CS 240 Programming in C

Conditional Statements

Feb 9, 2023

Conditional Statements

- Conditional statements are used to execute a block of code based on whether a certain condition is met or not.
- In C language, we have three main types of conditional statements: if, if-else, and switch.

If Statement

- The if statement is the simplest form of a conditional statement.
- The basic syntax of an if statement is:

```
if (condition) {
   // code to be executed if a condition is true
}
```

Example of If Statement

- Consider the following example where we want to check if a number is positive or not:

```
#include <stdio.h>

int main() {
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);
    if (num > 0) {
        printf("The number is positive.");
    }
    return 0;
}
```

If-Else Statement

- The if-else statement is an extension of the if statement, where we can execute a block of code if the condition is true and a different block of code if the condition is false.
- The basic syntax of an if-else statement is:

```
if (condition) {
   // code to be executed if a condition is true
} else {
   // code to be executed if a condition is false
}
```

Example of If-Else Statement

- Consider the following example where we want to check if a number is positive or negative:

```
#include <stdio.h>

int main() {
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);
    if (num > 0) {
        printf("The number is positive.");
    } else {
        printf("The number is negative.");
    }
    return 0;
}
```

Switch Statement

- The switch statement is used to execute a block of code based on the value of a variable or expression.
- The basic syntax of a switch statement is:

```
switch (expression) {
    case value1:
        // code to be executed if expression == value1
        break;
    case value2:
        // code to be executed if expression == value2
        break;
    ...
    default:
        // code to be executed if the expression does
        // not match any case
}
```

Example of Switch Statement

- Consider the following example where we want to determine the day of the week based on a number:

```
#include <stdio.h>
int main() {
    int day;
    printf("Enter a number (1-7): ");
    scanf("%d", &day);
    switch (day) {
        case 1:
            printf("Monday");
            break;
        ...
        case 7:
          printf("Sunday");
          break;
}
```

Conclusion - Conditional Statements

- Conditional statements are used to make decisions in a program based on the result of a condition.
- The if statement is used to execute a block of code only if the condition is true.
- The if-else statement is used to execute one block of code if the condition is true and another block of code if the condition is false.
- The switch statement is used to execute a block of code based on the result of a matching expression.
- By using these statements, you can control the flow of your program and make it more dynamic and responsive to different inputs and conditions.

Loops

- A loop is a control structure that allows code to be executed repeatedly based on a certain condition.
- There are two main types of loops in programming: for loops and while loops.

For Loop

- Syntax:
  for (initialization; condition; increment/decrement)
  { code to be executed }
- Example:

```
for (int i = 0; i < 10; i++) {
    printf("\%d\n", i);
}
```

- In this example, the loop will start with i=0, then it will check if i is less than 10. If it is, it will execute the code inside the loop (printf statement), then it will increment i by 1 and repeat the process.

While Loop

- Syntax:
  while (condition)
  { code to be executed }
- Example:

```
int i = 0;
while (i < 10) {
    printf("\%d\n", i); i++;
}
```

- In this example, the loop will start with i=0, then it will check if i is less than 10. If it is, it will execute the code inside the loop (printf statement), then it will increment i by 1 and repeat the process until i is no longer less than 10.

Do-While Loop

- Syntax:
  do { code to be executed }
  while (condition);
- Example:

```
int i = 0;
do {
    printf("\%d\n", i);
    i++;
} while (i < 10);
```

- In this example, the code inside the loop (printf statement) will be executed once before checking the condition (i $<$ 10). If the condition is true, the code inside the loop will be executed again and the process will repeat until the condition is false.

Nested Loops

- Syntax:
  do { code to be executed }
  while (condition);
- Example:

```
for (int i = 0; i < 10; i++) {
    for (int j = 0; j < 5; j++) {
        printf("%d %d\n", i, j);
    }
}
```

- In this example, there are two for loops nested inside each other. The outer loop will execute 10 times and for each iteration, the inner loop will execute 5 times. The final output will be 10 pairs of numbers, with the first number being the iteration number of the outer loop and the second number being the iteration number of the inner loop.

Break and Continue

- The break statement is used to exit a loop early, while the continue statement is used to skip the current iteration of a loop and move on to the next one.
- Syntax:

```
for (int i = 0; i < 10; i++) {
    if (i == 5) {
        break;
    }
    printf("\%d\n", i);
}
```

- In this example, the loop will start with i=0 and continue until i reaches 5. When i reaches 5, the break statement will be executed and the loop will exit

Goto

- ‘Goto‘ allows a program to transfer control directly to another line of code, bypassing any intermediate code.
- Syntax:

```
goto label;
...
label: statement;
```

- "label" is the identifier that represents the target statement where control will be transferred.
- The "goto" statement can be used in a variety of situations where direct control transfer is needed.
- For example, it can be used to escape from nested loops, to transfer control to an error-handling routine, or to simplify complex control structures.
- It can simplify complex control structures, but it can also lead to code that is difficult to maintain and understand.

Conclusion - Loops

- Loops are an essential aspect of programming and are used to repeat a block of code multiple times.
- There are various types of loops available in C, including for, while, and do-while loops.
- The conditional statements like if-else, switch, and the ternary operator can be used to control the flow of loops.
- The break and continue statements can be used to interrupt or skip the execution of a loop.
- Loops are an indispensable part of C programming, and an understanding of the various types of loops and control statements helps in writing efficient and effective code.