CHAPTER 5

CONDITIONAL STRUCTURES

Chapter Overview

5.1 Control Structure

A statement used to control the flow of execution in a program is called **control structure**. The instructions in a program can be organized in three kinds of control structures to control execution flow. The control structures are used to implement the program logic.

5.1.1 Types of Control Structures

Different types of control structures are as follows:

1. Sequence

In **sequential structure**, the statements are executed in the same order in which they are specified in program. The control flows from one statement to other in a logical sequence. All statements are executed exactly once. It means that no statement is skipped and no statement is executed more than once

Example

Suppose a program inputs two numbers and displays average on screen. The program uses two statements to input numbers, one statement to calculate average and one statement to display average number. These statements are executed in a sequence to find the average number. All statements are executed once when the program is executed. It is an example of sequence control structure.

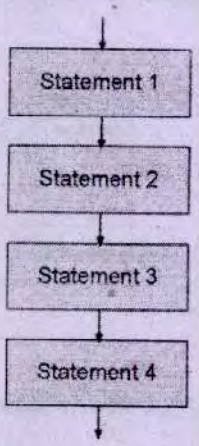


Figure 5.1: Execution of Sequential Statements

2. Selection

A **selection structure** selects a statement or set of statements to execute on the basis of a condition. In this structure, statement or set of statements is executed when a particular condition is **true** and ignored when the condition is **false.** There are different types of selection structures in C++. These are **if, if...else,** and **switch.**

Example

Suppose a program inputs the marks of a student and displays a message on screen whether the student is pass or fail. It displays **Pass** if the student gets 40 or more than 40 marks. It displays **Fail** when the marks are below 40. The program checks the marks before displaying the message. It is an example of selection structure.

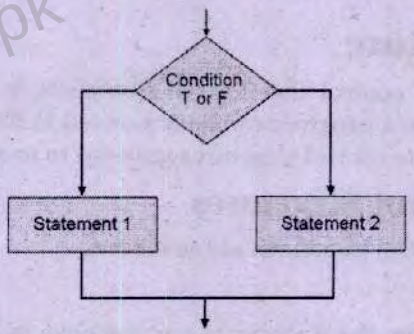


Figure 5.2: Execution of Conditional Statements

3. Repetition

A **repetition structure** executes a statement or set of statements repeatedly. It is also known as **iteration structure** or **loop**. There are different types of repetition structures in C++. These are **while, do while** and **for.**

Example

Suppose we want to display a message "I love Pakistan" on screen for 100 times. It is time consuming to perform this task using control sequence structure. However, it is very easy to perform this task using repetition structure. A single loop statement can display the message for 100 times.

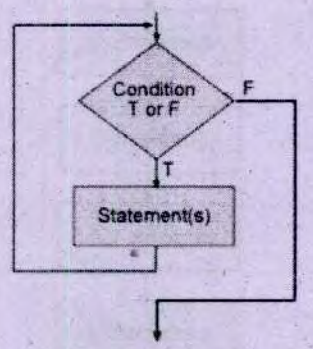


Figure 5.3: Execution of Iterative Statements

4. Function call

Function call is a type of statement that moves the control to another block of code. The control returns back after executing all statements in the block. The remaining statements are executed immediately after the function call when the control is returned.

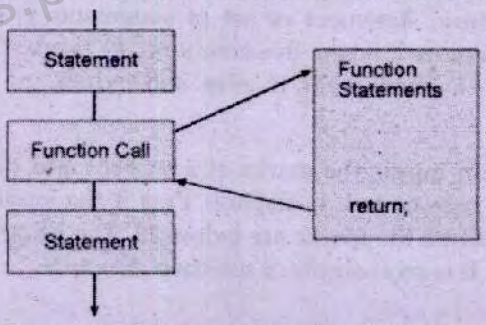


Figure 5.4: Execution of Function Call Statements

5.2 Relational Operators

The relational operators are used to specify conditions in programs. A relational operator compares two values. It produces result as **true** or **false**. The relational operators are sometimes called the **conditional operators** or **comparison operators** as they test conditions that are either **true** or **false**. C++ provides the following relational operators:

Operator	Description
>	**Greater than operator** returns **true** if the value on left side of > is greater than the value on the right side. Otherwise returns **false**.
<	**Less than operator** returns **true** if the value on left side of < is less than the value on right side. Otherwise returns **false**.
= =	**Equal to operator** returns **true** if the values on both sides of == are equal. Otherwise returns **false**.
>=	**Greater than or equal to operator** returns **true** if value on left side of >= is greater than or equal to the value on right side. Otherwise returns **false**.
<=	**Less than or equal to operator** returns **true** if the value on left side of <= is less than or equal to the value on right side. Otherwise returns **false**.
!=	The not equal to operator. Returns **true** if the value on the left side of <> is not equal to the value on the right. Otherwise returns **false**.

Table 5.1: Relational Operators

5.2.1 Relational Expression

A **relational expression** is a statement that uses relational operators to compare two values. The result of a relational expression can be **true** or **false**. Both sides of a relational expression can be a **constant, variable** or **arithmetic expression.**

Examples

Some examples of different relational expressions are as follows:

Relation Expression	Result
100 > 15	True
25 < 5	False
30 <= 12	False
40 >= 20	True
!('A' > 'B')	True
0 >= 0	True
0 <= 0	True
1 != 2	True
5 != 5	False

Table 5.2: Relational Expressions

5.3 'if' Statement

if is a keyword in C++ language. **if** statement is a decision-making statement. It is the simplest form of selection constructs. It is used to execute or skip a statement or set of statements by checking a **condition.**

The condition is given as a relational expression. If the condition is **true**, the statement or set of statements after **if statement** is executed. If the condition is **false**, the statement or set of statements after **if statement** is not executed.

Syntax

The syntax of **if** statement is as follows:

```
if (condition)
  statement;
```

The above syntax is used for single statement. A set of statements can also be made conditional. In this case, these statements are written in curly brackets { }. The set of statements is also called **compound statements**.

The syntax for compound statements in **if statement** is as follows:

```
if (condition)
{
  statement 1;
  statement 2;
  :
  statement N;
}
```

Flowchart

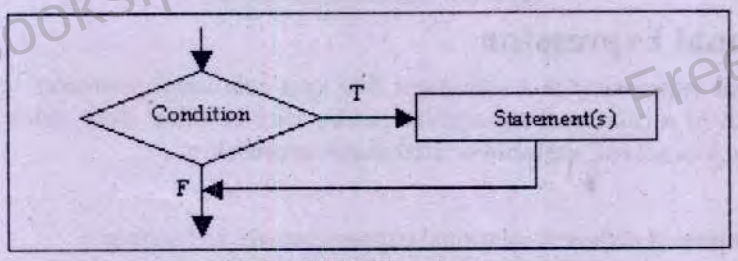


Figure 5.5: If...Else structure

5.3.1 Limitation of simple 'if' Statement

if statement is the simplest selection structure but it is very limited in its use. The statement or set of statements is executed if the condition is **true**. But if the condition is **false** then nothing happens. A user may want to:

- Execute one statement or set of statements if the condition is **true**.
- Execute other statement or set of statements if the condition is **false**.

In this situation, simple **if** statement cannot be used effectively.

Example

For example, a program should display '**Pass**' if the student gets 40 or more marks. It should display '**Fail**' if the student gets less then 40 marks. Simple if statement cannot be used to handle this situation.

Program 5.1

Write a program that inputs marks and displays "Congratulations! You have passed." if the marks are 40 or more.

```
#include <iostream.h>
#include <conio.h>
```

```
void main()
{
  int marks;
  clrscr();
  cout<<"Enter your marks: ";
  cin>> marks;
  if(marks >= 40)
    cout<<"Congratulations! You have passed.";
  getch();
}
```

Output:
Enter your marks: 50
Congratulation! You have passed.

Program 5.2

Write a program that inputs two numbers and finds whether both are equal.

```
#include <iostream.h>
#include <conio.h>
void main()
{
  int a, b;
  clrscr();
  cout<<"Enter a number: ";
  cin>>a;
  cout<<"Enter a number: ";
  cin>>b;
  if(a == b)
    cout<<"Both numbers are equal.";
  getch();
}
```

Output:
Enter a number: 15
Enter a number: 15
Both numbers are equal.

Program 5.3

Write a program that inputs two numbers and finds if second number is square of first.

```
#include <iostream.h>
#include <conio.h>
void main()
{
  int a, b;
  clrscr();
  cout<<"Enter a number: ";
  cin>>a;
  cout<<"Enter a number: ";
  cin>>b;
  if(a * a == b)
    cout<<"2nd number is square of 1st number.";
  getch();
}
```

Output:
Enter a number: 5
Enter a number: 25
2nd number is square of 1st number.

Program 5.4

Write a program that inputs marks of three subjects. If the average of marks is more than 80, it displays two messages "You are above standard!" and "Admission granted!"

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

```
{
  int sub1, sub2, sub3;
  float avg;
  clrscr();
  cout<<"Enter marks of first subject: ";
  cin>>sub1;
  cout<<"Enter marks of second subject: ";
  cin>>sub2;
  cout<<"Enter marks of third subject: ";
  cin>>sub3;
  avg = (sub1+sub2+sub3)/3.0;
  if(avg>80)
  {
    cout<<"You are above standard! \n";
    cout<<"Admission granted!";
  }
  getch();
}
```

Output:
Enter marks of first subject: 90
Enter marks of second subject: 80
Enter marks of third subject: 80
You are above standard!
Admission granted!

Program 5.5

Write a program that inputs three numbers and displays the maximum number.

```
#include <iostream.h>
#include <conio.h>
void main()
{
  int a, b, c, max;
  cout<<"Enter first  number: ";
  cin>>a;
  cout<<"Enter second number: ";
  cin>>b;
  cout<<"Enter third number: ";
  cin>>c;
  max = a;
  if(b > max)
    max = b;
  if(c > max)
    max = c;
  cout<<"The maximum number is "<<max;
  getch();
}
```

Output:
Enter first number: 20
Enter second number: 30
Enter third number: 12
The maximum number is 30

Program 5.6

Write a program to input a number and determine whether it is positive, negative or 0.

```
#include <iostream.h>
#include<conio.h>
void main( )
{
  int n;
  cout<<"Enter a number";
  cin>>n;
  if(n>0)
    cout<<"The number is positive";
```

Output:
Enter a number: 20
The number is positive.

```
    if (n<0)
      cout<<"The number is negative";
    if (n==0)
      cout<<"The number is zero";
    getch();
}
```

Program 5.7

Write a program that inputs five integers. It finds and prints the largest and smallest integer.

```
#include <iostream.h>
#include <conio.h>
void main()
{
  int a, b, c, d, e, max, min;
  clrscr();
  cout<<"Enter five integers: ";
  cin>>a>>b>>c>>d>>e;
  min = max = a;
  if(b < min) min = b;
  if(c < min) min = c;
  if(d < min) min = d;
  if(e < min) min = e;
  if(b > max) max = b;
  if(c > max) max = c;
  if(d > max) max = d;
  if(e > max) max = e;
  cout<<"Largest number is "<<max<<endl;
  cout<<"Smallest number is "<<min<<endl;
  getch();
}
```

Output:
Enter five integers: 10 32 76 61 40
Largest number is 76
Smallest number is 10

5.4 'if-else' Statement

if else statement is another type of **if** statement. It executes one block of statement(s) when the **condition** is **true** and the other when it is **false**. In any situation, one block is executed and the other is skipped. In **if else** statement:

- Both blocks of statement can never be executed.
- Both blocks of statements can never be skipped.

Syntax

Its syntax is as follows:

```
if (condition)
  statement;
else
  statement;
```

Two or more statements are written in curly brackets { }. The syntax for compound statements in **if else** statement is as follows:

```
if (condition)
{
  statement 1;
```

```
    statement 2;
        :
    statement N;
}
else
{
    statement 1;
    statement 2;
        :
    statement N;
}
```

Flowchart

The flowchart of **if else** statement is as follows:

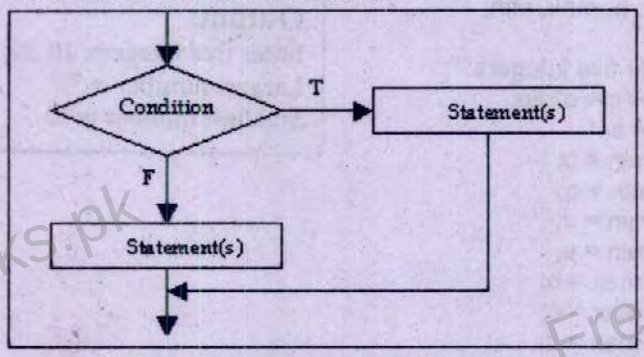


Figure 5.6: Flowchart of if-else structure

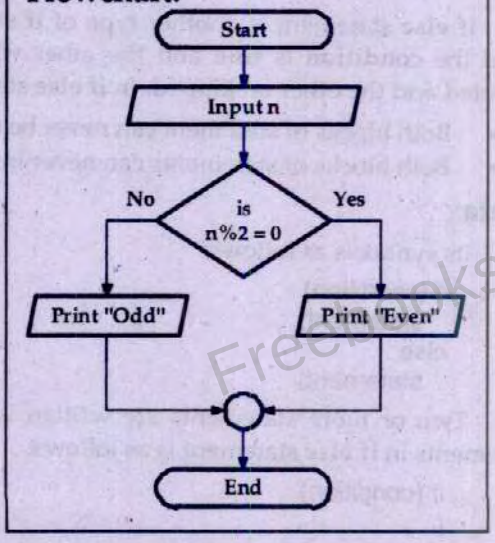
Program 5.8

Write a program that inputs a number and finds whether it is even or odd using if-else structure.

```
#include <iostream.h>
#include <conio.h>
void main()
{
  int n;
  clrscr();
  cout<<"Enter a number: ";
  cin>>n;
  if(n%2 == 0)
    cout<<n<<" is even.";
  else
    cout<<n<<" is odd.";
  getch();
}
```

Output:
Enter a number: 10
10 is even.

Flowchart:



Program 5.9

Write a program that inputs a year and finds whether it is a leap year or not using if-else structure.

```
#include <iostream.h>
#include <conio.h>
void main()
{
  int y;
  clrscr();
  cout<<"Enter a year: ";
  cin>>y;
  if(y % 4 == 0)
    cout<<y<<" is a leap year.";
  else
    cout<<y<<" is not a leap year.";
  getch();
}
```

Output:
Enter a year: 2003
2003 is not a leap year.

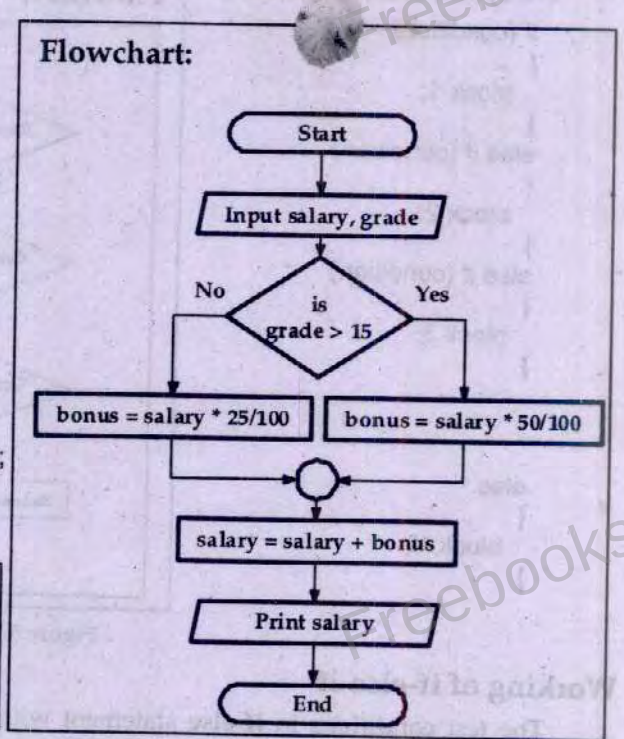
Program 5.10

Write a program that inputs salary and grade. It adds 50% bonus if the grade is greater than 15. It adds 25% bonus if the grade is 15 or less and then displays the total salary.

```
#include <iostream.h>
#include <conio.h>
void main()
{
  float salary, bonus;
  int grade;
  clrscr();
  cout<<"Enter your salary: ";
  cin>>salary;
  cout<<"Enter your grade: ";
  cin>>grade;
  if(grade>15)
    bonus = salary * 50.0/100.0;
  else
    bonus = salary * 25.0/100.0;
  salary = salary + bonus;
  cout<<"Your total salary is Rs. ";
  cout<<salary;
  getch();
}
```

Output:
Enter your salary: 16000
Enter your grade: 17
Your total salary is 24000

Flowchart:



Program 5.11

Write a program that inputs two integers. It determines and prints if the first integer is a multiple of second integer.

```
#include < iostream.h>
#include <conio.h>
void main()
{
  int a, b;
  clrscr();
  cout<<"Enter first integer: ";
  cin>>a;
  cout<<"Enter second integer: ";
  cin>>b;
 if(a % b == 0)
    cout<<"The first number is a multiple of second.";
 else
    cout<<"The first number is not a multiple of second.";
  getch();
}
```

Output:
Enter first integer: 8
Enter second integer: 2
The first number is a multiple of second.

5.5 Multiple 'if-else-if' Structure

if-else-if statement can be used to choose one block of statements from many blocks of statements. It is used when there are many options and only one block of statements should be executed on the basis of a condition.

Syntax

The syntax of this structure is:

```
if (condition)
{
  block 1;
}
else if (condition)
{
  block 2;
}
else if (condition)
{
  block 3;
}
.
.
.
else
{
  block N;
}
```

Flowchart

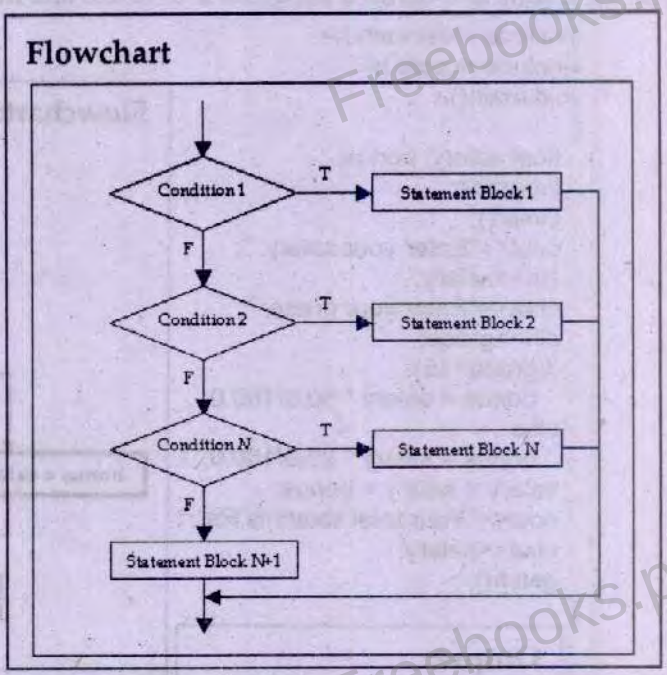


Figure 5.7: Flowchart of multiple if-else-if structure

Working of if-else-if

The test conditions in **if-else** statement with multiple alternatives are executed in a sequence until a **true** condition is reached. If a condition is **true**, the block of statements following the condition is executed. The remaining blocks are skipped. If a condition is **false**, the block of statements following the condition is skipped. The statement after the last **else** are executed if all conditions are **false**.

Program 5.12

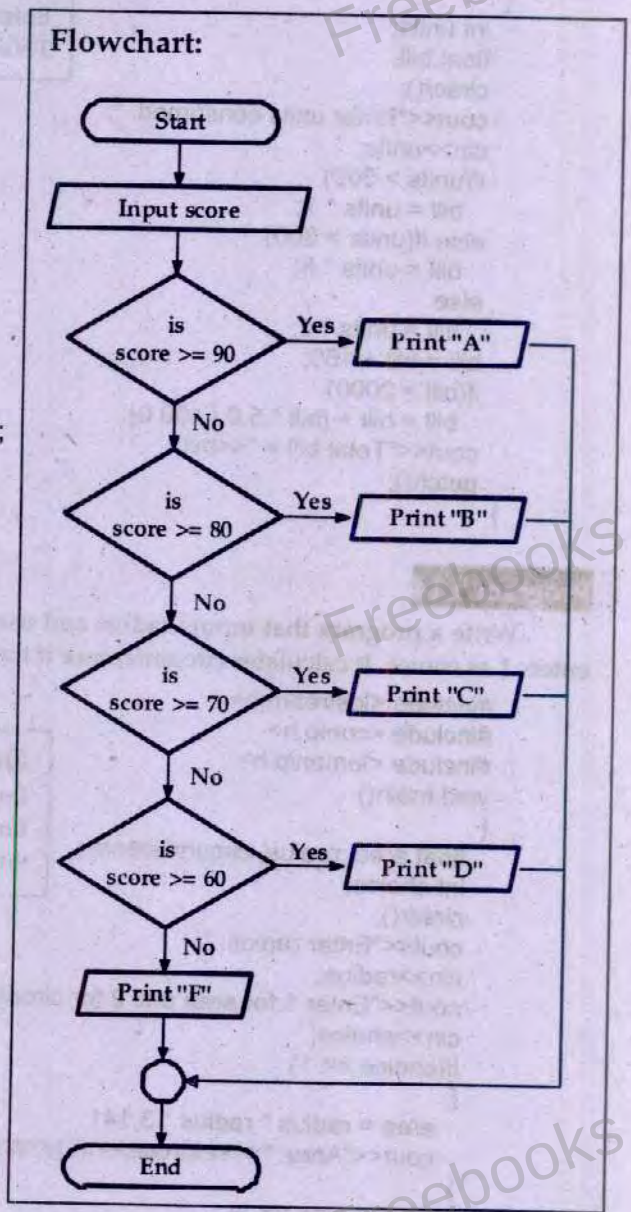
Write a program that inputs test score of a student and displays his grade according to the following criteria:

Test Score	Grade
>= 90	A
80 – 89	B
70 – 79	C
60 – 69	D
Below 60	F

```
#include <iostream.h>
#include <conio.h>
void main()
{
  int score;
  clrscr();
  cout<<"Enter your test score: ";
  cin>>score;
  if(score>=90)
    cout<<"Your grade is A.";
  else if(score>=80)
    cout<<"Your grade is B.";
  else if(score>=70)
    cout<<"Your grade is C.";
  else if(score>=60)
    cout<<"Your grade is D.";
  else
    cout<<"Your grade is F.";
  getch();
}
```

Output:
Enter your test score: 74
Your grade is C.

