**Condition or Logical expression**

In C++ a Condition or a Logical expression compares two values using relational operators. Relational operators supported by C++ are >, >=, <, <=, == and !=. Either two integer values or two floating point values or two characters can be compared using relational operators. Two characters are compared by comparing the ASCII codes of two the characters. Two strings cannot be compared using relational operators. String comparison will be discussed later. In C++ if the condition is **TRUE** => logical expression has value 1 and if the condition is **FALSE** => logical expression has value 0. List of relational operators are given below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operator** | **Meaning** | **Condition** | **Result** | **Meaning** |
| > | Greater than | 20>10 | 1 | TRUE |
| 10>20 | 0 | FALSE |
| 2.5 > 13.5 | 0 | FALSE |
| 'T' > 'B' | 1 | TRUE |
| >= | Greater than equal to | 20>=10 | 1 | TRUE |
| 20>=20 | 1 | TRUE |
| 20>=40 | 0 | FALSE |
| 13.5 >= 10.25 | 1 | TRUE |
| 'A' >= 'f' | 0 | FALSE |
| < | Less than | 10<20 | 1 | TRUE |
| 20<10 | 0 | FALSE |
| 2.5 < 13.5 | 1 | TRUE |
| 'T' < 'B' | 0 | FALSE |
| <= | Less than equal to | 10<=20 | 1 | TRUE |
| 10<=10 | 1 | TRUE |
| 40<=10 | 0 | FALSE |
| 13.5 <= 10.25 | 0 | FALSE |
| 'A' <= 'f' | 1 | TRUE |
| == | Equal to | 40==40 | 1 | TRUE |
| 50==40 | 0 | FALSE |
| 'B' == 'B' | 1 | TRUE |
| 2.5 == 13.5 | 0 | FALSE |
| != | Not equal to | 30!=10 | 1 | TRUE |
| 40!=40 | 0 | FALSE |
| 'B' != 'B' | 0 | FALSE |
| 13.5 != 10.25 | 1 | TRUE |

**if-else**

In C++ condition or logical expression is used with **if-else**. **if-else** statement provides a way to change program flow based on a condition. We can have **if** statement without **else** but we cannot have **else** without **if**.

Rule1: **if** (condition)

statement1 / block1

**else**

statement2 / block2

Rule2: **if** (condition)

statement / block

1. If the condition is **TRUE** then the statement1 or block1 is executed and the statement or the block after the **else** is ignored.
2. If the condition is **FALSE** then the statement or block after the condition is ignored and the statement2 or block2 is executed.
3. If there is no **else,** then statement immediately after **if** is executed.

Usage of **if-else**

#include<iostream.h>

**void** main()

{

**double** marks;

cout<<"Input marks[0-100]? "; cin>>marks;

**if** (marks>=40)

cout<<"Pass"<<endl;

**else**

cout<<"Fail"<<endl;

}

Running of the program

Input marks[0-100]? 85

Pass

**Explanation of output**: Inputted marks is 85, that is, variable marks has a value 85. **if** condition is tested (marks>=40), condition is **TRUE**. Therefore cout<<"Pass"; is executed and the statement after **else**, cout<<"Fail"; is ignored.

Running of the program

Input marks[0-100]? 35

Fail

**Explanation of output**: Inputted marks is 35, that is, variable marks has a value 35. **if** condition is tested (marks>=40), condition is **FALSE**. Therefore cout<<"Pass"; is ignored and the statement after **else**, cout<<"Fail"; is executed.

Usage of **if** without **else**

#include<iostream.h>

**void** main()

{

**double** marks;

cout<<"Input marks[0-100]? "; cin>>marks;

**if** (marks>=40)

cout<<"Pass";

**if** (marks<40)

cout<<"Fail";

}

Running of the program

Input marks[0-100]? 73

Pass

**Explanation of output**: Inputted marks is 73. Condition marks>=40 is **TRUE**. cout<<"Pass"; is executed. Condition marks<40 is **FALSE**. cout<<"Fail"; is ignored.

Running of the program

Input marks out of 100? 37

Fail

**Explanation of output**: Inputted marks is 37. Condition marks>=40 is **FALSE**. cout<<"Pass"; is ignored. Condition marks<40 is **TRUE**. cout<<"Fail"; is executed.

Programs using **if**-**else** statement are given below:

1. Write a complete C++ program to input two integer values and display the largest value on the screen.

|  |
| --- |
| #include<iostream.h>  **void** main()  {  **int** x, y, max;  cout<<"Input 1st integer value? "; cin>>x;  cout<<"Input 2nd integer value? "; cin>>y;  **if** (x>y)  max=x;  **else**  max=y;  cout<<"Max="<<max<<endl;  } |

2. Write a complete C++ program to input 3 coefficient of a quadratic equation (ax2+bx+c=0); calculates two roots of the quadratic equation. Display two real roots on the screen, otherwise display an error message on the screen.

|  |
| --- |
| #include<iostream.h>  #include<math.h>  **void** main()  {  **double** a, b, c;  cout<<"Coefficient of x^2? "; cin>>a;  cout<<"Coefficient of x ? "; cin>>b;  cout<<"Constant Term ? "; cin>>c;  **double** disc=b\*b-4\*a\*c;  **if** (disc>=0)  {  **double** x1=(-b+sqrt(d))/(2\*a);  **double** x2=(-b-sqrt(d))/(2\*a);  cout<<"Two real root are "<<x1<<" and "<<x2<<endl;  }  **else**  cout<<"Complex roots"<<endl;  } |

3. Write a complete C++ program to input two integers; swap the two values and display the output on the screen.

|  |
| --- |
| #include<iostream.h>  **void** main()  {  **int** x, y;  cout<<"Input 1st integer value? "; cin>>x;  cout<<"Input 2nd integer value? "; cin>>y;  **if** (x>y)  {  **int** t=x;  x=y;  y=t;  }  cout<<x<<','<<y<<endl;  } |

4. Write a complete C++ program to input four integer values and display the largest value on the screen.

|  |
| --- |
| #include<iostream.h>  **void** main()  {  **int** x1, x2, x3, x4;  cout<<"Input 1st integer value? "; cin>>x1;  cout<<"Input 2nd integer value? "; cin>>x2;  cout<<"Input 3rd integer value? "; cin>>x3;  cout<<"Input 4th integer value? "; cin>>x4;  **int** max=x1;  **if** (x2>max)  max=x2;  **if** (x3>max)  max=x3;  **if** (x4>max)  max=x4;  cout<<"Max="<<max<<endl;  } |

**&&** **Operator**

Consider the program segment given below:

**double** marks;

cout<<"Input marks[0-100]? "; cin>>marks;

cout<<"Inputted marks="<<marks;

It is expected that a user will input marks between 0 and 100. But if a user inputs either -20 or 150, inputted marks will be stored in variable marks. So how to ensure that marks inputted between 0 and 100 is to be accepted only and inputted marks either less than 0 or more than 100 is to be ignored. So we have to combine two conditions, marks>=0 and marks<=100. This can be done by using && operator. && is used to combine two or more conditions (sub-conditions) as one condition. **All** the sub-conditions have to be **TRUE** for the entire condition to be **TRUE**.

