

# 9

## Control Statements

### 9.1 Introduction

The normal flow of execution in a high level language is sequential, i.e., each statement is executed in the order of its appearance in the program. However, depending on the requirements of a problem it might be required to alter the normal sequence of execution in a program. The statements which specify the order of execution of statements are called control flow statements. There are many control flow statements provided in C++ that will be discussed in this lesson.

### 9.2 Objectives

After going through this lesson, you would be able to:

- differentiate between a statement and compound statement
- learn about the conditional statements
- familiarize with the loop statement
- differentiate between *while* and *do-while* loop
- identify the jump statements
- explain how to exit from the program

### 9.3 Statements

Statements are the instructions given to the computer to perform any kind of action. Action may be in the form of data movement, decision making etc. Statements form

---

the smallest executable unit within a C++ program. Statements are always terminated by semicolon.

## 9.4 Compound Statement

A compound statement is a grouping of statements in which each individual statement ends with a semi-colon. The group of statements is called block. Compound statements are enclosed between the pair of braces ( { } ). The opening brace ( { ) signifies the beginning and closing brace ( } ) signifies the end of the block.

## 9.5 Null Statement

Writing only a semicolon indicates a null statement. Thus ';' is a null or empty statement. This is quite useful when the syntax of the language needs to specify a statement but the logic of the program does not need any statement. This statement is generally used in for and while looping statements.

## 9.6 Conditional Statements

Sometimes the program needs to be executed depending upon a particular condition. C++ provides the following statements for implementing the selection control structure.

- (i) 'if' statement
- (ii) 'if else' statement
- (iii) 'nested if' statement
- (iv) 'switch' statement

### 'if' statement

syntax of the 'if' statement

```
if (condition)
```

```
{
```

```
statement(s);
```

```
}
```

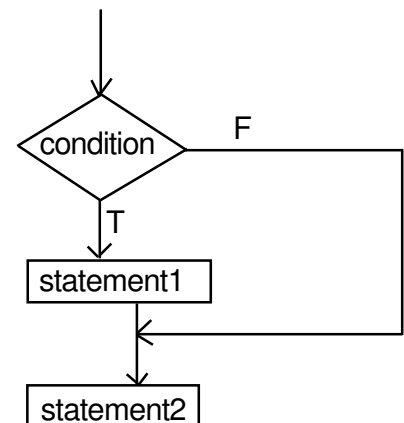


Fig. 9.1

---

From the flowchart it is clear that if the 'if condition' is true, statement 1 is executed; otherwise it is skipped. The statement 1 may either be a single or compound statement.

### Example 1

#### 'if else' statement

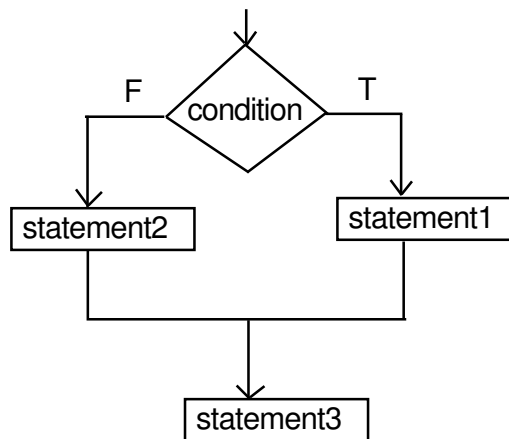
syntax of the if- else statement

if (condition)

statement1;

else

statement2;



**Fig. 9.2**

From the above flowchart it is clear that the given condition is evaluated first. If the condition is true, statement1 is executed, followed by statement3. If the condition is false, statement2 is executed, followed by statement 3. It should be kept in mind that statement1 and statement2 can be single or compound statement.

