DSA Study Notes Day 3

Chapter 3: Conditional Statements and Loops

1. Conditional Statements

Conditional statements allow you to execute certain code blocks based on specific conditions.

If-Else Statement

The `if-else` statement checks a condition. If the condition is `true`, the code inside the `if` block is executed. Otherwise, the code inside the `else` block is executed.

Syntax:

```
if (condition) {
    // code to execute if condition is true
} else {
    // code to execute if condition is false
}
```

Example:

```
if (n >= 0) {
    cout << "positive";
} else {
    cout << "negative";
}
```

In this example, if `n` is greater than or equal to 0, the program prints "positive." Otherwise, it prints "negative."

Ternary Operator (Conditional Operator)

The ternary operator is a shorthand way to write an `if-else` statement. It is used when you have a simple condition that leads to two outcomes.

Syntax:

```
condition ? expression1 : expression2;
```

- If the `condition` is `true`, `expression1` is executed.
- If the `condition` is `false`, `expression2` is executed.

Example:

```
n >= 0 ? cout << "positive" : cout << "negative";
```

This example performs the same task as the `if-else` above but in one line using the ternary operator.

---

2. Loops

Loops allow you to repeat a block of code multiple times, either for a set number of times or until a certain condition is met.

While Loop

The `while` loop repeatedly executes a block of code as long as the specified condition remains `true`.

Syntax:

```
while (condition) {
    // code to be executed
}
```

Example:

```
int i = 0;
while (i < 5) {
    cout << i << " ";
    i++;
}
```

In this example, the loop prints numbers from 0 to 4, as long as `i` is less than 5.

For Loop

The `for` loop is used when the number of iterations is known beforehand. It has three parts:

1. Initialization
2. Condition
3. Increment/Decrement

Syntax:

```
for (initialization; condition; increment/decrement) {
    // code to be executed
}
```

Example:

```
for (int i = 0; i < 5; i++) {
    cout << i << " ";
}
```

This loop will print numbers from 0 to 4. The loop starts with `i = 0`, and continues while `i < 5`, incrementing `i` after each iteration.

Do-While Loop

The `do-while` loop is similar to the `while` loop, except that it guarantees that the block of code will execute at least once before checking the condition.

Syntax:

```
do {
    // code to be executed
} while (condition);
```

Example:

```
cpp
Copy code
int i = 0;
do {
    cout << i << " ";
    i++;
} while (i < 5);
```

This loop will print numbers from 0 to 4. The key difference from the `while` loop is that the condition is checked after the code block is executed.

Nested Loops

A nested loop is a loop inside another loop. Each time the outer loop is executed, the inner loop runs its full cycle.

Example:

```
for (int i = 1; i <= 3; i++) {
    for (int j = 1; j <= 2; j++) {
        cout << "i = " << i << ", j = " << j << endl;
    }
}
```

This example will print the values of `i` and `j` as the loops progress. The outer loop runs 3 times, and for each iteration of the outer loop, the inner loop runs 2 times.

---

3. Home Task

Q1. Sum of All Numbers from 1 to n that are Divisible by 3

To solve this problem, you can use a loop to go through all numbers from 1 to `n` and check if each number is divisible by 3.

Example Code:

```
int sum = 0;
for (int i = 1; i <= n; i++) {
    if (i % 3 == 0) {
        sum += i;  // Add the number to the sum if it's divisible by 3
    }
}
cout << "Sum of numbers divisible by 3: " << sum << endl;
```

This code checks if each number is divisible by 3 using the modulus operator (`%`), and if true, adds the number to the `sum`.

Q2. Print the Factorial of a Number n

The factorial of a number `n` is the product of all positive integers less than or equal to `n`. You can compute this using a loop.

Example Code:

```
int factorial = 1;
for (int i = 1; i <= n; i++) {
    factorial *= i;  // Multiply factorial by i
}
cout << "Factorial of " << n << " is " << factorial << endl;
```

This code calculates the factorial by multiplying `factorial` by each number from 1 to `n`.

Day 3 Notes

Prepared by Munawar Johar