Increment and Decrement Operators, First Loops

https://csci-1301.github.io/about#authors

May 22, 2024 (01:30:40 PM)

Contents

1 Increment and Decrement Operators **1**
- 1.1 Preamble . . . 1
- 1.2 Observe and Confirm . . . 2
- 1.3 Exercise . . . 2

2 First While Loops **3**

3 Pushing Further (Optional) **3**

This lab serves multiple goals:

- To introduce increment and decrement operators,
- To understand the difference between prefix and postfix operators,
- To help you implement your first `while` loops, and
- (Optional) to write your first complex loops.

1 Increment and Decrement Operators

1.1 Preamble

Start by answering the following, assuming an `int` variable `i` has been initialized and that its current value is 5.

1. What does `i++` do to `i`?
2. What does `i--` do to `i`?
3. In your own words, can you explain the difference between `++i` and `i++`?

Solution:

1. `i++` increments the value of `i` by 1; if the value of `i` was 5, it would become 6.
2. `i--` decrements the value of `i` by 1; if the value of `i` was 5, it would become 4.
3. An explanation can be found in this post[1]. In short, `++i` "gives back" the value of `i` *after* it had been incremented by 1, while `i++` "gives back" the value of `i` *before* it has been incremented. This makes a difference if `i++` or `++i` is part of a larger statement. The next exercises will illustrate this principle.

[1] https://stackoverflow.com/q/24853

1.2 Observe and Confirm

Download, extract, open in your IDE, and read the following code[2]. Then, compile and execute it. Study the output carefully to make sure you understand the mechanism of the increment and decrement operators.

Compare your answers from the previous section to what you observe in the output. Do your answers correspond to what you observe in the output?

1.3 Exercise

For each of the following, determine the final value of `n` and `x`.

```
int x = 5;
int n = x++;
```

```
int x = 5;
int n = ++x;
```

```
int x = 5;
int n = x + x++;
```

```
int x = 5;
int n = ++x + ++x;
```

Solution:

You can download, extract, open in your IDE, and execute the following code[3] to check that the answers are:

```
n is now 5.
x is now 6.
n is now 6.
x is now 6.
n is now 10.
x is now 6.
n is now 13.
x is now 7.
```

If you read the source code, you can see that the order of evaluation is quite difficult to follow. Actually, the way the sum operator `+` and the increment operator `++` interact is sometimes *undefined*, meaning that the result may vary from one programming language to another (or worst, from one *version* to another): you can read more about this here[4] of there[5] (Disclaimer: those are *very* technical discussions, much more advanced than the official documentation[6] could lead to think).

In short: software developers generally avoid this type of "nested" operations altogether, and use increment or decrement operators more scarsely!

[2] labs/IncrementDecrement/IncrementExample.zip
[3] labs/IncrementDecrement/IncrementSolution.zip
[4] https://stackoverflow.com/a/3458842
[5] https://stackoverflow.com/a/4176333
[6] https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/arithmetic-operators#increment-operator-

2 First While Loops

The following asks you to write a series of **while** loops performing simple tasks. The solution to the first question is in this archive[7], but it is recommended to try it on your own first.

1. Write a **while** loop that displays the integers between 1 and 100 on the screen with a space between each number.
2. Write a **while** loop that displays the integers between 100 and -100 on the screen, in decreasing order, with a space between each number.
3. Write a **while** loop that displays the $*$ (asterisk symbol) character 100 times on the screen.
4. Modify your previous loop, so that a new line character is displayed on the screen every time 10 asterisk symbols have been displayed on the screen.

To clarify, your program should display the following on the screen (this example has a space after each asterisk symbol for display purposes):

```
* * * * * * * * * *
* * * * * * * * * * *
* * * * * * * * * * *
* * * * * * * * * * *
* * * * * * * * * * *
* * * * * * * * * * *
* * * * * * * * * * *
* * * * * * * * * * *
* * * * * * * * * * *
* * * * * * * * * * *
```

3 Pushing Further (Optional)

Here are additional pattern problems, similar to the last one from the previous section. You are asked to generate these patterns using a **while** loop for each.

1. Triangle:

```
*
* *
* * *
* * * *
* * * * *
* * * * * *
* * * * * * *
* * * * * * * *
* * * * * * * * *
* * * * * * * * * *
```

1. Triangle of numbers

```
1
222
33333
4444444
555555555
```

[7] labs/IncrementDecrement/FirstLoop.zip

1. Upside-down binary triangle

```
1010101
10101
101
1
```