### Example 2

Write a program that will print the greatest of two numbers.

```

#include <iostream.h>
void main ()
{
int x, y;
cout << "Enter two numbers" << "\n";
cin >> x >> y ;
if ( x > y)
cout << "The bigger number is " << X;
else
cout << "The bigger number is" << y;
}
  
```

### Example 3

Write a program that will calculate the sum of odd numbers and sum of even numbers for numbers 1 to 10.

```
#include <iostream.h>
void main ()
{
int sum 1, sum2;
sum1 = sum2 = 0;
for (int i=1; i <= 10; i++)
if (i % 2 == 0)
sum 1 += i ;
else
sum2+= i ;
cout << "sum of odd number = " << sum 2 << "\n";
cout << "sum of even number = " << sum 1 << "\n";
}
```

#### **Example 4**

Write a program that will print the greatest of three numbers.

```
#include <iostream.h>
void main ()
{
int x, y, z, l;
cout << "Enter three numbers" << "\n";
cin >> x >> y >> z;
if (x > y )
l = x ;
else
l = y ;
if (l > z)
cout << "The greatest number is " << l;
else
cout << "The greatest number is " << z;
}
```

#### **'Nested if' statement**

Syntax of the nested if statement

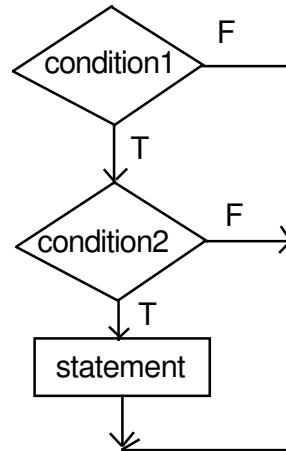
---

- (i) The if block may be nested in another if or else block. This is called nesting of if or else block.

```

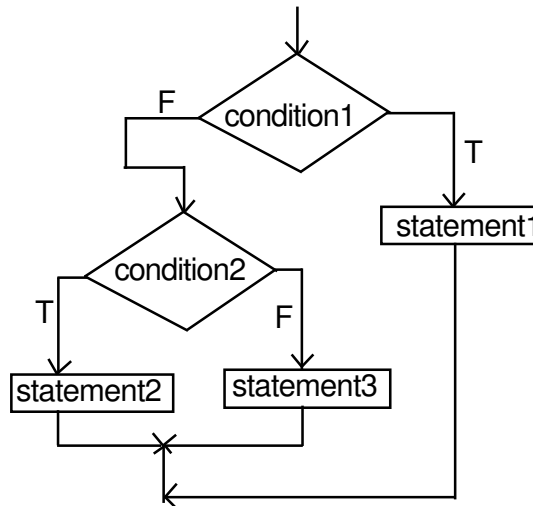
If (condition1)
{
    if (condition2)
    {
        statement(s);
    }
}

```



**Fig. 9.3**

- (ii) if (condition 1)  
statement1;  
else if (condition2)  
statement2;  
else statement 3;



**Fig. 9.4**

From the flowchart it is clear that condition 1, condition2, etc. are nested in sequence until a condition is reached that evaluates to TRUE. Only one statement will be executed. If condition1 is TRUE, then statement1 is executed. If none of the conditions is TRUE, then statement 3 is executed. Regardless of which statement is executed, control is passed to the statement following statement3. Each statement can be single or compound statement.

### **Example 5**

Write a program that will find the greatest of three numbers.

```

#include < isostream.h>
void main ( )

```

```
{
int x, y, z;
cout << "Enter three numbers" << "\n";
cin >> x >> y >> z ;
if ( x > y )
    {
    if (x >z)
        cout << "The greatest number is " << x;
    else
        cout << "The greatest number is " << z;
    }
else
    {
    if (y > z)
        cout << "The greatest number is " << y;
    else
        cout << "The greatest number is " << z;
    }
}
```

**Example 6**

Write a program using nested if condition that will enter total marks and assign grades according to the following:

Mark	Grade
$\geq 90$	A
$\geq 80$ and $< 90$	B
$\geq 70$ and $< 80$	C
$\geq 60$ and $< 70$	D
$< 60$	Fail

```
#include <iostream.h>
void main ( )
{
int marks;
```

---

```
cout << "Enter total marks" << "\n";
cin >> marks;
if (marks >= 90)
cout << "A Grade";
else
if (marks >= 80)
cout << "B Grade";
else
if (marks >= 70)
cout << "C Grade";
else
if (marks >= 60)
cout << "D Grade";
else
Cout << "Failed";
}
```

