Week 3

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 3 of the course.

This course is designed to give you an introduction to C++, code design, and general computer science concepts. These resources will be available every week to offer a review of some sections of the course material, so make use of them as needed. However, this document cannot cover everything in a given week, so make sure you get help on any specific concepts you struggle with. Remember, these resources should be used as a review and to help answer any general questions, and NOT as a substitute for reading the textbook or attending lectures. If you have any questions regarding CS 1430 or computer science in general, feel free to reach out to the tutoring center; but don't forget that your course instructor is the most important resource you have in this introductory programming course.

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.

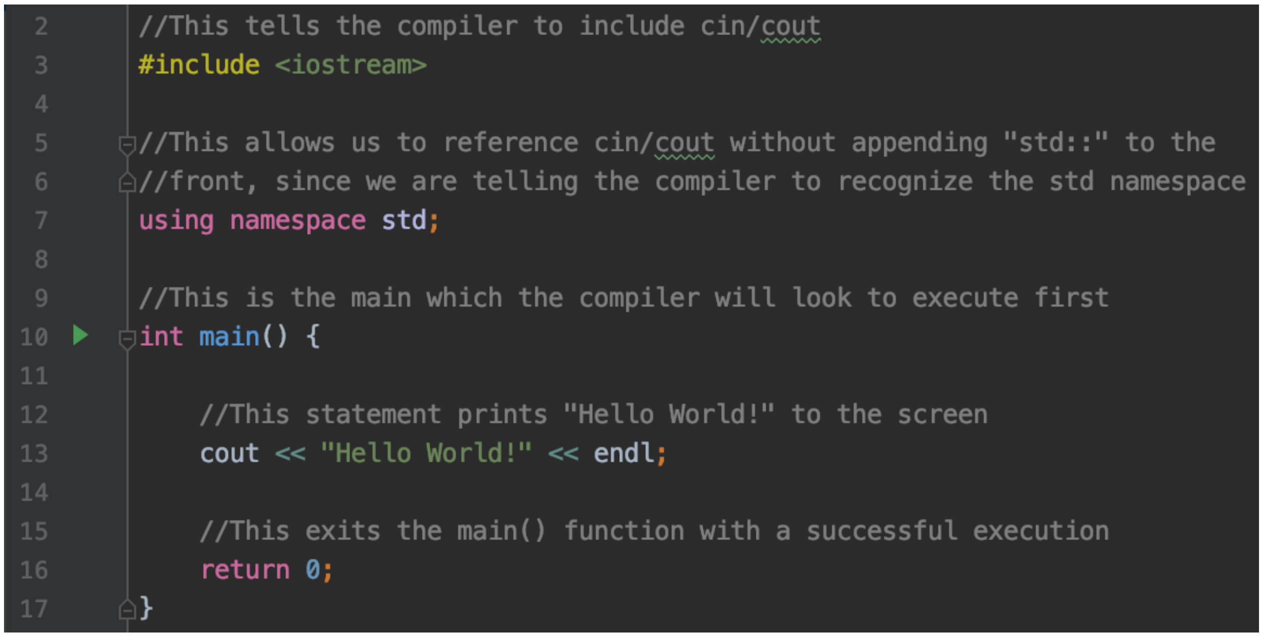
Keywords: Basic Input & Output, Variables, Arithmetic Operators

