Flow of Control

Overview

Statement normally execute sequentially. The first statement in a block is executed first, followed by the second, and so on. Programming languages provides various flow-of-control statements that allow for more complicated execution paths.

While Statement

A while statement repeatedly executes a section of code as long as a given condition is true. We can use a `while` to write a program to sum the numbers from 1 to 10:

```
#include <iostream>
int main()
{
    int sum = 0 , val = 1;
    while (val <= 10){
        sum += val;
        val++;
    }
    std::cout << "Sum of 1 to 10 is " << sum << std::endl;
    return 0;
}
```

The program will prints

```
Sum of 1 to 10 is 55
```

Key points

The `+=` operator add its right-hand operand to its left-hand operand and stores the results in the left-hand operand. It is equivalent to

```
sum = sum + val;
```

The `++` operator aka the increment operator adds `1` to its operand. `val++` is the same as writing `val = val + 1`

> **Note**
>
> The `val++` (Post-increment), while the `++val` (Pre-increment) although, they seems pretty similar they behave differently.

For Statement

The for statement, is usually used when you know exactly how many times you want to loop through a block of code, use the `for` loop instead of a `while` loop. We can rewrite the previous program using a `for` loop to sum the numbers from to 10:

```
#include <iostream>
int main()
{
    int sum;
    for(int val = 1; val <= 10; val++)
        sum += val;
    std::cout << "Sum of 1 to 10 is " << sum << std::endl;
    return 0;
}
```

The program will print

```
Sum of 1 to 10 is 55
```

Each `for` statement has two parts, a *header* and a *body*. The header controls how often the body is executed. The header itself consist of three parts:

- init-statement
- condition
- expression

The overall execution flow of this `for` is:

1. Create `val` and initialize it to 1.
2. Test whether `val` is less than or equal to 10. If the test succeeds, execute the `for` body. If the test fails, exit the loop and continue execution with the first statement following the `for` body.
3. Increment `val`.

4. Repeat the test in step 1, continuing with the remaining steps as long as the condition is true.

If Statement

Like most languages, C++ provides an if statement that support conditional execution.

```
using namespace std;
int main()
{
        int choice;
        cout << "Select 1 or 2 ";
        cin >> choice;
        if(choice == 1){
                cout << "Nice";
        }
        else if(choice == 2){
                cout << "Not bad";
        }
        else{
                cout << "Wrong choice son";
        }
}
```

Warning

C++ uses = for assignment and == for equality. Both operator can appear inside a condition. It is a common mistake to write = when you mean == inside a condition.