Programming Fundamentals

Muhammad Ateeq

[Updated: 28-02-2023]

Making Decisions

Building Blocks of Logic

What is three main building blocks of logic used to write computer programs?
Sequence, selection, and iteration are three fundamental programming concepts that are used to control the flow of execution in a program:

> **Sequence**: Sequence refers to the order in which instructions are executed in a program. In a sequential program, each instruction is executed in order, one after the other.
> **Selection**: Selection is the process of making decisions in a program based on certain conditions. This is typically done using conditional statements such as "if-else" statements, which allow the program to execute different instructions based on whether a condition is true or false.
> **Iteration**: Iteration, also known as looping, is the process of repeating a sequence of instructions a certain number of times, or until a certain condition is met. This is typically done using loop structures such as "while" loops or "for" loops, which allow the program to repeat a block of code until a certain condition is met.

Together, these three concepts form the basic building blocks of most computer programs and allow programmers to control the flow of execution in a program based on certain conditions and requirements.

What is sequence?
In a C++ program, sequence refers to the order in which statements are executed. In a sequential program, each statement is executed in the order in which it appears in the program, one after the other.
For example, consider the following C++ program:

```
#include <iostream>
Using namespace stdl
int main() {
    cout << "Hello, ";
    cout << "world!";
    return 0;
}
```

In this program, the first statement is `cout << "Hello, ";`, which outputs the string `"Hello, "` to the console. The second statement is `cout << "world!";`, which outputs the string `"world!"` to the console. Finally, the program ends with the statement `return 0;`, which terminates the program. When the program is executed, the statements are executed in the order in which they appear, resulting in the output `"Hello, world!"` being displayed in the console.

Therefore, in C++, sequence is an important concept that determines the order in which statements are executed in a program, and it is crucial for creating programs that perform the desired operations in the correct order.

What is selection?

The steps to solve a problem are not always sequential. Sometime, based on the situation, we need to execute different statements. Selection is the construct that allows us to do that. Selection is important in C++ because it allows programmers to write programs that can make decisions based on certain conditions, and execute different blocks of code accordingly. This allows for more flexible and versatile programs that can adapt to different situations and input values.

In C++, selection is the process of making decisions in a program based on certain conditions. This is typically done using conditional statements such as `if`, `else if`, and `else`. The `if` statement in C++ allows the program to execute a certain block of code if a certain condition is true. For example:

```
int x = 10;
if (x > 5) {
    cout << "x is greater than 5";
}
```

In this example, the program will execute the code inside the curly braces `{}` only if the condition `x > 5` is true. Since `x` is equal to `10`, which is greater than 5, the code inside the curly braces will be executed and the message `"x is greater than 5"` will be printed to the console.

The `else if` and `else` statements allow the program to execute different blocks of code depending on whether certain conditions are true or false. For example:

```
int x = 10;
if (x > 5) {
    cout << "x is greater than 5";
} else if (x == 5) {
    cout << "x is equal to 5";
} else {
    cout << "x is less than 5";
}
```

In this example, the program will first check whether the condition `x > 5` is true. Since it is true, the code inside the first block of the `if` statement will be executed, and the message `"x is greater than 5"` will be printed to the console.

If the condition `x > 5` had been false, the program would have moved on to the `else if` statement to check whether the condition `x == 5` is true. If that condition had been true, the code inside the second block of the `if` statement would have been executed, and the message `"x is equal to 5"` would have been printed to the console.

If neither the first nor the second condition had been true, the program would have executed the code inside the `else` block, and the message `"x is less than 5"` would have been printed to the console.

Is there any other way to carry out conditional execution in C++?
In C++, `switch` is a control statement that allows the program to execute different blocks of code based on the value of a variable or expression. `switch` statements are often used as an alternative to multiple `if` statements, particularly when there are a large number of possible conditions.

The basic syntax of a `switch` statement is as follows:

```
switch(expression) {
    case value1:
        // code to be executed if expression == value1
        break;
    case value2:
        // code to be executed if expression == value2
        break;
    // more cases can be added here
    default:
        // code to be executed if none of the cases match
        break;
}
```

In a `switch` statement, the `expression` inside the parentheses is evaluated, and then the program jumps to the `case` that matches the value of the `expression`. If none of the `case` values match the expression, the program executes the code inside the `default` block.

For example, consider the following C++ program that uses a `switch` statement:

```
#include <iostream>
Using namespace std;
int main() {
    int day = 3;

    switch(day) {
        case 1:
            cout << "Monday";
            break;
        case 2:
            cout << "Tuesday";
            break;
        case 3:
            cout << "Wednesday";
            break;
        case 4:
            cout << "Thursday";
            break;
        case 5:
            cout << "Friday";
            break;
        default:
            cout << "Invalid day";
            break;
    }

    return 0;
}
```

In this program, the variable `day` is set to `3`, and the switch statement checks the value of `day`. Since `day` is equal to `3`, the program jumps to the third `case`, which prints the string `"Wednesday"` to the console. The program then executes the `break` statement, which exits the `switch` statement.

`switch` statements can be useful for organizing code that would otherwise require multiple `if` statements, particularly when there are a large number of possible conditions. However, it is important to note that `switch` statements can be less efficient than `if` statements for small numbers of conditions, and should be used carefully.