Flowchart:



Program 5.13

Write a program that calculates the electricity bill. The rates of electricity per unit are as follows:

- If the units consumed are <=300, then the cost is Rs. 2 per unit
- If the units consumed are >300 and <=500, then the cost is Rs. 5 per unit.
- If the units consumed exceed 500 then the cost per unit is Rs. 7

A line rent Rs. 150 is also added to the total bill and a surcharge of 5% extra if the bill exceeds Rs.2000. Calculate the total bill with all the conditions given above.

```
#include <iostream.h>
#include <conio.h>
void main()
{
  int units;
  float bill;
  clrscr();
  cout<<"Enter units consumed: ";
  cin>>units;
  if(units > 500)
    bill = units * 7;
  else if(units > 300)
    bill = units * 5;
  else
    bill = units * 2;
  bill = bill + 150;
  if(bill > 2000)
    bill = bill + (bill * 5.0 / 100.0);
  cout<<"Total bill = "<<bill;
  getch();
}
```

Output:
Enter units consumed: 300
Total bill = 750

Program 5.14

Write a program that inputs radius and user's choice. It calculates area of circle if user enters 1 as choice. It calculates circumference if the user enters 2 as choice.

```
#include <iostream.h>
#include <conio.h>
#include <iomanip.h>
void main()
{
  float area, radius, circumference;
  int choice;
  clrscr();
  cout<<"Enter radius: ";
  cin>>radius;
  cout<<"Enter 1 for area and 2 for circumference: ";
  cin>>choice;
  if(choice == 1)
  {
    area = radius * radius * 3.141;
    cout<<"Area: "<<setprecision(2)<<area;
  }
  else if(choice == 2)
  {
    circumference = 2.0 * 3.141 * radius;
    cout<<"Circumference: "<<circumference;
  }
  else
    cout<<"Invalid choice.";
  getch();
}
```

Output:
Enter radius: 5
Enter 1 for area and 2 for circumference: 1
Area: 78.53

Program 5.15

Write a program that inputs salary. If the salary is 20000 or more, it deducts 7% of salary. If the salary is 10000 or more but less than 20000, it deducts 1000 from the salary. If salary is less than 10000, it deducts nothing. It finally displays the net salary.

```
#include <iostream.h>
#include <conio.h>
void main()
{
  int salary;
  float net;
  clrscr();
  cout<<"Enter salary: ";
  cin>>salary;
  if(salary >= 20000)
    net = salary - (salary * 7.0/100);
  else if(salary >= 10000)
    net = salary - 1000;
  else
    net = salary;
  cout<<"Your net salary is "<<net;
  getch();
}
```

Output:
Enter your salary: 15000
Your net salary is 14000

5.6 Nested 'if' Structure

An if statement within an if statement is called **nested if** statement. In nested structure, the control enters into the **inner if** only when the outer condition is **true**. Only one block of statements are executed and the remaining blocks are skipped automatically.

The user can use as many **if** statements inside another **if** statement as required. The increase in the level of nesting increases the complexity of **nested if** statement.

Syntax

The syntax of nested If is as follows:

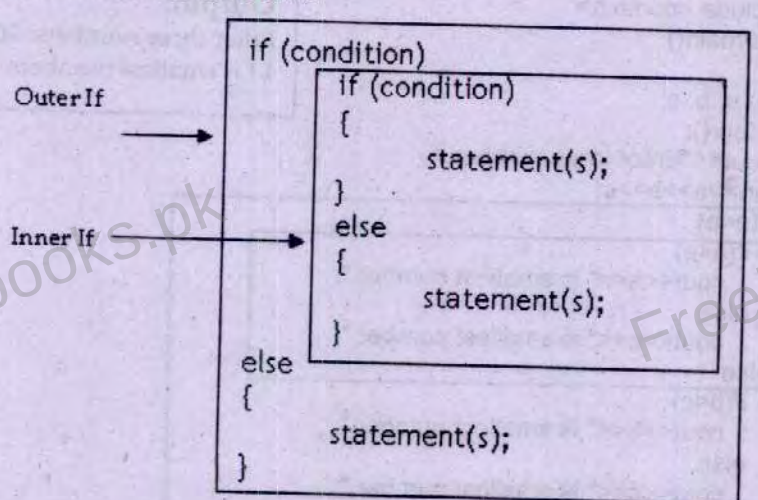


Figure 5.8: Syntax of nested if structure

Working of Nested IF

In **nested if** statement, the condition of **outer if** is evaluated first. If it is **true**, the control enters in the **inner if** block. If the condition is **false**, the **inner if** is skipped and control directly moves to the **else** part of **outer if**. If **outer if** is **true** then control enters in the **inner if** statement. The **inner if** evaluated according to **simple if** statement.

Flowchart

The flowchart of **nested if** statement is as follows:

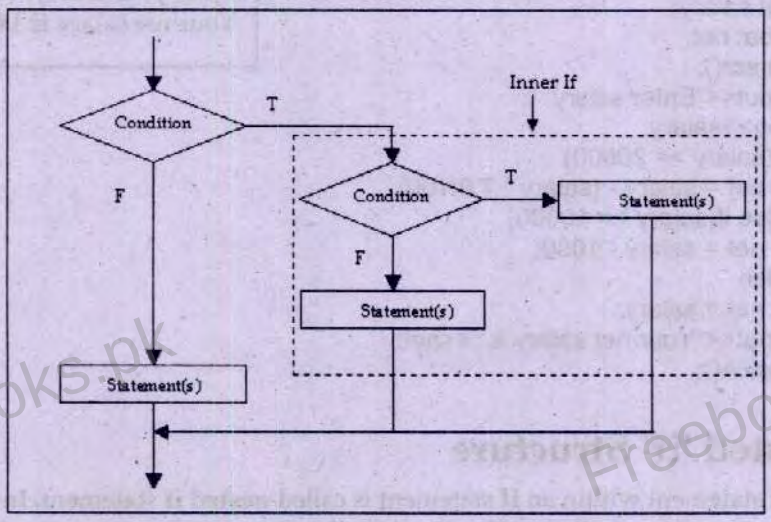


Figure 5.9: Flowchart of nested if structure

Program 5.16

Write a program that inputs three numbers and displays the smallest number by using nested if condition.

```
#include <iostream.h>
#include <conio.h>
void main()
{
  int a, b, c;
  clrscr();
  cout<<"Enter three number:";
  cin>>a>>b>>c;
  if(a<b)
    if(a<c)
      cout<<a<<" is smallest number.";
    else
      cout<<c<<" is smallest number.";
  else
    if(b<c)
      cout<<b<<" is smallest number.";
    else
      cout<<c<<" is smallest number.";
  getch();
}
```

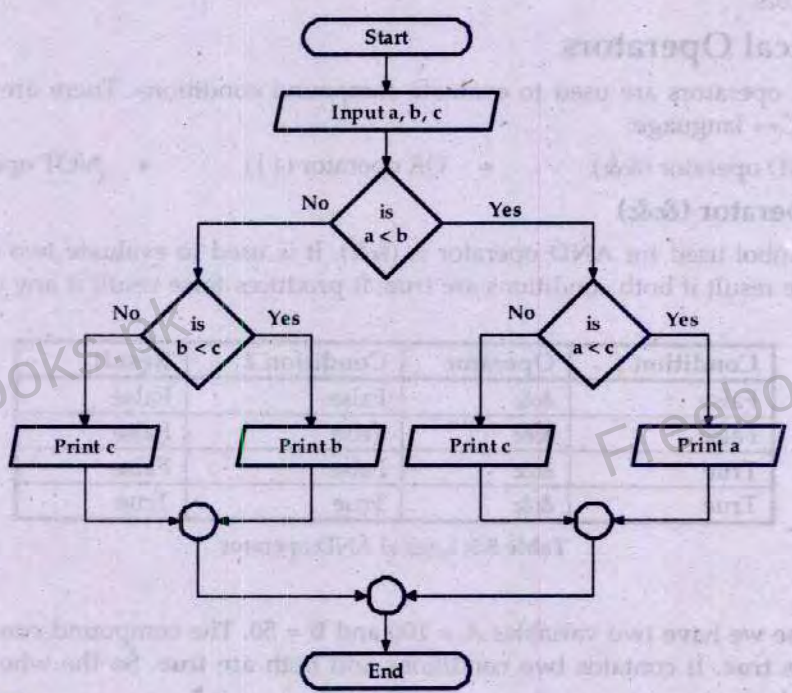
Output:

```
Enter three numbers: 20 35 13
13 is smallest numbers.
```

How it Works?

The above program inputs three numbers and finds the smallest one. When the control enters the outer box, first condition **if(a<b)** is evaluated. If it is true, the controls enters in the smaller box and the condition **if(a<c)** is evaluated. If it is also **true**, the value of **a** is displayed, otherwise the value of **c** is displayed. If the first condition **if(a<b)** is **false**, the control shifts to **else** part of **if** statement. Now the condition **if(b<c)** is evaluated. If it is **true** the value of **b** is displayed, otherwise the value of **c** is displayed.

Flowchart



Program 5.17

Write a program that inputs three numbers and displays whether all numbers are equal or not by using nested if condition.

```
#include <iostream.h>
#include <conio.h>
void main()
{
  int a, b, c;
  cout<<"Enter three number:";
  cin>>a>>b>>c;
  if(a==b)
    if(a==c)
      cout<<"All numbers are equal.";
    else
      cout<<"Numbers are different.";
  else
    cout<<"Numbers are different.";
  getch();
}
```

Output:
Enter three numbers: 15 25 30
Numbers are different.

5.7 Compound Condition

A type of comparison in which more than one conditions are evaluated is called **compound condition**. It is used to execute a statement or set of statements by testing many conditions.

Example

For example, a program inputs two numbers. It displays **OK** if one numbers is greater than 100 and second number is less than 100. Compound condition is executed by using logical operators.

5.7.1 Logical Operators

Logical operators are used to evaluate compound conditions. There are three logical operators in C++ language:

- AND operator (&&)
- OR operator (||)
- NOT operator (!)

1. AND Operator (&&)

The symbol used for **AND** operator is (&&). It is used to evaluate two conditions. It produces **true** result if both conditions are **true**. It produces **false** result if any one condition is **false**.

Condition 1	Operator	Condition 2	Result
False	&&	False	False
False	&&	True	False
True	&&	False	False
True	&&	True	True

Table 5.3: Logical AND operator

Example

Suppose we have two variables A = 100 and B = 50. The compound condition **(A>10) && (B>10)** is **true**. It contains two conditions and both are **true**. So the whole compound condition is also **true**.

The compound condition **(A>50) && (B>50)** is **false.** It contains two conditions. One condition **(A>50)** is **true** and second condition **(B>50)** is **false**. So the whole compound condition is **false**.

Program 5.18

Write a program that inputs three numbers and displays the maximum number by using logical operators.

```
#include <iostream.h>
#include <conio.h>
void main()
{
  int a, b, c;
  cout<<"Enter three numbers: ";
  cin>>a>>b>>c;
  if(a>b && a>c)
    cout<<"Maximum number is "<<a;
  else if(b>a && b>c)
    cout<<"Maximum number is "<<b;
```

Output:
Enter three numbers: 10 20 30
Maximum number is 30

```
    else
      cout<<"Maximum number is "<<c;
    getch();
}
```

2. OR Operator (||)

The symbol used for **OR** operator is (||). It is used to evaluate two conditions. It produces **true** result if either condition is **true**. It produces **false** result if both conditions are **false**.

Condition 1	Operator	Condition 2	Result
False	||	False	False
False	||	True	True
True	||	False	True
True	||	True	True

Table 5.4: Logical OR operator

Example

Suppose we have two variables A = 100 and B = 50. The compound condition **(A>50) || (B>50)** is **true**. It contains two conditions and one condition **(A>50)** is **true**. So the whole compound condition is also **true**. The compound condition **(A>500) || (B>500)** is **false** because both conditions are **false**.

Program 5.19

Write a program that inputs a character and displays whether it is a vowel or not.

```
#include <iostream.h>
#include <conio.h>
void main()
{
  char ch;
  clrscr();
  cout<<"Enter any character: ";
  cin>>ch;
  if(ch=='A' || ch=='a' || ch=='E' || ch=='e' || ch=='I' || ch=='i' || ch=='O' || ch=='o' ||
  ch=='U' || ch=='u')
    cout<<"You entered a vowel: "<<ch;
  else
    cout<<"You did not enter a vowel: "<<ch;
  getch();
}
```

Output:
Enter any character: A
Your entered a vowel: A

Program 5.20

Write a program that allows the user to enter any character through the keyboard and determines whether it is a capital letter, small case letter, a digit number or a special symbol.

```
#include <iostream.h>
#include <conio.h>
void main()
{
  clrscr();
  char ch;
```

Output:
Enter any character: #
The character # is a symbol.

```
    cout << "Enter any character: "<<endl;
    cin>>ch;
    if ((ch >= 'A') && (ch <= 'Z'))
      cout<< "The character "<<ch<<" is a capital letter."<<endl;
    else if ((ch >= 'a') && (ch <= 'z'))
      cout<< "The character "<<ch<<" is a small case letter."<<endl;
    else if ((ch >= '0') && (ch <= '9'))
      cout<< "The character "<<ch<<" is a digit."<<endl;
    else
      cout<<"The character "<<ch<<" is a symbol."<<endl;
    getch();
}
```

3. NOT Operator (!)

The symbol used for **NOT** operator is (!). It is used to reverse the result of a condition. It produces **true** result if the condition is **false**. It produces **false** result if the condition is **true**.

Operator	Condition	Result
!	True	False
!	False	True

Table 5.5: Logical NOT operator

Example

Suppose we have two variables A = 100 and B = 50. The condition !(A==B) is true. The result of (A==B) is false but NOT operator converts it into true. The condition !(A>B) is false. The condition (A>B) is true but NOT operator converts it into false.

Program 5.21

Write a program that inputs a number and displays whether it is even or odd by using logical operator "!".

```
#include <iostream.h>
#include <conio.h>
void main()
{
  int n;
  clrscr();
  cout<<"Enter any number: ";
  cin>>n;
  if(!(n%2==0))
    cout<<"You entered odd number.";
  else
    cout<<"You enter even number.";
  getch();
}
```

Output:
Enter any number: 20
You entered even number.

Program 5.22

Write a program that inputs three digits and displays all possible combinations of these digits.

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

```
int a,b,c;
clrscr();
cout<<"Enter three digits:";
cin>>a>>b>>c;
if((a!=b) && (b!=c) && (c!=a))
{
  cout<<a<<b<<c<<"\t";
  cout<<a<<c<<b<<"\t";
  cout<<b<<a<<c<<"\t";
  cout<<c<<a<<b<<"\t";
  cout<<c<<b<<a;
}
else
{
  if((a==b) && (a==c))
 cout<<a<<b<<c;
  else
  {
   if(a==b)
   {
     cout<<a<<b<<c<<"\t";
     cout<<a<<c<<b<<"\t";
     cout<<c<<b<<a;
   }
   else
   {
    if(a==c)
    {
       cout<<a<<c<<b<<"\t";
       cout<<a<<b<<c<<"\t";
       cout<<b<<a<<c;
    }
    else
    {
       cout<<b<<c<<a<<"\t";
       cout<<b<<a<<c<<"\t";
       cout<<a<<b<<c;
    }
   }
  }
}
getch();
}
```

Output:

```
Enter three digits: 1 2 3
123     132     213     312     321
```

5.8 'switch' Structure

The **switch** statement is another conditional structure. It is a good alternative of **nested if-else**. It can be used easily when there are many choices available and only one should be executed. Nested if becomes very difficult in such situation.

Syntax

The syntax for writing this structure is as follows:

```
switch (expression)                    <- Integer or character variable
{                                      <- Integer or character constant
  case constant 1:
    statement(s);
    break;                             <- First case body -> Causes exit from case body
  case constant 2:
    statement(s);
    break;                             <- Second case body
    .
    .
  case constant N:
    statement(s);
    break;                             <- N case body
  default:
    statement(s);                      <- Default body
}
```

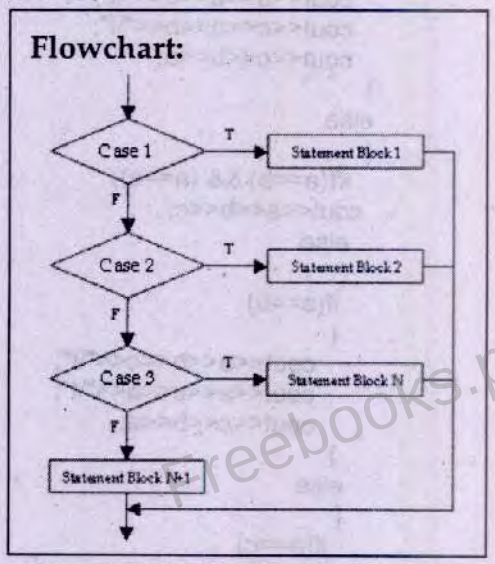


Figure 5.10: Syntax of switch statement

Working of 'switch' Structure

switch statement compares the result of a single expression with multiple cases. Expression can be any valid expression that results in integer or character value. The expression is evaluated at the top of switch statement and its result is compared with different cases. Each case represents one choice. If the result matches with any case, the corresponding block of statements is executed. Any number of cases can be used in one switch statement.

The **default** label appears at the end of all **case** blocks. It is executed only when the result of **expression** does not match with any case label. Its use is optional. The position of default label is not fixed. It may be placed before the first case statement or after the last one.

The **break** statement in each case is used to exit from **switch** body. It is used at the end of each **case** block. When the result of the expression matches with a **case** block, the corresponding statements are executed. The **break** statement comes after these statements and the control exits from **switch** body. If **break** is not used, all case blocks that come after the matching case, will also be executed.

Program 5.23

Write a program that inputs number of week's day and displays the name of the day. For example if user enters 1, it displays "Friday" and so on.

```
#include <iostream.h>
#include <conio.h>
```

```
void main()
{
  int n;
 clrscr();
 cout<<"Enter number of a weekday: ";
 cin>>n;
 switch(n)
 {
   case 1:
    cout<<"Friday";
    break;
   case 2:
    cout<<"Saturday";
    break;
   case 3:
    cout<<"Sunday";
    break;
   case 4:
    cout<<"Monday";
    break;
   case 5:
    cout<<"Tuesday";
    break;
   case 6:
    cout<<"Wednesday";
    break;
   case 7:
    cout<<"Thursday";
    break;
   default:
    cout<<"Invalid number";
 }
 getch();
}
```

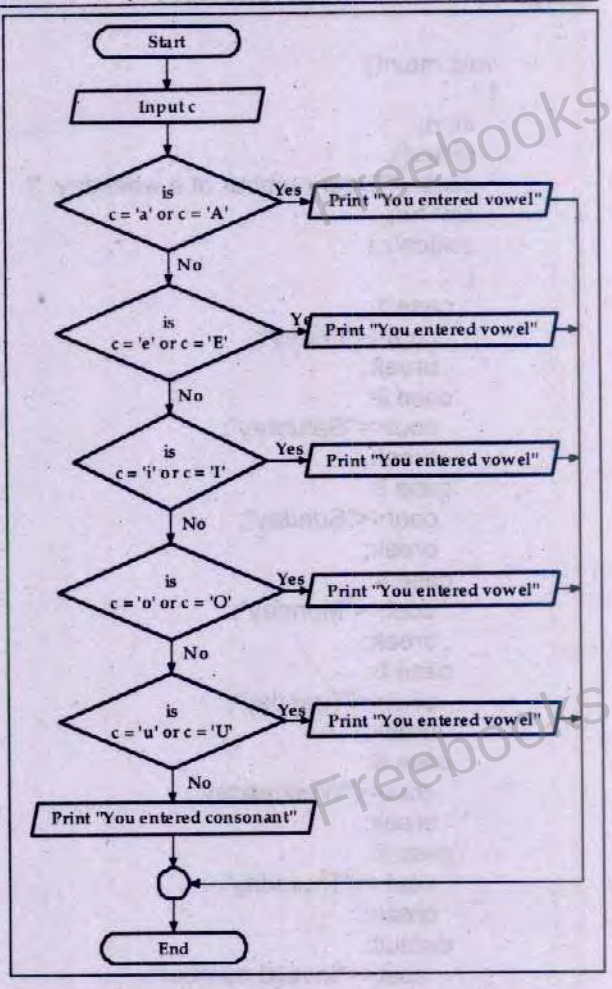
Output:
Enter number of a weekday: 3
Sunday

Program 5.24

Write a program that inputs a character from the user and checks whether it is a vowel or consonant.

```
#include <iostream.h>
#include <conio.h>
void main()
{
 char c;
 clrscr();
 cout<<"Enter an alphabet: ";
 cin>>c;
 switch(c)
 {
   case 'a':
   case 'A':
    cout<<"You entered vowel.";
    break;
```

```
    case 'e':
    case 'E':
      cout<<"You entered vowel.";
      break;
    case 'i':
    case 'I':
      cout<<"You entered vowel.";
      break;
    case 'o':
    case 'O':
      cout<<"You entered vowel.";
      break;
    case 'u':
    case 'U':
      cout<<"You entered vowel.";
      break;
    default:
      cout<<"Not vowel";
  }
  getch();
}
```



Output:
Enter an alphabet: u
You entered a vowel.

