Week 5

Intro to Computer Science - CSI 1430

Hello and welcome to the weekly resources for CSI 1430: Introduction to Computer Science! This week's resource will be covering the course material that will be covered in week 5 of the course.

This week's review will be a little bit lighter since there is not a lot that is new to go over, but make sure you are preparing for upcoming exams and staying on top of coding assignments. **Just as a reminder:** These documents tend to be super good in going over concepts right before exams, as they are a concise and to the point review of the materials. But remember, this document cannot cover everything, so make sure you get help on any specific concepts you struggle with as well as go over lecture notes, programs, and the textbook.

Reminder: If you have any questions about these study guides, group tutoring sessions, private 30 minutes tutoring appointments, the Baylor Tutoring Youtube channel or any tutoring services we offer, please visit our website https://baylor.edu/tutoring or call our drop in center during open business hours. Monday-Thursday 9am - 8pm on class days (254) 710-4135.

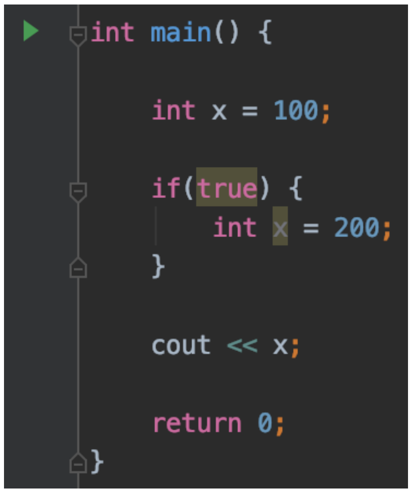
Keywords: *Variable Scope, Pre & Post Incrementing and Decrementing*

TOPIC OF THE WEEK

Variable Scope

Variable scope refers to the visibility of your variables to certain parts of your programs, as well as how long they stick around before they go "out of scope". Understanding this is crucial to writing code that does what it is intended to do.

Where you declare a variable is super important. Take this chunk of code for example:



```
int main() {

    int x = 100;

    if(true) {
        int x = 200;
    }

    cout << x;

    return 0;
}
```

In this example, the variable x is being declared at the beginning of the program with the value of 100. Then, as soon as the conditional statement is stepped into, the variable x will be reinitialized, and set to the value of 200. However, when the conditional body is stepped out of...the variable x, which was declared to 200, will go out of scope, and in a way revert back to the previous variable named x, which has the value of 100 stored in it. Therefore, once the program goes to display the value of x to stdout...then the value printed will be 100.

This can be a very confusing concept, and you have to be super careful to not let anything like this happen when writing your programs. Luckily, there are two really good steps that I tend to follow to make sure these issues are combatted before they are ever created.

1. Make sure you declare all of the variables you use in your program at the very top of the main(). This helps keep all of the variables declared in the same scope as one another, and there is no confusion with "one variable stepping on another variable's toes".
2. If you run into a situation where you have to create a variable that cannot be put at the beginning of a program, then make sure it is named in a way where it does not collide with the naming of another variable (and that it is properly commented on its intentions).

OTHER IMPORTANT TOPICS

Pre & Post Incrementing and Decrementing

There are two other basic operators that are worth mentioning, the increment and decrement operators. The purpose of these two operators is to basically perform the same function as adding one, or taking one away from a non-floating point number.

Operator	Explanation
++	Adds 1 to the current number
--	Subtracts 1 from the current number

These operators can be appended to the front or back of a variable to automatically change the value of a variable (without having to assign it back to itself). For example x++ or ++x achieve the same thing as x = x + 1.

While these 2 operators seem to be acting with the same exact behavior, they do in fact have 2 different functionalities. When either the increment or decrement operator is appended to the front of a variable, the operation will happen **before** the variable is evaluated. However, if it is appended to the back, then the variable will first be evaluated, **and then** the operation will occur. While this does not make much of a difference in the programs that you are writing right now, you will start to see how these can be different once loops are introduced.

CHECK YOUR LEARNING

1. **How would you pre-decrement the value of a variable named x?**
2. **Where should most variables be defined in order to eliminate variable scoping issues?**
3. **Write an equivalent statement to the following:** *x*++

THINGS STUDENTS MAY STRUGGLE WITH

1. Most of the time, when you are writing conditional statements (and later loops) it does not matter whether or not you are pre or post incrementing/decrementing. However, understanding the difference in how they operate is crucial in situations where code might be a little more complex and depend on which you are using.
2. "Scope" of a program is a term that is hard to understand without a lot of practice debugging code. There is a lot of program behavior that can happen completely different than what you would expect it to do intuitively. I suggest spending some time not only learning the different areas in which scope is applied during programs, but also stepping through the debugger to see how a variable is being affected every step of the way throughout a program.

Thanks for checking out these weekly resources! Don't forget to check out our website for group tutoring times, video tutorials and lots of other course resources: https://baylor.edu/tutoring. The answers to the 'Check Your Learning' questions are below.

Answers:

1. ++x
2. At the beginning of the main, a.k.a at the top of the program.
3. x = x + 1;