Rule: **if** (Condition1 && Condition2 [&& Condition3 … ])

Statement1 / Block1

**else**

Statement2 / Block2

Truth tables for && operator are given below:

|  |  |  |
| --- | --- | --- |
| Cond1 | Cond2 | Cond1 && Cond2 |
| FALSE | FALSE | FALSE |
| FALSE | TRUE | FALSE |
| TRUE | FALSE | FALSE |
| TRUE | TRUE | TRUE |

|  |  |  |  |
| --- | --- | --- | --- |
| Cond1 | Cond2 | Cond3 | Cond1 && Cond2 && Cond3 |
| FALSE | FALSE | FALSE | FALSE |
| FALSE | FALSE | TRUE | FALSE |
| FALSE | TRUE | FALSE | FALSE |
| FALSE | TRUE | TRUE | FALSE |
| TRUE | FALSE | FALSE | FALSE |
| TRUE | FALSE | TRUE | FALSE |
| TRUE | TRUE | FALSE | FALSE |
| TRUE | TRUE | TRUE | TRUE |

Usage of && operator with **if–else** statement

C++ program to validate inputted marks (marks out of 100)

#include<iostream.h>

**void** main()

{

**double** m;

cout<<"Input marks[0-100]? "; cin>>m;

**if** (m>=0 && m<=100)

cout<<"Marks="<<m;

**else**

cout<<"Input Error";

}

**Explanation of output**

Two sub-conditions are m>=0 and m<=100. **First run**: Marks 78; m>=0 and m<=100 are **TRUE** and therefore **if** condition is **TRUE**, cout<<"Marks="<<m; is executed. **Second run**: Marks -35; m>=0 is **FALSE** but m<=100 is **TRUE** and therefore **if** condition is **FALSE**, cout<<"Input error"; is executed. **Third run**: Marks 130; m>=0 is **TRUE** but m<=100 is **FALSE** and therefore **if** condition is **FALSE**, cout<<"Input error"; is executed.

Running of the program

Input marks[0-100]? 78

Marks=78.5

Input marks[0-100]? -35

Input Error

Input marks[0-100]? 130

Input Error

C++ program to input three values and display the largest value on the screen.

#include<iostream.h>

**void** main()

{

**int** a, b, c, max;

cout<<"1st value? "; cin>>a;

cout<<"2nd value? "; cin>>b;

cout<<"3rd value? "; cin>>c;

**if** (a>=b && a>=c)

**Explanation of the output**

**First run**: Conditions a>=b and a>=c are **FALSE,** first **if** condition is **FALSE** and therefore max=a is ignored. Conditions b>=a and b>=c are **TRUE,** second **if** condition is **TRUE** and therefore max is assigned the value 65. Condition c>=a is **TRUE** but c>=b is **FALSE,** third **if** condition is **FALSE** and therefore max=c is ignored. Hence program displays Max=65. **Second run**: Conditions a>=b and a>=c are **TRUE,** first **if** condition is **TRUE** and therefore max is assigned the value 40. Conditions b>=a and b>=c are **FALSE,** second **if** condition is **FLSE** and therefore max=b is ignored. Condition c>=a is **FALSE** but c>=b is **TRUE,** third **if** condition is **FALSE** and therefore max=c is ignored. Hence program displays Max=40. **Third run**: Conditions a>=b and a>=c are **FALSE,** first **if** condition is **FALSE** and therefore max=a is ignored. Conditions b>=a is **TRUE** but b>=c is **FALSE,** second **if** condition is **FLSE** and therefore max=c is ignored. Conditions c>=a and c>=b are **TRUE,** third **if** condition is **TRUE** and therefore max is assigned the value 80. Hence program displays Max=80.

max=a;

**if** (b>=a && b>=c)

max=b;

**if** (c>=a && c>=b)

max=c;

cout<<"Max="<<max;

}

Running of the program

1st value? 34

2nd value? 65

3rd value? 49

Max=65

1st value? 40

2nd value? 20

3rd value? 30

Max=40

1st value? 50

2nd value? 60

3rd value? 80

Max=80

C++ program to input a character and check whether inputted character is uppercase or not.

#include<iostream.h>

**void** main()

{

**char** ch;

cout<<"Input character? "; cin>>ch;

**if** (ch>='A' && ch<='Z')

cout<<"Uppercase";

**else**

cout<<"Not Uppercase";

}

**Explanation of output**

Two conditions are ch>='A' and ch<='Z'. **First run**: Inputted character F; ch>='A' and ch<='Z' are **TRUE** and therefore **if** condition is **TRUE**, cout<<"Uppercase"; is executed. **Second run**: Inputted character e; ch>='A' is **TRUE** but ch<='Z' is **FALSE** and therefore **if** condition is **FALSE**, cout<<"Not Uppercase"; is executed.

