Notes

14

CONTROL STATEMENTS

In the previous lesson you have learnt about general concepts of OOP. Now you will learn about control statements which help you to control movements of data and also help you to make decisions based on the conditions. 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.

OBJECTIVES

After reading this lesson, you will be able to:

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

14.1 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, which are always terminated by semicolon.

14.1.1 Compound Statement

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

14.1.2 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.

14.1.3 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) 'if-else if.. else

(iv) 'nested if' statement

(v) 'switch' statement

(i) 'if' statement

syntax of the 'if' statement

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

From the Fig 14.1 it is clear that if the 'if condition' is true, statement 1 is executed, otherwise it is skipped and directly statement 2 will be executed. The statement 1 may either be a single or compound statement.

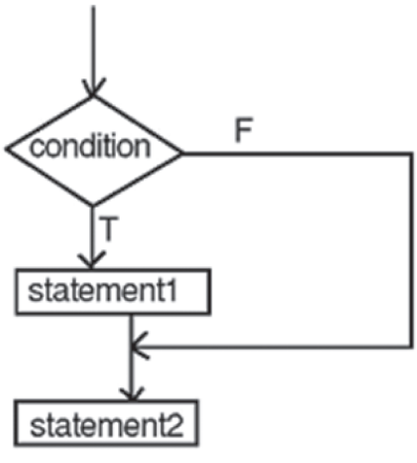


Fig. 14.1: if statement

(ii) 'if else' statement

syntax of the if- else statement

```
if (condition)
{
statement1;
}
else
{
statement 2;
}
statement 3;
```

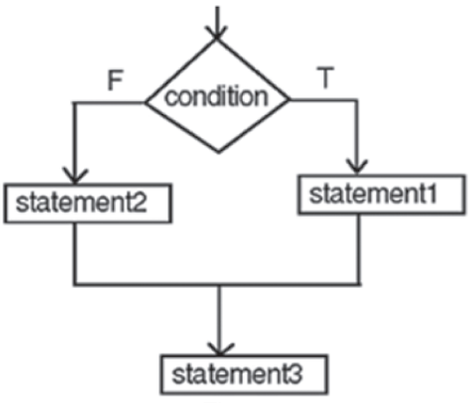


Fig. 14.2: if – else statement

From the Fig 14.2, it is clear that the given condition is evaluated first. If the condition is true, statement1 is executed, followed by statement 3. 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 1

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)                          // if condition
cout << "The bigger number is " << x;   // if true this statement will execute
else                                 // if false this statement will execute
cout << "The bigger number is" <<y;
}
```

The output of the above program is:

```
Enter two numbers
10
15
The bigger number is 15
```

Example 2

Write a program that will print the greatest of three numbers (use multiple if – else)

```
# include <iostream.h>
void main ( )
{
int x, y, z, l;
cout << "Enter three numbers" <<"\n";
cin >> x >> y >> z;
if (x > y )                          // 1st if condition
l = x ;
else                                 // 1st else condition
l = y ;
if (l > z)                           //2nd if condition
cout << "The greatest number is" << l;
else
cout << "The greatest number is" <<z;   //2nd else condition
}
```

The output of the above program is

```
Enter three numbers
10
15
20
The greatest number is 20
```

iii) 'Nested if' statement

Syntax of the nested if statement

The if or else if statement can be used inside another if or else if statement(s).

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

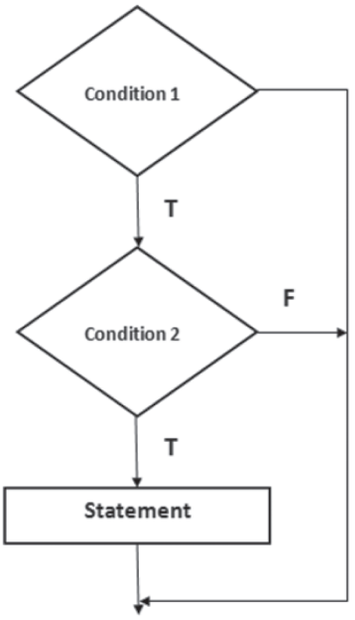


Fig. 14.3: Nested if Statement

(iv) if – else if – else

Syntax of if else if else

