

1. Learning Objectives

After completing this lesson, you will be able to:

- Use logical operators inside `if` and `if-else` statements.
- Combine multiple conditions using `&&`, `||`, and `!`.
- Understand how changing variable values affects program output.
- Write programs to solve real-life problems.

2. Quick Revision: Logical Operators

`&&`

AND

Both conditions must be true.

Example: `marks >= 50 && attendance >= 75`

`||`

OR

At least one condition must be true.

Example: `marks >= 90 || attendance == 100`

`!`

NOT

Reverses a condition from true to false or false to true.

Example: `!(marks >= 50)`

Example 1: Student Pass or Fail Using AND (`&&`)

Problem: A student passes only if marks are at least 50 and attendance is at least 75%.

Step 1 — Understand the problem

Suppose a student obtains 65 marks and has 80% attendance.

We need to check two conditions:

1. Are marks greater than or equal to 50?
2. Is attendance greater than or equal to 75%?

Both conditions must be true for the student to pass.

Step 2 — Write the code

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

int main() {
    int marks = 65;
    int attendance = 80;

    if (marks >= 50 && attendance >= 75) {
        cout << "Pass";
    } else {
        cout << "Fail";
    }

    return 0;
}
```

Step 3 — Understand the code

Code	Explanation
<code>int marks = 65;</code>	Stores the student's marks.
<code>int attendance = 80;</code>	Stores attendance percentage.
<code>marks >= 50</code>	True because 65 is greater than 50.
<code>attendance >= 75</code>	True because 80 is greater than 75.
<code>&&</code>	Checks whether both conditions are true.
<code>if</code>	Executes when the complete condition is true.
<code>else</code>	Executes when the complete condition is false.

Step 4 — Trace the conditions

Marks >= 50

TRUE

Attendance >= 75

TRUE

TRUE && TRUE

TRUE

Output

Pass

Step 5 — Modify the code

Change only the variable values and predict the output.

Your task: Set marks to 70 and attendance to 60. Explain why the output is Fail.

Example 2: Scholarship Eligibility Using AND (&&)

Problem: A student can receive a scholarship if family income is below Rs. 50,000 per month and marks are at least 80.

Step 1 — Understand the problem

Both conditions must be satisfied:

- Family income is less than 50,000.
- Marks are at least 80.

Step 2 — Write the code

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

int main() {
    int income = 40000;
    int marks = 85;

    if (income < 50000 && marks >= 80) {
        cout << "Eligible for scholarship";
    } else {
        cout << "Not eligible";
    }

    return 0;
}
```

Step 3 — Understand the code

- `income < 50000` → True.
- `marks >= 80` → True.
- `true && true` → True.

Output:

```
Eligible for scholarship
```

Step 4 — Modify the code

Try these changes one by one.

Income	Marks	Expected output
40,000	85	Eligible for scholarship
60,000	85	Not eligible
40,000	70	Not eligible
60,000	70	Not eligible

Learning point: If either condition becomes false, the AND expression becomes false.

Example 3: Bonus Eligibility Using OR (||)

Problem: An employee receives a bonus if sales are at least Rs. 100,000 OR the employee has completed 10 years of service.

Step 1 — Understand the problem

Only one condition needs to be true.

Step 2 — Write the code

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

int main() {
    int sales = 80000;
    int serviceYears = 12;

    if (sales >= 100000 || serviceYears >= 10) {
        cout << "Bonus approved";
    } else {
        cout << "Bonus not approved";
    }

    return 0;
}
```

Step 3 — Understand the code

- `sales >= 100000` → False.
- `serviceYears >= 10` → True.
- `false || true` → True.

Output:

Bonus approved

Step 4 — Modify the code

Change the values and predict the output.

Sales	Service years	Expected output
120,000	5	Bonus approved
80,000	12	Bonus approved
120,000	12	Bonus approved
80,000	5	Bonus not approved

Learning point: OR returns false only when both conditions are false.

Example 4: NOT (!) — Checking Registration

Problem: Display a message if a student is not registered.

Step 1 — Write the code

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

int main() {
    bool registered = false;

    if (!registered) {
        cout << "Please register first";
    } else {
        cout << "Registration complete";
    }

    return 0;
}
```

Step 2 — Understand the code

The variable `registered` is a Boolean variable. It can store `true` or `false`.

- `registered = false`
- `!registered` becomes `true`.
- The first block executes.

Output:

```
Please register first
```

Step 3 — Modify the code

Change:

```
bool registered = false;
```

to:

```
bool registered = true;
```

Expected output:

```
Registration complete
```

Learning point: The NOT operator reverses the Boolean value.

5. Practice Coding Exercises

Important: When combining `&&` and `||` in the same condition, use parentheses to make the intended logic clear. For example:

```
if (marks >= 60 && (hasTestResult || hasExemption))
```

This means the student needs marks of at least 60 and must have either a valid test result or an approved exemption.