Lab Objectives

By the end of this lab, students will be able to:

- Understand operators and operands.
- Use arithmetic operators.
- Use assignment and compound assignment operators.
- Use increment and decrement operators.
- Understand pre-increment and post-increment.
- Use relational operators.
- Use logical operators.
- Combine operators in simple C++ programs.
- Use operators with `if` statements.
- Modify existing programs and write similar programs independently.

Part 1 — Arithmetic Operators

Arithmetic operators are used to perform mathematical calculations.

Operator	Meaning
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Remainder

Example

```cpp
#include <iostream>
using namespace std;

int main() {

    int a = 20;
    int b = 6;

    cout << "Addition: " << a + b << endl;
    cout << "Subtraction: " << a - b << endl;
    cout << "Multiplication: " << a * b << endl;
    cout << "Division: " << a / b << endl;
    cout << "Remainder: " << a % b << endl;
```

```
    return 0;
}
```

Output

```
Addition: 26
Subtraction: 14
Multiplication: 120
Division: 3
Remainder: 2
```

Understand the Code

```
a + b
```

adds two numbers.

```
a - b
```

subtracts `b` from `a`.

```
a * b
```

multiplies two numbers.

```
a / b
```

performs division.

```
a % b
```

gives the remainder after division.

For example:

```
20 / 6 = 3
20 % 6 = 2
```

Modify the Code

Change:

```
int a = 20;
int b = 6;
```

to:

```
int a = 50;
int b = 8;
```

Run the program and observe the results.

Practice

Try the program with:

```
a = 100, b = 15
a = 25, b = 4
a = 75, b = 10
```

Task

Write a program that takes two integers from the user and displays:

- Addition
- Subtraction
- Multiplication
- Division
- Remainder

Part 2 — Expressions

An expression is a combination of values, variables, and operators that produces a result.

Example

```
#include <iostream>
using namespace std;

int main() {

    int a = 10;
    int b = 5;

    int result = a + b * 2;

    cout << "Result = " << result;
```

```
    return 0;
}
```

Output

```
Result = 20
```

Understand the Code

The expression is:

```
a + b * 2
```

Multiplication is performed before addition.

```
5 * 2 = 10
10 + 10 = 20
```

Modify the Code

Change:

```
int result = a + b * 2;
```

to:

```
int result = (a + b) * 2;
```

Observe the difference.

Task

Write a program that calculates:

```
total = price + tax
```

Use:

```
price = 1000
tax = 150
```

Display the total.

Part 3 — Assignment Operator

The assignment operator `=` is used to assign a value to a variable.

Example

```
#include <iostream>
using namespace std;

int main() {

    int marks = 75;

    cout << "Marks = " << marks;

    return 0;
}
```

Understand the Code

```
int marks = 75;
```

creates a variable named `marks` and assigns `75` to it.

The `=` operator means:

Store the value on the right side in the variable on the left side.

Modify the Code

Change:

```
int marks = 75;
```

to:

```
int marks = 85;
```

Run the program.

Task

Create variables for:

```
studentAge
semester
marks
```

Assign suitable values and display them.

Part 4 — Compound Assignment Operators

Compound assignment operators provide a shorter way to update a variable.

Operator	Example	Equivalent
+=	x += 5	x = x + 5
-=	x -= 5	x = x - 5
*=	x *= 5	x = x * 5
/=	x /= 5	x = x / 5
%=	x %= 5	x = x % 5

Example

```cpp
#include <iostream>
using namespace std;

int main() {

    int x = 20;

    x += 10;
    cout << "After += : " << x << endl;

    x -= 5;
    cout << "After -= : " << x << endl;

    x *= 2;
    cout << "After *= : " << x << endl;

    x /= 5;
    cout << "After /= : " << x << endl;

    return 0;
}
```

Understand the Code

When we write:

```
x += 10;
```

it means:

```
x = x + 10;
```

If `x` is `20`:

```
20 + 10 = 30
```

So `x` becomes `30`.

Modify the Code

Change:

```
int x = 20;
```

to:

```
int x = 100;
```

Change the values used with `+=`, `-=`, `*=`, and `/=`.