Program 5.25

Write a program that inputs a floating point number, an operator and another floating point number. It displays the result by performing the operation on the given numbers. If the operator is a division, it should check to make sure that the divisor is not equal to zero. If the operator is not a +, -, *, or / then the program should print an error message.

```
#include <iostream.h>
#include <conio.h>
void main()
{
  float a, b;
  char op;
  clrscr();
  cout<<"Enter a floating point number: ";
  cin>>a;
  cout<<"Enter an operator: ";
  cin>>op;
  cout<<"Enter a floating point number: ";
  cin>>b;
  switch(op)
  {
    case '+':
```

Output:
Enter a floating point number: 5.2
Enter an operator: *
Enter a floating point number: 1.3
6.759999

```
            cout<<a+b<<endl;
            break;
        case '-':
            cout<<a-b<<endl;
            break;
        case '*':
            cout<<a*b<<endl;
            break;
        case '/':
            if (b == 0)
              cout<<"Division by zero!"<<endl;
            else
              cout<<a/b<<endl;
            break;
        default:
            cout<<"Invalid operator!"<<endl;
    }
    getch();
}
```

Program 5.26

Write a program that displays the following menu of a health club:

1. Standard Adult Membership
2. Child Membership
3. Senior Citizen Membership
4. Quit the Program

It inputs choice and number of months and calculates membership charges as follows:

Choice	Charges per month
1. Standard Adult Membership	Rs. 50
2. Child Membership	Rs. 20
3. Senior Citizen Membership	Rs. 30

```
#include <iostream.h>
#include <iomanip.h>
void main()
{
  int choice, months;
  double charges;
  cout << "   Health Club Membership Menu\n\n";
  cout << "1. Standard Adult Membership\n";
  cout << "2. Child Membership\n";
  cout << "3. Senior Citizen Membership\n";
  cout << "4. Quit the Program\n\n";
  cout << "Enter your choice: ";
  cin  >> choice;
  if (choice >= 1 && choice <= 3)
  {
    cout << "For how many months? ";
    cin  >> months;
    switch (choice)
    {
```

Output:

```
Health Club Membership Menu
1. Standard Adult Membership
2. Child Membership
3. Senior Citizen Membership
4. Quit the Program

Enter your choice: 2
For how many months? 6
Total charges are Rs. 120
```

```
            case 1:   charges = months * 50.0;
            break;
            case 2:   charges = months * 20.0;
            break;
            case 3:   charges = months * 30.0;
          }
          cout<<"The total charges are Rs. "<<charges<< endl;
        }
        else if (choice != 4)
        {
          cout << "The valid choices are 1 to 4.\n";
          cout << "Run the program again and select one of these.\n";
        }
      }
```

Program 5.27

Write a program that converts ASCII number to character and vice versa. The program should display the following menu:

1. Convert ASCII value to Character
2. Convert Character to ASCII value

Output:

1. Convert ASCII value to Character
2. Convert Character to ASCII value
Enter your choice: 1
Enter a number: 97
The corresponding character is a

```
#include<iostream.h>
#include <stdlib.h>
#include<conio.h>
void main()
{
    int number, option;
    char letter;
    cout<<"1. Convert ASCII value to Character"<<endl;
    cout<<"2. Convert Character to ASCII value"<<endl;
    cout<<"Enter your choice: ";
    cin>>option;
    switch (option)
    {
      case 1:
        cout<<"Enter a number : ";
        cin >> number;
        cout<<"The corresponding character is: "<<char(number)<<endl;
        break;
      case 2:
        cout<<"Enter a letter : ";
        cin >> letter;
        cout<<"ASCII value of the letter is: "<<int(letter)<<endl;
        break;
      default:
        cout<<"Invalid Option!";
        break;
    }
    getch();
}
```

5.8.1 Difference between nested 'if-else' and 'switch'

The switch statement and nested if-else statements are used when there are multiple choices and only one is to be executed. Difference between if-else and switch is as follows:

	Switch	Nested-If
1.	It is easy to use when there are multiple choices.	It is difficult to use when there are multiple choices.
2.	It uses single expression for multiple choices.	It uses multiple expressions for multiple choices.
3.	It cannot check a range of values.	It can check a range of values.
4.	It checks only constant values. You cannot use variables with case statement.	It can check variables also. You can use a constant or variable in relational expressions.

Table 5.6: Difference between switch and nested if

5.9 Conditional Operator

Conditional operator is a decision-making structure. It can be used in place of simple **if-else** structure. It is also called **ternary operator** as it uses three operands.

Syntax

The syntax of conditional operator is as follows:

```
(condition) ? true-case statement: false-case statement;
```

condition The condition is specified as relational or logical expression. The condition is evaluated to **true** or **false**.

true-case It is executed if expression evaluates to **true**.

false-case It is executed if expression evaluates to **false**.

Example

Suppose we have a variable A. The following statement:

```
X = (A>50) ? 1 : 0;
```

will assign 1 to X if the condition **A>50** is **true**. It will assign 0 to X the condition is **false**. The above statement can be written using **if-else** statement as follows:

```
if (A>50)
  X = 1;
else
  X = 0;
```

Program 5.28

Write a program that inputs marks of a student and displays "Pass" if marks are more than 40 and "Fail" otherwise by using conditional operator.

```
#include <iostream.h>
#include <conio.h>
void main()
{
  int marks;
  clrscr();
  cout<<"Enter your marks:";
  cin>>marks;
```

Output:
Enter your marks: 92
Result is Pass

```
    cout<<"Result is "<<(marks>40 ? "Pass" : "Fail");
    getch();
}
```

Program 5.29

Write a program that inputs a number and displays whether it is divisible by 3 or not by using conditional operator.

```
#include <iostream.h>
#include <conio.h>
void main()
{
  int n;
  clrscr();
  cout<<"Enter number:";
  cin>>n;
  (n%3==0 ? cout<<"Divisible by 3" : cout<<"Not divisible by 3");
  getch();
}
```

Output:
Enter number: 15
Divisible by 3

5.10 'goto' Statement

The 'goto' statement is used to move the control directly to a particular location of the program by using label. A label is a name given to a particular line of the program. A label is created with a valid identifier followed by a colon (:).

Syntax

```
goto Label;
```

The 'Label' indicates the lable to which the control is transferred.

Program 5.30

Write a program that displays "C++" five times using goto statement.

```
#include <iostream.h>
#include <conio.h>
void main ()
{
  clrscr();
  int n=1;
  loop:
  cout<<n<<": C++"<<endl;
  n++;
  if (n<=5) goto loop;
  cout<<"End of Program";
  getch();
}
```

Output:
1: C++
2: C++
3: C++
4: C++
5: C++
End of Program

How above Program Works?

The above program uses **goto** statement to repeat a statement. The **if** statement checks the value of n. If the value is 5 or less, the control moves back to the **loop** label. In this way, the required message appears five times.

Programming Exercises

1. Writing a program that accepts a character and determines whether the character is a lowercase letter. A lowercase letter is any character that is greater than equal to 'a' and less than or equal to 'z'. If the entered character is a lowercase letter, display the message "Entered character is a lowercase letter", otherwise display the message "Entered character is not a lowercase letter".
2. Senior salesperson is paid Rs. 400 a week, and a junior salesperson is paid Rs. 275 a week. Write a program that accepts as input a salesperson's status in the character variable *status*. If *status* is 's' or 'S', the senior person's salary should be displayed; if status is 'j' or 'J', the junior person's salary should be displayed, otherwise display error message.
3. Write a program to get three numbers from user for integer variables a, b and c. If a is not zero, find out whether it is the common divisor of b and c.
4. Write a program that contains an if statement that may be used to compute the area of a square (area = side * side) or a triangle (area = ½ * base * height) after prompting the user to type the first character of the figure name (S or T).
5. Write a program that gets the number and a letter. If the letter is 'f', the program should treat the number entered as temperature in degrees Fahrenheit and convert it to the temperature in degree Celsius and print a suitable message. If the letter is 'c', the program should consider the number as Celsius temperature and convert it to Fahrenheit temperature and print a suitable message. The program should display error message and then exit if the user enters any other letter.
6. Write a program that accepts the code number as an input and display the correct disk drive manufacture as follows:

Code	disk drive manufacture
1	Western Digital
2	3M Corporation
3	Maxell Corporation
4	Sony Corporation
5	Verbatim Corporation

7. Write a program that uses the following categories of movies:

   A for Adventure movies
   C for Comedy movies
   F for Family movies
   H for Horror movies
   S for Science Fiction movies

   The program inputs code for movie type and displays its category. For example if the user enters H, it displays "Horror Movies". The program should also display a menu of movie categories.
8. Write a program that inputs a value and type of conversion. The program should then output the value after conversion. The program should include the following conversions:

   1 inch = 2.54 centimeters
   1 gallon = 3.785 liters

1 mile = 1.609 kilometers
1 pound = .4536 kilograms

Make sure that program accepts only valid choices for type of conversion to perform.

9. A year is a leap year if it is divisible by four, except that any year divisible by 100 is a leap year only if it is divisible by 400. Write a program that inputs a year such as 1996, 1800 and 2010 and display "Leap year" if it is a leap year otherwise displays "Not a leap year".

10. Write a program that inputs temperature and displays a message as follows:

Temperature	Message
Greater than 35	Hot day
Between 25 and 35 (inclusive)	Pleasant day
Less than 25	Cool day

11. Write a program that inputs obtained marks of a student, calculates the percentage (assuming total marks are 1100) and displays grade according to the following rules:

Percentage	Grade
More than or equal to 80	A+
Between 70 (inclusive) and 80	A
Between 60 (inclusive) and 70	B
Between 50 (inclusive) and 60	C
Between 40 (inclusive) and 50	D
Between 33 (inclusive) and 40	E
Less than 33	F

12. Write a program that converts MILITARY time to STANDARD time.

13. Write a program that will take three values a, b and c and print the roots, if real, of the quadratic equation $ax^2 + bx + c = 0$. Sampel input 1: (a=1, b=1, c=-6 the output should be "Roots of the equation are 2 and -3"). Sample input 2 (if the input is a=1, b=0, c=9, the output should be "Sorry the roots are not real.")

14. Write a program that inputs the salary of an employee from the user. It deducts the income tax from the salary on the following basis :

- 20% income tax if the salary is above Rs. 30000.
- 15% income tax if the salary is between Rs. 20000 and Rs. 30000.
- 10% income tax if the salary is below Rs. 20000.

The program finally displays salary, income tax and the net salary.

15. Write a program that inputs year and month. It displays the number of days in the month of the year entered by the user. For example, if the user enters 2010 in year and 3 in month, the program should display "March 2010 has 31 days".

16. Write a program that displays the following menu for a parking area:

- M = Motorcycle
- C = Car
- B = Bus

The program inputs the type of vehicle and number of days to park the vehicle. It finally displays the total charges for the parking according to the following:

- Motorcycle Rs. 10 per day
- Car Rs. 20 per day
- Bus Rs. 30 per day

17. Write a program that inputs a value and type of conversion. The program should then display the output after conversion. The program should include the following conversions:

- 1 cm = .394 inches
- 1 liter = .264 gallons
- 1 kilometer = .622 miles
- 1 kilogram = 2.2 pounds

Make sure that the program accepts only valid choices for the type of conversion.

Exercise Questions

Q.1. What is a control structure?

A statement used to control the flow of execution in a program or function is called control structure. The control structures in C are used to combine individual instruction into a single logical unit. The logical unit has one entry point and one exit point.

Q.2. What are the basic control structures for writing programs?

The basic control structures for writing programs are sequence, selection and repetition.

Q.3. Name and describe three types of branching mechanisms.

Three types of branching mechanisms include if...else statement, multi-way if...else statement and switch statement. An if...else statement selects between two alternative statements based on a condition. A multi-way if...else statement evaluates several conditions and selects one block of statements from many blocks. The switch statement evaluates an expression and executes statements in the matching case label.

Q.4. Write three selection statements and three repetition statements.

Selection statements are if, if...else, switch. The repetition statements are while, do...while, for.

Q.5. What is Relational Operators? List out different relational operators in C++ language?

The relational operators are used to specify conditions in programs. A relational operator compares two values. It produces result as true or false. The relational operators are sometimes called the conditional operators or comparison operators as they test conditions that are either true or false. C++ provides different relational operators. These are >, <,==, >=, <=, and !=.

Q.6. What is relational expression? Give some examples of relational expressions

Relational expression is a statement that uses relational operators to compare two values. Examples of relational expressions are A>B, A<B, A<=B, A>=B, A==B and A!=B.

Q.7. When must you use curly braces { } with a selection or repetition statement?

The curly braces are required if the selection or repetition statement has more than one statement in its body.

Q.8. What is Compound Condition? Give its example.

A type of comparison in which more than one conditions are evaluated is called **compound condition**. It is used to execute a statement or set of statements by testing many conditions. For example, a program inputs two numbers. It displays **OK** if one numbers is greater than 100 and second number is less than 100. Compound condition is executed by using logical operators.

Q.9. What is dangling else?

When an if-else statement is used as the statement part of another if statement, it is not clear that else is associated with which if statement. It is known as dangling else.

Q.10. What happens if break is missed in case block?

If break is not used, all case blocks coming after matching case will also be executed.

Q.11. Which data types can be used in the expression in the switch statement?

The data types that can be used in the expression in switch statement are int and char.

Q.12. What is the alternative of if-else statement in C++?

Conditional operator is an alternative of if-else statement in C. It is a decision-making structure. It is also called ternary operator as it uses three operands.

Q.13. Describe the importance of break & default statement in switch statement.

break statement in each case label is used to exit from switch body. If break is not used, all case blocks coming after matching case will also be executed. The default block is used to execute a statement or set of statements when the result of expression in switch statement does not match with any case.

Q.14. Why break statement is used in a switch ( ) structure?

The break statement in each case label is used to exit from switch body. If break is not used, all case blocks coming after matching case will also be executed.

Q.15. Why should you use a default label in a switch statement?

The default block is used to execute a statement or set of statements when the result of expression in switch statement does not match with any case. It makes sure that switch executes properly if the selector is not in the range of case labels.

Q.16. What is conditional operator? Write the syntax of conditional operator.

Conditional operator is a decision-making structure. It can be used in place of simple "if-else" structure. It is also called ternary operator because it uses three operands. The syntax of conditional operator is as follows:

(condition) ? true-case statement : false-case statement;

Q.17. Write one similarity and one difference between if...else and conditional operator.

The similarity between if...else and conditional operator is that both select a statement depending on a condition. The difference is that conditional operator can choose between two values but if...else can be extended to check multiple conditions to select one statement from many available statements.

Q.18. Convert the following code to an equivalent if-else statement:

```
Answer= (x >y)? x*y : x+y;
```

```
if (x>y)
  Answer = x*y;
else
  Answer = x+y;
```

Q.19. Write C++ expressions that represent the following English expressions. Assume that all variables have been properly defined and initialized.

1. x is even and y is odd (x and y are integers)
2. x is not between –47.5 and +132.0 (x is double)
3. x is either greater than 10, or less than or equal to -142
4. w is more than 200, and one or more of the following three conditions is fulfilled: x is less than 98.6; y is less than 60.0; z is more than 160.0
5. age is from 18 to 25
6. Temperature is less than 40.0 and greater than 25.0
7. Year is divisible by 4
8. Y is greater than x and less than z.
9. W is either equal to 6 or not greater than 3
10. p and q are less than zero
11. score is between 50 and 60 inclusive
12. y is equal to –5 or 5
13. age is outside the range p to q
14. digit is either equal to 50 or not less than 100
15. at least one of x or y is odd
16. The hit is no more than 6.7 units away from target
17. ch is an upper case alphabetic character (between 'A' and 'Z')

Answer:

1. (x%2 == 0) && (y%2 == 1)
2. !((-47.5 < x) && (x < 132.0)) or: !(-47.5 < x) || !(x < 132.0)
3. (x > 10) || (x <= -142)
4. (w > 200.0) && ((x < 98.6) || (y < 60.0) || (y > 160.0))
5. (age>18 && age<25)

6. (temperature<40.0 && temperature>25.0)
7. (year%4==0)
8. (y>x && y<z)
9. (w==6 || (!(w>3))
10. (p < 0) && (q < 0)
11. (score >= 50) && (score <= 60)
12. (y = = -5) || (y = = 5)
13. (age < p) || (age > q)
14. digit == 50 || !(digit < 100)
15. x%2 || y%2
16. ((hit - target) <= 6.7) && ((hit - target) <= -6.7)
17. ('A' <= ch) && (ch <= 'Z')

Q.20. Write C++ statements for the following:

a. **Assign the value of 1 to variable test if k is in the range -m through +m inclusive. Otherwise, assign a value of zero.**

```
if (k >= -m && k <= +m)
        test = 1;
else
        test = 0;
```

b. **Assign a value of 1 to variable lowercase if ch is a lowercase letter; otherwise, assign a value of zero.**

```
if (ch >= 97 && ch <= 122)
        lowercase = 1;
else
        lowercase = 0;
```

c. **Assigns a value of 1 to variable divisor if m is a divisor of n; otherwise assign a value of 0.**

```
if (m%n == 0)
        divisor = 1;
else
        divisor = 0;
```

Q.21. Do the following expressions evaluate to true or false?

a. (3<4) || (3>4)
b. (3!=3) && (4 == 4)
c. (15>=15) && (16==16) || (14<2)
d. 6 <= 6) && (5 < 3)
e. (6 <= 6) || (5 < 3)
f. (5 != 6)
g. (5 < 3) && (6 <= 6) || (5 != 6)
h. (5 < 3) && ((6 <= 6) || (5 != 6))
i. !((5 < 3) && ((6 <= 6) || (5 != 6)))
j. !(12 > 25) && !(18 < 17)

Answer:

a. True	b. False	c. True	d. False	e. True
f. True	g. True	h. False	i. True	j. True

Q.22. What is the output by the following code segment?

a)
```
int p = 1, q = 9;
if ((p >= 3) || (q < 11 && p + q < 15))
```

```
        cout<<("hello\n");
    else
        cout<<("goodbye\n");
```

Answer: hello

b)
```
    int n, k = 5;
    n = (100 % k ? k + 1 : k - 1);
    cout << "n = " << n << " k = " << k << endl;
```

Answer: n = 4 k = 5

c)
```
    int found = 0, count = 5;
    if (!found || --count == 0)
      cout << "danger" << endl;
    cout << "count = " << count << endl;
```

Answer: danger
count = 5

Q.23. What is the output by the following code segment when score has the value 62?

```
if (score < 50)
        cout << "Failing" ;
if (score < 60)
        cout << "Below average" ;
if (score < 70)
        cout << "Average" ;
if (score < 80)
        cout << "Above average" ;
if (score < 90)
        cout << "Very good" ;
if (score < 100)
        cout << "Excellent" ;
```

Answer: **AverageAbove averageVery goodExcellent**

Q.24. What is the output of the following statement, if grade = 'B'?

```
if (grade == 'A' || grade == 'B' && grade == 'C')
   cout << "Fail";
else
   cout << "Pass";
```

Answer: Pass

Q.25. What is the output of the following program?

```
int b=6, c=5;
if(b++==7 && ++c==5)
{
        b*=c;
        cout<<++b<<endl;
}
else
        cout<<b--<<endl;
```

Answer: 7

Q.26. What is the output of the following program?

```
int n1, n2;
n1 = 25;
n2 = 50;
if (n1 < n2 && !(n2 != 2 * n1))
{
  n1 = n1 * 2;
  n2 = n2 * 2;
```

```
    cout<<n1<<n2;
  }
  if (n2 < 2 * n1 || n1 == n2)
    n1 = n1 / 2;
    n2 = n2 / 2;
  cout<<n1<<n2;
```

Answer: 501005050

Q.27. What is the output of the following?

```
int n = 0;
  if('A' == 'a')
    n = 1;
  else if('A' > 'a')
    n = 2;
  else
    n = 3;
  cout << n << endl;
```

Answer: 3

Q.28. What is the output of the following?

```
int n = 5, a = -1, b = 3;
switch((--n)%4) {
  case 0: n++;
  break;
  case 1: n *= a++;
  case 2: n +=--b;
  case 3: --n;
  default:n++;
}
cout<<n;
```

Answer: 5

Q.29. What is the output of the following?

```
int n = 5, a = 4, b = 0, c = 8;
if (b<=6 && c>4 || a++==6)
n ++;
if (a!=4 || b>5 || c<=9)
n += a++;
if (c<=2 || a>=5 && b--==0)
n *= a++;
```

Answer: 50

Q.30. What is the output of the following?

```
int a = 4, b = 2, c = 5;
if(a > b)
{
        a = 5;
}
if(c == a)
{
        a = 6;
}
else
{
        a = 7;
}
cout << a;
```

Answer: 6

Q.31. What is the output of the following?

```
char choose = 'b';
switch (choose)
{
        case 'a':
        case 'A':
                cout << "you win!\n";
        case 'b':
        case 'B':
                cout << "you lose!\n";
        case 'c':
        case 'C':
                cout << "you draw!\n";
        default:
                cout << "I don't know you!\n";
}
```

Answer:

```
you lose!
you draw!
I don't know you!
```

Q.32. What is the output of the following?

```
int x = 5 + 7/2;
switch (x)
{
        case 5: cout << 'p'; break;
        case 7: cout << 'q'; break;
        case 8: cout << 'r'; break;
        default: cout << "No output";
}
```

Answer: r

Q.33. Write C++ code to accomplish each of the following (just the code fragment required to achieve the output - no headers, declarations, etc):

a. Display "child" if x is less than 10, "youth" if x is between 10 and 20 (inclusive) and "adult" if x is greater than 20. The statement should output "error" if x is negative or zero.