### **switch statement**

The if and if-else statements permit two way branching whereas switch statement permits multiple branching. The syntax of switch statement is:

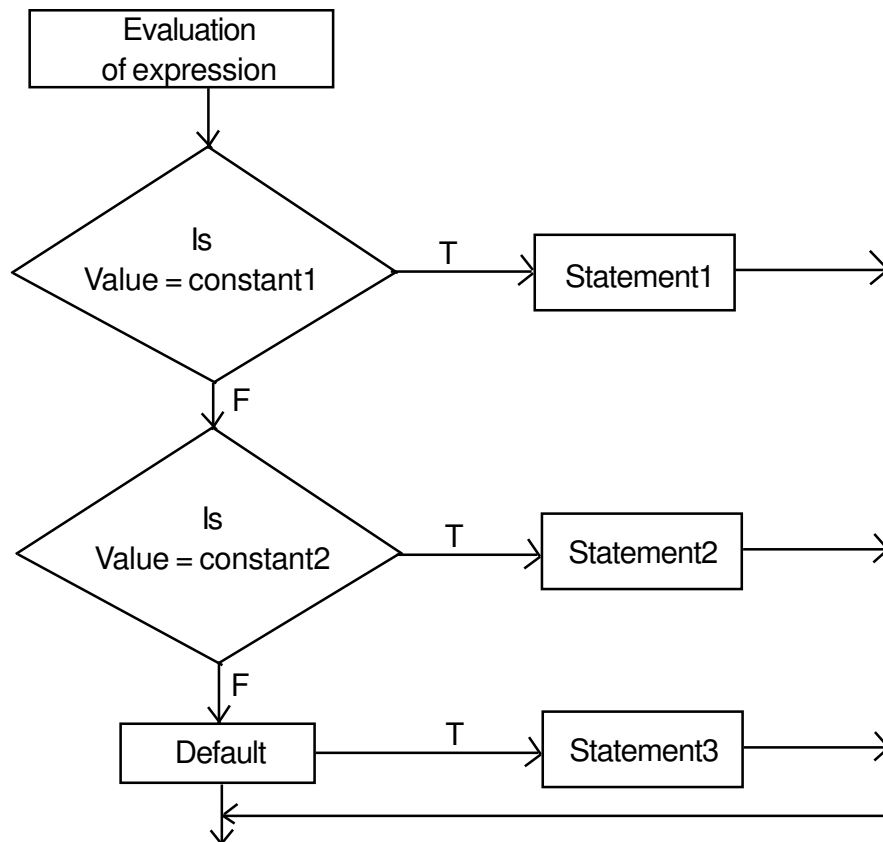
```
switch (var / expression)
{
case constant1 : statement1;
                |
                |
                break;

case constant2 : statement2;
                |
                |
                break;

default :      statement3;
}
}
```

---

The execution of switch statement begins with the evaluation of expression. If the values of expression matches with the constant then the statements following this statement execute sequentially till it executes break. The break statement transfers control to the end of the switch statement. If the value of expression does not match with any constant, the statement with default is executed.



**Fig.9.5**

**Example 7**

```
#include <iostream.h>
void main ( )
{
char ch;
cout << "Enter a character";
cin >> ch;
switch (ch)
{
```

```
case 'a'   :   cout << "vowel a";
            break ;
case 'e'   :   cout << "vowel e";
            break ;
case 'o'   :   cout << "vowel o";
            break ;
case 'i'   :   cout << "vowel i"
            break ;
case 'u'   :   cout << "vowel u";
            break;
default    :   cout << "not a vowel";
}
}
```

The program needs the break statement to confine execution to particular portion of switch. If the break statement is removed from the above program then it starts executing the statements associated with case labels specified till it comes across either the break statement or end of switch statement. The above switch statement can be written as:

```
switch (ch)
{
    case 'a'   :
    case 'A'   :   cout << "vowel a";
    case 'e'   :
    case 'E'   :   cout << "vowel e";
    case 'i'   :
    case 'I'   :   cout << "vowel i";
                break;
    case 'o'   :
    case 'O'   :   cout << "vowel o";
                break;
    default    :   cout << "not a vowel";
}
}
```

The input character either in small letter or capital will be accepted.