Observe the results.

Task

Start with:

```
int balance = 1000;
```

Perform these operations:

1. Add `500`
2. Subtract `200`
3. Multiply by `2`
4. Divide by `4`

Use compound assignment operators.

Display the balance after every operation.

Part 5 — Increment Operator ++

The increment operator increases a value by 1.

Example

```
#include <iostream>
using namespace std;

int main() {

    int students = 30;

    students++;

    cout << students;

    return 0;
}
```

Output

```
31
```

Understand the Code

Before:

```
students = 30
```

After:

```
students++;
```

the value becomes:

```
students = 31
```

The following two statements have the same effect when used alone:

```
students++;
```

and:

```
students = students + 1;
```

Modify the Code

Change:

```
int students = 30;
```

to:

```
int students = 50;
```

Run the program.

Task

Create:

```
int books = 10;
```

Use `++` to increase the number of books by one.

Display the result.

Part 6 — Decrement Operator `--`

The decrement operator decreases a value by `1`.

Example

```
#include <iostream>
using namespace std;

int main() {

    int seats = 50;

    seats--;

    cout << seats;

    return 0;
}
```

Output

```
49
```

Understand the Code

```
seats--;
```

is equivalent to:

```
seats = seats - 1;
```

Modify the Code

Change:

```
int seats = 50;
```

to:

```
int seats = 100;
```

Run the program.

Task

Create:

```
int lives = 5;
```

Use `--` twice and display the final value.

Part 7 — Pre-Increment and Post-Increment

There are two forms of increment:

```
++x
```

and:

```
x++
```

Example

```
#include <iostream>
using namespace std;

int main() {

    int x = 5;

    cout << x++ << endl;
    cout << x << endl;

    return 0;
}
```

Output

```
5
6
```

Understand the Code

With:

```
x++
```

the current value is used first, and then the value is increased.

Now compare it with:

```
#include <iostream>
using namespace std;

int main() {

    int x = 5;

    cout << ++x << endl;
    cout << x << endl;

    return 0;
}
```

Output

```
6
6
```

With:

```
++x
```

the value is increased first, and then the new value is used.

Modify the Code

Change:

```
int x = 5;
```

to:

```
int x = 10;
```

Predict the output before running the program.

Task

Write a program that demonstrates the difference between:

```
x++;
```

and:

```
++x;
```

Use `x = 10`.

Record the output of both programs.

Part 8 — Relational Operators

Relational operators are used to compare values.

Operator	Meaning
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
==	Equal to
!=	Not equal to

Example

```
#include <iostream>
using namespace std;

int main() {

    int a = 20;
    int b = 10;

    cout << (a > b) << endl;
    cout << (a < b) << endl;
    cout << (a == b) << endl;

    return 0;
}
```

Output

```
1
0
0
```

In C++:

```
1 = true
0 = false
```

Understand the Code

```
a > b
```

asks:

Is a greater than b?

Because:

```
20 > 10
```

is true, the result is:

```
1
```

Modify the Code

Change:

```
int a = 20;
int b = 10;
```

to:

```
int a = 5;
int b = 10;
```

Predict the output before running the program.

Task

Take two numbers from the user and display the result of:

```
>
<
>=
<=
==
!=
```

Part 9 — Pass or Fail Using >=

Example

```
#include <iostream>
using namespace std;

int main() {
```

```
    int marks;

    cout << "Enter marks: ";
    cin >> marks;

    if (marks >= 50) {
        cout << "Pass";
    }

    return 0;
}
```

Understand the Code

The condition:

```
marks >= 50
```

checks whether the marks are greater than or equal to `50`.

If the condition is true, the program displays:

```
Pass
```

Modify the Code

Change the passing marks from:

```
50
```

to:

```
40
```

Test the program again.

Task

Write a program that checks whether a student has obtained **60 or more marks**.

Display:

```
Qualified
```

if the condition is true.

Part 10 — Logical AND &&

The && operator is used when **all conditions must be true**.