Answer:

```
if( x <= 0 )
        cout << "error" << << "\n";
else if ( x < 10 )
        cout << "child" << "\n";
else if ( x <= 20 )
        cout << "youth" << "\n";
else
        cout << "adult" << "\n";
```

b. Display "x is even" if the integer variable x is even and non-zero, if it is odd, display "x is odd" and if it is zero output "x is zero".

Answer:

```
if( x == 0 )
        cout << "x is zero" << "\n";
else if( x % 2 == 0 )
        cout << "x is even" << "\n";
else
        cout << "x is odd" << "\n";
```

c. A passing score is 60 or higher, below 60 is considered failing.

Answer:

```
if (score >= 60)
```

```
            cout << "Passing score.";
else
            cout << "Failing score.";
```

d. Display a message telling whether or not letter follows 'M' in the alphabet

```
if (letter > 'M' && letter <= 'Z')
            cout <<"Yes, it follows the letter 'M' in the alphabet" <<endl;
else
            cout <<"No, it does not" <<endl;
```

e. Use an if - else structure to display a student's course result. The inputs are lab_grade and theory_grade. Both are character variables and can take the values 'p' or 'f'. A student passes the course only if both lab_ grade and theory_grade are equal to 'p'.

Answer:

```
if( lab_grade == 'p' && theory_grade == 'p' )
            cout << "pass" << endl;
else
            cout << "fail" << endl;
```

f. Write a nested if statement to display a message indicating the educational level of a student based on the number of years of schooling (0 is none, 1-6 is elementary, 7-8 is middle school, 9-12 is high school, greater than 12 is college). Display a message to indicate if the data is bad as well.

```
if (level < 0)
            cout <<"Invalid data has beeen supplied" <<endl;
else if (level == 0)
            cout <<"No education level" <<endl;
else if ( level <= 6)
            cout <<"An elementary education level" <<endl;
else if (level <= 8)
            cout <<"A middle school education level" <<endl;
else if (level <= 12)
            cout <<"A high school education level" <<endl;
else
            cout <<"A college education level" <<endl;
```

Q.34. Rewrite the following if, else if, else statement as a switch statement:

a)

```
if ((x=='I') || (x=='i'))
 cout<<"Roman Number 1"<<endl;
else if ((x=='V') || (x=='v'))
 cout<<"Roman Number 2"<<endl;
else if ((x=='X') || (x=='x'))
 cout<<"Roman Number 3"<<endl;
else
 cout<<"Invalid Number"<<endl;
```

b)

```
if (option==1)
            cout<<"The option was 1";
else if (option ==2)
            cout<<The option was 2";
else if (option ==3)
            cout<<The option was 3";
else if (option ==4)
            cout<<The option was 4";
else if (option ==5)
            cout<<The option was 5";
```

Answer:

```
switch (x)
{
 case 'I':
 case 'i':
  cout<<"Roman Number 1"<<endl;
  break;
 case 'V':
 case 'v':
  cout<<"Roman Number 2"<<endl;
  break;
 case 'X':
 case 'x':
  cout<<"Roman Number 3"<<endl;
  break;
 default:
   cout<<"Invalid Number"<<endl;
}
```

Answer:

```
switch (option)
{
 case 1:
        cout<<"The option was 1";
  break;
 case 2:
        cout<<"The option was 2";
        break;
 case 3:
        cout<<"The option was 3";
        break;
 case 4:
        cout<<"The option was 4";
        break;
 case 5:
        cout<<"The option was 5";
        break;
}
```

Q.35. Rewrite the following switch statement as if, else if, else statement:

```
switch (grade)
{
        case 'A' :
        gradePt = 4.0;
        break;
        case 'B' :
        gradePt = 3.0;
        break;
        case 'C' :
        gradePt = 2.0;
        break;
        case 'D' :
        gradePt = 1.0;
        break;
        case 'F' :
        gradePt = 0.0;
        break;
        default :
        cout << "Invalid grade";
```

Answer:

```
if (grade = 'A')
  gradePt = 4.0;
else if (grade = 'B')
  gradePt = 3.0;
else if (grade = 'C')
  gradePt = 2.0;
else if (grade = 'D')
  gradePt = 1.0;
else if (grade = 'F')
  gradePt = 0.0;
else
  cout << "Invalid grade";
```

Q.36. Find the errors in the following codes:

a.

```
if ( a > b );
  a = b;
else
  b = a;
```

b.

```
if ( a > b )
  a = b;
else
  b = a;
```

c.

```
if (a > b)
  a = b
else
```

```
    b = a;
```

d.

```
if ( x != 0 )
  a = a / x;
switch ( number % 2 )
{
  case 0 :
    cout << " The number " << number << " is divisible by 3 " << endl ;
  default
    cout << " The number " << number << " is not divisible by 3 " << endl ;
}
```

Answer:

a. Compilation Error. Semicolon is not allowed after condition.

b. Correct

c. Semicolon missing after the statement a = b

d. Semicolon missing after the statement a = a / x

e. default statement needs a colon (:) – syntax error. case 0 needs a break statement and number % 2 must be number % 3 – logic errors

Multiple Choice

1. **The three programming structures are:**
   a. Sequence, decision, and repetition. b. Process, decision, and alternation.
   c. Sequence, definition, and process. d. Relation, comparison, and process.
2. **Which programming structure executes program statements in order?**
   a. Relation b. Decision c. Sequence d. Repetition
3. **Another term for a computer making a decision is:**
   a. Sequential. b. Selection. c. Repetition. d. Iteration.
4. **Which programming structure makes a comparison?**
   a. Relation b. Decision c. Sequence d. Repetition
5. **An expression that uses a relational operator is known as:**
   a. Condition. b. Decision. c. Series. d. Relation.
6. **Relational operators allow you to _____ numbers.**
   a. Compare b. Add c. Multiply d. Divide
7. **When a relational expression is false, it has the value _____.**
   a. Zero b. one c. Less than 0 d. None
8. **You can use a decision statement to:**
   a. Run a series of statements if a test fails. b. Test a series of conditions.
   c. Test whether a condition is true or false. d. All
9. **Operators that are used to compare operands and decide whether the relation is true or false are called:**
   a. Arithmetic operators. b. Logical operators.
   c. Syntax operators. d. Relational operators
10. **Which of the following statements is the simplest form of a decision structure?**
    a. Select...Case b. If statement c. Try...Catch...Finally d. Nested if
11. **In if statement, false is represented by:**
    a. 0 b. 1 c. 2 d. 3

12. **In if statement, true is represented by:**
    a. 0 b. 1 c. 2 d. 3
13. **Which of the following is used for making two way decision**
    a. if-else b. if c. nested if d. switch
14. **Conditional operator is an alternative of:**
    a. if b. if-else c. nested if d. None
15. **The conditional portion of the If statement can contain:**
    a. Any valid constant.
    b. Any expression that can be evaluated to a Boolean value.
    c. Any valid variable.
    d. Any valid constant or variable.
16. **An if statement within an if statement is called:**
    a. A nested if statement. b. A branched if statement.
    c. A infinite if statement d. None
17. **What does a compound condition use to join two conditions?**
    a. Relational Operator b. Logical Operator
    c. Relational Results d. Logical Results
18. **Which is an example of multiple branches from single expression?**
    a. If statement b. switch statement c. while loop d. All
19. **Which of the following is valid case statement in a switch?**
    a. case 1: b. case x<4: c. case 'ab': d. case 1.5:
20. **A case can be defined as:**
    a. An individual condition to test. b. A set of conditions to test.
    c. A statement that includes an If statement. d. A block of selection code.
21. **The first line in switch block contains:**
    a. The value of the first criterion.
    B. The statement to be executed if the first criterion is true.
    c. The expression to be evaluated.
    d. The statement to be executed if none of the criteria are true.
22. **The Case block ends with:**
    a. End Select. b. End Case. c. break d. Case Else.
23. **Which of the following is equivalent to (p >= q)?**
    a. p < q b. !(p < q) c. p > q d. !p < q
24. **Another term for a conditional operator is:**
    a. Ternary b. Binary c. Byte d. Iteration.
25. **Conditional operator takes:**
    a. Three operands b. Two operands c. One operand d. None
26. **Which operator in C++ is called a ternary operator?**
    a. if b. ++ c. ? d. ( )
27. **All of the following are relational operators EXCEPT:**
    a. != b. ||. c. = d. >=.
28. **Which of the following returns True if A = 25 and B = 35:**
    a. A+B b. A>=B c. A!=B A=B
29. **For A = 4 and B = 4 which evaluates as true?**
    a. A <> B b. A < B c. A > B d. A >= B
30. **What operator is used in a compound condition to join two conditions?**
    a. Relational Operator b. Logical Operator c. Relational Results d. Logical Results
31. **Operators that perform operations on Boolean data are:**
    a. Arithmetic operators. b. Logical operators.
    c. Relational operators. d. Math operators.

32. **Which of the following symbol is used for OR operator?**
a. || b. && c. ! d. None

33. **Which of the following symbol is used for AND operator?**
a. || b. && c. ! d. None

34. **Which of the following symbol is used for NOT operator?**
a. || b. && c. ! d. None

35. **All of the following are logical operators EXCEPT:**
a. && b. || c. ! d. =

36. **Which of the following is not a C++ relational operator?**
a. == b. < c. != d.&&

37. **Which C++ logical expression correctly determines whether the value of beta lies between 0 and 100?**
a. 0 < beta < 100
b. 0 < beta && beta < 100
c. (0 < beta) && (beta < 100)
d. b and c above

38. **If the int variables i, j, and k contain the values 3, 20, and 100, respectively, what is the value of the following logical expression:**

```
j < 4 || j == 5 && i <= k3 < 4 || 20 == 5 && 3 < 100
```

a. 3 b. false c. 20 d. True

39. **If p is a Boolean variable, which of the following logical expressions always has the value false ?**
a. p && p b. p || p c. p && !p d. p || !p

40. **What is output of the following code?**

```
if ((1==1) || (2==2) || (3==3))
  cout<<"Good";
else
   cout<<"Bad";
```

a. Good b. Bad c. Goodbad d. No output

41. **What is output of the following code?**

```
int x=1;
int y=2;
int z=3;
if ((x==y) || (y==z) || (z==2))
  printf("YES");
else
  printf("NO");
```

a. No b. YES c. yes d. No output

42. **What is output of the following code?**

```
if (1==2)
  cout<<"Hello";
else if (2==1)
  cout<<"World";
```

a. No output b. Hello c. World d. Helloworld

43. **After execution of the following code, what will be the value of angle if the input value is 0?**

```
cin >> angle;
if (angle > 5)
   angle = angle + 5;
else if (angle > 2)
   angle = angle + 10;
else
```

```
angle = angle + 15;
```

a. 15 b. 25 c 35 d. 40

44. What is output of the following code?

```
int p, q, r;
p = 10;
q = 3;
r = 2;
if ((p + q) < 14 && (r < q – 3))
  cout << r;
else
  cout << p;
```

a. –2 b. 4 c. 9 d. -4

45. What is output of the following code?

```
if(('A'=='A') && ('B'=='B') && ('C'=='c'))
cout<<"Equal";
else
 cout<<"Not Equal";
```

a. Not Equal b. Equal c. No output d. Both a and b

46. What is output of the following

```
if(9<5)
cout<<"x";
else
  if(5==6)
  cout<<"#";
else
   cout<<"@";
```

a. x b. # c. @ d. None

47. What is the output of the following C++ Code for the input data 25 35 20

```
cin >> a >> b >> c;
if (a > b && a > c)
  m = a;
else if (b > c)
  m = b;
else
  m = c;
cout << m << endl;
```

a. 35 b. 25 c. 34 d. 31

48. What is the output of the following C++ Code for the input data 35 30 36 24

```
cin >> a >> b >> c >> d;
if (a > b || a > c && c != d )
  m = a;
else if (b > d)
  m = b;
else
  m = c;
cout << m << endl;
```

a. 35 b. 45 c. 31 d. 40

49. What is the opposite of the boolean expression: (1 < x && x < 100)

a. x <= 1 || 100 <= x) b. !(1 < x && x < 100)

c. Both a or b d. None

50. What would be the value of x if y is 10 and z is 4?

```
switch (y - z)
{
case 8: x = y + z; break;
case 9: x = y; break;
case 10: x = z; break;
default: x = y * 2;
}
```

a. 20 b. 42 c. 35 d. 45

51. Which of the following has the highest precedence?

a. ++ b. && c. || d. -

Answers

1. a	2. c	3. b	4. b	5. a
6. a	7. a	8. d	9. d	10. c
11. a	12. b	13. a	14. b	15. a
16. a	17. a	18. b	19. a	20. d
21. b	22. c	23. b	24. a	25. b
26. c	27. b	28. c	29. d	30. d
31. b	32. a	33. b	34. c	35. d
36. d	37. d	38. d	39. c	40. a
41. a	42. a	43. a	44. a	45. a
46. c	47. a	48. a	49. c	50. a
51. a				

Fill in the Blanks

1. A statement that is executed when a particular condition is true and ignored otherwise is called _____ statement.
2. The statements that are executed in the same order in which they are written are called _____.
3. A _____ is a group of statements enclosed in braces.
4. A _____ is a pictorial representation of a program.
5. _____ are statement used to control the flow of execution in a program.
6. _____ refers to a structure with one if statement inside another if statement.
7. _____ statements can be used as an alternative to if-else if statement.
8. In switch statement the value of expression must be _____ or _____.
9. The _____ statement switches the control outside the block in which it is used.
10. The purpose of _____ label in switch case statement is the same as that of else in if-else statement.
11. A condition is an expression that is either _____ or _____.
12. _____ statement executes one block of statements if the condition is true and the other if condition is false.
13. _____ statement is the simplest form of decision constructs.
14. The statements in the block of _____ statement are executed if the condition is true otherwise skipped.
15. The _____ label in switch statement is used to provide code go be executed if none of the case label is matched.
16. In nested-if structure, control enters in the inner if only when ____ condition is true.
17. _____ condition is a type of comparison in which more than one conditions are evaluated.

18. _____ statement is best to use when there are many choices available and only one should be executed.
19. Each ____ in the switch statement represents one choice.
20. The _____ statement is used at end of each case label to exit form switch body.
21. If ____ statement is not used in switch structure, all case labels will be executed that come after the matching case block.
22. _____ operator also called ternary operator can be used in the place of simple if-else structure.
23. Ternary operator requires _____ operands.
24. Conditional operator is also called_____ operator.
25. _____ operators are used to compare two values.
26. The relational operator always evaluate to _____ or _____.
27. The relational expression is assigned a value of _____ if it is true.
28. The relational expression is assigned a value of _____ if it is false.
29. If a=1, b=2, the value of the relational expression (a>b) is_____.
30. If a=1, b=2, the value of the relational expression (a<b) is_____.
31. Operators used to combine two or more relational expression to construct compound expression are called _____ operators.
32. && is a logical operator while != is a ____.
33. C++ has three logical operators !, && and _____.
34. _____operator produces the true result only if both the conditions are true.
35. _____ operator produces the true result only if anyone of the conditions are true.
36. Logical operators have _____ precedence than relational operators.

Answers

1. Conditional	2. Sequential statements	3. Compound statement
4. Flow chart	5. Control structures	6. Nested if statement
7. switch	8. integer or character	9. break
10. default	11. true(1), false (0)	12. if-else
13. if	14. if	15. default
16. Outer	17. Logical	18. More
19. case	20. break	21. break
22. Conditional	23. three	24. Ternary
25. Relational	26. true, false	27. 1
28. 0	29. 0	30. 1
31. Logical	32. Logical NOT	33. || (Logical OR)
34. && (Logical AND)	35. || (Logical OR)	36. High

True / False

1. No statement is skipped in execution of sequential statements.
2. A logical operator is used to compare two values.
3. There are six relational operators.
4. ! is a relational operator used in C++ language.
5. >= returns false if value on left side of operator is greater than or equal to value on right side.
6. A statement that uses relational operators to compare two values is called relational expression.
7. All conditions of an If statement can evaluate to a Boolean expression.
8. The only decision structure available to a computer program is IF statement.
9. 'if statement' is used to execute a statement by checking a condition.
10. Every 'else' statement has a condition after it.

11. 'if else' statement is a type of 'if' statement that executes one block of statement when the condition is true and executes no thing if condition is false.
12. The 'else' is attached with the far most 'if' when there are many 'if-else' used.
13. An 'if' statement within another 'else' statement is called nested if statement.
14. In nested if statement, control enters the inner if block if outer if block returns true.
15. Every If statement has two or more conditions.
16. A compound condition has at least three or more conditions joined by a logical operator.
17. Conditional operator is also called ternary operator.
18. Relational operators have less precedence than logical operators.
19. AND operator returns true if both conditions are true.
20. OR operator returns true if any one of the conditions is true.
21. != is used to reverse the results of a condition.
22. Switch is a good choice when many choices are available and at most two should be executed.
23. Missing break statements are a cause of compilation error in switch statement.
24. Default block is executed after each case of switch statement.
25. Switch statement does not work without the presence of 'default' block.
26. An arithmetic expression cannot be used as condition in if statement.
27. The conditional operator is a ternary operator.
28. Logical operators are used to compare values.
29. A switch statement cannot be used within the block of an if statement.
30. The break statement stops the execution of a program for a moment and then resumes.
31. In sequence structure, statements are executed in same order in which they appear in program.
32. A false condition always evaluates to zero.
33. Use of the statement terminator at the end of an if statement causes a syntax error.
34. The switch expression cannot be of float or double type.
35. The integer zero has a boolean value of false. The boolean value true is represented by the integer(s) non-zero integers.
36. Floating point numbers cannot be used as case values for switch statements.
37. The switch structure is used to handle multiway selection.
38. Control structure alters the normal flow of control.
39. Including space between the relational operators create syntax error.
40. Selection structure incorporates decisions in a program.
41. The result of logical expression cannot assigned to an int variable.
42. Every if statement must have a corresponding else.
43. The expression ! (a>0) is true only if a is a negative number.
44. All switch structures can be converted to if/else structures.
45. The controlling value in a switch structure can be an expression.
46. Every compound condition in an it/else structure can be translated into an if/else structure without compound conditions.
47. The controlling value in a switch structure can only be of type integer.
48. If x has the value 3, the expression (x > 3 && x < 10) returns TRUE.

Answers

1. T	2. F	3. T	4. F	5. F	6. T
7. T	8. F	9. T	10. F	11. F	12. F
13. F	14. T	15. F	16. F	17. T	18. T
19. T	20. T	21. F	22. F	23. F	24. F
25. F	26. T	27. T	28. F	29. F	30. T
31. T	32. F	33. F	34. T	35. T	36. T
37. T	38. T	39. F	40. F	41. F	42. F
43. F	44. T	45. T	46. T	47. F	48. F

CHAPTER 6

LOOPING STRUCTURES

Chapter Overview

6.1 Loops

A type of control structure that repeats a statement or set of statements is known as **looping** structure. It is also known as **iterative** or **repetitive** structure. In sequential structure, all statements are executed once. On the other hand, conditional structure may execute or skip a statement on the basis of some given condition. In some situations, it is required to repeat a statement or number of statements for a specified number of times. Looping structures are used for this purpose..There are different types of loops available in C++.

Loops are basically used for two purposes:

- To execute a statement or number of statements for a specified number of times. For example, a user may display his name on screen for 10 times.
- To use a sequence of values. For example, a user may display a set of natural numbers from 1 to 10.

There are three types of loops available in C++. These are as follows:

- while loop
- do-while loop
- for loop

6.1.1 Counter-Controlled Loops

This type of loop depends on the value of a variable known as **counter variable**. The value of counter variable is incremented or decremented each time the body of the loop is executed. This loop terminates when the value of counter variable reaches a particular value. The number of iterations of counter-controlled loop is known in advance. The iterations of this loop depends on the following:

- Initial value of the counter variable
- Terminating condition
- Increment or decrement value

6.1.2 Sentinel-Controlled Loops

This type of loop depends on special value known as **sentinel value**. The loop terminates when the sentinel value is encountered. These types of loops are also known as **conditional loops**. For example, a loop may execute while the value of a variable is not –1. Here –1 is the sentinel value that is used to terminate the loop.

The number of iterations of sentinel-controlled loop is unknown. It depends on the input from the user. Suppose the sentinel value to terminate a loop is –1. If the user enters –1 in first iteration, the loop will execute only once. But if the user enters –1 after entering many other values, the number of iterations will vary accordingly. A sentinel value is commonly used with **while** and **do-while** loops.

6.2 'while' Loop

while loop is the simplest loop of C++ language. This loop executes one or more statements while the given condition remains **true**. It is useful when the number of iterations is not known in advance.

Syntax

The syntax of **while** loop is as follows:

```
while (condition)
        statement;
```

condition The condition is given as a relational expression. The statement is executed only if the given condition is **true**. If the condition is **false**, the statement is never executed.

statement Statement is the instruction that is executed when the condition is **true**. If two or more statements are used, these are given in braces { }. It is called the **body** of the loop.