Running of the program

Input character? F

Uppercase

Input character? e

Not Uppercase

1. Write a complete C++ program to input theory marks out of 70 and practical marks out of 30; check that the inputted marks are valid then calculate total marks (theory marks + practical marks) and display the total marks on then screen. If inputted marks are invalid then display an error message.

|  |
| --- |
| #include<iostream.h>  **void** main()  {  **double** theo, prac;  cout<<"Theory marks [0-70]? "; cin>>theo;  cout<<"Practical marks [0-30]? "; cin>>prac;  **if** (theo>=0 && theo<=70 && prac>=0 && prac<=30)  {  **double** total=theo+prac;  cout<<"Total Marks="<<total;  }  **else**  cout<<"Inputted marks out of range";  } |

2. Write a complete C++ to input three angles of a triangle and check whether inputted angles form a valid triangle or not.

|  |
| --- |
| #include<iostream.h>  **void** main()  {  **double** a, b, c;  cout<<"1st angle? "; cin>>a;  cout<<"2nd angle? "; cin>>b;  cout<<"3rd angle? "; cin>>c;  **if** (a>0 && b>0 && c>0 && a+b+c==180)  cout<<"Angles Form a Triangle";  **else**  cout<<"Angles don't Form a Triangle";  } |

3. Write a complete C++ to input three angles of a triangle and check whether inputted angles form an equilateral triangle or not.

|  |
| --- |
| #include<iostream.h>  **void** main()  {  **double** a, b, c;  cout<<"1st angle? "; cin>>a;  cout<<"2nd angle? "; cin>>b;  cout<<"3rd angle? "; cin>>c;  **if** (a==60 && b==60 && c==60)  cout<<"Equilateral Triangle";  **else**  cout<<"Not Equilateral Triangle";  } |

4. Write a complete C++ to input three angles of a triangle and check whether inputted angles form a scalene triangle or not.

|  |
| --- |
| #include<iostream.h>  **void** main()  {  **double** a, b, c;  cout<<"1st angle? "; cin>>a;  cout<<"2nd angle? "; cin>>b;  cout<<"3rd angle? "; cin>>c;  **if** (a!=b && b!=c && c!=a)  cout<<"Scalene Triangle";  **else**  cout<<"Not Scalene Triangle";  } |

5. Write a complete C++ program to input a character and check whether inputted character is digit or not.

|  |
| --- |
| #include<iostream.h>  **void** main()  {  **char** ch;  cout<<"Input character? "; cin>>ch;  **if** (ch>='0' && ch<='9')  cout<<"Digit";  **else**  cout<<"Not Digit";  } |

**||** **Operator**

Program given below checks that the inputted marks lies between 0 and 100. If input is valid, inputted marks is displayed otherwise an error message is displayed on the screen.

#include<iostream.h>

**void** main()

{

**double** m;

cout<<"Input marks[0-100]? "; cin>>m;

**if** (m>=0 && m<=100)

cout<<"Marks="<<m;

**else**

cout<<"Input Error";

}

Marks either less than 0 or more than 100, is invalid. Now we have two conditions marks<0 and marks>100, if either one of the condition is true then marks is invalid. The two conditions marks<0 and marks>100 are to be combined in a different way. This is done by using || operator. || operator combines two or more conditions (sub-conditions) as one condition. **At** **least** one of the sub-conditions has to be **TRUE** for the entire condition to be **TRUE**.

Rule: **if** (Condition1 || Condition2 [|| Condition3 … ])

Statement1 / Block1

**else**

Statement2 / Block2

Truth tables for || operator are given below:

|  |  |  |
| --- | --- | --- |
| Cond1 | Cond2 | Cond1 || Cond2 |
| FALSE | FALSE | FALSE |
| FALSE | TRUE | TRUE |
| TRUE | FALSE | TRUE |
| TRUE | TRUE | TRUE |

|  |  |  |  |
| --- | --- | --- | --- |
| Cond1 | Cond2 | Cond3 | Cond1 || Cond2 || Cond3 |
| FALSE | FALSE | FALSE | FALSE |
| FALSE | FALSE | TRUE | TRUE |
| FALSE | TRUE | FALSE | TRUE |
| FALSE | TRUE | TRUE | TRUE |
| TRUE | FALSE | FALSE | TRUE |
| TRUE | FALSE | TRUE | TRUE |
| TRUE | TRUE | FALSE | TRUE |
| TRUE | TRUE | TRUE | TRUE |

Usage of || operator with **if–else** statement

C++ program to validate inputted marks (marks out of 100)

#include<iostream.h>

**void** main()

{

**double** m;

cout<<"Input marks[0-100]? ";

cin>>m;

**if** (m<0 || m>100)

cout<<"Input error";

**else**

{

cout<<"Valid input"<<endl;

cout<<"Marks="<<m;

}

}

Running of the program

**Explanation of output**

**First run**: Inputted marks -5; m<0 is **TRUE** and m>100 is **FALSE** and therefore **if** condition is **TRUE**, cout<<"Input error"; is executed. **Second run**: Inputted marks 115; m<0 is **FALSE** but m>100 is **TRUE** and therefore **if** condition is **TRUE**, cout<<"Input Error"; is executed. **Third run**: Inputted marks 66; m<0 and m>100 are **FALSE** and therefore **if** condition is **FALSE**, block after **else** is executed.

Input marks[0-100]? -5

Input error

Input marks[0-100]? 115

Input error

Input marks[0-100]? 66

Valid input

Marks=78.5

1. Write a complete C++ program to input three angles of a triangle and check whether inputted angles form a right-angled triangle or not.

|  |
| --- |
| #include<iostream.h>  **void** main()  {  **double** a, b, c;  cout<<"1st angle? "; cin>>a;  cout<<"2nd angle? "; cin>>b;  cout<<"3rd angle? "; cin>>c;  **if** (a==90 || b==90 || c==90)  cout<<"Right-angled Triangle";  **else**  cout<<"Not Right-angled Triangle";  } |

2. Write a complete C++ program to input three angles of a triangle and check whether inputted angles form a isosceles triangle or not.

|  |
| --- |
| #include<iostream.h>  **void** main()  {  **double** a, b, c;  cout<<"1st angle? "; cin>>a;  cout<<"2nd angle? "; cin>>b;  cout<<"3rd angle? "; cin>>c;  **if** (a==b || b==c || c==a)  cout<<"Isosceles Triangle";  **else**  cout<<"Not Isosceles Triangle";  } |

**Explanation of output**

Since a==b, b==c and c==a are **TRUE**, **if** condition is **TRUE** and hence program displays Isosceles Triangle. But in an isosceles only two angles are equal. Edited Isosceles triangle program is given below where **if** condition contains && and || operator. && **has higher precedence than** ||.

Running of the program

1st angle? 60

2nd angle? 60

3rd angle? 60

Isosceles Triangle

|  |
| --- |
| #include<iostream.h>  **void** main()  {  **double** a, b, c;  cout<<"1st angle? "; cin>>a;  cout<<"2nd angle? "; cin>>b;  cout<<"3rd angle? "; cin>>c;  **if** (a==b && c!=60 || b==c && a!=60 || c==a && b!=60)  cout<<"Isosceles Triangle";  **else**  cout<<"Not Isosceles Triangle";  } |

**Nested if-else**

The program segment given below test whether inputted angles form an isosceles triangle or not.