---

**Example 8**

Write a program that will accept a one character grade code and depending on what grade code is input, display the Basic salary according to the table given below:

Grade	Basic salary
A	15000
B	12000
C	10000
D	9000

```
#include <iostream.h>
void main ( )
{
char grade;
cout << "Enter Grade" << "\n";
cin >> grade;
switch (grade)
{
case 'A' : cout << "Basic salary is 15000";
           break;
case 'B' : cout << "Basic salary is 12000;";
           break;
case 'C' : cout << "Basic salary is 10000";
           break;
case 'D' : cout << "Basic salary is 9000";
           break;
default  : cout << "Invalid grade code";
           }
}
```

---

**In-Text Questions 9.1**

---

1. What is a compound statement?
-

2. What is a selection statement? Write down the name of selection statements provided by C++.
3. What is the significance of default clause in switch statement?
4. Identify the error(s), if any, in the following programs and write them in correct format.

a) 

```
#include <iostream.h>
void main ( )
int a, b;
cin << b; <<a;
if (a > b) Max = a
}
```

b) 

```
#include <iostream.h>
void main( )
{
int x, y;
cin >> x ;
for (y = 0 ; y < 10, y ++ )
if x == y
cout << y + x ;
else
cout >> y;
}
```

5. What will be the output of the following program segments.

a) 

```
cin >> x;
if ( x == 10 )
cout << "x is 10" ;
else
cout << " x is not 10" ;
```

if the input given is

- (i) 11
- (ii) 10

---

```
b) int year ;
    cin >> year ;
    if (year % 100 == 0)
    if (year % 400 == 0)
    cout << "Leap" ;
    else
    cout << "Not century year" ;
    if the input is
    (i) 2000
    (ii) 1971
c) int year ;
    cin >> year ;
    if (year % 100 == 0)
    {
    if (year % 400 == 0)
    cout << "Leap" ;
    }
    else
    cout << "Not century year" ;
    if the input given is
    (i) 2000
    (ii) 1971
```

6. Write equivalent switch statement for the following:

```
if (code == 'a')
cout << "season is summer" ;
else
if (code == 'r')
cout << "season is rainy" ;
else
if (code == 'w')
cout << "season is winter";
else
cout << "wrong code" ;
```

---

7. What would be the value of c?

```
{
int c;
float a, b;
a = 245.05;
b = 40.02 ;
c = a + b;
}
```

a) 285.09   b) 2850   c) 285   d) 285.0

8. What would be the value of i and k?

```
{
int i, j, k ;
j = 5 ;
i = 2 * j / 2 ;
k = 2 * (j/2) ;
}
```

a) i=4, k=4      b) i=4, k=5      c) i=5, k=4      d) i=5, k=5

9. What is the output of the following program?

```
void main ( )
{
int x = 5, y = 6, z = 7;
if (x < y + z)
cout << "Hello \n" ;
else
{
cout << "How \n" ;
cout << "Are you \n";
}
}
```

a) Hello      b) How      c) Are you      d) None of the above

---

10. Consider the program fragment.

```
switch (choice)
{
case 'W' : cout << "WHITE" ;
case 'R' : cout << "RED" ;
case 'B' : cout << "BLUE" ;
default  : cout << "error" ;
          break ;
}
```

What would be the output if choice = 'R'

- a) RED
- b) RED BLUE error
- c) RED error
- d) WHITE RED BLUE

### 9.7 Loop Construct

It is also called a Repetitive control structure. Sometimes we require a set of statements to be executed a number of times by changing the value of one or more variables each time to obtain a different result. This type of program execution is called looping. C++ provides the following constructs.

- (i) while loop
- (ii) do - while loop
- (iii) for loop

#### While loop

Syntax of while loop

```
while (condition)
{
statement(s);
}
```

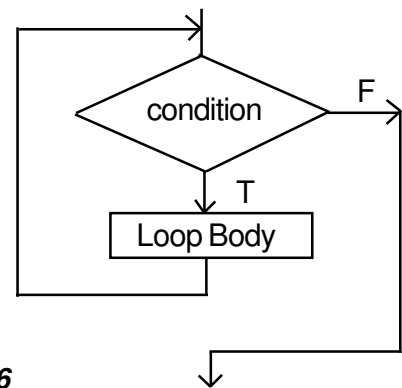


Fig. 9.6

---

The flow diagram indicates that a condition is first evaluated. If the condition is true, the loop body is executed and the condition is re-evaluated. Hence, the loop body is executed repeatedly as long as the condition remains true. As soon as the condition becomes false, it comes out of the loop and goes to the statement next to the 'while' loop. To make it more clear, we take the following example.