The syntax for compound statements is as follows:

```
while (condition)
{
        statement 1;
        statement 2;
        :
        :
        statement N;
}
```

Working of 'while' Loop

First of all, the condition is evaluated. If it is **true**, the control enters the body of the loop and executes all statements in the body. After executing the statements, it again moves to the start of the loop and evaluates the condition again. This process continues as long as the condition remains **true**. When the condition becomes **false**, the loop is terminated. **while** loop terminates only when the condition becomes **false**. If the condition remains **true**, the loop never ends. A loop that has no end point is known as an **infinite loop**.

Flowchart

The flowchart of **while** loop is as follows:

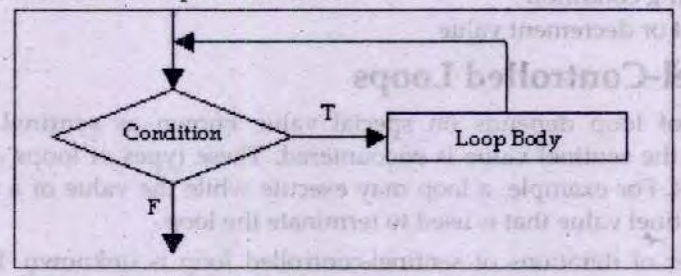


Figure 6.1: Flowchart of while loop

Program 6.1

Write a program that displays "Pakistan" for five times using **while** loop.

```
#include <iostream.h>
#include <conio.h>
void main ()
{
        int n;
        clrscr();
        n = 1;
        clrscr();
        while(n<=5)
        {
                cout<<"Pakistan"<<endl;
                n++;
```

Output:

```
Pakistan
Pakistan
Pakistan
Pakistan
Pakistan
```

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

How above Program Works?

The above program uses a variable **n** to control the iterations of the loop. It is known as **counter variable**. The variable **n** is initialized to 1. When the condition is evaluated for the first time, the value of **n** is less than 5 so the condition is **true**. The control enters the body of the loop. The body of loop contains two statements. The first statement prints **Pakistan** on the screen. The second statement increments the value of **n** by 1 and makes it 2. The control moves back to the condition after executing both statements. This process continues for five times. When the value of **n** becomes 6, the condition becomes **false** and the loop terminates.

Program 6.2

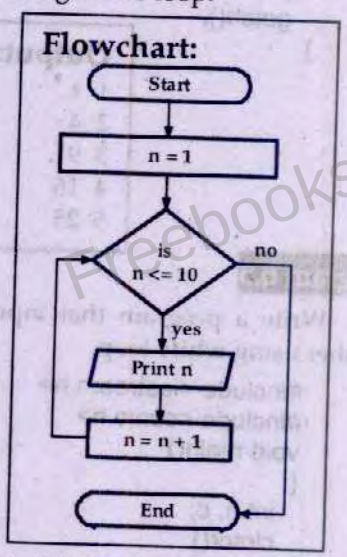
Write a program that displays counting from 1 to 10 using **while** loop.

```
#include <iostream.h>
#include <conio.h>
void main()
{
  int n;
  n = 1;
  clrscr();
  while(n<=10)
  {
    cout<<n<<endl;
    n++;
  }
  getch();
}
```

Output:

```
1
2
3
4
5
6
7
8
9
10
```

Flowchart:



Program 6.3

Write a program that displays first five numbers and their sum using **while** loop.

```
#include <iostream.h>
#include <conio.h>
void main()
{
  int c, sum;
  clrscr();
  c = 1;
  sum = 0;
  while(c <= 5)
  {
    cout<<c<<endl;
    sum = sum + c;
    c = c + 1;
  }
  cout<<"Sum is "<<sum;
  getch();
}
```

Output:

```
1
2
3
4
5
Sum is: 15
```

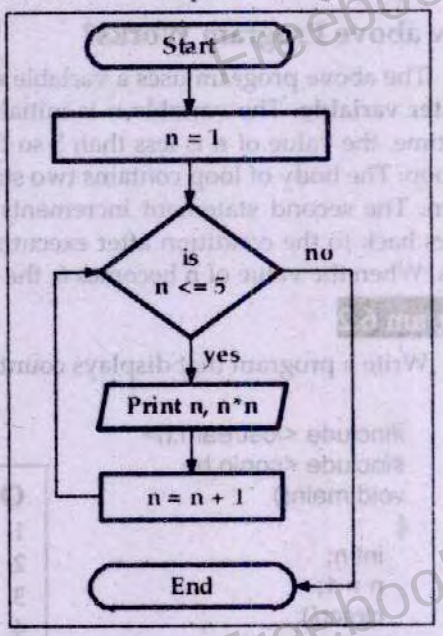
Program 6.4

Write a program that displays first five numbers with their squares using **while** loop.

```
#include <iostream.h>
#include <conio.h>
void main()
{
  int n;
  n = 1;
  clrscr();
  while(n<=5)
  {
    cout<<n<<" "<<n*n<<endl;
    n++;
  }
  getch();
}
```

Output:

```
1 1
2 4
3 9
4 16
5 25
```



Program 6.5

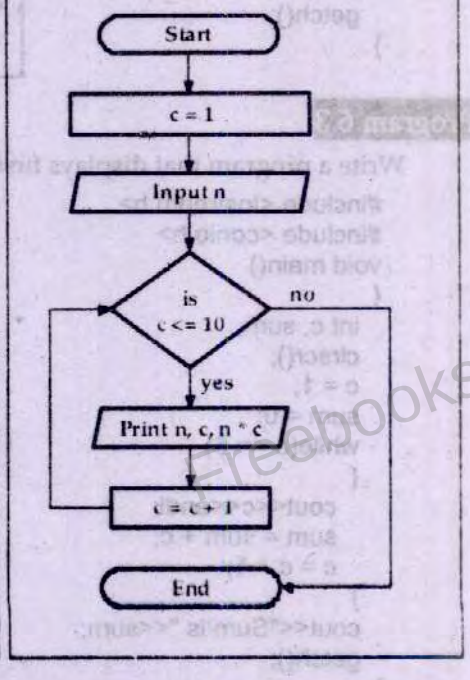
Write a program that inputs a number from the user and displays a table of that number using **while** loop.

```
#include <iostream.h>
#include <conio.h>
void main()
{
  int n, c;
  clrscr();
  c = 1;
  cout<<"Enter a number: ";
  cin>>n;
  while(c <= 10)
  {
    cout<<n<<" * "<<c<<" = "<<n*c<<endl;
    c = c + 1;
  }
  getch();
}
```

Output:

```
Enter a number: 3
3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
.
.
.
3 * 10 = 30
```

Flowchart:



Program 6.6

Write a program that inputs an integer and displays the sum of its digits. For example the program should display 9 if the user enters 135.

```
#include<iostream.h>
#include<conio.h>
void main()
{
  clrscr();
  int x, a, r, sum = 0;
  cout<<"Enter an integer: ";
  cin>>x;
  a = x;
  while (x != 0)
  {
    r = x % 10;
    if (r == 0)
      sum = sum+x;
    else
      sum = sum+r;
    x = x / 10;
  }
  cout<<"The sum of digits of "<<a<<"= "<<sum;
  getch();
}
```

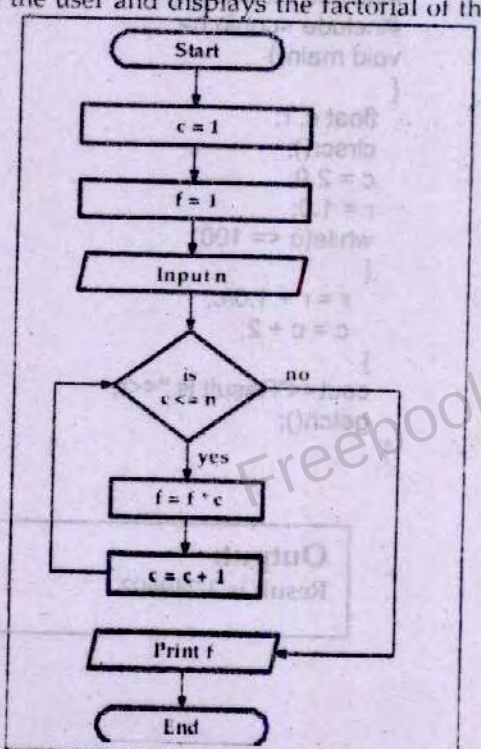
Output:
Enter an integer: 512
The sum of digits of 512 = 8

Program 6.7

Write a program that inputs a number from the user and displays the factorial of that number using **while** loop.

```
#include <iostream.h>
#include <conio.h>
void main()
{
  long int n, c, f;
  clrscr();
  c = 1;
  f = 1;
  cout<<"Enter a number: ";
  cin>>n;
  while(c <= n)
  {
    f = f * c;
    c = c + 1;
  }
  cout<<"Factorial of "<<n<<" is "<<f;
  getch();
}
```

Output:
Enter a number: 8
Factorial of 8 is 40320



Program 6.8

Write a program that displays the degree-to-radian tabel using **while** loop.

```
#include <iostream.h>
#include<conio.h>
#include <iomanip.h>
const double PI = 3.141593;
void main()
{
  clrscr();
  int degrees=0;
  double radians;
  cout.setf(ios::fixed);
  cout.precision(6);
  cout << "Degrees to Radians \n";
  while (degrees <= 360)
  {
    radians = degrees*PI/180;
    cout << setw(6) << degrees << setw(10) << radians << endl;
    degrees += 10;
  }
  getch();
}
```

Output:

```
Degree to Radians
     0          0
    10   0.174533
    20   0.349066
     .
     .
     .
   360   6.283186
```

Program 6.9

Write a program that displays the sum of the following series using **while** loop.

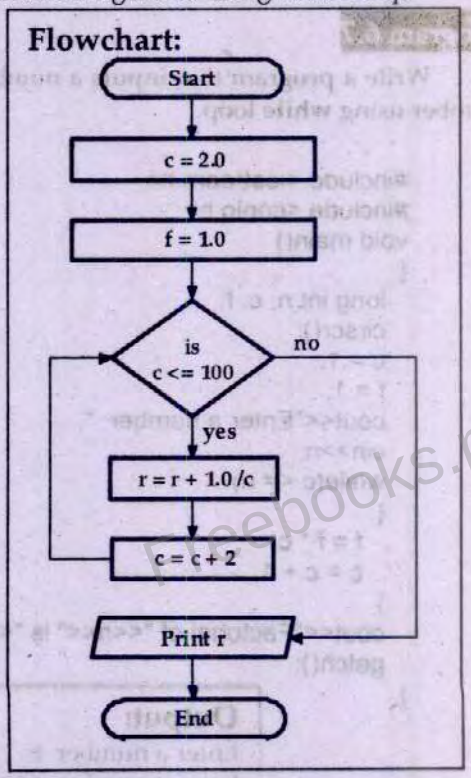
$1 + 1/2 + 1/4 + 1/6 + \ldots + 1/100$

```
#include <iostream.h>
#include <conio.h>
void main()
{
  float c, r;
  clrscr();
  c = 2.0;
  r = 1.0;
  while(c <= 100)
  {
    r = r + 1.0/c;
    c = c + 2;
  }
  cout<<"Result is "<<r;
  getch();
}
```

Output:

Result is 3.249602

Flowchart:



Program 6.10

Write a program that inputs a positive number. It then displays the sum of all odd numbers and the sum of all even numbers from 1 to the number entered by the user.

```
#include<iostream.h>
#include<conio.h>
void main()
{
  int n,oddsum=0,evensum=0;
  clrscr();
  cout<<"Enter a positive number:";
  cin>>n;
  while (n >= 0)
  {
    if(n%2==0)
      evensum=evensum+n;
    else
      oddsum=oddsum+n;
    n--;
  }
  cout<<"The sum of even digits is:" <<evensum<<endl;
  cout<<"The sum of odd digits is:" <<oddsum;
  getch();
}
```

Output:
Enter a positive number: 5
The sum of even digits is: 6
The sum of odd digits is: 9

Program 6.11

Write a program that inputs a number and checks whether it is an Armstrong number or not. A number is an Arstrong number if the sum of the cubes of its digits is equal to the number itself. For example, 371 is an Armstrong number since $3^3 + 7^3 + 1^3 = 371$

```
#include <iostream.h>
#include<conio.h>
void main()
{
  clrscr();
  int num, n, r, sum;
  cout<<"Enter a number:";
  cin>>num;
  n = num;
  sum = 0;
  while(n != 0)
  {
    r = n % 10;
    sum = sum + (r * r * r);
    n /= 10;
  }
  if (sum == num)
    cout<<num<<" is an Armstrong number.";
  else
    cout<<num<<" is not an Armstrong number.";
  getch();
}
```

Output:
Enter a number: 371
371 is an Armstrong number.

Program 6.12

Write a program that inputs numbers until the user enters a negative number. The program calculates the average, maximum and minimum of all positive numbers.

```
#include <iostream.h>
#include<conio.h>
void main()
{
 clrscr();
 float num, sum;
 float avg, min, max;
 int count;
 sum = 0.0;
 count = 0;
 cout<<"Enter positive number ";
 cin >> num;
 min = num;
 max = num;
 while (num >= 0.0)
 {
   sum += num;
   count++;
   if (num > max)
    max = num;
   else if (num < min)
    min = num;
   cout << "Enter positive number ";
   cin >> num;
 }
 if (count == 0)
   cout << "No positive number entered " << endl;
 else
  {
    avg = sum / count;
    cout << "You entered "<<count<<" numbers "<<endl;
    cout << "Average = "<<avg <<endl;
    cout << "Minimum = "<<min <<endl;
    cout << "Maximum = "<<max<<endl;
  }
 getch();
}
```

Output:

```
Enter a positive number: 3
Enter a positive number: 5
Enter a positive number: 7
Enter a positive number: 9
Enter a positive number: 10
Enter a positive number: 9
Enter a positive number: -1
You entered 6 numbers.
Average = 7.166667
Minimum = 3
Maximum = 10
```

Program 6.13

Write a program that inputs a sentence from the user and counts the number of words and characters in the sentence.

```
#include <iostream.h>
#include <conio.h>
void main()
{
 clrscr();
 int countch=0;
 int countwd=1
```

```
    cout << "Enter a sentence: " << endl;
    char ch='a';
    while(ch!='\r')
    {
      ch=getche();
      if(ch==' ')
        countwd++;
      else
        countch++;
    }
    cout<< "\nWords = "<<countwd<<endl;
    cout<<"Characters = "<<countch-1<<endl;
    getch();
}
```

Output:
Enter a sentence:
I love my Pakistan
Words = 4
Characters = 15

Program 6.14

Write a program that inputs starting and ending number from the user and displays all even numbers in the given range using **while** loop.

```
#include <iostream.h>
#include <conio.h>
void main()
{
  int n, s, e;
  clrscr();
  cout<<"Enter starting number: ";
  cin>>s;
  cout<<"Enter ending number: ";
  cin>>e;
  n = s;
  while(n<=e)
  {
    if(n%2==0)
        cout<<n<<endl;
    n++;
  }
  getch();
}
```

Output:
Enter starting number: 1
Enter ending number: 10
2
4
6
8
10

How above Program Works?

The above program inputs two numbers from the user in **s** and **e**. It uses a counter variable **n** to control the iterations of the loop. The value of **s** is assigned to variable **n**. The number of iterations depends on the values entered in **s** and **e**. For example, if the user enters 11 in **s** and 20 in **e**, loop will execute for ten times and program displays 12, 14, 16, 18 and 20.

Program 6.15

Write a program that uses a **while** loop to enter number from the user and then display it. The loop is terminated when the user enters -1.

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

```
    int n;
    n = 1;
    while(n != -1)
    {
      cout<<"Enter a number: ";
      cin>>n;
      cout<<"You entered "<<n<<endl;
    }
    cout<<"End of Program";
    getch();
}
```

Output:
Enter a number: 5
You entered 5
Enter a number: 10
You entered 10
Enter a number: -1
You entered -1
End of Program...

How above Program Works?

The above example uses –1 to terminate the loop. The variable **n** is initialized to 1. First of all, the condition is evaluated and the control enters the body of the loop because the condition is **true**. The first statement in the body gets the input from the user and prints that value on the screen. Then the control moves the beginning of the loop and the condition is evaluated again. This process continues and the value entered by the user is displayed. The loop is terminated when the user enters -1.

Program 6.16

Write a program that inputs a number from user and displays **n** fibonacci terms. In Fibonacci sequence, sum of two successive terms gives the third term.

```
#include<iostream.h>
#include<conio.h>
void main()
{
  int a,b,next,n,count;
  clrscr();
  cout<<"How many Fibonacci terms required:";
  cin>>n;
  a=0;
  b=1;
  cout<<"Fibonacci terms are"<<endl;
  cout<<a<<"\t"<<b;
  count=2;
  while(count<n)
  {
    next=a+b;
    cout<<"\t"<<next;
    count++;
    a=b;
    b=next;
  }
  getch();
}
```

Output:
How many Fibonacci terms required: 6
Fibonacci terms are:
0 1 1 2 3 5

Program 6.17

Write a program that inputs a number and checks if it is a Fibonacci number or not.

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

```
{
 clrscr();
 int a,b,next,n;
 cout<<"Enter the number :";
 cin>>n;
 if ((n==0) || (n==1))
  cout<<n<<" is a Fibonacci number.";
 else
 {
    a=0;
    b=1;
    next=a+b;
 while(next<n)
 {
   a=b;
   b=next;
   next=a+b;
 }
 if (next==n)
   cout<<n<<" is Fibonacci number.";
 else
   cout<<n<<" is not a Fibonacci number.";
 }
 getch();
}
```

Output:
Enter a number: 8
8 is a Fibonacci number.

6.3 'do-while' Loop

do-while is an **iterative** control in C++ language. This loop executes one or more statements while the given condition is **true**. In this loop, the condition comes after the body of the loop. It is an important loop in a situation when the loop body must be executed at least once.

Syntax

The syntax of while loop is as follows:

```
do
{
        statement 1;
        statement 2;
        :
        :
        statement N;
}
while (condition);
```

do — It is the keyword that indicate the beginning of the loop.

Condition — The condition is given as a relational expression. The statement is executed only if the given condition is **true**. If the condition is **false**, the statement is never executed.

Statement — Statement is the instruction that is executed when the condition is **true**. Two or more statements are written in braces {}. It is called the **body** of the loop.

Working of 'do-while' Loop

First of all, the body of loop is executed. After executing the statements in loop body, the condition is evaluated. If it is **true**, the control again enters the body of the loop and executes all statements in the body again. This process continues as long as the condition remains **true**. The loop terminates when the condition becomes **false**. This loop is executed at least once even if the condition is **false** in the beginning.

Flowchart

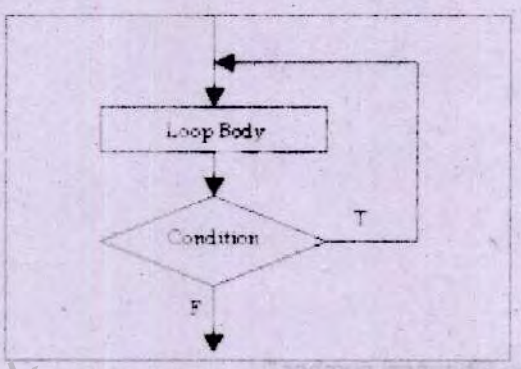


Figure 6.2: Flowchart of do-while loop

6.3.1 Difference between 'while' and 'do-while' Loop

The difference between 'while' and 'do-while' loop is as follows:

while loop	do-while loop
In while loop, condition comes before the body of the loop.	In do-while loop, condition comes after the body of the loop.
If condition is false in the beginning, while loop is never executed.	do-while is executed at least once even if condition is false in the beginning.

