Lecture – 9-

Conditionals and loops

The if Statement

- The **if** statement is used to execute some code if a condition is true.

Syntax:

```
if (condition) {
//statements
}
```

- The **condition** specifies which expression is to be evaluated.
- If the condition is **true**, the statements in the curly brackets are executed.
- If the condition is **false**, the statements in the curly brackets are ignored, and the program continues to run after the if statements body.
- Use **relational operators** to evaluate conditions.
- **Example:**

```
if (7 > 4) {
cout << "Yes";
}
// Outputs "Yes"
```

- The **if** statement evaluates the condition (7>4), finds it to be **true**, and then executes the **cout** statement.
- If we change the greater operator to a less than operator (**7<4**), the statement will not execute.
- Note: A condition specified in an if statement does not require a semicolon.

Relational operators:

Operator	Description	Example
>=	Greater than or equal to	7 >= 4 True
<=	Less than or equal to	7 <= 4 False
==	Equal to	7 == 4 False
!=	Not equal to	7 != 4 True

Example:

```
if (10 == 10) {
cout << "Yes";
}
// Outputs "Yes"
```

- The **not equal to** operator evaluates the operands, determines whether they are equal or not.
- If the operands are not equal, the condition is evaluated to **true**.

- **Example:**

```
if (10 != 10) {
cout << "Yes";
}
```

- The above condition evaluates to **false** and the block of code is not executed.
- You can use relational operators to compare variables in the **if** statement.

- **Example:**

```
int a = 55;
int b = 33;
if (a > b) {
cout << "a is greater than b";
}

// Outputs "a is greater than b"
```

The else Statement

- An **if** statement can be followed by an optional **else** statement, which executes when the condition is **false**.

Syntax:

```
if (condition) {
//statements
}
else {
//statements
}
```

- The compiler will test the condition:
  - If it evaluates to **true**, then the code inside the **if** statement will be executed.
  - If it evaluates to **false**, then the code inside the **else** statement will be executed.
- When only **one** statement is used inside the if/else, then the curly braces can be omitted.
- **Example:**

```
int mark = 90;
if (mark < 50) {
cout << "You failed." << endl;
}
else {
cout << "You passed." << endl;
}

// Outputs "You passed."
```

- In all previous examples only one statement was used inside the if/else statement, but you may include as many statements as you want.

Example:

```
int mark = 90;
if (mark < 50) {
cout << "You failed." << endl;
cout << "Sorry" << endl;
}
else {
cout << "Congratulations!" << endl;
cout << "You passed." << endl;
cout << "You are awesome!" << endl;
}
/* Outputs
Congratulations!
You passed.
You are awesome!
*/
```

Nested if Statements

- You can also include, or **nest**, if statements within another if statement.
- **Example:**

```
int mark = 100;
if (mark >= 50) {
cout << "You passed." << endl;
if (mark == 100) {
cout <<"Perfect!" << endl;
}
}
else {
cout << "You failed." << endl;
}

/*Outputs
You passed.
Perfect!
*/
```

- C++ provides the option of nesting an unlimited number of if/else statements.

Example:

```
int age = 18;
if (age > 14) {
if(age >= 18) {
cout << "Adult";
}
else {
cout << "Teenager";
}
}
else {
if (age > 0) {
cout << "Child";
}
else {
cout << "Something's wrong";
}
}
```

- Remember that all **else** statements must have corresponding **if** statements.
- In if/else statements, **a single statement** can be included without enclosing it into curly braces.
- Example:

```
int a = 10;
if (a > 4)
cout << "Yes";
else
cout << "No";
```