while & do-while loops

This lesson introduces the while and do-while loops in C++. It uses coding examples to show their implementation and explain their workings.

We'll cover the following

Loops allow a programmer to execute the same block of code repeatedly. We will make heavy use of conditional statements in this section.

The while Loop

The `while` *loop* is really the only necessary repetition construct. The `for` loop, coming up, can be duplicated using a `while` loop, and with more control. A simple negation can perform the same behavior as *do-while* loop.

The syntax is as follows:

```
while ( condition ) {
    //body
}
```

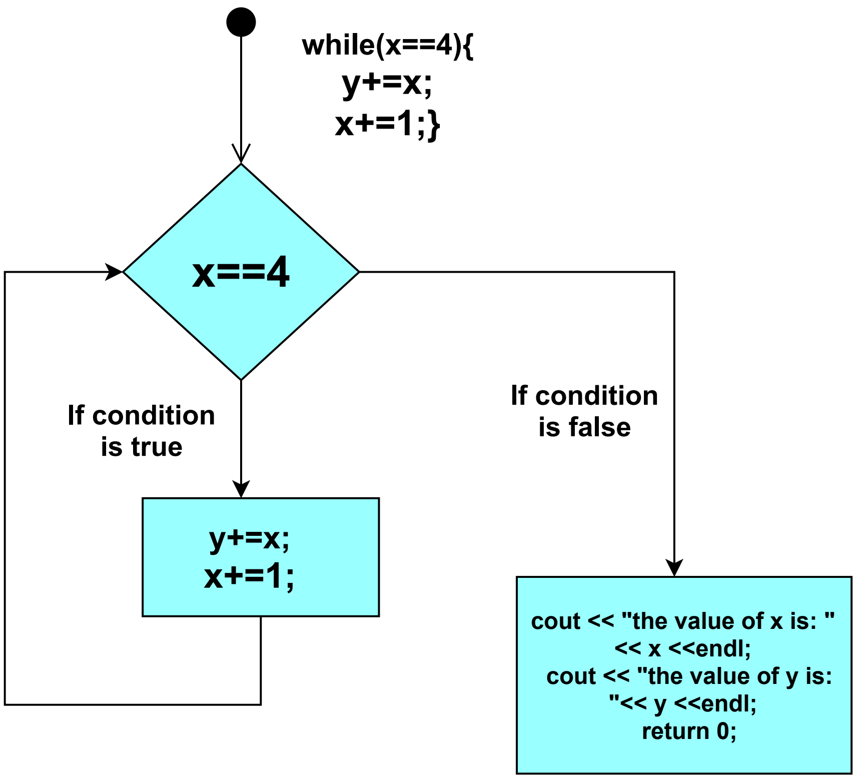
Again, the **curly braces** surrounding the *body* of the `while` *loop* indicate that *multiple* statements will be executed as part of this *loop*.

Here's a look at the `while` loop code:

```
#include <iostream>
using namespace std;
int main(){
  int x=4;
  int y;
  int iterations=1;
  cout << "value of x at start is: " << x <<endl;
  cout << "value of y at start is: " << y <<endl;
  while ( x == 4 ) {   // the while loop will run till x==4 condition is being met
    y += x;
    cout << "value of y in iteration " << iterations << " is: " << y <<endl;
    x += 1;
    cout << "value of x in iteration " << iterations << " is: " << x <<endl;

    iterations++;
  }
  cout<< "while loop breaks"<<endl;
  cout << "the value of x is: "<< x <<endl;
  cout << "the value of y is: "<< y <<endl;
  return 0;
}
```

Below is an *illustration* of the code above to help you better understand the logic.



Flow Chart For While Loop Code

If the `while` *loop* code looked like this instead:

```
while ( x == 4 ) { //since x is not being changed inside the while loop you will get stuck
    y += x;        // in an infinte loop as the condiiton will always be met
}
x += 1;
```

There would be a problem.

According to what is written, even though the *second* line after the `while` was *incremented*, only the *first* line corresponds to the `while` *loop*. This is a huge problem because the variable involved in the condition `(x)` does **not** change, so it problem because the variable involved in the condition `(x)` does **not** change, so it

will always evaluate to *true*, making this an **infinite** loop. This could be alleviated by containing all statements intended to be a part of the loop body in `{ }`.

The do...while Loop

The **do...while** loop is nearly identical to the `while` loop, but instead of checking the *conditional* statement before the loop starts, the **do...while** loop checks the *conditional* statement **after** the *first* run, then continuing onto another iteration.

The *syntax* is as follows:

```
do {
    //body
} while (condition);
```

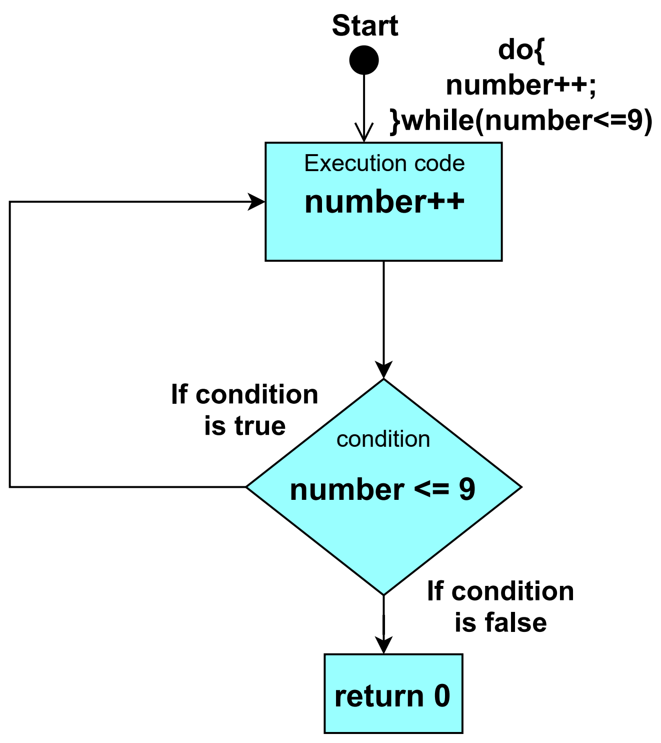
As you can see, it will run the loop **at least** once before checking the conditional.

> **Note:** The **do...while** loop is still haunted by *infinite* loops, so exercise the same caution with the **do...while** loop as you would with the `while` loop. Its usefulness is much more limited than the `while` loop, so use this only when necessary.

Below is an example showing how to implement the **do...while** loop in C++.

```
#include <iostream>
using namespace std;
int main() {
    int number=5;
    do {
        cout<<"Value of number is: "<<number<<endl;
        number++;
    } while(number<=9);   // the contition is being checked after the first run
    return 0;
}
```

Below is an *illustration* of the code above to help you better understand the logic.



Flow Chart For Do While Loop Code

When is do-while used?

A *do-while* loop is used where your loop should execute **at least one** time.

For example, let’s consider a scenario where we want to take an *integer* input from the user until the user has entered a **positive** number. In this case, we will use a **do-while** as we have to run loop **at-least once** because we want input from user at-least once. This loop will continue running until the user enters a **positive** number.

That’s all the major stuff you needed to know about the workings of `while` and **do…while** loops in C++. Let’s learn about `for` loops in the next lesson.