Program 6:18

Write a program that displays back-counting from 10 to 1 using **do-while** loop.

```
#include <iostream.h>
#include <conio.h>
void main()
{
  int c;
  clrscr();
  c = 10;
  do
  {
    cout<<c<<endl;
    c = c - 1;
  }
  while(c>=1);
  getch();
}
```

Output:

```
10
9
8
7
6
5
4
3
2
1
```

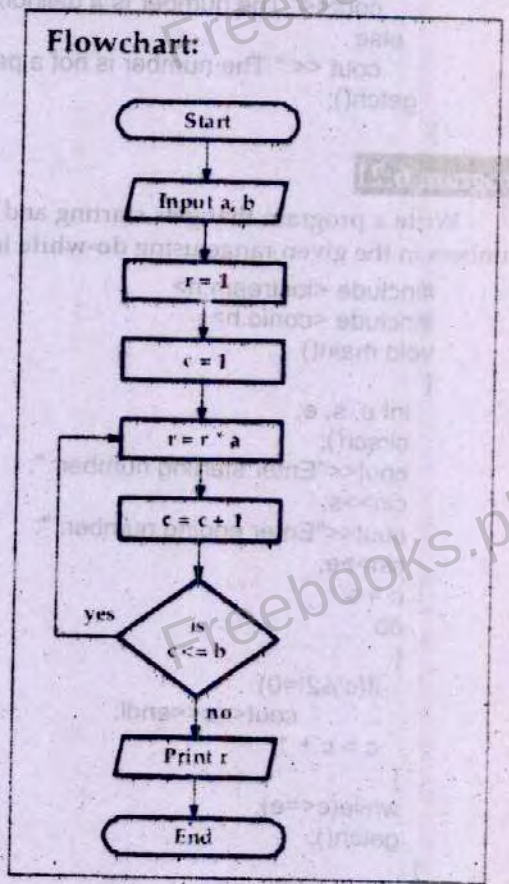
Program 6.19

Write a program that gets two numbers from the user and displays the result of first number raise to the power of second number using **do-while** loop.

```
#include <iostream.h>
#include <conio.h>
void main()
{
  int a, b, c, r;
  clrscr();
  cout<<"Enter first number: ";
  cin>>a;
  cout<<"Enter second number: ";
  cin>>b;
  c = 1;
  r = 1;
  do
  {
    r = r * a;
    c = c + 1;
  }
  while(c<=b);
  cout<<"Result is "<<r;
  getch();
}
```

Output:
Enter first number: 2
Enter second number: 3
Result is 8

Flowchart:



Program 6.20

Write a program that inputs a number and checks whether it is a palindrome or not. A palindrome is a number that reads the same backwards as forwards such as 62526 and 4994.

```
#include<iostream.h>
#include<conio.h>
void main()
{
  clrscr();
  long int n, num, digit, rev = 0;
  cout << "Enter a positive number: ";
  cin >> num;
  n = num;
  do
  {
    digit = num%10;
    rev = (rev*10) + digit;
    num = num/10;
  }
```

Output:
Enter a positive number: 62526
The reverse of the number is 62526
The number is a palindrome

```
    while (num!=0);
      cout<<" The reverse of the number is: " << rev << endl;
    if (n == rev)
      cout<<" The number is a palindrome";
    else
      cout << " The number is not a palindrome";
    getch();
}
```

Program 6.21

Write a program that gets starting and ending point from the user and displays all odd numbers in the given range using **do-while** loop.

```
#include <iostream.h>
#include <conio.h>
void main()
{
  int c, s, e;
  clrscr();
  cout<<"Enter starting number: ";
  cin>>s;
  cout<<"Enter ending number: ";
  cin>>e;
  c = s;
  do
  {
    if(c%2!=0)
          cout<<c<<endl;
    c = c + 1;
  }
  while(c<=e);
  getch();
}
```

Output:

```
Enter starting number: 5
Enter ending number: 15
5
7
9
11
13
15
```

Program 6.22

Write a program that reads the current state of a telephone line. The user should enter **w** for working state and **d** for dead state. Any other input should be invalid. Use do-while loop to force the user to enter a valid input value.

```
#include <iostream.h>
#include <conio.h>
void main()
{
  char s;
  clrscr();
  do
  {
    cout<<"Enter the state of phone (\'w\' for working \'d\' for dead): ";
    cin>>s;
  }
  while(s != 'w' && s != 'd');
  getch();
}
```

Output:

```
Enter the state of phone ('w' for working 'd' for dead): a
Enter the state of phone ('w' for working 'd' for dead): m
Enter the state of phone ('w' for working 'd' for dead): w
Telephone is working.
```

How above Program Works?

The above program does not proceed until the user enters a valid input of d or w. The program repeatedly shows a message to the user to enter a valid input. The condition indicates that the loop continues if the user enters a value other than w and d.

6.4 Pretest and Posttest in Loops

A **pretest** is a condition that is used to test for the completion of loop at the top of loop. In this type of loop, the statements inside the loop body can never execute if the terminating condition is **false** the first time it is tested.

Example

Following is an example of pretest condition. The statements in the body of loop cannot be executed if the condition is **false**.

```
n = 0;
while(n == 0)
{
  loop body
}
```

A **posttest** is a condition that is used to test for the completion of loop at the bottom of loop. It means that the statements in the loop will always be executed at least once.

Example

Following is an example of posttest condition. The statements in the body of loop will be executed at least once even if the condition is **false**.

```
do
{
  loop body
}
while(n == 0)
```

6.5 'for' Loop

for loop executes one or more statements for a specified number of times. This loop is also called **counter-controlled loop**. It is the most flexible loop. That is why, it is the most frequently-used loop by the programmers.

Syntax

The syntax of **for** loop is as follows:

```
for (initialization; condition; increment/decrement)
{
        statement 1;
        statement 2;
        :
        statement N;
}
```

Initialization It specifies the starting value of counter variable. One or many variables can be initialized in this part. To initialize many variables, each variable is separated by comma.

Condition The condition is given as a relational expression. The statement is executed only if the given condition is **true**. If the condition is **false**, the statement is never executed.

Increment/decrement This part of loop specifies the change in counter variable after each execution of the loop. To change many variables, each variable must be separated by comma.

Statement Statement is the instruction that is executed when the condition is **true**. If two or more statements are used, these are given in braces { }. It is called the **body** of the loop.

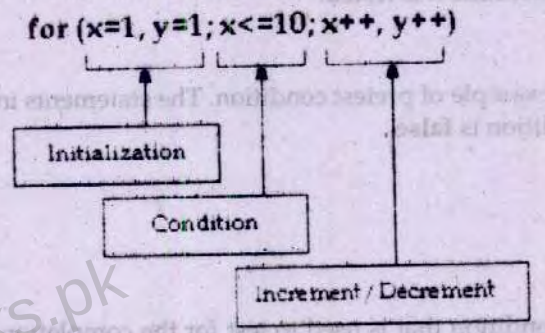


Figure 6.3: Parts of for loop

Working of 'for' Loop

The number of iterations depends on the initialization, condition and increment/decrement parts. The initialization part is executed only once when control enters the loop. After initialization, the given condition is evaluated. If it is **true**, the control enters the body of loop and executes all statements in it. Then the increment/decrement part is executed that changes the value of counter variable. The control again moves to condition part. This process continues while the condition remains **true**. The loop is terminated when the condition becomes **false**.

Flowchart

The flowchart of **for** loop is as follows:

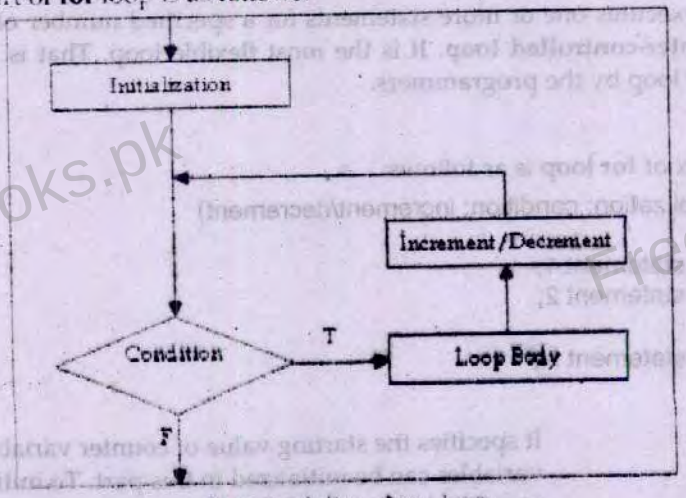


Figure 6.4: For Loop

Program 6.23

Write a program that displays counting from 1 to 5 using **for** loop.

```
#include <iostream.h>
#include <conio.h>
void main()
{
  int n;
  clrscr();
  for(n=1; n<=5; n++)
      cout<<n<<endl;
  getch();
}
```

Output:

```
1
2
3
4
5
```

How above Program Works?

In the above example, the variable **n** is initialized 1. The termination condition is **n <= 10**. It means that the loop will continue while the value of counter is less than or equal to 10. The loop will terminate when the value of counter is greater than 10. The third part contains n++. It indicates that the value of **n** will be incremented by 1 after each execution of the loop.

Program 6.24

Write a program that displays product of all odd numbers from 1 to 10 using **for** loop.

```
#include <iostream.h>
#include <conio.h>
void main()
{
  long int product = 1;
  int c;
  clrscr();
  for(c=1; c<=10; c=c+2)
        product *= c;
  cout<<"Result is "<<product;
  getch();
}
```

Output:

```
Result is: 945
```

Program 6.25

Write a program that inputs table number and length of table and then displays the table using **for** loop.

```
#include <iostream.h>
#include <conio.h>
void main()
{
  int tab, len, c;
  cout<<"Enter number for table: ";
  cin>>tab;
  cout<<"Enter length of table: ";
  cin>>len;
  for(c=1; c<=len; c++)
    cout<<tab<<" * "<<c<<" = "<<tab*c<<endl;
  getch();
}
```

Output:

```
Enter number for table: 2
Enter number for table: 8
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
```

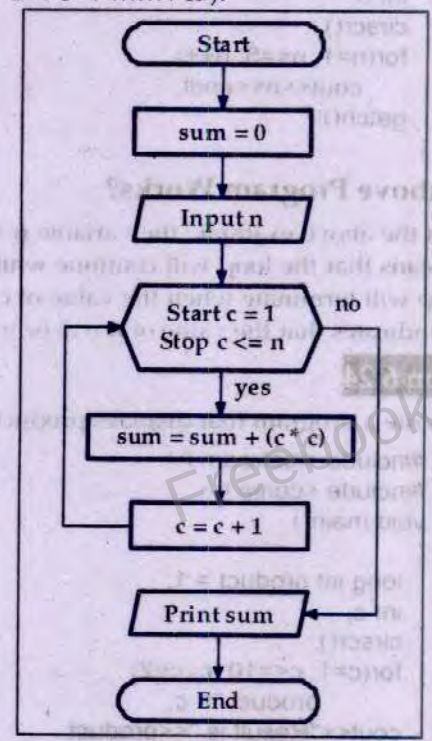
Explanation

The above program uses a variable **tab** for table instead of a constant number. It gets a number from the user as a table. It also inputs a number in variable **len** up to which the table is displayed. It execute the loop for **len** times by using the condition **c <= len**.

Program 6.26

Write a program that finds the sum of the squares of integers from 1 to n. Where n is a positive value entered by the user (i.e. Sum = $1^2 + 2^2 + 3^2 + \ldots\ldots + n^2$).

```
#include <iostream.h>
#include <conio.h>
void main()
{
  int n, c;
  long int sum;
  clrscr();
  sum = 0;
  cout<<"Enter a number: ";
  cin>>n;
  for(c=1; c<=n; c++)
    sum = sum + (c * c);
  cout<<"Sum is "<<sum;
  getch();
}
```



Output:
Enter a number: 5
Sum of squares: 55

Explanation

The above program declares three variables **c**, **n** and **sum**. It inputs a positive number from the user and stores this number in the variable **n**. In the initialization part of loop, **c** is initialized to 1. The condition statement **c<=n** uses **n** that is entered by the user. The loop will execute **n** times. The increment part increments the counter variable by 1.

In each iteration, the statement **sum = sum + (c * c)** is executed. It takes the square of the counter variable **c** and adds it to the variable **sum**. The loop runs **n** times and the squares of numbers from 1 to **n** are summed up. After completing **for** loop the **cout** statement is executed that displays the sum of the squares of number from 1 to **n.**

Program 6.27

Write a program that inputs a number from the user and prints the lowest and highest digit in the number.

```
#include<iostream.h>
#include<conio.h>
void main()
{
  int n;
  clrscr();
```

```
    cout<< "\n Enter a number ";
    cin>> n;
    int num = n;
    int high = n%10, low = n%10, rem;
    n = n/10;
    for(int i = n; i>=1; i=i/10)
    {
      rem =  i%10;
      if (rem>high)
        high = rem;
      if (rem < low)
        low = rem;
    }
    cout<< "\n The highest digit of "<< num <<" is " << high;
    cout<< "\n The lowest digit of "<< num <<" is " << low;
    getch();
}
```

Output:
Enter a number: 4591
The highest digit of 4591 is 1
The lowest digit of 4591 is 9

Program 6.28

Write a program that inputs the value of x and range. It then calculates and prints the sum of the following series:

1 + 1/x + 1/x^2 + 1/x^3 + ...

```
#include <iostream.h>
#include <math.h>
#include<conio.h>
void main ()
{
  clrscr();
  int i,n,x;
  float sum=0,den;
  cout<<"\nEnter the value of x : ";
  cin>>x;
  cout<<"\nEnter the range ; ";
  cin>>n;
  for (i=0;i<n;i++)
  {
    den=pow (x,i);
    sum = sum + ( 1/den);
  }
  cout<<"Sum of series:"<<sum;
  getch();
}
```

Output:
Enter the value of x: 2
Enter the range: 5
Sum of series: 1.9375

Program 6.29

Write a program to print the following series:

1 4 7 10 ... 40

```
# include<iostream.h>
#include<conio.h>
void main()
{
  int a = 1,i , incre = 3;
```

```
    clrscr();
    cout<<"\n The Series is as follows:";
    for( i = 1 ; a<=40; i++)
    {
      cout<< a << " " ;
      a = a+incre;
    }
    getch();
}
```

Program 6.30

Write a program to print the following series:

1 -4 7 -10 ..... -40

```
# include<iostream.h>
#include<conio.h>
void main()
{
  int a = 1, i,p , incre = 3;
  cout<<"\n The Series:";
  for( i = 1 ; a<=40; i++)
  {
    if(i%2 == 0)
    {
      p=-a;
      cout<< p << " " ;
    }
    else
      cout<< a << " ";
    a = a+incre;
  }
  getch();
}
```

Program 6.31

Write a program that inputs a number and checks whether it is a perfect number or not. A perfect number is a number that is numerically equal to the sum of its divisors. For example, 6 is a perfect number because the divisors of 6 are 1, 2, 3 and 1+2+3 = 6.

```
#include <iostream.h>
#include<conio.h>
void main()
{
  clrscr();
  int i, num, mid, sum=0;
  cout<<"Enter the number:";
  cin>>num;
  mid = num / 2;
  for(i=1;i<=mid;i++)
  {
    if ((num%i)==0)
      sum=sum+i;
  }
```

Output:
Enter a number: 6
6 is a perfect number

```
    if (sum==num)
      cout<<num<<"is a perfect number";
    else
      cout<<num<<"is not a perfect number";
    getch();
  }
```

Program 6.32

Write a program that inputs an integer and prints if it is prime or composite. A number is prime if it has factors 1 and itself, otherwise it is a composite number.

```
#include <iostream.h>
#include<conio.h>
void main()
{
  int c, num, p = 1;
  clrscr();
  cout<<"Enter an integer: ";
  cin>>num;
  for(c=2; c<=num/2; c++)
    if(num%c==0)
    {
        p = 0;
        break;
    }
  if(p==1)
    cout<<num<<" is a prime number.";
  else
    cout<<num<<" is a composite number.";
  getch();
}
```

Output:
Enter an integer: 97
97 is a prime number.

6.5.1 'continue' Statement

The **continue** statement is used in the body of the loop. It is used to move the control to the start of the loop body. When this statement is executed in the loop body, the remaining statements of current iteration are not executed. The control directly moves to the next iteration.

Example

```
int x;
for(x=1; x<=5; x++)
{
  cout<<"Hello World! \n";
  continue;
  cout<<"Knowledge is power.";
}
```

Explanation

The above example has two **cout** statements. One statement is before the **continue** statement and one is after **continue** statement. The second statement is never executed. This is because each time **continue** statement is executed, the control moves back to the start of the body. So "Knowledge is power" is never displayed.

Program 6.33

Write a program that inputs a number from the user using **for** loop. It displays the number if it is greater than 0 otherwise it inputs next number using **continue** statement.

```
#include <iostream.h>
#include <conio.h>
void main()
{
  int x, num;
  for(x=1; x<=5; x++)
  {
    cout<<"Enter a number: ";
    cin>>num;
    if(num<=0)
        continue;
    cout<<"You entered "<<num<<endl;
  }
  getch();
}
```

Output:
```
Enter a number: 5
You entered 5
Enter a number: -1
Enter a number: 0
Enter a number: -5
Enter a number: 100
You entered 100
```

Program 6.34

Write a program that displays the sum of the following series:

1+3+5+7+..100

```
#include<iostream.h>
#include<conio.h>
void main()
{
  int sum = 0;
  for(int i=1;i<100;i++)
  {
    if(i % 2 == 0)
      continue;
    sum = sum + i;
  }
  cout<<"The sum is "<<sum;
  getch();
}
```

Output:
```
The sum is 2500
```

6.5.2 'break' Statement

The **break** statement is used in the body of the loop to exit from the loop. When this statement is executed in the loop body, the remaining iterations of the loop are skipped. The control directly moves outside the body and the statement that comes after the body is executed.

Example

```
for(int x=1; x<=5; x++)
{
  cout<<"Questioning\n";
  break;
  cout<<"Gateway to knowledge";
}
cout<<"Bye";
```

Explanation

The above program uses **break** statement in **for** loop. The counter variable x indicates that the loop should execute for five times. But it is executed only once. In first iteration, the cout statement in the loop displays "Questioning" and control moves to **break** statement. This statement moves the control out of the loop. So the message appears only once.

Program 6.35

Write a program that inputs number from the user using **for** loop. If the number is greater than 0, it is displayed and the next number is input. The program exits the loop if the number is 0 or negative using **break** statement.

```
#include <iostream.h>
#include <conio.h>
void main()
{
  int x, num;
  for(x=1; x<=5; x++)
  {
    cout<<"Enter a number: ";
    cin>>num;
    if(num<=0)
        break;
    cout<<"You entered "<<num<<endl;
  }
  getch();
}
```

Output:
Enter a number: 5
You entered 5
Enter a number: 10
You entered: 10
Enter a number: -5

Explanation

The above program uses **for** loop to get five numbers from the user. If the number is greater than 0, it is displayed on the screen. Otherwise, the loop is terminated and the control moves out of the body.

Program 6.36

Write a program that inputs two numbers and displays the greatest common divisor of both numbers.

```
#include <iostream.h>
#include<conio.h>
void main()
{
  int n1, n2, d;
  cout << "Enter the first number: ";
  cin >> n1;
  cout << "Enter the second number: ";
  cin >> n2;
  d = (n1 < n2) ? n1 : n2;
  for ( ; d >= 1; d--)
    if ((n1 % d == 0) && (n2 % d == 0))
      break;
  cout << "GCD of " << n1 << " and " << n2 << " is " << d;
  getch();
}
```

Output:
Enter the first number: 18
Enter the second number: 27
GCD of 18 and 27 is 9

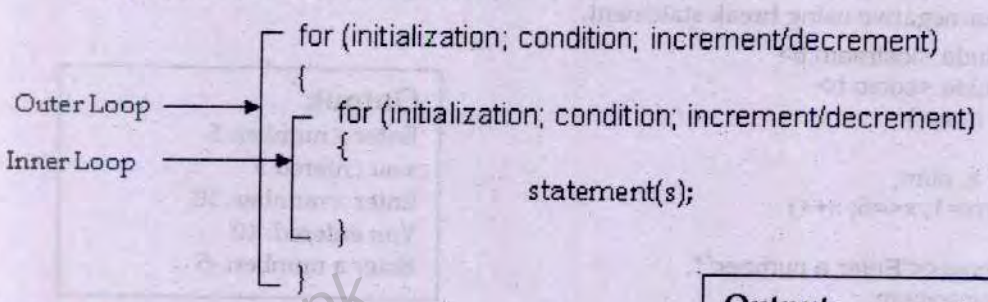
6.6 Nested Loops

A loop within a loop is called **nested loop**. In nested loops, the inner loop is executed completely with each change in the value of counter variable of outer loop.

The nesting can be done up to any level. The increase in level of nesting increases the complexity of **nested loop**. Any loop can be used as inner loop of another loop. For example, **while** loop can be used as outer loop and **for** loop can be used as inner loop in nested loop.

Syntax

The syntax of nested loop is as follows:



Example

```
int i, j;
for(i=1; i<=2; i++)
  for(j=1; j<=3; j++)
      cout<<"Outer: "<<i<<" Inner: "<<j<<endl;
```

Output:

```
Outer: 1 Inner: 1
Outer: 1 Inner: 2
Outer: 1 Inner: 3
Outer: 2 Inner: 1
Outer: 2 Inner: 2
Outer: 2 Inner: 3
```

Explanation

The above example uses nested **for** loop. The outer loop executes two times and the inner loop executes three times with each iteration of outer loop. It means that the inner loop executes six times in total. When the value of **i** is 1 in first iteration of outer loop, the value of **j** changes from 1 to 3 in three iterations of inner loop. Similarly, when the value of **i** is 2 in second iteration of outer loop, the value of **j** again changes from 1 to 3 in three iterations of inner loop.