Example

```
#include <iostream>
using namespace std;

int main() {

    int age = 20;
    int marks = 70;

    if (age >= 18 && marks >= 50) {
        cout << "Eligible";
    }

    return 0;
}
```

Understand the Code

The program checks two conditions:

```
age >= 18
```

and:

```
marks >= 50
```

Both must be true.

```
Age = 20     → true
Marks = 70   → true
```

Therefore:

```
Eligible
```

Modify the Code

Change:

```
int marks = 70;
```

to:

```
int marks = 40;
```

Run the program.

Task

A student is eligible for an exam if:

- Attendance is at least `75`
- Marks are at least `50`

Write a program using `&&`.

Part 11 — Logical OR `||`

The `||` operator is used when **at least one condition must be true**.

Example

```
#include <iostream>
using namespace std;

int main() {

    int semester;

    cout << "Enter semester: ";
    cin >> semester;

    if (semester == 1 || semester == 2) {
        cout << "Junior Semester";
    }

    return 0;
}
```

Understand the Code

The student belongs to the junior semesters if:

```
Semester = 1
OR
Semester = 2
```

Only one condition needs to be true.

Modify the Code

Change the condition so that semesters `1`, `2`, or `3` are considered junior semesters.

Task

Write a program that checks whether a student is eligible for a special lab if they are in:

- Semester 5
- OR Semester 6

Use `||`.

Part 12 — Logical NOT `!`

The `!` operator reverses a Boolean value.

Example

```cpp
#include <iostream>
using namespace std;

int main() {

    bool isStudent = true;

    cout << !isStudent;

    return 0;
}
```

Output

```
0
```

Understand the Code

The original value is:

```
true
```

The `!` operator changes it to:

```
false
```

C++ displays `false` as `0`.

Modify the Code

Change:

```
bool isStudent = true;
```

to:

```
bool isStudent = false;
```

Observe the result.

Task

Create:

```
bool isLoggedIn = false;
```

Use `!` to display the opposite value.

Part 13 — Combining Operators

Different operators can be used together in one program.

Example

```
#include <iostream>
using namespace std;

int main() {

    int age;
    int marks;

    cout << "Enter age: ";
```

```
    cin >> age;

    cout << "Enter marks: ";
    cin >> marks;

    if (age >= 18 && marks >= 50) {
        cout << "Student is eligible";
    } else {
        cout << "Student is not eligible";
    }

    return 0;
}
```

Understand the Code

This program uses:

- Assignment operator
- Input operator `>>`
- Relational operators
- Logical `&&`
- `if`
- `else`

The important condition is:

```
age >= 18 && marks >= 50
```

Both conditions must be true.

Modify the Code

Add attendance:

```
int attendance;
```

Then change the condition to:

```
age >= 18 && marks >= 50 && attendance >= 75
```

Task

Write a program that checks whether a student can sit in the final examination.

Conditions:

```
Attendance >= 75
AND
Marks >= 50
AND
Fee Paid = Yes
```

Part 14 — Student Result Program

Example

```cpp
#include <iostream>
using namespace std;

int main() {

    int marks;

    cout << "Enter marks: ";
    cin >> marks;

    if (marks >= 50) {
        cout << "Pass";
    } else {
        cout << "Fail";
    }

    return 0;
}
```

Understand the Code

The program takes marks from the user.

If:

```
marks >= 50
```

is true, it displays:

```
Pass
```

Otherwise:

```
Fail
```

Modify the Code

Add grades:

```
80 or above  → A
70–79        → B
60–69        → C
50–59        → D
Below 50     → Fail
```

Solve a Similar Task

Write a complete student result program that takes:

```
Student Name
Marks
```

and displays:

```
Student Name
Marks
Result
Grade
```

Part 15 — Shopping Bill

Example

```cpp
#include <iostream>
using namespace std;

int main() {

    double price;
    int quantity;

    cout << "Enter price: ";
    cin >> price;

    cout << "Enter quantity: ";
    cin >> quantity;
```

```
    double total = price * quantity;

    cout << "Total Bill = " << total << endl;

    return 0;
}
```

Understand the Code

The expression:

```
price * quantity
```

calculates the total bill.