```
if (condition 1)
statement1;
else if (condition 2)
statement2;
else statement 3;
```

In a program, the statement x = 5; assigns the value 5 to the variable x. If you write a statement if (x = = 5) - here it checks whether the value of x is equal to 5 or not.

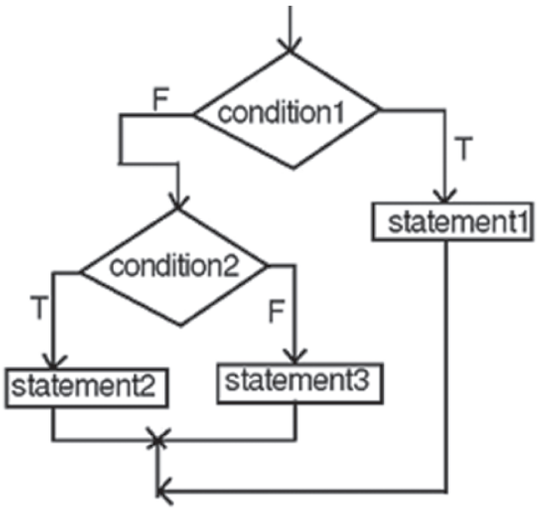
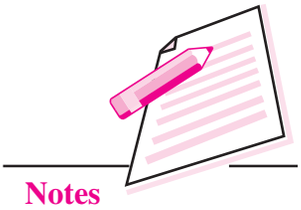


Fig. 14.4: if else if else statement



Notes

If condition1 is TRUE, then statement1 is executed. Else condition 2 is checked, if condition 2 is true then statement 2 is executed. If both the condition fails then statement 3 will be executed. For any given condition only one of the statements (Statement1, Statement2, or Statement3) will be executed. Regardless of which statement is executed, control is passed to the statement following statement3. Each statement can be single or compound statement.

Example 3

Write a program that will find the greatest of three numbers (use nested if - else).

```
# include < iostream.h>
void main ( )
{
int x, y, z;
cout << "Enter three numbers" << "\n";
cin >> x >> y >> z ;
if ( x > y )                                    //1st if
     {
     if (x >z)                                  //2nd if nested in 1st else
     {cout << "The greatest number is" << x; }
     else                                       //2nd else
     cout << "The greatest number is" <<z;
     }
else                                            //1st else
     {
     if (y > z)                                 // 3rd if nested in else
     {cout << "The greatest number is" << y; }
     else                                       // 3rd else
     {cout "The greatest number is" << z; }
     }
}
```

Example 4

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

Mark	Grade
> = 90	A
> = 80 and < 90	B
> = 70 and < 80	C
> = 60 and < 70	D
< 60	Fail