**double** a, b, c;

cout<<"1st angle? "; cin>>a;

cout<<"2nd angle? "; cin>>b;

cout<<"3rd angle? "; cin>>c;

**if** (a==b && c!=60 || b==c && a!=60 || c==a && b!=60)

cout<<"Isosceles Triangle";

**else**

cout<<"Not Isosceles Triangle";

Running of the program segment

1st angle? 40

2nd angle? 40

3rd angle? 20

Isosceles Triangle

1st angle? 120

2nd angle? 80

3rd angle? 80

Isosceles Triangle

When we are inputting three angles of a triangle we are assuming that the sum of three angles will add up to 180. But the program cannot stop the user from inputting three angles where sum does not add up to 180. So there is a logical error in the program. We have to make program smart enough to ignore inputs where sum does not add up to 180. This possible with the help of nested **if**-**else** statement. In a nested **if**-**else** statement, either if part or the else part contain another **if**-**else** statement, that is, **if**-**else** statement contains another **if**-**else** statement.

**Explanation of nested if**-**else syntax**

Outer **if** contains inner **if**-**else** statement and outer **else** contains another inner **if**-**else** statement.

If OuterCondition is **TRUE** then, block after the outer **if** part is executed. Outer **if** block contains inner **if**-**else** statement. If InnerCondition1 is **TRUE** then Statement1 or Block1 is executed. If InnerCondition1 is **FALSE** then Statement2 or Block2 is executed.

If OuterCondition is **FALSE** then, block after **else** part is executed. Outer **else** block contains another inner **if**-**else** statement. If InnerCondition2 is **TRUE** then Statement3 or Block3 is executed. If InnerCondition2 is **FALSE** then Statement4 or Block4 is executed.

Rule: **if** (OuterCondition)

{

//C++ Statements

**if** (InnerCondition1)

Statement1/Block1

**else**

Statement2/Block2

//C++ statements

}

**else**

{

//C++ Statements

**if** (InnerCondition2)

Statement3/Block3

**else**

Statement4/Block4

//C++ Statements

}

Usage of Nested **if**-**else**

a) Program to check right-angled triangle. Outer **if** part containing **if**-**else** statement.

#include<iostream.h>

**void** main()

{

**double** a, b, c;

cout<<"1st angle? "; cin>>a;

cout<<"2nd angle? "; cin>>b;

cout<<"3rd angle? "; cin>>c;

**if** (a+b+c==180)

**if** (a==90 || b==90 || c==90)

cout<<"Right-angled Triangle";

**else**

cout<<"Not Right-angled Triangle";

**else**

cout<<"Input error";

**Explanation of output**

**First run**: Inputted angles 40, 90 and 50 => a+b+c==180 => outer **if** condition is **TRUE** => inner **if**-**else** is executed. Since b==90 => inner **if** condition is **TRUE** and program display Right-angled Triangle. **Second run**: Inputted angles 50, 60 and 70 => a+b+c==180 => outer **if** condition is **TRUE** => inner **if**-**else** is executed. Since a==90, b==90 and c==90 are **FALSE** => inner **if** condition is **FALSE** (inner **else** part is executed) and program display Not Right-angled Triangle. **Third run**: Inputted angles 50, 50 and 50 => a+b+c!=180 => outer **if** condition is **FALSE** => outer **else** part is executed and program displays Input error.

}

Running of the program

1st angle? 40

2nd angle? 90

3rd angle? 50

Right-angled Triangle

1st angle? 50

2nd angle? 60

3rd angle? 70

Not Right-angled Triangle

1st angle? 50

2nd angle? 50

3rd angle? 50

Input error

b) Program to check right-angled triangle. Outer **else** part containing **if**-**else** statement.

#include<iostream.h>

**void** main()

{

**double** a, b, c;

cout<<"1st angle? "; cin>>a;

cout<<"2nd angle? "; cin>>b;

cout<<"3rd angle? "; cin>>c;

**if** (a+b+c!=180)

cout<<"Input error";

**else**

**if** (a==90 || b==90 || c==90)

cout<<"Right-angled Triangle";

**else**

cout<<"Not Right-angled Triangle";

}

Running of the program

**Explanation of output**

**First run**: Angles 40, 90 & 50 => a+b+c!=180 => outer **if** condition is **FALSE** => outer **if**-**else** is executed. Since b==90 => inner **if** condition is **TRUE** and program display Right-angled Triangle. **Second run**: Angles 50, 60 & 70 => a+b+c!=180 => outer **if** condition is **FALSE** => outer **if**-**else** is executed. Since a==90, b==90 and c==90 are **FALSE** => inner **if** condition is **FALSE** and program display Not Right-angled Triangle. **Third run**: Angles 50, 50 & 50 => a+b+c!=180 => outer **if** condition is **TRUE** and program displays Input error. **Fourth run**: Angles -90, 180 & 90 => a+b+c!=180 => outer **if** condition is **FALSE** => outer **else** is executed. Since c==90 => inner **if** condition is **TRUE** and program display Right-angled Triangle.

1st angle? 40

2nd angle? 90

3rd angle? 50

Right-angled Triangle

1st angle? 50

2nd angle? 60

3rd angle? 70

Not Right-angled Triangle

1st angle? 50

2nd angle? 50

3rd angle? 50

Input error

1st angle? -90

2nd angle? 180

3rd angle? 90

Right-angled Triangle

Sum of the three angles add up to 180 but every angle does not store correct value. Valid input means every angle should be positive and a+b+c==180. Edited programs are given below.

|  |
| --- |
| #include<iostream.h>  **void** main()  {  **double** a, b, c;  cout<<"Input 3 angles? "; cin>>a>>b>>c;  **if** (a>0 && b>0 && c>0 && a+b+c==180)  **if** (a==90 || b==90 || c==90)  cout<<"Right-angled Triangle";  **else**  cout<<"Not Right-angled Triangle";  **else**  cout<<"Input error";  } |

|  |
| --- |
| #include<iostream.h>  **void** main()  {  **double** a, b, c;  cout<<"Input 3 angles? "; cin>>a>>b>>c;  **if** (a<=0 || b<=0 || c<=0 || a+b+c!=180)  cout<<"Input error";  **else**  **if** (a==90 || b==90 || c==90)  cout<<"Right-angled Triangle";  **else**  cout<<"Not Right-angled Triangle";  } |

The last program inner **if**-**else** is with the outer **else** part, that is, an **else** is followed by an **if** statement. In a programming terminology it is called **if**-**else**-**if** ladder. In an **if**-**else**-**if** ladder, every **else** is followed by an **if** except the last **else** in the ladder. Few programs are given below using **if**-**else**-**if** ladder.