Example

The following program explains the use of nested **while** loops.

```
int i, j;
i = 1;
while(i<=2)
{
  j = 1;
  while(j<=3)
  {
      cout<<"Outer: "<<i<<" Inner: "<<j<<endl;
      j++;
  }
  i++;
}
```

Output:

```
Outer: 1 Inner: 1
Outer: 1 Inner: 2
Outer: 1 Inner: 3
Outer: 2 Inner: 1
Outer: 2 Inner: 2
Outer: 2 Inner: 3
```

Program 6.37

Write a program that displays the product components of a number wihtout repeating them. For example, if the user enters 24, it displays 24*1, 12*2, 8*3 and 4*6.

```
#include<iostream.h>
#include<conio.h>
void main()
{
  int n,i,j, s = 1;
  cout<<"\n Enter a number:";
  cin>>n;
  clrscr();
  cout<<"\n Product Components of "<< n << " are\n";
  for(i = n; i>= s;i--)
    for(j = 1; j<=n;j++)
    {
      if(i*j == n)
      {
        cout<< i << "*" << j << endl;
        s = j+1;
      }
    }
  getch();
}
```

Output:
Enter a number: 24
Product Components of 24 are:
24*1
12*2
8*3
6*4

Program 6.38

Write a program that displays the following output:

```
#include <iostream.h>
#include<conio.h>
void main()
{
  clrscr();
  int i, j, sum;
  for(i=1; i<=5 ;i++)
  {
    cout<<'1';
    sum = 1;
    for(j=2; j<=i ;j++)
    {
      cout<<'+'<<j;
      sum = sum + j;
    }
    cout<<"="<<sum<<endl;
  }
  getch();
}
```

```
1 = 1
1 + 2 = 3
1 + 2 + 3 = 6
1 + 2 + 3 + 4 = 10
1 + 2 + 3 + 4 + 5 = 15
```

Program 6.39

Write a program that inputs the hieght of triangle and displays a triangle of characters. For example, if the user enters 5, it displays the following:

```
#include<iostream.h>
#include<conio.h>
void main()
{
  char ch='A';
  int n,i,j;
  clrscr();
  cout<< "\n Enter the height of triangle :";
  cin>> n;
  for(i=1; i<= n ; i++)
  {
    for(j= 1;j<=i;j++)
    {
      cout<< ch<< " ";
      ch++;
    }
    cout<< "\n";
  }
  getch();
}
```

```
Enter the height of triangle: 5
A
B C
D E F
G H I J
K L M N O
```

Program 6.40

Write a program that displays the following shape using nested loops. The outer loop should be for loop and inner loop should be **while** loop.

```
#include <iostream.h>
#include <conio.h>
void main()
{
  int i, j;
  clrscr();
  for(i=1; i<=7; i++)
  {
    j = i;
    while( j<=7 )
    {
        cout<<"*";
        j++;
    }
    cout<<"\n";
  }
  getch();
}
```

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

Program 6.41

Write a program that displays the following block using nested **for** loop.

```
#include <iostream.h>
#include <conio.h>
void main()
{
  int m, n;
  clrscr();
```

```
for(m=1; m<=5; m++)
{
  for(n=1; n<=5; n++)
      cout<<"*";
  cout<<endl;
}
getch();
}
```

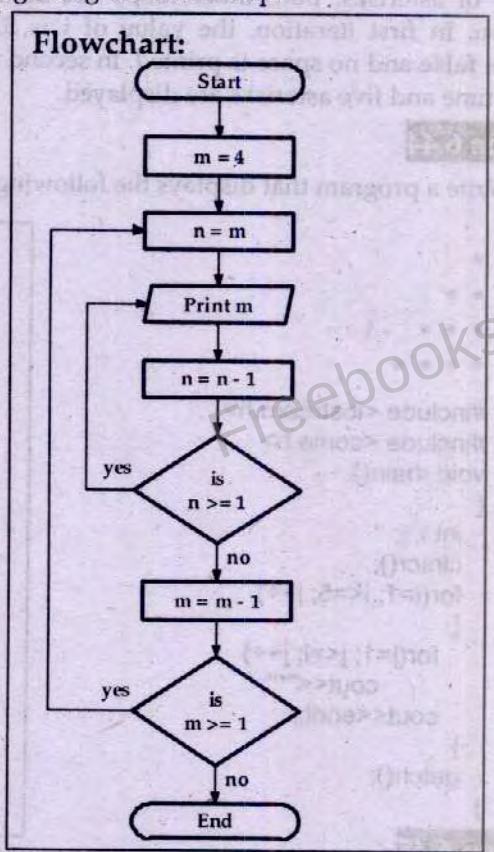
```
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

Program 6.42

Write a program that displays the following using **do-while** loop.

```
4    4    4    4
3    3    3
2    2
1
```

```
#include <iostream.h>
#include <conio.h>
void main()
{
  int m, n;
  clrscr();
  m = 4;
  do
  {
    n = m;
    do
    {
        cout<<m<<"\t";
        n--;
    }
    while(n>=1);
    cout<<endl;
    m--;
  }
  while(m>=1);
  getch();
}
```



Program 6.43

Write a program that displays the following shape using nested **for** loops.

```
#include <iostream.h>
#include <conio.h>
void main()
{
  int i, j, s;
  clrscr();
  for(i=5; i>=1; i--)
  {
    for(s=1; s<=5-i; s++)
        cout<<" ";
```

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

```
        for(j=1; j<=i; j++)
            cout<<"*";
        cout<<endl;
    }
    getch();
}
```

How above Program Works?

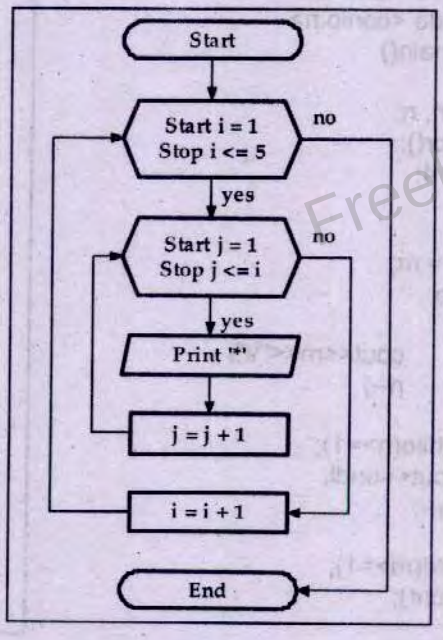
The above program uses three **for** loops. The outer loop runs five times. The first inner loop displays required number of spaces and the second inner loop displays required number of asterisks. Both inner loops use counter variable of outer loop in termination condition. In first iteration, the value of **i** is 5. The condition **s<=5-1** in first inner loop becomes **false** and no space is printed. In second inner loop, the condition **j<=i** executes loop for five time and five asterisks are displayed.

Program 6.44

Write a program that displays the following shape using **nested for** loops.

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

```
#include <iostream.h>
#include <conio.h>
void main()
{
  int i, j;
  clrscr();
  for(i=1; i<=5; i++)
  {
    for(j=1; j<=i; j++)
        cout<<"*";
    cout<<endl;
  }
  getch();
}
```



Program 6.45

Write a program that displays the following output using loops.

```
#include <iostream.h>
#include<conio.h>
void main()
{
  clrscr();
  int i,j;
  for (i = 1; i <= 6; i++)
  {
    for (j = i; j > 1; j--)
      cout << " ";
    for (j = 1; j <= 6 + 1 - i; j++)
      cout << j << " ";
```

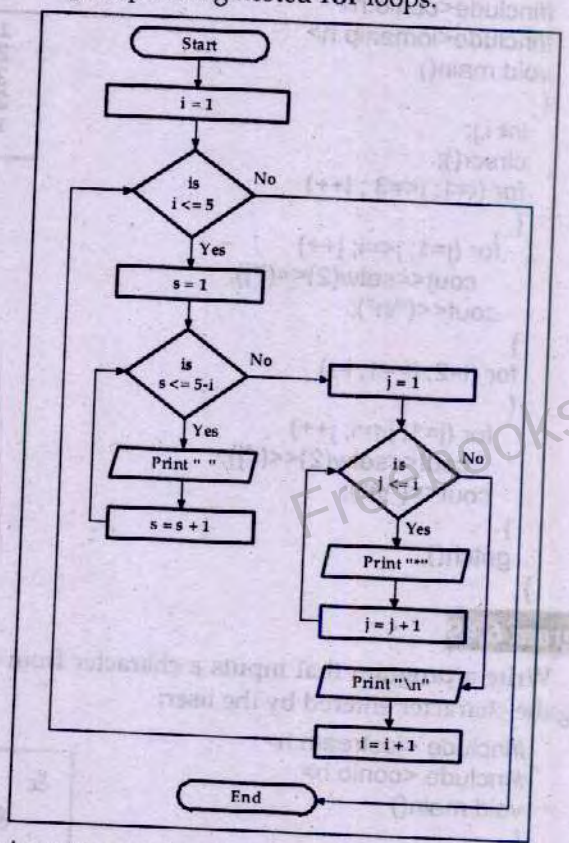
```
1 2 3 4 5 6
  1 2 3 4 5
    1 2 3 4
      1 2 3
        1 2
          1
```

```
    cout << endl;
    }
    getch();
}
```

Program 6.46

Write a program that displays the following shape using nested **for** loops.

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




```
#include <iostream.h>
#include <conio.h>
void main()
{
  int i, j, s;
  clrscr();
  for(i=1; i<=5; i++)
  {
    for(s=1; s<=5-i; s++)
        cout<<" ";
    for(j=1; j<=i; j++)
        cout<<"*";
    cout<<endl;
  }
  getch();
}
```

Program 6.47

Write a program that displays the following output using loops.

```
    1
   2 2
  3 3 3
 4 4 4 4
5 5 5 5 5
```

```
#include<iostream.h>
#include<conio.h>
#include<iomanip.h>
void main()
{
  clrscr();
  int i, j, k, c = 5;
  for (i = 1; i <= 5; i++)
  {
    for (k = 1; k <= c; k++)
      cout<<(" ");
    for (j = 1; j <= i; j++)
      cout<<setw(2)<<i;
    cout<<("\n");
    c--;
  }
```

```
    getch();
}
```

Program 6.48

Write a program that displays the following output:

```
#include <iostream.h>
#include<conio.h>
#include<iomanip.h>
void main()
{
  int i,j;
  clrscr();
  for (i=1; i<=3 ; i++)
  {
    for (j=1; j<=i; j++)
      cout<<setw(2)<<(i*j);
    cout<<("\n");
  }
  for (i=2; i>=1; i--)
  {
    for (j=1; j<=i; j++)
      cout<<setw(2)<<(i*j);
    cout<<("\n");
  }
  getch();
}
```

```
1
2 4
3 6 9
2 4
1
```

Program 6.49

Write a program that inputs a character from the user and draws the following shape using the character entered by the user:

```
#include <iostream.h>
#include <conio.h>
void main()
{
  char choice;
  int x, y, count;
  cout << "\nEnter any character: ";
  cin >> choice;
  count=1;
  for (x=0;x<5;++x)
  {
    cout << endl;
    for (y=0;y<5*2-1;++y)
        if (y==x || y==((5*2-1)-count))
          cout << choice;
        else
          cout << " ";
        ++count;
    }
    getch();
}
```

```
&         &
  &     &
    &  &
      &
    &  &
  &     &
&         &
```

Program 6.50

Write a program that displays all possible combinations of 1, 2 and 3.

```
#include<iostream.h>
#include<conio.h>
void main()
{
  clrscr();
  int i, j, k;
  for (i=1; i<=3; i++)
  {
    for (j=1; j<=3; j++)
    {
      for (k=1; k<=3; k++)
        cout<<i<<j<<k<<"\t";
    }
  }
  getch();
}
```

Output:

```
111  112  113  121  122  123  131  132  133  211
212  213  221  222  223  231  232  233  311  312
313  321  322  323  331  332  333
```

Programming Exercises

1. Write a program to display the following format using **while** loop:

a	b
1	5
2	4
3	3
4	2
5	1

2. Write a program to display the following format using **while** loop:

num	sum
1	1
2	3
3	6
4	10
5	15

3. Write a program that displays the sum of the following series using do-while loop.
   1 + 1/4 + 1/8 + ... + 1/100
4. Write a program to display alphabets from A to Z using **for** loop.

5. Write a program to find the largest, smallest, and average of n whole numbers. You can assume that "n" has already been set by the user.
6. Write a program that will ask the user a question with four possible answers. The question should be asked 20 times. After all the input is gathered, the program should output the number of times each answer was selected.
7. Write a program that inputs a series of 20 numbers and displays the minimum value.
8. Write a program that inputs a number from the user and displays Fibonacci series up to the number entered.
9. Write a program that inputs a number from the user and displays all Armstrong numbers up to the number entered.
10. Write a program that inputs a number from the user and displays all perfect numbers up to the number entered.
11. Write a program that inputs the number of students in the class. It then inputs the marks of these students and displays the highest and second highest marks.
12. Write a program that calculates and prints the average of several integers. Assume that the last value read is sentinel 9999. A typical input sequence might be 10 8 6 7 2 9999 indicating that average of all values preceding 9999 is required.
13. Write a program that inputs a number from the user and displays all prime numbers which are less than the input number using any loop.
14. Write a program that inputs a number from the user and displays its factorial. It asks the user whether he wants to calculate another factorial or not. If the user inputs 1, it again inputs number and calculates factorial. If user inputs 0, program terminates.
15. Write a program that inputs an integer and displays whether it is a prime number or not.
16. Write a program that continuously inputs positive integer values from the user. The user enters a zero to show that he has no more values to enter. The program should finally display the second largest number entered.
17. Write a program that takes n numbers as input. It displays total positive and negative numbers.
18. Write a program to calculate and display sum of the following series using **for** loop:
    $x + x^2 + x^3 \ldots x^n$
19. Write a program to calculate and display sum of the following series using **for** loop:
    1! + 2! + 3! + 4! + 5!
    Where the symbol "!" represents the factorial of the number.
20. Write a program to calculate and display the sum of the following series using for loop:
    $1 + 2x + 3x^2 + 4x^3 + 5x^4$
21. Write a program to calculate and display the sum of the following series using for loop:
    1/2 + 2/3 + 3/4 + ... + 99/100
22. Write a program to print the following sequence:
    64 32 16 8 4 2
23. Write a program to print the following sequence:
    1 3 9 27 81 ... 200
24. Write a program to print the following sequence:
    8 12 17 24 28 33 ... 100
25. Write a program to add the first seven terms of the following series using **for** loop:
    1/1! + 2/2! + 3/3! ...

26. Write a program that sums the sequence of integers assuming that first integer read specifies the number of values remaining to be entered. The program should read one value per input statement. A typical input sequence might be 5 100 200 150 300 500. The first integer 5 indicates that subsequent five values are to be summed.
27. A person invests $1000.00 in a saving account yielding 5% interest. Assuming all interest is left deposit in the account, calculate and print the amount of money in the accounts at the end of each year for ten years. Formula: $(a = p(1+r)^n)$. where
    - p Original amount invested.
    - r Annual Interest rate
    - n Number of years
    - a Amount on deposit at the end of nth years.
28. Write a program to calculate the sum of the first n odd integers.
29. Write a program that could find whether the number entered through keyboard is odd or even and should also tell that whether it is prime or not. The program should keep on taking the value till the user ends and before termination should find the total number of odds, evens and primes entered.
30. Write a loop that will calculate the sum of every third integer, beginning with i = 2 (i.e., calculate the sum 2 + 5 + 8 + 11 + ...) for all values of i that are less than 100. write the loop in each of the following ways:
    (1) Using a for loop
    (2) Using a do while loop
    (3) Using a while loop
31. Write a program to add first nine terms of following series using for and while loop: 1/3! + 5/4! + 9/5! + ... where ! indicates factorial.
32. Write a program to calculate sum of the following series using for and do while loop: 1/3 + 3/5 + 5/7 + ... + 97/99
33. Write a program to generate all possible combinations of 1, 2, 3 and 4.
34. Write a program that inputs the starting and ending numbers and displays all prime number ending with digit 7 between the given range in descending order.
35. Write a program that displays all prime numbers between 100 and 500 that are also palindrome.
36. Write a program to display the following output using nested for loop:

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

37. Write a program to print the following output using loop:

```
        1
      1 2
    1 2 3
  1 2 3 4
1 2 3 4 5
```

38. Write a program to print the following output using loop:

```
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```

39. Write a program to print the following output using loop:

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

40. Write a program that uses nested for loops to display multiplication table as follows:

```
1   2   3   4   5
2   4   6   8   10
3   6   9   12  15
4   8   12  16  20
5   10  15  20  25
```

41. Write a program that uses nested loops to display the following lines

```
1 2 4 6
2 2 4 6
3 2 4 6
4 2 4 6
```

42. Write a program to print the following output using loop:

```
#####*#####
####*#*####
###*###*###
##*#####*##
#*#######*#
*#########*
```

43. Write a program to print the following output using loop

```
BBBBBBBBB
. BBBBBBB
. . BBBBB
. . . BBB
. . . . B
```

44. Write a program that inputs the height of a triangle and displays it using loop. For example, if the user enters height as 5, the program should print the following:

```
& & & & & & & & &
  & & & & & & &
    & & & & &
      & & &
        &
```

45. Write a program that displays a diamond of asterisks using loop.

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

46. Write a program to generate the following pyramid of digits using nested loop:

```
1
232
34543
4567654
567898765
67890109876
7890123210987
890123454321098
90123456765432109
0123456789876543210
```

47. Write a program that inputs the height of triangle and displays a triangle of alphabets. For example, if the user enters 5, it displays the following:

```
A
A B
A B C
A B C D
A B C D E
```

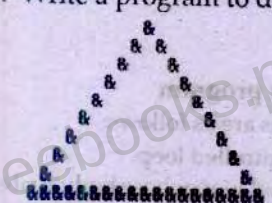
48. Write a program to display the following output using while loop:

```
1
3 5
7 9 11
13 15 17 19
21 23 25 27 29
31 33 35 37 39 41
43 45 47 49 51 53 55
```

49. Write a program to display the following output using loop:

```
1 2 3 4 5 6 7 6 5 4 3 2 1
1 2 3 4 5 6     6 5 4 3 2 1
1 2 3 4 5         5 4 3 2 1
1 2 3 4             4 3 2 1
1 2 3                 3 2 1
1 2                     2 1
1                         1
```

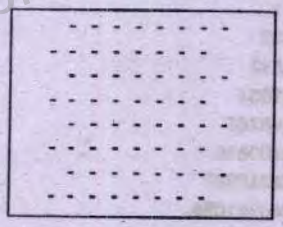
50. Write a program to display the following output using loop:



51. Write a program to display the following output using loop:

```
0
0 1
0 1 2
0 1 2 3
0 1 2 3 4
0 1 2 3 4 5
```

52. Write a program that generates the following checker board by using loop.



Exercise Questions

Q.1. What is loop? What are two uses of loop?

A control structure that executes a statement or number of statements repeatedly is known as loop. Loops can be used to execute a statement or number of statements for a specified number of times. Loops can also be used to access a sequence of values such as 1,2,3 so on.

Q.2. What are the program control statements used to control iterations?

The program control statements used to control iterations are while loop, do-while loop and for loop.

Q.3. What is the part of the loop that contains the statements to be repeated?

The loop body is the part of the loops that contains the statements to be repeated.

Q.4. What are the different phases of loop?

- **Loop Entry:** It occurs when the control reaches the first statement in the loop body.
- **Iteration:** It is execution of the body of the loop.
- **Loop Test:** It is the point at which the test expression is evaluated to decide whether to start iteration or not.
- **Loop Exit:** It is the point at which the last iteration is complete and the control is passed to the first statement after the loop.
- **Exit Condition:** It is the condition that causes the loop to exit.

Q.5. What are the important points for Loop Design

- What is the condition that ends the loop?
- How should the condition be initialized?
- How should the condition be updated?
- What is the process being repeated?
- How should the process be initialized?
- How should the process be updated?
- What is the state of the program on exiting the loop?

Q.6. What are some important points in selecting a Loop for the program

Some important points in selecting a proper loop for the programs are as follows:

- The for loop is the best choice if the loop is a simple counter-controlled loop.
- The do while loop is appropriate if the loop is sentinel-controlled loop whose body is always executed at least once.
- The while and for loops are appropriate if the loop is an event-controlled loop and nothing is known about the first execution.

Q.7. Explain the difference between a pretest and a posttest.

A pretest is a condition that is used to test for the completion of loop at the top of loop. In this type of loop, the statements inside the loop body can never execute if the terminating condition is true the first time it is tested. A posttest is a condition that is used to test for the completion of loop at the bottom of loop. It means that the statements in the loop will always be executed at least once.

Q.8. What is difference between while and do-while loops?

In while loop, condition comes before the body of the loop. In do-while loop, condition comes after the body of the loop. If condition is false in the beginning, while loop is never executed. do-while is executed at least once even if condition is false in the beginning.

Q.9. Differentiate between counter and conditional loops?

Counter loop depends on the value of a variable known as counter variable. The value of counter variable is incremented or decremented each time the body of the loop is executed. This loop terminates when the value of counter variable reaches a particular value.

Conditional loop depends on special value known as sentinel value. Sentinel value indicates that the loop should continue or terminate. For example, a loop may execute while the value of a variable is not –1. Here –1 is the sentinel value used to terminate loop.

Q.10. How does a programmer decide to use a While loop versus a For loop?

For loop is used when the number of iterations are known before the loop is entered. Suppose the user wants to calculate the salary of 100 employees. In this situation, For loop is the best option.

While loop is used when the number of iterations is not known at the beginning of loop. Suppose a loop inputs a number from the user and then displays its square. It then asks the user whether he wants to repeat it or not. In this situation, the iterations are decided by the user. So While loop is better.

Q.11. What is sentinel value?

Sentinel value is a value that is used to terminate a loop. It is used to control loops when the number of repetitions is unknown. A sentinel value is commonly used with while and do while loops.

Q.12. Discuss the difference between break and continue statements used in loops.

When the break statement is executed in a loop, it terminates the loop. When continue statement is executed in a loop, it terminates the current iteration of the loop and the execution moves to the next iteration of the loop.

Q.13. What are the three steps that must be performed using the loop control variable?

The three steps that must be performed using the loop control variable are initialization, test and increment/decrement.

Q.14. What is nested loop?

A loop within a loop is called nested loop. In nested loops, the inner loop is executed completely with each change in the value of counter variable of outer loop. Any loop can be used as inner loop of another loop.

Q.15. What is an infinite loop?

A loop in which the ending condition never occurs is called indefinite loop. It repeats forever until the user intervenes to stop the loop.

Q.16. Describe the syntax of while loop with example.

The syntax of **while** loop is as follows:

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

Example

```
int count = 0;
while (count < 10)
{
  cout<< "C++ programming";
  count++;
}
```

Q.17. Describe the syntax of do-while loop with example.

The syntax of while loop is as follows:

```
do
{
        statement 1;
        statement 2;
        :
        statement N;
}
while (condition);
```

Example

```
int n = 4;
do
{
  n = n * 3;
}
while(n <= 100);
```

Q.18. Describe the syntax of for loop with example.