```
# include <iostream.h>
void main ( )
{
int marks;
cout << "Enter total marks" << "\n";
cin >> marks;
if (marks > = 90)                          //1st if
cout << "A Grade";
else                                       // 1st else
if (marks > = 80)                          //2nd if nested in 1st else
cout << "B Grade";
else                                       //2nd else
if (marks > = 70)                          //3rd if nested in 2nd else
cout << "C Grade";
else                                       // 3rd else
if (marks > = 60)                          //4th if nested in 3rd else
cout << "D Grade";
else                                       //4th else
cout << "Fail";
}
```

v. 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 value of the expression matches with the constant then the statements following the expression will be executed sequentially till the break statement. 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 will be executed.

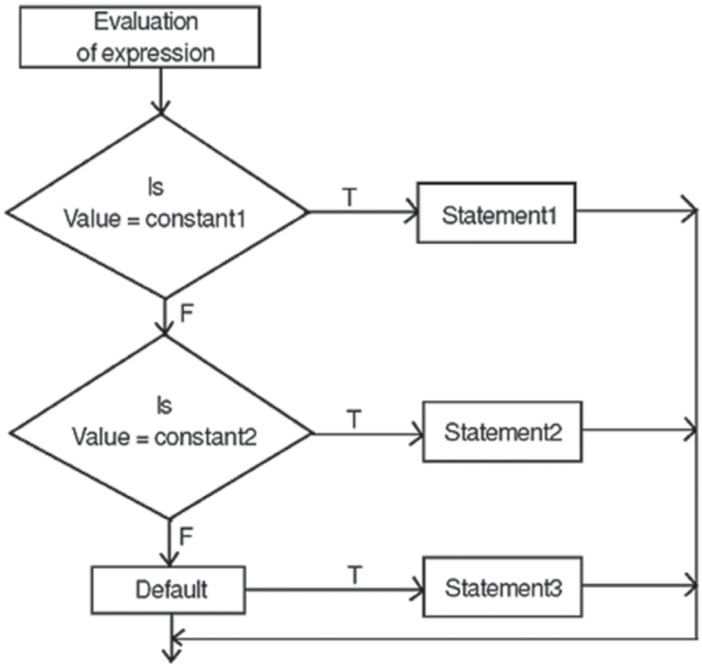
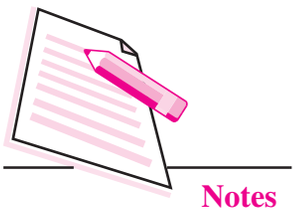


Fig. 14.5: Switch statement



Notes

Example 5

Write a program to show the use of switch statement

```
# include < iostream.h>
void main ( )
{
char ch;
cout << "Enter a character";
cin >> ch;
switch (ch)                                    //switch statement
{
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 a 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";
break;
case 'e'    :
case 'E'    :    cout << "vowel e";
break;
```

```
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 6

Write a program that will accept a one character grade code and depending on the grade code 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)                                  // switch statement
        {
        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";
        }
}
```

INTEXT QUESTIONS 14.1

1 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;
}
```

2. What will be the output of the following program segment?

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

3. 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

4 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

5. 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

6. Consider the program segment.

```
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

14.2 LOOP CONSTRUCT

It is also called a repetitive / iterative 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

(i) while loop

Syntax of while loop

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

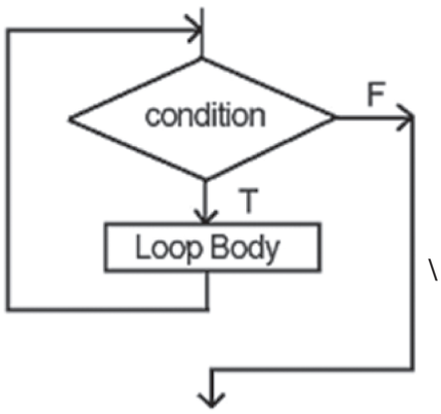
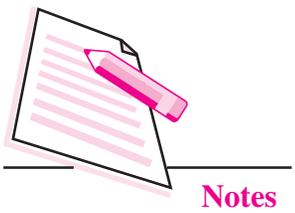


Fig. 14.6: while loop

The Fig 14.6 indicates that a condition is first evaluated. If the condition is true, then the loop body will be executed and the condition will be 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.



Notes

Example 7

Write a program 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)                                    // while condition
{
total + = n ;
n + + ;
}
 cout << "sum of first ten natural number is" << 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 will not change and loop exit will never occur.

(ii) do-while loop

Syntax of do-while loop

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

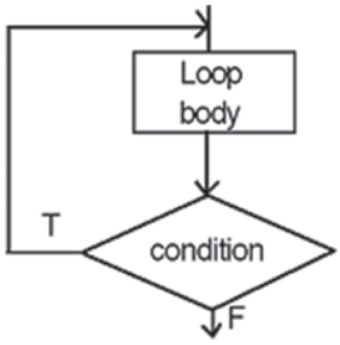


Fig. 14.7: do – while loop

The Fig 14.7 indicates that after each execution of the loop body, the condition will be checked. If the condition is true, then the loop body will be executed again. If the condition evaluates to false, loop exit occurs and the next program statement is executed.

Note : The loop body is always executed at least once in the do-while loop.

One important difference between while loop and 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.

Example 8

Write a program 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;
}
```

