Control Statements and Conditions in C Programming

C Programming Journey

Introduction

In C programming, effective decision-making is vital for directing program flow. By testing variables, programs can determine which tasks to perform. For instance, if X is greater than Y, execute task 1; otherwise, perform task 2. C offers various tests, with the if-else construct being the simplest. However, more complex conditions may require nested if-else statements. Alternatively, the switch construct is suitable when tasks need selection based on variable values.

Types of Control Statements:

- if
- if else
- nested if else
- if-else-if
- switch Case

What is an If Statement in C?

An If statement in C is a fundamental decision-making construct that allows the execution of a block of code based on the evaluation of a condition. It enables the

program to make choices and execute different code paths depending on whether a specified condition is true or false. The If statement accepts boolean values, executing the block of statements below it if the condition evaluates to true; otherwise, it skips the block of code. Essentially, an If statement provides a way to control the flow of execution in a C program based on the outcome of logical expressions.

1. **if Statement:**
    - Syntax:

```
if (condition) {
    // Code to execute if condition is true
}
```

    - Example:

```
int x = 10;
if (x > 5) {
    printf("x is greater than 5\n");
}
```

What is an If-Else Statement in C?

An If-Else statement in C is a fundamental control flow mechanism that allows for conditional execution of code segments. It consists of two main blocks: the `if` block and the `else` block. When the condition specified in the `if` block evaluates to true, the code within that block is executed. However, if the condition is false, the code within the `else` block is executed instead. This construct provides a way to execute different blocks of code based on the outcome of a condition, effectively branching the program's execution path.

*if-else Statement:**

- Syntax:

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

```
    // Code to execute if condition is false
}
```

- Example:

```
int x = 3;
if (x % 2 == 0) {
    printf("x is even\n");
} else {
    printf("x is odd\n");
}
```

What is nested if-else statement in C?

In C programming, a nested `if-else` statement is a construct where an `if-else` statement is nested inside another `if` or `else` block. This means that there is an `if` statement within the block of code controlled by another `if` or `else` statement. `Nested if-else` statements are used when more complex decision-making is required, allowing for multiple conditions to be evaluated sequentially.

Here's a simple example of a nested if-else statement in C:

```
#include <stdio.h>

int main() {
    int num = 10;

    if (num > 0) {
        if (num % 2 == 0) {
            printf("The number is positive and even.\n");
        } else {
            printf("The number is positive and odd.\n");
        }
    } else {
        printf("The number is non-positive.\n");
    }

    return 0;
}
```

In this example, the outer if statement checks if the number `num` is positive. If it is positive, the inner if-else statement checks whether the number is even or odd. Depending on the conditions, different messages are printed to the console. If the

number is not positive, a different message is printed. This demonstrates the concept of nesting if-else statements in C to handle different scenarios based on multiple conditions.

Nested if-else Statement:

- Syntax:

```
if (condition1) {
    // Code to execute if condition1 is true
    if (condition2) {
        // Code to execute if both condition1 and condition2 are true
    } else {
        // Code to execute if condition1 is true but condition2 is false
    }
} else {
    // Code to execute if condition1 is false
}
```

- Example:

```
int x = 10;
if (x > 0) {
    if (x < 20) {
        printf("x is between 0 and 20\n");
    } else {
        printf("x is greater than or equal to 20\n");
    }
} else {
    printf("x is less than or equal to 0\n");
}
```

if-else-if Statement?

An `if-else-if` statement in C is a control flow statement used to evaluate multiple conditions sequentially. It provides an alternative set of code blocks to execute based on the result of each condition. Here's how it works:

- The `if` statement checks the first condition. If it's true, the corresponding code block executes, and the control exits the `if-else-if` ladder.
- If the first condition is false, the control moves to the next `else if` condition. If this condition is true, its associated code block executes, and the control exits the

`if-else-if` ladder.

- This process continues until a true condition is found or until the `else` block is reached, which serves as the default option if none of the preceding conditions are true.