1. Write a complete C++ program to input 3 coefficient of a quadratic equation (ax2+bx+c=0); calculates the discriminant; display the nature of the roots and display the real roots.

|  |
| --- |
| #include<iostream.h>  #include<math.h>  **void** main()  {  **double** a, b, c;  cout<<"Coefficient of x^2? "; cin>>a;  cout<<"Coefficient of x ? "; cin>>b;  cout<<"Constant Term ? "; cin>>c;  **double** d=b\*b-4\*a\*c;  **if** (d==0)  {  **double** x=-b/(2\*a);  cout<<"Real and equal roots"<<endl;  cout<<"Two root are "<<x<<" and "<<x<<endl;  }  **else**  **if** (d>0)  {  **double** x1=(-b+sqrt(d))/(2\*a), x2=(-b-sqrt(d))/(2\*a);  cout<<"Real and distinct roots"<<endl;  cout<<"Two root are "<<x1<<" and "<<x2<<endl;  }  **else**  cout<<"Complex roots"<<endl;  } |

2. Write a complete C++ program to input a character and check the type of character inputted.

|  |
| --- |
| #include<iostream.h>  **void** main()  {  **char** ch;  cout<<"Input any character? "; cin>>ch;  **if** (ch>='A' && ch<='Z')  cout<<ch<<" is Uppercase"<<endl;  **else**  **if** (ch>='a' && ch<='z')  cout<<ch<<" is Lowercase"<<endl;  **else**  **if** (ch>='0' && ch<='9')  cout<<ch<<" is Digit"<<endl;  **else**  cout<<ch<<" is Special Character"<<endl;  } |

3. Write a complete C++ program to input two values and input an operator; simulate a simple calculator program, that is, if inputted operator is + then find sum or if inputted operator is \* then find product … and display the result on the screen. If an invalid operator is inputted then display an error message.

|  |
| --- |
| #include<iostream.h>  #include<math.h>  **void** main()  {  **char** op;  **double** a, b, result;  cout<<"Input 1st value? "; cin>>a;  cout<<"Input 2nd value? "; cin>>b;  cout<<"Input an operator [+,-,\*,/,^]? "; cin>>op;  **if** (op=='+')  {  result=a+b;  cout<<a<<'+'<<b<<'='<<result<<endl;  }  **else**  **if** (op=='-')  {  result=a-b;  cout<<a<<'-'<<b<<'='<<result<<endl;  }  **else**  **if** (op=='\*')  {  result=a\*b;  cout<<a<<'\*'<<b<<'='<<result<<endl;  }  **else**  **if** (op=='/')  {  **if** (b==0)  cout<<"Division by Zero"<<endl;  **else**  {  result=a/b;  cout<<a<<'/'<<b<<'='<<result<<endl;  }  }  **else**  **if** (op=='^')  {  result=pow(a, b);  cout<<a<<'^'<<b<<'='<<result<<endl;  }  **else**  cout<<"Invalid operator"<<endl;  } |

**Ternary Operator (Conditional Operator)**

Ternary operator is used in place of **if**-**else** statement. But all **if**-**else** statement cannot be replaced by Ternary operator. It is called ternary operator since an expression involving ternary operator requires three (3) operands and two (2) operators. The two Ternary operator is more compact compared to **if**-**else** statement.

Rule: Condition? Action1: Action2

Condition or Logical Expression is evaluated and if the Condition is **TRUE** then Action1 executed otherwise Action2 is executed.

Usage of Ternary Operator (Conditional Operator)

Program to input two values and displays the bigger value on the screen.

#include<iostream.h>

**void** main()

{

**int** a, b;

cout<<"Input 2 integers? ";

cin>>a>>b;

**int** max = a>b ? a : b;

**Explanation of output**

**First run**: Inputted values 20, 10; condition a>b is **TRUE**; action1 is executed; max is assigned the value 20 and therefore program displays Max value=20. **Second run**: Inputted values 25, 40; condition a>b is **FALSE**; action2 is executed; max is assigned the value 40 and therefore program displays Max value=40.

cout<<"Max value="<<max;

}

Running of the program

Input 2 integers? 20 10

Max value=20

Input 2 integers? 25 40

Max value=40

1. Write a complete C++ program to input a character; convert it onto an uppercase.

|  |
| --- |
| #include<iostream.h>  **void** main()  {  **char** ch;  cout<<"Input a character? "; cin>>ch;  ch = ch>='a' && c<='z' ? **char**(ch-32) : ch;  cout<<"Uppercase character="<<ch;  } |

2. Write a complete C++ program to input a character and whether it is digit or not.

|  |
| --- |
| #include<iostream.h>  **void** main()  {  **char** ch;  cout<<"Input a character? "; cin>>ch;  cout<<(ch>='0' && ch<='9' ? "Digit" : "Not Digit");  } |

**Functions from the header file <math.h>**

|  |  |  |
| --- | --- | --- |
| **Function Name** | **Return Value** | **Usage** |
| sqrt(x) | **double** | Finds square root of x |
| pow(b, x) | **double** | Finds b raised to the power x |
| pow10(x) | **double** | Finds 10 raised to the power x |
| exp(x) | **double** | Finds e raised to the power x, e is 2.71828 |
| log(x) | **double** | Finds logarithm of x to the base e |
| log10(x) | **double** | Finds logarithm of x to the base 10 |
| abs(x) | **int** | Finds absolute value of an integer x |
| labs(x) | **long int** | Finds absolute value of a long integer x |
| fabs(x) | **double** | Finds absolute value of a floating point x |
| sin(x) | **double** | Finds sine of x radian |
| cos(x) | **double** | Finds cosine of x radian |
| tan(x) | **double** | Finds tangent of x radian |

1. **double** sqrt(**double** x)

Function sqrt() calculates positive square root of x. If parameter x is negative then run-time error is triggered. Example of sqrt() is given below:

#include<iostream.h>

#include<math.h>

**void** main()

{

**double** x1=25.0, x2=19.5,

**double** r1=sqrt(x1), r2=sqrt(x2);

cout<<”x1= “<<x1<<” , r2=”<<r1<<endl;

}

2. **double** pow(**double** base, **double** expo)

**double** pow10(**int** expo)

**double** exp(**int** expo)

Function pow() calculates base raised to the power of expo. Sometimes the arguments passed to the function pow() produce results that are incalculable and results in run-time error. Function pow10() calculates 10 raised to the power expo. Function exp() calculates e (e is 2.71828) raised to the power expo. Examples of pow(), pow10() and exp() are given below:

#include<iostream.h>

#include<math.h>

**void** main()

{

**double** x1=5, x2=81;

**double** p1=pow(x1, 4), p2=pow(b, 0.25),

**double** p3=pow10(2), p3=exp(4);

cout<<”p1=”<<p1<<” , p2=”<<p2<<endl;

cout<<”p3=”<<p3<<” , p4=”<<p4<<endl;

}

3. **double** log(**double** x)

**double** log10(**double** x)

Function log10() calculates logarithm to the base 10. Function log() calculates logarithm to the base e (e is 2.71828). Logarithm to the base e is also known as **Natural** logarithm. Sometimes the arguments passed to the function log10() and log() produce results that are incalculable and results in run-time error. Examples of log10() and log() are given below:

#include<iostream.h>

#include<math.h>

**void** main()

{

**double** x1=100.0, x2=20.0855

**double** lg10=log10(x1), loge=log(x2);

cout<<”lg10=”<<lg10<<endl;

cout<<”loge=”<<loge<<endl;

}

4. **int** abs(**int** x)

**long** **int** labs(**long** **int** x)

**double** fabs(**double** x)

Function abs() calculates absolute value (magnitude) of an **integer** x. Function labs() calculates absolute value of a **long** **integer** x. Function fabs() calculates absolute value of a **floating** **point** x. In Borland C++ data type **int** and data type **long** **int** are same. Examples of abs(), labs() and fabs() are given below:

#include<iostream.h>

#include<math.h>

**void** main()

{

**int** x1=10, x2=-45, a1=abs(x1), a2=abs(x2);

**double** y1=25.75, y2=-100.45, f1=fabs(y1), f2=fabs(y2);

cout<<”a1=”<<a1<<” , a2=”<<a2<<endl;

cout<<”f1=”<<f1<<” , f2=”<<f2<<endl;

}

5. **double** sin(**double** x)

**double** cos(**double** x)

**double** tan(**double** x)

Function sin() calculates **sine** of x. Function cos() calculates **cosine** of x. Function tan() **tangent** of x. There are no functions for cosec, sec and cot. We can calculate cosec by taking reciprocal of sin, calculate sec by taking reciprocal of cos and cot is calculated as reciprocal of tan. Functions sin(), cos() and tan() assumes that x is in Radian. Hence cout<<sin(30.0); displays -0.988032 and not 0.5. Function sin() calculates sin of 30 radians and not sin of 30 degrees. Sometimes the arguments passed to the function sin() and tan() produce results that are incalculable and results in run-time error. Examples of sin(), cos() and tan() are given on the next page:

#include<iostream.h>

#include<math.h>

**void** main()

{

**double** sin1=sin(30), sin2=sin(M\_PI/4);

**double** cos1=cos(30), cos2=cos(M\_PI/4);

**double** tan1=tan(30), tan2=tan(M\_PI/4);

cout<<”sin1=”<<sin1<<” , sin2=”<<sin2<<endl;

cout<<”cos1=”<<cos1<<” , cos2=”<<cos2<<endl;

cout<<”tan1=”<<tan1<<” , tan2=”<<tan2<<endl;

}

**Functions from the header file <ctype.h>**

|  |  |  |
| --- | --- | --- |
| **Function Name** | **Return Value** | **Usage** |
| toupper(ch) | **int** | Convert a lowercase ch into uppercase |
| tolower(ch) | **int** | Convert a uppercase ch into lowercase |
| isupper(ch) | **int** | Checks if ch is uppercase |
| islower(ch) | **int** | Checks if ch is lowercase |
| isdigit(ch) | **int** | Checks if ch is digit |
| isalpha(ch) | **int** | Checks if ch is alphabet (letter) |
| isalnum(ch) | **int** | Checks if ch is either alphabet or digit |
| Isspace(ch) | **Int** | Checks if ch is either space or tab or new line |

Header file <ctype.h> contains functions related to character (**char**). It is to be noted that all the functions of <ctype.h> has an integer as a parameter instead of character. Also return value of every function is **int**. Now that may sounds little odd. But not really, the header file <ctype.h> is from C-library (even <math.h> is from C-Library). In C data type **char** and data type **int** are used interchangeably.

1. **int** toupper(**int** ch)

Function toupper() converts a lowercase character ch into uppercase (outputs uppercase). But if ch either uppercase or digit or special character then function toupper() outputs ch only. Example of toupper() is given below:

**Explanation of output**

Compiler flags a warning but the program execution gives correct output. Variable c1 stores 'T' since 'T' remains 'T'. Variable c2 stores 'D', since 'd' is converted to 'D'. Variable c3 stores '6' since '6' remains '6'. Variable c4 stores '$' since '$' remains '$'. Since the return value of the function toupper() is **int**, output is 68 ASCII code of 'D'.

#include<iostream.h>

#include<ctype.h>

**void** main()

{

**char** c1=toupper('T'), c2=toupper('d');

**char** c3=toupper('6'), c4=toupper('$');

cout<<"c1="<<c1<<" , c2="<<c2<<endl;

cout<<"c3="<<c3<<" , c4="<<c4<<endl;

cout<<toupper('d')<<endl;

}

Running of the program

c1=T , c2=D

c3=6 , c4=$

68

2. **int** tolower(**int** ch)

Function tolower() converts an uppercase character ch into lowercase (outputs lowercase). But if ch either lowercase or digit or special character then function tolower() outputs ch only. Example of tolower() is given below:

**Explanation of output**

Compiler flags a warning but the program execution gives correct output. Variable c1 stores 't' since 'T' is converted to 't'. Variable c2 stores 'd', since 'd' remains 'd'. Variable c3 stores '6' since '6' remains '6'. Variable c4 stores '$' since '$' remains '$'. Since the return value of the function tolower() is **int**, output is 116 ASCII code of 't'.