(iii) For loop

It is a count controlled loop in the sense that the program knows in advance how many times the loop has to be executed. Repeats a **statement** or group of statements while a given condition is true.

syntax of for loop

for (intialization; decision; increment/decrement)

{ statement(s);

}

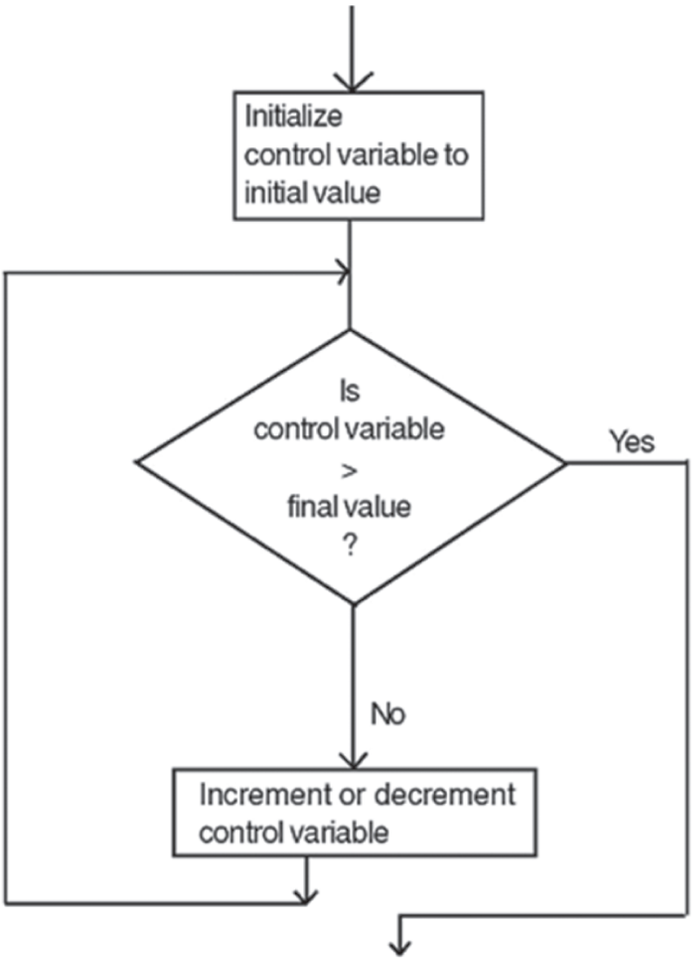


Fig. 14.8: for loop

The Fig 14.8 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 false, it gets out of the loop.

Use of "for" Statement

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

The output of the above program is


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

The output of the above program is

INTEXT QUESTIONS 14.2

1. How many times will the following loop be executed ?

   a)
   ```
   int s = 0, i = 0;
   while ( i + + < 5)
   s + = i;
   ```

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

2. 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 ;      break;
case 3 :
case 4 : + + b ;      break;
case 5 : + + c ;      break;
default : + + f ;     break;
}
}
cout << a << "\n" ;
cout << b << "\n" ;
cout << c << "\n" ;
cout << f << "\n" ; }
```

3. 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

4. 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

5. What will the following program do ?

```
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

6. What is the final value of digit variable?

```
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

14.3 JUMP STATEMENTS

The jump statements unconditionally transfer program control to another statement elsewhere in program code. You can use the following statements in a program to (Jump statements) to transfer program control from one statement to another statement.

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

(a) goto statement

Goto statement can be used in the program where you want to transfer your program control to another statement of the program. Let us see an example here.

syntax of goto statement

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

When program control reaches the statement goto pgr, it will transfer the control to the statement after label pgr. Now the program will start to exceute the statements mentioned after 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.

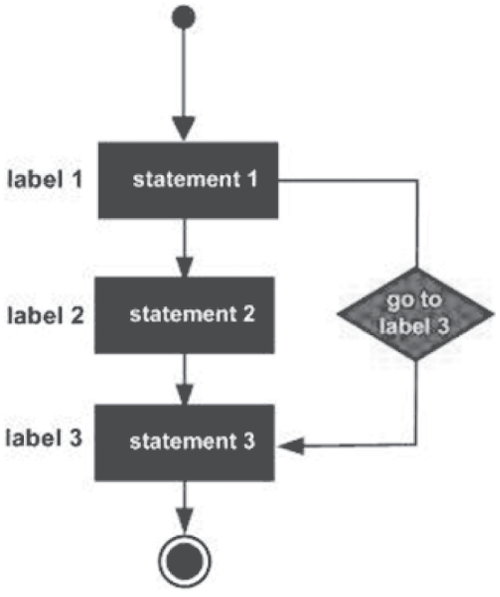


Fig. 14.9: goto statement

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

Example 9