Here's a basic syntax example of an `if-else-if` statement in C:

```
if (condition1) {
    // Code block to execute if condition1 is true
} else if (condition2) {
    // Code block to execute if condition2 is true
} else if (condition3) {
    // Code block to execute if condition3 is true
} else {
    // Code block to execute if none of the above conditions are true
}
```

In this structure, each `else if` block provides an alternative condition to be evaluated if the preceding conditions are false, allowing for more complex decision-making in the program.

if-else-if Statement:

- Example:

```
int x = 15;
if (x > 10) {
   printf("x is greater than 10\n");
} else if (x == 10) {
   printf("x is equal to 10\n");
} else {
   printf("x is less than 10\n");
}
```

What is Switch Case in C

In C programming, the switch case statement is a control flow statement used to select one choice among multiple options based on the value of a given expression. It provides an efficient way to write code when there are multiple conditions to be tested against a single variable. The switch statement evaluates the expression once and then compares the resulting value to the values of the case labels. If a match is found,

the corresponding block of code is executed until a break statement is encountered or until the end of the switch block. If no matching case is found, the default case (if provided) is executed, or the switch block exits. Switch case statements improve code readability and maintainability, especially when there are multiple options to consider.

Syntax of Switch Case in C

The syntax of a switch case statement in C is as follows:

```
switch(expression) {
    case constant1:
        // statements to be executed if expression matches constant1
        break;
    case constant2:
        // statements to be executed if expression matches constant2
        break;
    // additional case labels as needed
    default:
        // statements to be executed if expression does not match any constant
}
```

- The `switch` keyword is followed by an expression inside parentheses.
- Inside the switch block, each `case` label represents a constant value that the expression is compared against.
- If the expression matches a `case` label, the corresponding statements are executed until a `break` statement is encountered, which exits the switch block.
- If no `case` label matches the expression, the `default` case (if provided) is executed.
- The `default` case is optional and is executed when none of the `case` labels match the expression.
- Example:

```
#include <stdio.h>

int main() {
    int choice;

    printf("Enter a number between 1 and 3: ");
```

```
    scanf("%d", &choice);

    switch(choice) {
        case 1:
            printf("You entered 1\n");
            break;
        case 2:
            printf("You entered 2\n");
            break;
        case 3:
            printf("You entered 3\n");
            break;
        default:
            printf("Invalid choice\n");
    }

    return 0;
}
```

This code demonstrates the use of a switch statement in C. It prompts the user to enter a number between 1 and 3. Depending on the value entered, it prints a corresponding message. If the entered value doesn't match any of the cases (1, 2, or 3), the default case executes, indicating an invalid choice.

Common Programming Errors:

1. Lack of "then" in the if statement.
2. Missing parentheses around conditions in if statements.
3. Incorrect usage of semicolons after the else keyword.
4. Using a single equal sign (=) for comparison instead of the double equal sign (==).
5. Failure to enclose the integer expression following switch in parentheses.

Drill Note:

> Regular practice is essential for mastering programming concepts. Start with simple exercises and progress to more complex tasks. Without consistent practice, programming skills may deteriorate over time.

Lab-0003-Controls-Statments-Condions

C Programming Journey

Difference between break and continue statement in C:

Understanding `break`, `continue`, and `default`

1. `break` **Statement**:

- `break` is used within loops and switch statements.
- When encountered in a loop, it exits the loop immediately, jumping to the statement immediately following the loop.
- In a switch statement, `break` is used to exit the switch block, preventing fall-through to subsequent cases.
- Example:

```
for (int i = 0; i < 5; i++) {
    if (i == 3) {
        break; // exits the loop when i equals 3
    }
    printf("%d\n", i);
}
```

2. `continue` **Statement**:
   - `continue` is used within loops to skip the rest of the current iteration and proceed with the next iteration.
   - It's often used to skip specific conditions or values.
   - Example:

```
for (int i = 0; i < 5; i++) {
    if (i == 3) {
        continue; // skips printing when i equals 3
    }
    printf("%d\n", i);
}
```

3. `default` **Label**:
   - `default` is used in a switch statement as a catch-all case.
   - If none of the cases match the switch expression, the code block under `default` is executed.
   - Example:

```
switch (grade) {
    case 'A':
        printf("Excellent\n");
        break;
    case 'B':
        printf("Good\n");
        break;
    default:
        printf("Invalid grade\n");
}
```