For example:

```
Price = 500
Quantity = 4

Total = 500 × 4
      = 2000
```

Modify the Code

Add a 10% discount if the total bill is `5000` or more.

Hint:

```
if (total >= 5000)
```

Task

Write a complete shopping bill program.

The program should:

1. Take product price.
2. Take quantity.
3. Calculate total.
4. Apply 10% discount if total is `5000` or more.
5. Display discount.
6. Display final bill.

Part 16 — Predict the Output

Before running each program, predict the output.

Question 1

```
int a = 10;
int b = 5;

cout << a + b;
```

Output:

```
_____
```

Question 2

```
int x = 10;

x += 5;

cout << x;
```

Output:

```
_____
```

Question 3

```
int x = 5;

cout << x++;
cout << x;
```

Output:

```
_____
```

Question 4

```
int x = 5;
```

```
cout << ++x;
cout << x;
```

Output:

```
_____
```

Question 5

```
int a = 10;
int b = 20;

cout << (a < b);
```

Output:

```
_____
```

Question 6

```
int age = 20;
int marks = 70;

cout << (age >= 18 && marks >= 50);
```

Output:

```
_____
```

Part 17 — Operator Practice

Complete the following table.

Expression	Your Prediction	Actual Result
`10 + 5`		
`10 - 5`		
`10 * 5`		

Expression	Your Prediction	Actual Result
`10 / 5`		
`10 % 3`		
`10 > 5`		
`10 < 5`		
`10 == 10`		
`10 != 10`		
`10 > 5 && 5 > 2`		

`10 < 5 5 > 2`

Part 18 — Independent Tasks

Students should solve these tasks without looking at the examples.

Task 1 — Calculator

Create a calculator that takes two numbers and displays:

- Addition
- Subtraction
- Multiplication
- Division
- Remainder

Task 2 — Even or Odd

Take an integer from the user.

Use `%` to check whether the number is:

```
Even
```

or:

```
Odd
```

Hint:

```
number % 2
```

Task 3 — Age Check

Take the user's age.

If age is `18` or above, display:

```
Adult
```

Otherwise display:

```
Minor
```

Task 4 — Number Comparison

Take two numbers.

Display whether:

- First number is greater
- Second number is greater
- Both numbers are equal

Task 5 — Student Eligibility

Take:

```
Age
Marks
Attendance
```

A student is eligible when:

```
Age >= 18
Marks >= 50
Attendance >= 75
```

Use:

```
&&
```

Task 6 — Discount

Take the shopping bill amount.

If the bill is:

```
5000 or more → 10% discount
3000–4999   → 5% discount
Below 3000  → No discount
```

Display:

- Original bill
- Discount
- Final bill

Part 19 — Challenge Task

Student Admission Eligibility System

Create a program that takes:

```
Age
Marks
Attendance
Semester
```

The student is eligible if:

```
Age >= 18
AND
Marks >= 50
AND
Attendance >= 75
```

Additionally, if the semester is `1` or `2`, display:

```
Junior Semester
```

Otherwise display:

```
Senior Semester
```

The program should use:

- Arithmetic operators

- Assignment operators
- Relational operators
- Logical `&&`
- Logical `||`
- `if`
- `else`

Part 20 — Viva / Short Questions

Answer the following questions.

1. What is an operator?
2. What is an operand?
3. What is an expression?
4. What is a unary operator?
5. What is a binary operator?
6. What is the purpose of `+`?
7. What is the purpose of `%`?
8. What is the difference between `=` and `==`?
9. What is the difference between `++x` and `x++`?
10. What is the purpose of `--`?
11. What does `>=` mean?
12. What does `!=` mean?
13. What is the purpose of `&&`?
14. What is the purpose of `||`?
15. What does `!` do?
16. What is the difference between `x += 5` and `x = x + 5`?
17. What does `x *= 2` do?
18. What is the result of `20 % 6`?
19. What does a relational operator return?
20. Why are logical operators used with conditions?

Lab Submission Requirements

Students must submit:

1. Source code for all assigned tasks.
2. Output screenshots where required.
3. Answers to the prediction questions.
4. Answers to the viva questions.
5. Completed practice table.
6. Challenge task code.