### **Example 9**

To find the sum of first ten natural numbers i.e. 1, 2, 3, .....10.

```
# include <iostream.h>
void main ( )
{ int n, total = 0 ;
  n = 1 ;
  while (n <= 10)
  {
    total + = n ;
    n + + ;
  }
  cout << "sum of first ten natural number" << total :
}
```

The variable n is called a loop control variable since its value is used to control loop repetition. Normally, the three operations listed below must be performed on the loop control variable.

- (i) Initialize the loop control variable
- (ii) Test the loop control variable
- (iii) Update the loop control variable

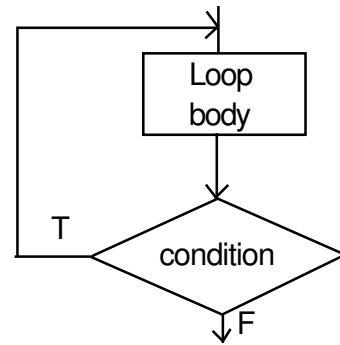
Operation (i) must be performed before the loop is entered. Operation (ii) must be performed before each execution of the loop body; depending on the result of this test, the loop will either be repeated or make an exit. Operation (iii) must be included as part of the loop body. Unless the loop control variable is updated in the loop body, its value cannot change and loop exit will never occur.

---

### do-while loop

Syntax of do-while loop

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



**Fig. 9.7**

The flow diagram indicates that after each execution of the loop body, the condition is true, the loop body is executed again. If the condition evaluates to false, loop exit occurs and the next program statement is executed.

Note : that the loop body is always executed at least once.

One important difference between the while loop and the do-while loop is the relative ordering of the conditional test and loop body execution. In the while loop, the loop repetition test is performed before each execution of the loop body; the loop body is not executed at all if the initial test fails. In the do-while loop, the loop termination test is Performed after each execution of the loop body; hence, the loop body is always executed at least once.

Let us take an example to explain it further.

#### **Example 10**

To find the sum of the first N natural number.

```
# include < iostream.h>  
void main ( )  
{  
int N, number, sum;  
cin >> N;  
sum = 0;  
number = 1;  
do  
{  
sum + = number;
```

```
    number ++ ;  
  }  
  while (number <= N) ;  
  cout << sum;  
}
```