```
#include <iostream.h>
int main( )
{
      int num = 1;
      STEP:do
      {
            if( num == 5)
      {
            num = num + 1;
            goto STEP;
      }
      cout << "value of num : " << num << endl;
      num = num + 1;
      }while( num < 10 );
      return 0;
}
```

(b) 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.

Syntax of break statement.

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

The difference between the break and continue statement is that break statement comes out of the loop and executes the next statement after the loop. Contine statement skips rest of the loop and starts new iteration of the same loop.

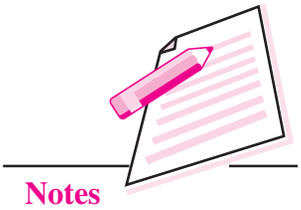
The break statement skips rest of the loop and goes out of the loop.

(c) 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.



Notes

14.4 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.

INTEXT QUESTIONS 14.3

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. Name the header file to be included in the program for using the exit ( ) function.
5. 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 ;
}
```

6 What will be the output of the above program ?

```
# include < iostream.h>
# include < stdio.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) ;
}
}
```

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

WHAT YOU HAVE LEARNT

- Statements are the instructions given to the computer to perform any kind of action.
- Compound statement is a grouping of statements in which each individual statement ends with a semicolon.
- 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.
- 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.
- Continue statement is used in loops and causes a program to skip the rest of the body of the loop.
- The execution of the program can be stopped at any point with exit() statement.

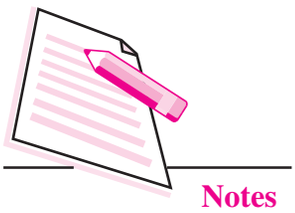
TERMINAL EXERCISE

1. Write a program to display the following menu and accept a choice number. If an invalid choice is entered, then 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 :



Notes

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

Grade	Mark
A	> = 90
B	> = 80 but < 90
C	> = 70 but < 80
D	> = 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) ?

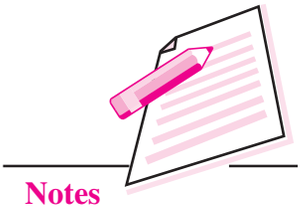
7. Write a program that will print the factorial of any number.

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. Write a program to find out the sum of the series $1 + x + x^2 + x^2 \ldots\ldots\ldots x^n$



Notes

11. 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!

12. Write a program that will find out whether the given number is even or odd. If odd number then find out whether it is prime or not.

13. Write a program that will ask the user to enter N integers. Find out the following:

    (i) Total number of positive integers

    (ii) Total number of negative integers

    (iii) Total number of integers equal zero

ANSWERS TO INTEXT QUESTIONS

14.1

1 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;
}
```

2. a) (i) x is not 10
   (ii) x is 10
   b) (i) Leap
   (ii) No output
   c) (i) Leap
   (ii) Not century year
3. c) 285
4. c) i = 5, k = 4
5. Hello
6. b) RED error

14.2

1. a) 5 times b) endless
2. a) output = 2 7 3 6 6 8 7 b) 2 2 1 1
3. d) 12
4. b) 2
5. b) Display integers from 0 to 10
6. a) 49

14.3

1. (ii) a label
2. (iv) from the innermost loop or switch
3. the decision statement of loop
4. process.h
5. 9
6. (b) FG