Logical and Relational Operators

Lecture 18: Conditional Statement if in C

1. Introduction to the if Statement

Conditional statements are fundamental in programming as they allow the execution of specific code blocks based on certain conditions. The if statement is the most basic and commonly used conditional statement in C. It enables your program to make decisions and execute certain parts of code only when specified conditions are true.

2. Basic Syntax of the if Statement

The if statement evaluates a condition enclosed in parentheses. If the condition is true (non-zero), the code block following the if statement is executed. If the condition is false (zero), the code block is skipped.

Syntax:

```
1. if (condition) {
2.     // Code to execute if the condition is true
3. }
```

Example:

```
1. int a = 10;
2.
3. if (a > 5) {
4.     printf("a is greater than 5.\n");
5. }
```

In this example, the condition a > 5 is true, so the message "a is greater than 5." is printed to the console.

3. The else Statement

The else statement can be used in conjunction with if to execute a different block of code if the condition is false.

Syntax:

```
1. if (condition) {
```

```
    // Code to execute if the condition is true
} else {
    // Code to execute if the condition is false
}
```

Example:

```
int a = 3;

if (a > 5) {
    printf("a is greater than 5.\n");
} else {
    printf("a is not greater than 5.\n");
}
```

In this case, the condition a > 5 is false, so the message "a is not greater than 5." is printed.

4. The else if Statement

For multiple conditions, you can use the else if statement. This allows you to check several conditions in sequence.

Syntax:

```
if (condition1) {
    // Code to execute if condition1 is true
} else if (condition2) {
    // Code to execute if condition2 is true
} else {
    // Code to execute if none of the above conditions are true
}
```

Example:

```
int a = 7;
```

```
2.
3.  if (a > 10) {
4.      printf("a is greater than 10.\n");
5.  } else if (a > 5) {
6.      printf("a is greater than 5 but less than or equal to 10.\n");
7.  } else {
8.      printf("a is 5 or less.\n");
9.  }
```

In this example, since a is 7, the second condition (a > 5) is true, so the message "a is greater than 5 but less than or equal to 10." is printed.

5. Practical Example: Using if Statements

Problem Statement:
Write a C program that determines if a number entered by the user is positive, negative, or zero using if, else if, and else statements.

Code Implementation:

```
1.  #include <stdio.h>
2.
3.  int main() {
4.      int num;
5.
6.      // Input a number from the user
7.      printf("Enter an integer: ");
8.      scanf("%d", &num);
9.
10.     // Use if-else statements to determine if the number is positive, negative, or zero
11.     if (num > 0) {
12.         printf("%d is a positive number.\n", num);
```

```
    } else if (num < 0) {
        printf("%d is a negative number.\n", num);
    } else {
        printf("The number is zero.\n");
    }

    return 0;
}
```

Explanation:

- The program takes an integer input from the user.
- It uses if, else if, and else statements to check if the number is positive, negative, or zero, and prints the appropriate message.

6. Question:

Test Your Understanding:

- **Q1**: Write a C program that checks if a given year is a leap year. Use if-else statements to implement the logic. (Hint: A year is a leap year if it is divisible by 4 but not by 100, or divisible by 400).
- **Q2**: Create a program that takes three integers from the user and prints the largest number using nested if statements.
- **Q3**: Explain how you would use the if statement to determine if a character entered by the user is a vowel or a consonant. Write the corresponding C code.

Conclusion:

The if statement is a fundamental control structure in C that allows your programs to make decisions based on conditions. By mastering if, else if, and else statements, you can write more dynamic and responsive code that adapts to different inputs and conditions.