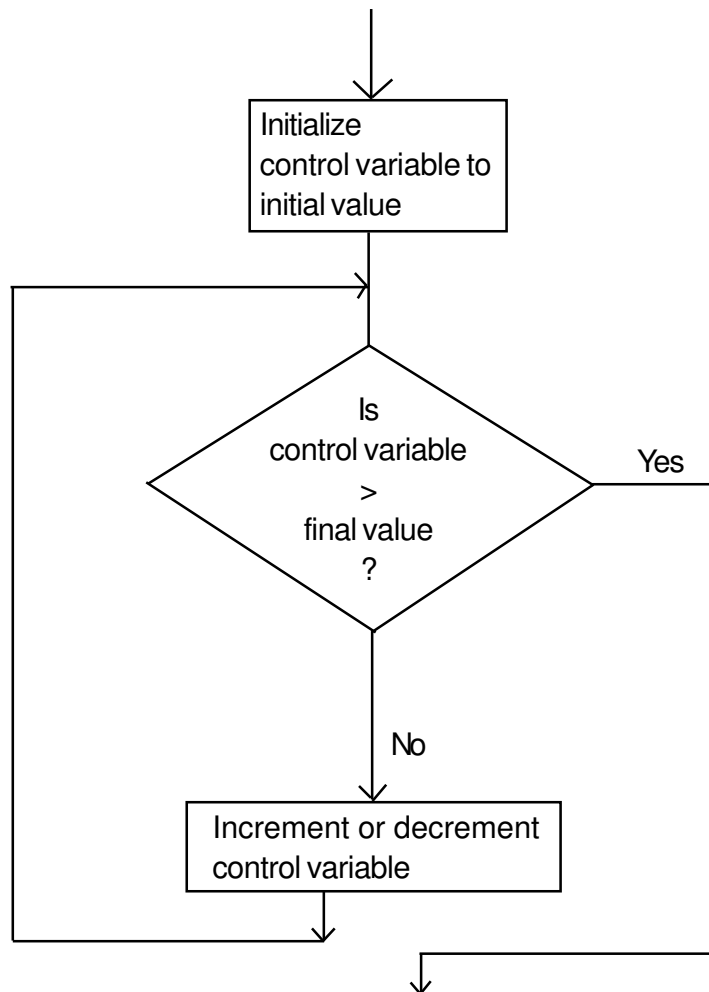
## For loop

It is a count controlled loop in the sense that the program knows in advance how many times the loop is to be executed.

### syntax of for loop

```
for (initialization; decision; increment/decrement)
```

```
{  
  statement(s);  
}
```



**Fig. 9.8**

The flow diagram indicates that in for loop three operations take place:

- (i) Initialization of loop control variable
- (ii) Testing of loop control variable
- (iii) Update the loop control variable either by incrementing or decrementing.

Operation (i) is used to initialize the value. On the other hand, operation (ii) is used to test whether the condition is true or false. If the condition is true, the program executes the body of the loop and then the value of loop control variable is updated. Again it checks the condition and so on. If the condition is true, it gets out of the loop.

**Example 11**

```
for (int i = 0 ; i < 5 ; i + + )  
    cout << i ;
```

The output of the above program is

0 1 2 3 4

```
for (char i = 'A'; i < 'E' ; i + + )  
    cout < < i ;
```

The output of the above program is

A B C D

---

**In-Text Questions 9.2**

---

- 1. What is the difference between while and do-while loops.
  - 2. Write a while loop that displays numbers 2, 4, 6, 8 ..... 12.
  - 3. Write a do-while loop that displays numbers 1, 3, 5, 7 ..... 11.
  - 4. Write a for-loop that displays numbers from 10 to 20.
  - 5. How many times is the following loop executed ?
    - a) `int s = 0, i = 0;`
-

```
while ( i + + < 5)
    s + = i;
```

```
b) int s = 0 ; i = 0;
    do
        s + = i ;
    while ( i < 5 ) ;
```

6. What will be the output of the following program ?

```
a) # include <iostream.h>
    void main ( )
    {
    long number = 7866372, result = 0 ;
    do
        {
        result * = 10 ;
        int digit = number % 10 ;
        result + = digit ;
        number / = 10 ;
        }
    while (number) ;
    cout << "output =" << result << endl;
    }
```

```
b) void main ( )
    {
    int i = 0, a = 0, b = 0, c = 0, f = 0 ;
    while ( i < = 5)
        {
        switch (i + +)
        {
        case 1 :
        case 2 : + + a ;
        case 3 :
        case 4 : + + b ;
```

```
        case 5 : + + c ;
        default : + + f ;
    }
    cout << a << "\n" ;
    cout << b << "\n" ;
    cout << c << "\n" ;
    cout << f << "\n" ;
}
```

7. What will be the value of counter after the following program is executed ?

```
void main ( )
{
    int counter ;
    int digit = 0 ;
    counter = 1 ;
    while (digit <= 10)
    {
        + + counter ;
        + + digit ;
    }
    cout << counter ;
}
```

a) 9    b) 10    c) 11    d) 12

8. What will be the value of sum after the following program is executed ?

```
void main ( )
{
    int sum, i ;
    sum = 1 ;
    i = 9 ;
    do
    {
        i = i - 1 ;
        sum = 2 * sum ;
    }
```

```
    }  
    while (i > 9) ;  
    cout << sum ;  
    }
```

- a) 1    b) 2    c) 9    d) 0.5

9. What is the following program doing ?

```
void main ( )  
{  
int digit = 0 ;  
do  
{  
cout << digit + + << "\n" ;  
}  
while (digit <= 10);  
}
```

- a) Display integers from 1 to 10  
b) Display integers from 0 to 10  
c) Adding 10 integers  
d) None of the above

10. What is the final value of digit ?

```
void main ( )  
{  
int digit ;  
for (digit = 0 ; digit <= 9 ; digit + + )  
cout << digit << "\n" ;  
digit = 5 * digit ;  
-- digit ;  
}
```

- a) 49    b) -1    c) 47    d) 46
-

## 9.8 Jump Statements

The jump statements unconditionally transfer program control within a function.