The syntax of this loop is as follows:

```
for (initialization; condition; increment/decrement)
{
        statement 1;
        statement 2;
        :
        statement N;
}
```

Example

```
int n = 4;
for(int i = n; i >= 0; i--)
  n -= i;
```

Q.19. Trace the output of the following piece of code:

```
int i;
for( i = 10 ; i > 0 ; i -= 1 )
{
  if ( i % 2 != 0 )
    cout<<"i = "<<i<<endl;
  else
    continue ;
}
```

Answer:

```
i = 9
i = 7
i = 5
i = 3
i = 1
```

Q.20. Trace the output of the following piece of code?

```
int j, k ;
for( j = 1 ; j < 6 ; j = j + 1 )
{
  for ( k = 0 ; k < j ; k++ )
  {
    cout<< " + ";
  }
  cout<<endl ;
}
```

Answer:

```
+
++
+++
++++
+++++
```

Q.21. Trace the output of the following piece of code?

```
int i, s = 0 ;
for( i = 10 ; i > 0 ; i -= 1 )
{
  s += i ;
}
cout<<"Sum is = "<<s ;
```

Answer:

Sum is = 55

Q.22. Trace the output of the following piece of code?

```
int i, j = 1 ;
for( i = 10 ; i > 0 ; i -= 1 )
{
  cout<<" \n i = "<<i<<"\t j = "<<j;
  j = j * 2;
}
```

Answer:

```
i = 10    j = 1
i = 9     j = 2
i = 8     j = 4
i = 7     j = 8
i = 6     j = 16
i = 5     j = 32
i = 4     j = 64
i = 3     j = 128
i = 2     j = 256
i = 1     j = 512
```

Q.23. Trace the output of the following piece of code:

```
int i, p = 1 ;
for( i = 1 ; i < 6 ; i += 1 )
{
  p *= 2 ;
}
cout<<"\np is = "<<p ;
```

Answer:

p is = 32

Q.24. Trace the output of the following of code:

```
int i, j;
for(i=1; i<=3; i++)
  for(j=1; j<=4; j++)
    cout<<"\n "<<i<<"\t "<<j;
```

Answer:

1	1
1	2
1	3
1	4
2	1
2	2
2	3
2	4
3	1
3	2
3	3
3	4

Q.25. Trace the output of the following of code:

a)

```
int i,j,k;
for(i=0,j=2,k=1;i<=4;i++)
   cout<<i+j+k;
```

Answer: 34567

b)

```
int i=1;
for(;i<=4;i++)
   cout<<i;
```

Answer: 1234

c)

```
int i;
for(i=1;i<=3;i++)
   cout<<"hi ";
cout<<i;
```

Answer: hi hi hi 4

Q.26. What is the output of the following while loops?

a.

```
int n = 45;
int m = 32;
while(m > 0)
{
   cout << n / m;
   n %= m;
   m /= 2;
}
```

Answer: 101101

b.

```
int x, y;
x = -1;
y = 0;
while(x < 3)
{
      y += 2;
      x += 1;
}
cout << "x = " << x << "\n"<<"y = " << y << endl;
```

Answer:

x = 3
y = 8

c.

```
int x = 1;
while(x != 12)
{
 cout << x << "\n";
 x+=2;
}
```

Answer: infinite loop (1, 3, 5, 7, 9 ...)

d.

```
int m = 3, n = 2;
while ( n <= 74 )
{
      n = 5 * n + m;
      m ++;
      if ( m >= 5 )
            m = m/2;
      cout << n << " " << m << endl;
}
```

Answer:

13	4
69	2
347	3

f.
```
int x = 1;
while ( x < 9 )
{
  x ++;
  if ( x%2 == 1 )
    cout << x << "+";
}
```
Answer: 3+5+7+9+

Q.27. What is the output of the following do-while loops?

a.
```
char c = 'A';
do
{
 cout << c << " ";
 c = c + 2;
} while ( c <= 'I' );
cout << endl;
```
Answer: A C E G I

b.
```
int c = 1;
do
cout << c << " ";
while(c++ <= 4 );
cout << endl;
```
Answer: 1 2 3 4 5

c.
```
int x = 1;
do {
if ( x % 2 == 0)
x -= 2;
else
x++;
cout << x << " + ";
} while (x >= 0 || x == -2);
```
Answer: 2 + 0 + -2 + -4 +

Q.28. What is the output of the following for loops?

a.
```
int c;
  for (c=0; c<0; c--)
    cout<<c<<endl;
```
Answer: No output

b.
```
int c1, c2;
for (c1=1, c2=1;c1<=5;c2++)
  cout<<c1++<<endl;
cout<<--c2 <<endl;
```
Answer:
1
2
3
4
5
5

c.
```
for (int i = 1; i < 30; i += 3)
{
  if (i % 2 == 1)  // If i is odd.
    continue;
  else if (i == 10)
    break;

  cout << i << endl;
}
```
Answer: 4

d.
```
int c;
  for(c=0; c>=0; c++)
    cout<<c<<endl;
```
Answer:
0
1
2
3
... infinite loop

e.

```
int count = 1;
for( ; ; count++)
        if (count < 5)
                cout << count;
        else
                break;
```

Answer: 1234

f.

```
int n = 0;
for(int m = 0; m < 10; m++)
{
  n += m++;
}
cout << n << endl;
```

Answer: 20

Q.29. What is the output of the following nested loops?

a.

```
for (int a = 0; a < 3; a++)
{
        cout << a << " ";
        for (int b = 0; b < 2; b++)
        cout << b << " ";
}
```

Answer: 0 0 1 1 0 1 2 0 1

b.

```
sum = 0;
outerCount = 1;
while (outerCount <= 5)
{
  innerCount = 1;
  while (innerCount <= outerCount)
  {
    sum = sum + innerCount;
    innerCount++;
  }
  outerCount++;
}
cout << sum << endl;
```

Answer: 35

c.

```
for (int outCount = 1 ; outCount < 2; outCount++)
        for (int inCount = 3 ; inCount > 0; inCount--)
                cout << outCount + inCount << endl;
```

Answer:
4
3
2

Q.30. What is the output of the following code if limit = 6?

```
int i, j;
for ( i = limit; i > 0; i-- )
{
  for ( j = 1; j <= i; j++ )
  {
    if ( i == 1 || i == limit || j == 1 || j == i )
    {
      cout << i;
    }
    else
    {
      cout << ' ';
    }
  }
  cout << endl;
}
```

Answer:
666666
5 5
4 4
3 3
2 2
1

Q.31. Consider the following code:

```
int x = 3;
int a = 12;
while(x <= a)
{
        x = x + 4;
        a++;
}
```

a. How many times will the following loop be executed? 4
b. Using the above code, what will the value of x be after execution?19
c. Using the above code, what will the value of a be after execution? 16

Q.32. Rewrite the following loops as do-while loops:

a)

```
int i = 1;
while (i <= 15)
{
  cout << 'a';
  i = i + 1;
}
```

Answer:

```
int i = 1;
do
{
 cout << 'a';
 i = i + 1;
} while ( i <= 15 );
```

b)

```
int c = 1;
while(c <= 10)
{
 if( c < 5 && c != 2 )
    cout << "hello";
  i++;
}
```

Answer:

```
int c = 1;
do
{
 if ( c < 5 && c != 2)
   cout << "hello";
 i++;
} while( c <= 10 );
```

c)

```
long n = 10;
while (n < 100)
{
  cout << 'a';
  n = n + 10;
}
```

Answer:

```
long n = 10;
do
{
 cout << 'a';
 n = n + 10;
} while ( n < 100 );
```

Q.33. Rewrite the following loops as for loops:

a)

```
int c=0;
while (c<10)
{
 cout<<c;
 c=c+2;
}
```

Answer:

```
int c;
for(c=0;c<10;c=c+2)
 cout<<c;
```

b)

```
int c = 1;
while(c <= 10)
{
  if( c < 5 && c != 2 )
    cout <<"hello";
  c++;
}
```

Answer:

```
for ( int c = 1; c <= 10; c++)
  if ( c < 5 && c != 2 )
     cout << "hello";
```

c)
```
long n = 10;
do
{
 cout << 'a';
 n = n + 10;
} while ( n < 100 );
```

Answer:
```
for( long n = 10; n < 100; n = n + 10 )
  cout << 'hello';
```

Q.34. Rewrite following loops as do while loops:
```
for( int i = 3; i <= 39; i += 6 )
{
        cout << i << "\n";
}
```

Answer:
```
int i = 3;
do
{
        cout << i << "\n";
        i += 6;
} while( i <= 39 );
```

Q.35. Rewrite the following loops as while loops:

a.
```
for(i = 0; i < num; i++)
{
  cout << i << "   Enter a value: ";
  cin >> val;
  cout << "\nYou entered " << val << endl;
}
```

Answer 21(a):
```
i = 0;
while( i < num )
{
  cout << i << "   Enter a value: ";
  cin >> val;
  cout << "\nYou entered " << val
<< endl;
  i++;
}
```

b.
```
int i, data, sum ;
sum = 0;
for (i=1 ; i<=10 ; i++)
{
        cout<<"\n Enter an integer : " ;
        cin>> data ;
        sum +=data ;
}
cout<<"\n Sum is : "<< sum<<endl;
```

Answer:
```
int sum =0;
int i = 1;
int data;
while ( i<= 10)
{
  cout<<"\n Enter an integer : ";
  cin>>data ;
  sum += data ;
  i++ ;
}
cout<<"\n The sum is: "<<sum<<endl ;
```

Q.36. What is wrong with the following code?

a.
```
int total, n;
int c = 0;
while (c < 100)
{
  cin >> n;
  total = total + n;
  c =  + 1;
}
```

Answer: total must be initialized to 0.

b.

```
for (int c = 0; c < 100; c++)
{
  cout <<"hello" << endl;
}
cout << c << " lines ";
```

Answer: c cannot be referenced outside the body of the for loop where it is defined.

c.

```
  For ( x = 50, x => 1, --x ) ;
   cout << x << endl ;
```

Answer.

For should be for. The commas (,) should be semicolons (;). The operator => should be >=. There should not be a semicolon at the end of first line. All of these errors are syntax errors.

d.

```
x = 2 ;
sum = 0 ;
while ( x <= 10 )
   sum += x ;
   x-- ;
```

Answer: The body of while loop should be enclosed in braces. x- - should be x++.

e.

```
while ( y > 0 )
{
  cout << y << endl;
  y= y + 1;
}
```

Answer: The variable y should be decremented (i.e., y=y-1;) not incremented (y=y+1;).

f.

```
while ( c <= 5 )
{
  product = product * c ;
  c= c + 1;
```

Answer: After the statement c = c + 1, closing right brace of while body should be added.

g.

This code is meant to calculate sum of all numbers entered at the keyboard, excluding the last entry, which will be –1. (Assume all appropriate prompts, variable declarations, etc.)

```
cin >> next;
while( next > 0 )
{
        sum = 0;
        sum += next;
        cin >> next;
}
```

Answer: "sum" should be initialized OUTSIDE while loop

Q.37. Write a C++ statement or a set of statements to accomplish each of the following:

a) Sum the odd integers between 1 and 99 using a for structure.

Answer:

```
Sum = 0 ;
for ( count = 1 ; count <= 99 ; count += 2 )
  Sum += count ;
```

b) Print integers from 1 to 20 using a **while** loop and counter variable x. Assume that the variable x has been declared, but not initialized. Print only 5 integers per line. Hint: Use the calculation x % 5. When the value of this is 0, print a new line character (endl), otherwise print a tab character ('\t').

Answer:

```
int x = 1 ;
while ( x <=20)
{
  cout << x ;
  if ( x % 5 == 0 )
    cout << endl ;
  else
    cout << '\t' ;
  x++ ;
}
```

c) Output all numbers between 1 and 20 inclusive except 10. Print each number on a new line.

```
for(int i = 1; i <= 20; i++ )
{
  if( i != 10 )
    cout << i << "\n";
}
```

d) Output all odd numbers between 0 and 100, printing 5 on a line with tabs as follows:

```
1       3       5       7       9
11      13      15      17      19 ..... etc.
for(int i = 1; i < 100; i += 2 )
{
  cout << i;
  if( (i+1) % 10 == 0 )
    cout << "\n";
  else
    cout << "\t";
}
cout << "\n";
```

Multiple Choice

1. **This is a control structure that causes a statement or group of statements to repeat:**
   a. Decision statement. b. Loop c. Sequential d. None
2. **One execution of a loop is known as a(n):**
   a. Cycle. b. Duration. c. Iteration. d. Test
3. **How many types of loop structure are available in C++?**
   a. 4 b. 3 c. 2 d. 6
4. **Which of the following loop is available in C language?**
   a. while b. do-while c. for d. All
5. **A counter can be defined as:**
   a. The final value of a loop b. A variable that counts loop iterations
   c. The initial value of a loop. d. The step value of a loop.
6. **Where should be the loop control variable initialized?**
   a. Before the loop starts b. In the first statement of the loop body
   c. In the last statement of the loop body d. Anywhere in the loop body

7. **Which loop structure always executes at least once?**
   a. do-while b. for c. while d. None
8. **In which loop the condition comes before the body of the loop?**
   a. while loop b. do-while loop c. for loop d. b and c
9. **In which loop the condition comes after the body of the loop?**
   a. while loop b. do-while loop c. for loop d. b and c
10. **Which of the following loop is called counter loop?**
    a. for b. while c. do-while d. None
11. **This loop is a good choice when you know how many times you want the loop to iterate in advance of entering the loop?**
    a. while b. do-while c. for d. nested
12. **This statement causes a loop to terminate early?**
    a. break b. terminate c. exit d. Both a and b
13. **A loop within a loop is called:**
    a. Nested loop b. Complex loop c. Infinite loop d. None
14. **The loop which never ends is called:**
    a. Infinite loop b. Running loop c. Continuous loop d. Nested loop
15. **Which statement is used to move the control to the start of loop body?**
    a. continue b. break c. switch d. None
16. **A special value that marks the end of a list of input data is called:**
    a. Terminal value b. sentinel value c. loop control value d. input value
17. **A for statement contains three expressions: initialization, test, and**
    a. Null b. validation c. increment/decrement d. None
18. **In a for statement, this expression is executed only once:**
    a. test b. validation c. initialization d. None
19. **This statement may be used to stop a loop's current iteration and begin next one:**
    a. continue b. break c. terminate d. None
20. **This means to increase a value by one:**
    a. Modulus b. increment c. decrement d. None
21. **If you want a user to enter exactly 20 values, which loop would be the best to use?**
    a. while b. do-while c. for d. infinite
22. **while loop is also called:**
    a. Conditional loop b. wend loop c. Counter loop d. None
23. **Semicolon is placed at the end of condition in:**
    a. while loop b. do-while loop c. for loop d. All
24. **Which is a loop statement?**
    a. if b. if-else c. switch d. None
25. **With respect to the loop in the following main function, what is missing?**

```
void main()
{
  int loopCount;
  while (loopCount <= 5)
  {
        cout << "Hi";
        loopCount++;
  }
  return 0;
}
```

a. The initialization of the loop control variable

b. The testing of the loop control variable
c. The incrementation of the loop control variable
d. Nothing is missing.

26. What is the output of the following code?

```
int n = 4;
do
{
  n = n * 3;
}
while(n <= 100);
```

a. 108 b. 110 c. 107 d. 100

27. What is the output of the following code?

```
int n = 8;
for(int i=1; i <= n*3; i++)
  n++;
```

a. infinite loop b. 9 c. 12 d. 16

28. What is the output of the following code?

```
int n = 4;
for(int i = n; i >= 0; i--)
  n -= i;
```

a. -6 b. 5 c. -7 d. -5

29. What is the output of the following code?

```
int n = 5;
while ( n <= 20 )
  if ( n%2 )
        n = n-1;
  else
        n = n+3;
```

a. 21 b. 20 c. 19 d. 22

30. What is the output of the following code?

```
int x = -1, y = 0;
while(x <= 3)
{
  y = y + 2;
  x = x + 1;
}
cout << y;
```

a. 10 b. 11 c. 9 d. 8

31. What is the output of the following code?

```
int n = 0;
for(int m = 0; m < 10; m++){
  n += m++;
}
cout << n << endl;
```

a. 20 b. 21 c. 22 d. 18

32. How many times the following code prints "OPP Using C++"?

```
int count = 0;
while (count < 10)
{
  cout << " OPP Using C++";
  count++;
```

```
}
```

a.9 b. 10 c. 0 d. 11

33. How many times the following code prints " OOP Using C++"?

```
int count = 0;
while (count++ < 10)
{
  cout << " OPP Using C++";
}
```

a. 12 b. 10 c. 9 d.0

34. How many times the following code prints " OOP Using C++"?

```
int count = 0;
do
{
  cout << " OPP Using C++";
} while (count++ < 10);
```

a. 11 b. 12 c. 9 d. 13

35. What is the value of variable s after the following loop terminates?

```
int s = 0;
int n = 0;
do
{
  n++;
  s+= n;
  if (s > 4) break;
}
while (n < 5);
```

a. 6 b. 5 c. –6 d. 0

36. What is the value of variable s after the following loop terminates?

```
int s = 0;
int n = 0;
do
{
  n++;
  s += n;
  if (s >= 4) continue;
}
while (n < 5);
```

a. 15 b. 14 c. 16 d. 18

Answers

1. c	2. c	3. b	4. d	5. b	6. a
7. a	8. a	9. b	10. a	11. c	12. a
13. a	14. a	15. a	16. b	17. c	18. c
19. a	20. b	21. c	22. a	23. b	24. d
25. a	26. a	27. a	28. a	29. a	30. a
31. a	32. b	33. b	34. a	35. a	36. a

Fill in the Blanks

1. There are _____ types of loop in C++.
2. The loop condition controls the loop_____.
3. Repetition of statements in a program is called _____.
4. The structure that repeats the statement(s) is known as _____ structure.
5. Three types of loop structures in C++ are for loop, while loop and _____ loop.
6. A variable whose value controls the number of iterations is known as _____.
7. The compound statements that is enclosed in braces are called_____
8. In _____ loop, the loop control variable is always initialized out of body of loop and is incremented or decremented inside the loop body.
9. _____ loop executes at least once.
10. In _____ loop, condition is checked before the execution of loop body.
11. A loop that never ends is known as _____ loop.
12. In _____ loop, first the body of the loop is executed and then the test condition is checked.
13. _____ loop is also called counter-controlled loop.
14. for loop consists of three parts: initialization, _____ and increment/decrement.
15. The three expressions in _____loop statements are separated by semicolons.
16. In for loop, _____ part is executed only once when the controls enters the loop.
17. If value of conditional expression in for loop is non-zero, the body of loop is _____.
18. A loop within a loop is called _____ loop.
19. In nested loop, the inner loop is executed completely with each change in the value of _____ variable of outer loop.
20. _____ statement moves control directly to next iteration wherever it is used in loop body.
21. One or more _____ loops can be placed in the body of while or do-while loop.
22. There is no restriction on type of loops that may be placed in the body of other _____.
23. This type of loop depends on special value known as _____.
24. _____ is an end marker that follows the last item in the list of items.
25. There are _____ expressions in for loop statement.

Answers

1. Three	2. iteration	3. Loop
4. iterative	5. do-while	6. loop control variable
7. loop body	8. while	9. do while
10. while	11. Infinite	12. do-while
13. for	14. Condition	15. for
16. Initialization	17. Executed	18. Nested
19. Counter	20. Continue	21. for
22. loops	23. sentinel value	24. sentinel value
25. three		

True / False

1. The number of statements that are executed repeatedly is called loop.
2. A repeating structure in C++ is called iterative structure.
3. There are three looping structures in C++.
4. In counting loops, counter must be initialized to zero before execution of the loop body begins.
5. Loop counter variables are usually of type double.

6. A counter is a variable that keeps a running total of a variable of interest such as the total of all the individual sales.
7. Each for loop has an initial, final and step value.
8. while loop executes the loop body even before checking the condition.
9. Loop body is executed at least once when do-while loop is used.
10. do-while is an iterative control in C++ language in which the body of while loop may not execute even once.
11. The body of for statement may not be executed at all.
12. The while loop is surely executed if the condition is false in the beginning.
13. Loop repetition condition of while or for statement can be false before loop begins to execute.
14. The loop repetition condition of for statement is tested at the end of each pass.
15. Counter-controlled loop never executes statements for a specified number of times.
16. The body of while statement must cause the loop repetition condition to become false after a finite number of passes to prevent an infinite loop.
17. The number of iterations in for loop depends on the initialization, condition and increment/decrement parts.
18. Initialization part is optional in for loop.
19. A semi-colon should be used in for loop if increment/decrement part is not given.
20. The continue statement is used in the body of loop for loop to move the control to the start of loop body.
21. The remaining statements of current iteration are not executed when 'start' statement is used in the loop body.
22. The remaining iterations of loop are skipped when 'break' statement is used in the loop body.
23. The count variable of an outer loop can be used as final value of an inner loop.
24. Only a loop of the same type can be used as an inner loop.
25. There is no difference between while and do-while loop.
26. The body of the while loop may or may not execute.
27. var++ is an example of prefix increment.
28. The condition of infinite loop never becomes true.
29. The initialization expression is optional in for loop.
30. for (m=1; m<=10 ; m++) is an infinite loop.
31. A loop is a decision making construct.
32. A while loop cannot be used in the body of for loop.
33. In type casting, a variable of one type behaves as a variable of another type temporarily.
34. The while structure and for structure are basically the same.
35. Loops are used when we need our program to make a choice between two or more things.
36. In C++, each statement must be on a separate line.

Answers

1. T	2. T	3. T	4. F	5. F	6. F
7. T	8. F	9. T	10. T	11. T	12. T
13. F	14. T	15. F	16. F	17. T	18. T
19. T	20. T	21. T	22. F	23. F	24. T
25. F	26. F	27. T	28. F	29. T	30. T
31. F	32. F	33. T	34. T	35. F	36. F