EXPERIMENT NO 3

Errors, Operators and Conditional Statements

Objectives:

- To know about different types of errors in C++
- To learn about different types of Operators in C++
- To learn about Conditional "If else" statements

Equipment required:

- Dev-C++/Eclipse/Visual Studio installed in PC/Windows

DISCUSSION

1. Pre-Lab

Errors in C++

Error is an illegal operation performed by the user which results in abnormal working of the program. Programming errors often remain undetected until the program is compiled or executed. Some of the errors inhibit the program from getting compiled or executed. Thus, errors should be removed before compiling and executing. The most common errors can be broadly classified as follows.

Syntax Errors

Errors that occur when you violate the rules of writing C++ syntax are known as syntax errors. The compiler error indicates something that must be fixed before the code can be compiled. All these errors are detected by compiler and thus are also known as compile-time errors. Most frequent syntax errors are: Missing Parenthesis "}", semicolon ";" or printing the value of variable without declaring it etc.

Run-time Errors

Errors which occur during program execution(run-time) after successful compilation are called run-time errors. One of the most common run-time errors is division by zero also known as Division error. These types of error are hard to find as the compiler does not point to the line at which the error occurs.

Linker Errors

These are errors generated when the executable of the program cannot be generated. This may be due to wrong function prototyping, incorrect header files. One of the most common linker errors is writing Main () instead of main ().

Logical Errors

On compilation and execution of a program, desired output is not obtained when certain input values are given. These types of errors which provide incorrect output but appears to be error free are called logical errors. These are one of the most common errors done by beginners of programming.

These errors solely depend on the logical thinking of the programmer and are easy to detect if we follow the line of execution and determine why the program takes that path of execution.

Semantic Errors

This error occurs when the statements written in the program are not meaningful to the compiler.

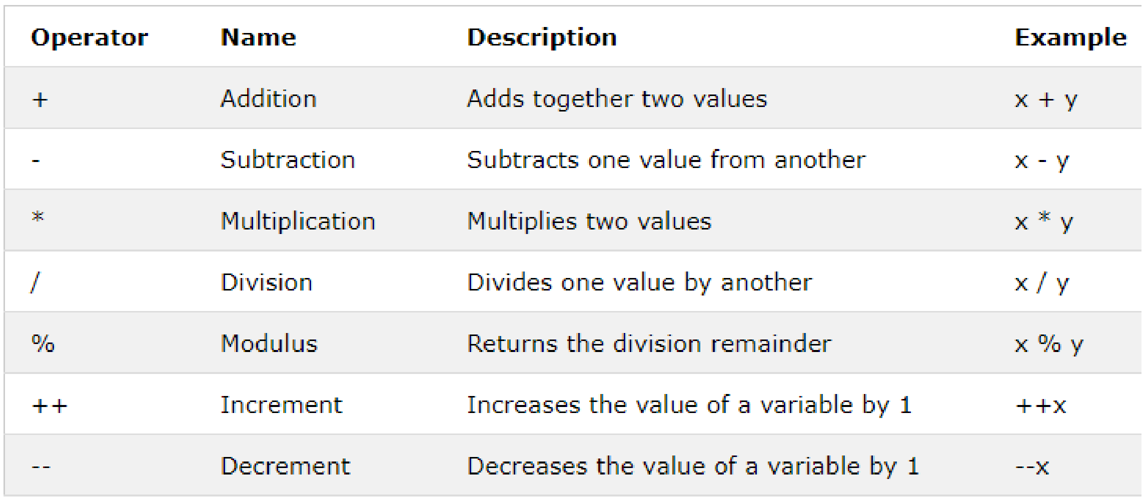
```
int main ()
{
  int a, b, c;
  a + b = c; //semantic error
}
```

Operators in C++

Operators are used to perform operations on variables and values. C++ divides the operators into different groups:

Arithmetic Operators

Arithmetic operators are used to perform common mathematical operations.



Operator	Name	Description	Example
+	Addition	Adds together two values	x + y
-	Subtraction	Subtracts one value from another	x - y
*	Multiplication	Multiplies two values	x * y
/	Division	Divides one value by another	x / y
%	Modulus	Returns the division remainder	x % y
++	Increment	Increases the value of a variable by 1	++x
--	Decrement	Decreases the value of a variable by 1	--x

Assignment Operators

Assignment operators are used to assign values to variables.

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3

Comparison Operators

Comparison operators are used to compare two values.

Note: The return value of a comparison is either true (1) or false (0).

Operator	Name	Example
==	Equal to	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Logical Operators

Logical operators are used to determine the logic between variables or values.