#include<iostream.h>

#include<ctype.h>

**void** main()

{

**char** c1=tolower('T'), c2=tolower('d');

**char** c3=tolower('6'), c4=tolower('$');

cout<<"c1="<<c1<<" , c2="<<c2<<endl;

cout<<"c3="<<c3<<" , c4="<<c4<<endl;

cout<<tolower('T')<<endl;

}

Running of the program

c1=t , c2=d

c3=6 , c4=$

116

3. **int** isupper(**int** ch)

**int** islower(**int** ch)

**int** isdigit(**int** ch)

Function isupper() checks whether character ch is uppercase or not. If ch is uppercase (ch>='A' && ch<='Z') then function isupper() returns positive value (**TRUE**) and isupper() returns zero (**FALSE**) if ch is not uppercase.

Function islower() checks whether character ch is lowercase or not. If ch is lowercase (ch>='a' && ch<='z') then function islower() returns positive value (**TRUE**) and islower() returns zero (**FALSE**) if ch is not lowercase.

Function isdigit() checks weather character ch is digit or not. If ch is digit (ch>='0' && ch<='9') then function isdigit() returns positive value (**TRUE**) and islower() returns zero (**FALSE**) if ch is not digit.

Examples of isupper(), islower() and isdigit() are given below:

#include<iostream.h>

#include<ctype.h>

**void** main()

{

**int** x1=isupper('T'), x2=isupper('d'), x3=isupper('6');

**int** y1=islower('T'), y2=islower('d'), y3=islower('6');

**int** z1=isdigit('T'), z2=isdigit('d'), z3=isdigit('6');

**int** w1=isupper('$'), w2=islower('$'), w3=isdigit('$');

cout<<"x1="<<x1<<" , x2="<<x2<<" , x3="<<x3<<endl;

cout<<"y1="<<y1<<" , y2="<<y2<<" , y3="<<y3<<endl;

cout<<"z1="<<z1<<" , z2="<<z2<<" , z3="<<z3<<endl;

cout<<"w1="<<w1<<" , w2="<<w2<<" , w3="<<w3<<endl;

}

4. **int** isalpha(**int** ch)

**int** isalnum(**int** ch)

Function isalpha() checks whether character ch is alphabet or not. If ch is an alphabet then function isalpha() returns positive value (**TRUE**) and returns zero (**FALSE**) otherwise. Function isalnum() checks whether character ch is either alphabet or digit. If ch is either alpabet or digit then fnction isalnum() returns positive value (**TRUE**) and returns zero (**FALSE**) if ch is special character. Examples of isalpha() and isalnum() are given below:

#include<iostream.h>

#include<ctype.h>

**void** main()

{

**int** x1=isalpha('T'), x2=isalpha('d'), x3=isalpha('6');

**int** y1=isalnum('T'), y2=isalnum('d'), y3=isalnum('6');

**int** z1=isalpha('$'), z2=isalnum('$');

cout<<"x1="<<x1<<" , x2="<<x2<<" , x3="<<x3<<endl;

cout<<"y1="<<y1<<" , y2="<<y2<<" , y3="<<y3<<endl;

cout<<"z1="<<z1<<" , z2="<<z2<<endl;

}

Return value of functions isupper(), islower(), isdigit(), isalpha() and isalnum() vary from compiler to compiler. Table is given below showing return value of isupper(), islower(), isdigit(), isalpha() and isalnum() using Borland C++ compiler:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function** | **Digit (**ch**)** | **Uppercase (**ch**)** | **Lowercase (**ch**)** | **Special (**ch**)** |
| isupper(ch) | 0 (False) | 4 (True) | 0 (False) | 0 (False) |
| islower(ch) | 0 (False) | 0 (False) | 8 (True) | 0 (False) |
| isdigit(ch) | 2 (True) | 0 (False) | 0 (False) | 0 (False) |
| isalpha(ch) | 0 (False) | 4 (True) | 8 (True) | 0 (False) |
| isalnum(ch) | 2 (True) | 4 (True) | 8 (True) | 0 (False) |

Program to input a character and check the type of character inputted using isalnum().

#include<iostream.h>

#include<ctype.h>

**void** main()

{

**char** ch;

cout<<"Input character? "; cin>>ch;

**if** (isalnum(ch)==2)

cout<<ch<<" is Digit"<<endl;

**else**

**if** (isalnum(ch)==4)

cout<<ch<<" is Uppercase"<<endl;

**else**

**if** (isalnum(ch)==8)

cout<<ch<<" is Lowercase"<<endl;

**else**

cout<<ch<<" is Special Character"<<endl;

}

**switch**-**case**

In the previous example we observed that each of the conditions that are tested are mutually exclusive (conditions do not overlap). The sequence of mutually exclusive alternatives can be delineated by **if**-**else**-**if** statement, can also be coded using **switch**-**case** construct.

Rule: **switch** (CaseSelector)

{

**case** Label1: StatementList1; **break**;

**case** Label2: Statementlist2; **break**;

**case** Label3: StatementList3; **break**;

:

**default**: DefaultStatementList;

}

Expression after **switch** is called **Case** **Selector**. A **Case** **Selector** is either an **int** integer (**int**) or character (**char**) expression. If the expression is of the type floating point (float/ double), compiler will flag syntax error. But Case Selector may contain a floating value but the final value of the case selector has be either integer type / character type. After the **Case** **Selector** comes a block, the block contains **Case** Labels. **Case** Labels represent all the possible values of **Case** **Selector**. The **switch** evaluates the **Case** **Selector** and looks for its matching value among the **Case** Labels. If a match is found, then the statements in StatementList immediately after the matching Case Label are executed until **break** is encountered or end of **switch*-*case** is reached. If no match is found then DefaultStatementList after **default** is executed. The **default** is optional and, if it is missing, no action takes place if all matches fail. A **break** statement terminates a **switch*-*case**, **break** takes you out of **switch**-**case**, to the next statement after **switch**-**case**. A **break** statement is statement is optional. If **break** statement is missing, then from the matching **Case** Label till the last **Case** Label are executed.