- a) goto statement
- b) break statement
- c) continue statement

### goto statement

#### syntax of goto statement

```
    goto pgr;  
    |  
    |  
    |  
pgr :
```

pgr is known as label. It is a user defined identifier. After the execution of goto statement, the control transfers to the line after label pgr.

**Note** : It is not a good programming to use goto statement in a program.

### Break statement

#### syntax of break statement

The break statement can be used in a switch statement and in any of the loops. It causes program execution to pass to the next statement following the switch or the loop.


```
    while (condition)  
    {  
    statement 1;  
    if (condition)  
    break ;  
    statement 2;  
    }  
    → statement 3;
```

The break statement skips rest of the loop and goes out of the loop.

### **continue statement**

The continue statement is used in loops and causes a program to skip the rest of the body of the loop.

```
while (condition) ←  
{  
  statement 1;  
  If (condition)  
  continue ;  
  statement 2;  
}  
statement 3;
```



The continue statement skips rest of the loop body and starts a new iteration.

## **9.9 exit ( ) function**

The execution of a program can be stopped at any point with exit ( ) and a status code can be informed to the calling program. The general format is

```
exit (code) ;
```

where code is an integer value. The code has a value 0 for correct execution. The value of the code varies depending upon the operating system. It requires a process.h header file.

---

### **In-Text Questions 9.3**

---

1. The goto statement causes control to go to
    - (i) an operation
    - (ii) a label
    - (iii) a variable
    - (iv) a function
  
  2. The break statement causes an exit
    - i) only from the innermost loop
-

- ii) only from the innermost switch
  - iii) from all loops and switches
  - iv) from the innermost loop or switch
3. A continue statement within a loop causes control to go to \_\_\_\_\_.
4. Which header file must be included in the program for using the exit ( ) function.
5. Rewrite the following program segment using goto statement i = 100 ;
- ```
while (i)
cout << i -- ;
cout << "\n Thank you" ;
```
6. Rewrite the following program segment using a while loop.
- ```
i = 2 ;
start :
cout << i ;
i + = 2 ;
if ( i < 51) goto start ;
cout << "\n Thank you" ;
```
7. What will be the output of following program ?
- ```
#include < iostream.h>
void main ( )
int x = 10, count = 0 ;
while (x)
{
x - - ;
if (x == 4)
continue ;
count + + ;
}
cout << "\n" <<count ;
}
```
-

```
8. #include <iostream.h>
void main ( )
{
char ch = 'A' ;
while (ch <= 'F' )
{
switch (ch)
{
case 'A' :
case 'B' :
case 'C' :
case 'D' : ch ++ ; continue ;
case 'E' :
case 'F' : ch ++ ;
}
putchar (ch) ;
}
}
```

What will be the output of the above program ?

- a) ABCDEF    b) FG    c) EFG    d) None of the above
- 

## 9.10 What you have learnt

In this lesson you learnt the statements which control the execution of a program. Now you should be in a position to write small programs in which flow control statements are used. You have learnt the meaning of opening braces '{' and closing braces '}'.

---

## 9.11 Terminal Questions

---

1. Write a program to display the following menu and accept a choice number. If an invalid choice is entered, appropriate error message must be displayed. Otherwise the choice number entered must be displayed.