Operator	Name	Description	Example
&&	Logical and	Returns true if both statements are true	x < 5 && x < 10
||	Logical or	Returns true if one of the statements is true	x < 5 || x < 4
!	Logical not	Reverse the result, returns false if the result is true	!(x < 5 && x < 10)

C++ Conditional Statements

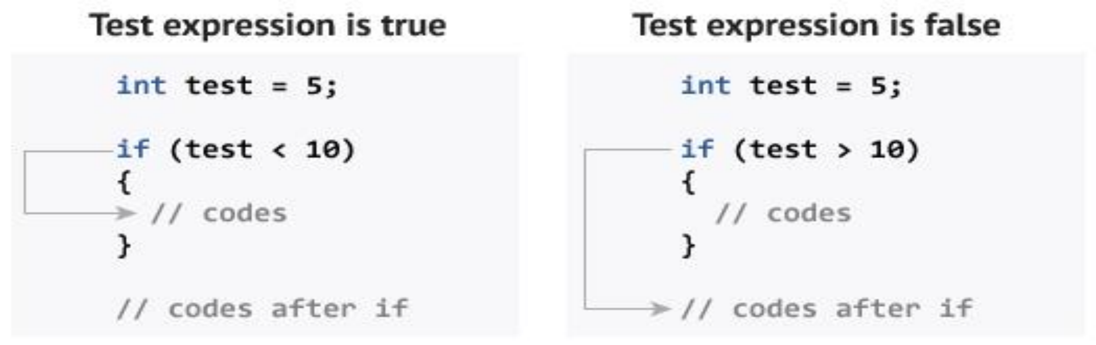
If Statements

Specify the conditions under which a statement or group of statements should be executed.

```
if (Test Expression)
{
  // statements
}
```

If test expression is true, statements inside the body of “if” are executed. If test expression is False, statements inside the body of “if” are skipped. We can also use multiple if statements, whichever test expression is True that particular “if” blocks are executed.

Example



Test expression is true

```
int test = 5;

if (test < 10)
{
  // codes
}

// codes after if
```

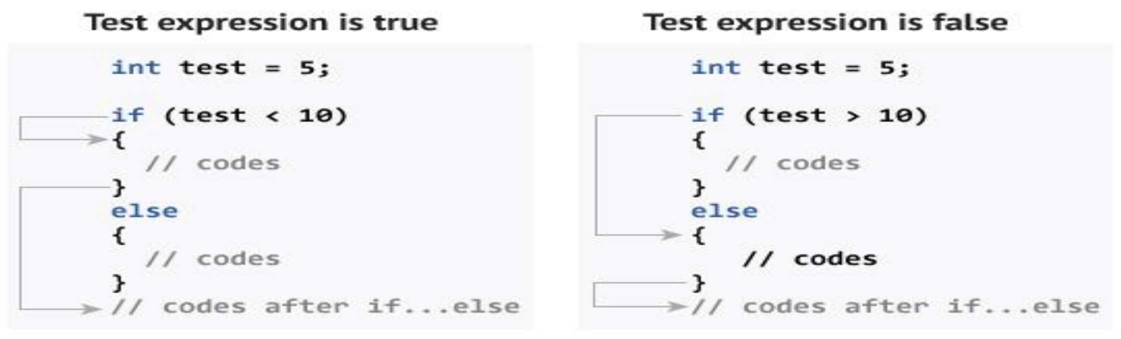
Test expression is false

```
int test = 5;

if (test > 10)
{
  // codes
}

// codes after if
```

If else Statements

If the test condition of “if” block is True, then the “if” block is executed and “else” block is skipped. If the test condition of “if” block is False, then the “else” block is executed and “if” block is skipped.

Example



Test expression is true

```
int test = 5;

if (test < 10)
{
  // codes
}
else
{
  // codes
}
// codes after if...else
```

Test expression is false

```
int test = 5;

if (test > 10)
{
  // codes
}
else
{
  // codes
}
// codes after if...else
```

If Else if Statements

Sometimes, a choice must be made from multiple possibilities, and we only want one possibility to execute. For that purpose, we use else if statement. It first checks the test condition of “if” block, if that comes out to be True, it will skip all other “else if” blocks even if their test conditions are correct. If the test condition of “if” is false it will skip the “if” block and check the next “else if” condition, if that comes out to be True, all other “else if” will be skipped. If the statement for all “else if” are False, then it will only execute the last “else” block.

Syntax

```
if (testExpression1)
{
  // statements to be executed if testExpression1 is true
}

else if(testExpression2)
{
  // statements to be executed if testExpression1 is false and testExpression2 is true
}

else if (testExpression 3)
{
  // statements to be executed if testExpression1 and testExpression2 is false and
testExpression3 is true
}
.
.
else
{
  // statements to be executed if all test expressions are false
}
```

2. Post-Lab (Lab Tasks)

1. Write a program to check whether a number is positive, negative, or zero. If the user enters a negative number, print number entered is negative, if the user enters a positive number, print number entered is positive, if the user enters 0, print number entered is zero.

2. Create a grading system, based on the following scheme. Take marks as input and display the grade at the output.
   - If marks are greater than and equal to 90, grade will be A.
   - If marks are between 80-89, grade will be B.
   - If marks are between 70-79, grade will be B-.
   - If marks are between 60-69, grade will be C.
   - If marks are between 50-59, grade will be C-.
   - If marks are between 40-49, grade will be D.
   - F grade for below 40.

3. Write a program that inputs a year and finds out whether it is a leap year or not, and also finds whether the entered year is even or odd.
4. Find the smallest and largest number among the five numbers entered by the users?
5. Write a program to calculate the monthly telephone bills as per the following rule:
   - Minimum Rs. 250 for up to 70 calls.
   - Plus Rs. 0.60 per call for the next 30 calls.
   - Plus Rs. 0.50 per call for the next 30 calls.
   - Plus Rs. 0.40 per call for any call beyond 130 calls.

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

END