Usage of **switch**-**case** with **break** and **default**:

#include<iostream.h>

**void** main()

{

**char** cho;

cout<<"Select a Shape"<<endl;

cout<<"T for [T]riangle"<<endl;

cout<<"C for [C]circle"<<endl;

cout<<"S for [S]quare"<<endl;

cout<<"Q for [Q]uadrilateral"<<endl;

cin>>cho;

**switch** (cho)

{

**case** 'T': cout<<"Triangle"<<endl; **break**;

**case** 'C': cout<<"Circle"<<endl; **break**;

**case** 'S': cout<<"Square"<<endl; **break**;

**case** 'Q': cout<<"Quadrilateral"<<endl; **break**;

**default**: cout<<"Wrong Choice"<<endl;

}

}

Running of the program

**Explanation of output**

**First run**: Input T, isalnum(ch) returns 4, **case** 4 matches, output T is Uppercase. **break** terminates **switch**-**case**. **Second run**: Input $, isalnum(ch) returns 0, no match is found, **default** label is executed and output is $ is Special Character. **Third run**: Input b, isalnum(ch) returns 8, **case** 8 matches, output b is Lowercase. **break** terminates **switch**-**case**. **Fourth run**: Input 6, isalnum(ch) returns 2, **case** 2 matches, output 6 is Digit. **break** terminates **switch**-**case**. **Fith run**: Input Bye, program accepts B and ignores ye, isalnum(ch) returns 4, **case** 4 matches, output B is Uppercase. **break** terminates **switch**-**case**.

Input character? T

T is Uppercase

Input character? $

$ is Special Character

Input character? b

b is Lowercase

Input character? 6

6 is Digit

Input character? Bye

B is Uppercase

Usage of **switch**-**case** with **break** but without **default**:

#include<iostream.h>

#include<ctype.h>

**void** main()

{

**char** ch;

cout<<"Input character? "; cin>>ch;

**switch** (isalnum(ch))

{

**case** 2: cout<<ch<<" is Digit"<<endl; **break**;

**case** 4: cout<<ch<<" is Uppercase"<<endl; **break**;

**case** 8: cout<<ch<<" is Lowercase"<<endl; **breal**;

**case** 0: cout<<ch<<" is Special Character"<<endl;

}

}

Running of the program

**Explanation of output**

**First run**: Input G, isalnum(ch) returns 4, **case** 4 matches, output G is Uppercase. **break** terminates **switch**-**case**. **Second run**: Input @, isalnum(ch) returns 0, **case** 0 matches, output @ is Special Character. **break** terminates **switch**-**case**. **Third run**: Input f, isalnum(ch) returns 8, **case** 8 matches, output f is Lowercase. **break** terminates **switch**-**case**. **Fourth run**: Input 3, isalnum(ch) returns 2, **case** 2 matches, output 3 is Digit. **break** terminates **switch**-**case**.

Input character? G

G is Uppercase

Input character? @

@ is Special Character

Input character? f

f is Lowercase

Input character? 3

3 is Digit

Since break and default are optional, we use switch-case with break and without default. Previous we have seen how to use switch-case without default. In most cases using swich-case without default will not create any problem during program execution. But using switch-case with break creates major problem during program execution. When break is missing, after a match is found, all the labels after the matching label(s) is(are) executed. So it safe to say, switch-case without break will create Logical error. An example is given in the next page showing use of switch-case without break:

#include<iostream.h>

#include<ctype.h>

**void** main()

{

**char** ch;

cout<<"Input character? "; cin>>ch;

**switch** (isalnum(ch))

{

**case** 2: cout<<ch<<" is Digit"<<endl;

**case** 4: cout<<ch<<" is Uppercase"<<endl;

**case** 8: cout<<ch<<" is Lowercase"<<endl;

**case** 0: cout<<ch<<" is Special Character"<<endl;

}

}

**Explanation of output**

**First run**: Input G, isalnum(ch) returns 4, **case** 4 matches, displays G is Uppercase. **break** is missing there **case** 8 is executed, displays G is Lowercase. **case** 0 is executed displays G is Special Character.No more displays since end of **switch**-**case** and. **Second run**: Input 3, isalnum(ch) returns 2, **case** 2 matches, displays 3 is Digit, 3 is Uppercase, 3 is Lowercase and 3 is Special character. End of **switch**-**case** and no more displays. **Third run**: Input f, isalnum(ch) returns 8, **case** 8 matches, displays f is Lowercase and f is Special Character. So it is very clear the missing **break** displays contradictory output.

Running of the program

Input character? G

G is Uppercase

G is Lowercase

G is Special Character

Input character? 3

3 is Digit

3 is Uppercase

3 is Lowercase

3 is Special Character

Input character? f

f is Lowercase

f is Special Character

1. Write a complete C++ program to input three angles of a triangle and display type of triangle.

|  |
| --- |
| #include<iostream.h>  **void** main()  {  **double** a, b, c;  cout<<"Input 3 angles? "; cin>>a>>b>>c;  **if** (a>0 && b>0 && c>0 && a+b+c==180)  **if** (a==60 && b==60)  cout<<"Equilateral Triangle"<<endl;  **else**  {  **if** (a==90 || b==90 || c==90) cout<<"Right-angled ";  **if** (a==b || b==c || c==a) cout<<"Isosceles ";  **if** (a!=b && b!=c && c!=a) cout<<"Scalene ";  cout<<" Triangle"<<endl;  }  **else**  cout<<"Input Error"<<endl;  } |

2. Write a complete C++ program to input date and check whether inputted date is valid or not. A non century year (year not divisible by 100) divisible 4 is a leap year or century year divisible by 400 is a leap year. In a leap year there are 29 days in February. In a non leap year February has 28 days.

|  |
| --- |
| #include<iostream.h>  **void** main()  {  **int** dd, mm, yy, maxdays=0;  cout<<"Input Day [1-31]? "; cin>>dd;  cout<<"Input Month[1-31]? "; cin>>mm;  cout<<"Input Year [yyyy]? "; cin>>yy;  cout<<"Inputted date "<<dd<<'-'<<mm<<'-'<<yy;  **if** (yy>0)  {  **switch** (mm)  {  **case** 2:  **if** (yy%400==0 || yy%4==0 && yy%100!=0)  maxdays=29;  **else**  maxdays=28;  **break**;  **case** 4:  **case** 6:  **case** 9:  **case** 11: maxdays=30; **break**;  **case** 1:  **case** 3:  **case** 5:  **case** 7:  **case** 8:  **case** 10:  **case** 12: maxdays=31;  }  **if** (dd>=1 && dd<=maxdays)  cout<<" Is Valid";  **else**  cout<<" Is Invalid";  }  **else**  cout<<" Is Invalid";  } |