MENU

---

1. Create a data file
2. Display a data file
3. Edit a data file
4. Exit

Choice :

2. Write a program that will accept a mark and assign letter grade according to the following table.

| Grade | Mark                 |
|-------|----------------------|
| A     | $\geq 90$            |
| B     | $\geq 80$ but $< 90$ |
| C     | $\geq 70$ but $< 80$ |
| D     | $\geq 60$ but $< 70$ |
| F     | $< 60$               |

3. Write a program that will print sum of N integers entered by the user.
4. Write a program to generate the members of fibonacci series upto 500.
5. Write a program to reverse a given number.
6. Write a program that will print the table of any number upto any number.

input :

Enter the table of which number : 4

Enter upto which number : 5

Output

$4 \times 1 = 4$

$4 \times 2 = 8$

$4 \times 3 = 12$

$4 \times 4 = 16$

$4 \times 5 = 20$

Continue (Y/N) ?

7. Write a program that will print the factorial of any number.

input : 5

Output :  $5 \times 4 \times 3 \times 2 \times 1 = 120$

---

8. Write a program to print all the tables between 2 and 20 upto 10.
9. Write a program that will find pythagorean triplets between 1 to 100. (Pythagorean triplets are such that the square of one number is equal to the sum of the squares of the other two numbers).

Example : 3, 4, 5

$$5^2 = 3^2 + 4^2$$

$$25 = 9 + 16 = 25$$

10. Determine the roots of the quadratic equation

$$ax^2 + bx + c = 0$$

using the formula  $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

11. Write a program to find out the sum of the series

$$1 + x + x^2 + x^3 + \dots + x^n$$

12. Write a program that will take a number N as input and print the factorial of all numbers from 1 to N as follows :

| Number | Factorial |
|--------|-----------|
| 1      | 1         |
| 2      | 2         |
| 3      | 6         |
|        |           |
|        |           |
|        |           |
| N      | N!        |

13. Write a program that will find out whether the given number is even or odd. If odd number then whether it is prime or not.
14. Write a program that will ask the user to enter N integers. Find out the following:
- Total number of positive integers
  - Total number of negative integers
  - Total number of integers equal zero

## 9.12 Feedback to In-Text Questions

---

### In-text Questions 9.1

1. Compound statement - Group of statements enclosed by a pair of braces ( { } ).
2. Selection statement - statement that allows us to choose a set of instructions for execution, depending upon the expression. Examples If, If-else switch.
3. If none of the constant present in case labels is equal to the value of the variable or expression in switch statement, default label is executed.

4. a) 

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

```
{
    int a, b, max;
    cin >> b >> a;
    if (a > b) Max = a;
}
```

- b) 

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

```
{
    int x, y;
    cin >> x;
    for (y = 0 ; y < 10 ; y + + )
        if (x == y)
            cout << y + x;
        else
            cout << y;
}
```

5. a) (i) x is not 10  
(ii) x is 10  
b) (i) Leap  
(ii) No output
-

- c) (i) Leap
- (ii) Not century year

## 6. Switch (code)

```
{  
case 'a' : cout << "season is summer" ; break;  
case 'r' : cout << "season is rainy"; break;  
case 'w' : cout << "season is winter"; break ;  
default : cout << "wrong code";
```

- 7. c)
- 8. c)
- 9. Hello
- 10. b)

**In-text Questions 9.2**

1. The while loop checks the condition first and then executes the loop whereas do-while loop executes the loop first and then checks the condition so it is executed at least once even if the condition is false.
  2. 

```
int i = 2 ;  
while (i <= 12)  
{  
cout << i ;  
i += 2 ;  
}
```
  3. 

```
int i = 1 ;  
do  
{  
cout << i ;  
i += 2 ;  
} while (i <= 11);
```
  4. 

```
for (int i = 10; i <= 20 ; i ++ )  
cout << i ;
```
-

5. a) 5 times  
b) endless
6. a) output = 2 7 3 6 6 8 7  
b) 2  
4  
5  
6
7. d)
8. b)
9. b)
10. a)

### In-text Questions 9.3

1. (ii) 2. (iv)
  3. the decision statement of loop
  4. process.h
  5. 

```
i = 100 ;  
xyz : i -- ;  
if i > 0 goto xyz ;  
cout << "\n Thank you" ;
```
  6. 

```
i = 2 ;  
while ( i < 51)  
{  
cout << i ;  
i + = 2, ;  
}  
out << "\n Thank you" ;
```
  7. 9
  8. (b)
-