This program will also display the values of the variables **i** and **j** on the screen.

```

//Write a program that terminates the execution of the inner loop when the
//values of the control variables of the outer and the inner loop are equal.
#include<iostream>
using namespace std;
int main()
{
    int i,j;
    for(i=0;i<=2;i++)//outer loop
    {
        for(j=2;j>=0;j--)//inner loop
        {
            if(i==j)//Innermost statement
            {
                cout<<"The values of i and j are";
                cout<<" equal"<<endl;
                continue;
            }
            cout<<" The value of i is: " << i <<endl;
            cout<<" The value of j is: " << j <<endl;
        }
        cout<<"Exit the inner loop"<<endl;
    }
    return 0;
}

```

Explanation

The working of Example 12 is the same as Example 11 except that in the former the **continue** statement is used. When the condition (**i==j**) is true, the statements associated with the **if** statement will be executed. Next, the **continue** statement will be executed, which ends the current iteration of the inner loop. Also, the control is transferred back to the decrement part of the inner loop for the next iteration.

The output of the program is as follows:

```

The value of i is: 0
The value of j is: 2
The value of i is: 0
The value of j is: 1
The values of i and j are equal
Exit from the inner loop
The value of i is: 1
The value of j is: 2
The values of i and j are equal
The value of i is: 1
The value of j is: 0
Exit from the inner loop

```

The values of i and j are equal
The value of i is: 2
The value of j is: 1
The value of i is: 2
The value of j is: 0
Exit from the inner loop

The goto Statement

The **goto** statement allows the program control to jump unconditionally to another part of the program, which is associated with the named label. A label is an identifier followed by a colon (:). The syntax is as follows:

```
goto label;
```

The following example illustrates the use of the **goto** statement.

Example 13

Write a program to illustrate the working of the **goto** statement.

The following program will terminate the execution of the loop when the value of the control variable is equal to five. It will then display a message on the screen.

```
//Write a program that terminates the loop
//when the value of the control variable is equal to five
#include<iostream>
using namespace std;
int main()
{
    int i;//Control variable
    for(i=1;i<=10;i++)
    {
        if(i==5)
            goto stop;
        cout<<"The value of i is: "<<i<<endl;
    }
    stop:
    cout<<"Label is encountered";
    return 0;
}
```

Explanation

Line 7

for(i=1;i<=10;i++)

When the compiler executes this line, the variable **i** is initialized to one and the condition **i<=10** is checked. If the condition is true, the body of the loop will be executed and the control will be transferred to line 9. Otherwise, the body of the loop will be skipped.

Line 9

if(i==5)

In this line, the value of variable **i** is checked to see if it is equal to five. If this condition is true, the control is transferred to the next line, which is a **goto** statement.

Line 10

goto stop;

In this statement, **stop** is used as a label. When this statement is executed, the control will be transferred to the statement (line 14) associated with the label.

Line 11

cout<<"The value of i is: "<<i<<endl;

This line will display the following on the screen:

The value of i is: value of the variable i

Line 14

cout<<"Label is encountered";

This line will display the following on the screen:

Label is encountered

The output of the program is as follows:

The value of i is: 1

The value of i is: 2

The value of i is: 3

The value of i is: 4

Label is encountered

The return Statement

The **return** statement stops the execution of a function and the control returns to the calling function. Now, the execution will begin from the next statement that immediately follows the call in the calling function. The syntax is as discussed next.

```
return expression;
```

In the above syntax, the expression represents the value that is returned to the calling function.

The return statement is of two types, one returns some value and the other returns **void** (nothing).

The functions whose return type is **void** does not have any return statement. For example:

```
void sum();//Calling function
{
    int c= a+b;
    cout<<c;
    return;//Optional because of the void function
}
```

In this example, the **void** specifies to the compiler that no value is returned to the calling function by the **return** statement. You can also use the **return** statement without specifying any value or expression.

The functions whose return type is not **void**, must have the **return** statement that returns some value or expression. For example:

```
int main()
{
    body of the program;
    return 0;
}
```

In the above example, the return type of the **main** function is **int** (Integer). So, the **return** statement returns some integer value to the **main** function.

LOGICAL OPERATORS

You have already learned about the **logical operators** in the earlier chapters. In this section, you will learn about the usage of **logical operators** with the help of a programming example. The syntax for the logical **AND** operator is as follows:

expression1 && expression2

In the above syntax, if both the expressions, **expression1** and **expression2**, are true the operator will return true, otherwise false.

The syntax for the logical **OR** operator is as follows:

expression1 || expression2

In the above syntax, if either of the expression1 or expression2 is true, the operator will return true, otherwise false.

The syntax for the logical **NOT** operator is as follows:

! expression

In the above syntax, if the expression is true, the operator will return false. If the expression is false, the operator will return true. This operator reverses the resultant value of the expression.

The following example illustrates the use of the logical operators.

Example 14

Write a program to find the greatest of the three numbers.

The following program will prompt the user to enter three numbers, compare them, and display the greatest number on the screen.

```
//Write a program to find the greatest among the three numbers      1
#include<iostream>                                                    2
using namespace std;                                                3
int main()                                                            4
{                                                                      5
    int a,b,c;                                                        6
    cout<<"Enter three numbers"<<endl;                               7
    cin>>a>>b>>c;                                                    8
    if( a>b && a>c)                                                  9
        cout<<"a is the greatest number"<<endl;                    10
    else if(b>c)                                                      11
        cout<<"b is the greatest number"<<endl;                    12
    else                                                              13
        cout<<"c is the greatest number"<<endl;                    14
    return 0;                                                         15
}                                                                      16
```

Explanation

Line 9

if(a>b && a>c)

In this statement, the logical **AND** operator is used. If both the conditions **a>b** and **a>c** given in the statement are true, the control will be transferred to line 10. Otherwise, line 10 will be skipped and the control will be transferred to line 11.

The output of the program is as follows:

Enter three numbers

23

34

45

c is the greatest number

Self-Evaluation Test

Answer the following questions and then compare them to the answers given at the end of this chapter:

1. A _____ is a graphical representation of the steps that constitute a program.
2. The _____ control the flow of execution in a program.
3. The _____ statement is a case control statement.
4. The _____ is used to repeat a particular block of code for a specific number of times.
5. The _____ statement is used to exit from any kind of loop or **switch** statement.

Review Questions

Answer the following questions:

1. The control structures specify the order of the execution of the statement. (T/F)
2. The **if** statement is a single path statement. (T/F)
3. In the **switch** statement, the flow of execution is controlled by a variable or an expression. (T/F)
4. In the **do-while** loop, the body of the loop must be executed atleast once. (T/F)
5. The **continue** statement transfers the control outside the loop. (T/F)

Exercises

Exercise 1

Using the **if-else** statement, write a program to find whether the number is even or odd.

Exercise 2

Write a program to find the square of the first ten natural numbers using the **for** loop.

Exercise 3

Write a program to display the day of a week according to the value entered by the user, using the **switch** statement.

Answers to Self-Evaluation Test

1. flowchart, 2. control structures, 3. switch, 4. for loop, 5. break