---

TOPIC OF THE WEEK

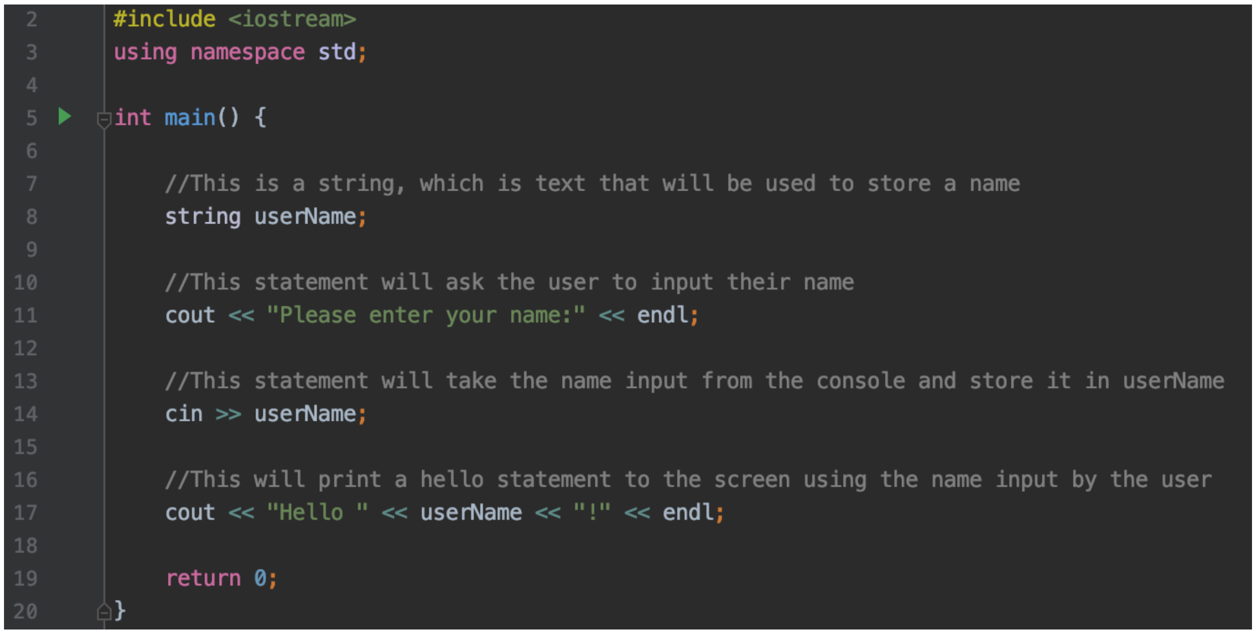
Basic Input & Output

Whether this is your first time programming or just your first time using C++, there are a few new standards and things to understand that you will need to get accustomed to. The first instance of that in most new coding languages is how input and output are handled. In C++, the object *cout* is used to display things to the screen (output), and the object *cin* is used to receive user input. Output can be shown from a basic hello world program where we can explore some of those principles, as well as some other structural components within the C++ language.



```cpp
//This tells the compiler to include cin/cout
#include <iostream>

//This allows us to reference cin/cout without appending "std::" to the
//front, since we are telling the compiler to recognize the std namespace
using namespace std;

//This is the main which the compiler will look to execute first
int main() {

    //This statement prints "Hello World!" to the screen
    cout << "Hello World!" << endl;

    //This exits the main() function with a successful execution
    return 0;
}
```

An example of input can be seen through another simple program that tells a user hello based on their name.



```cpp
#include <iostream>
using namespace std;

int main() {

    //This is a string, which is text that will be used to store a name
    string userName;

    //This statement will ask the user to input their name
    cout << "Please enter your name:" << endl;

    //This statement will take the name input from the console and store it in userName
    cin >> userName;

    //This will print a hello statement to the screen using the name input by the user
    cout << "Hello " << userName << "!" << endl;

    return 0;
}
```

Pay special attention to the places in the code where "//" appears. These places in the code are called comments. Comments have absolutely no effect on how a program runs, however they are super useful to include, as they are a way to write notes about how a program works.

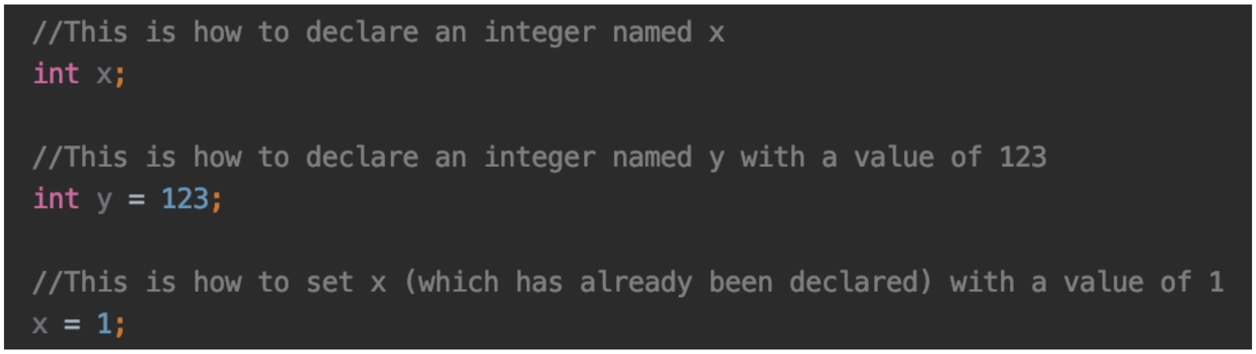
OTHER IMPORTANT TOPICS

Variables

Another important concept within most programming languages is the concept of a variable. A variable is responsible for holding a specific type of value(s). The most common forms of variables you will use are:

Variable Type	**Description**	**Examples**
bool	Stores a value of *true* or *false*	true, false
char	Stores a single character	‘a’, ‘1’, ‘$’
string	Stores a string of characters a.k.a. A text value	“Hello”, “CS1430”
int	Stores a integer, whole number value	1, 123, -12
double	Stores a floating point, decimal value	3.14, 1.0, -123.0

To initialize a variable, you put the variable type, followed by the variable name. Once a variable is declared, you can change the value of it. You can do this on the same line as the declaration, or on a separate line (excluding the variable type since it has already been declared).



```
//This is how to declare an integer named x
int x;

//This is how to declare an integer named y with a value of 123
int y = 123;

//This is how to set x (which has already been declared) with a value of 1
x = 1;
```

One thing to note, is that if you declare a variable and don’t set a value to it, it will generate a junk value to it (in most cases) and be properly set to what you want it to contain before it is used.

Arithmetic Operators

Setting variables alone in most instances will not be enough. To perform any sort of computation, arithmetic operators will have to be invoked. A list of the most common operators are described in the table below:

+	Adds two variables
-	Subtracts two variables
*	Multiplies two variables
/	Divides two variables
%	Returns the remainder between two numbers (after division)
==	Returns *true* if two values are equal to each other, *false* otherwise
!=	Returns *true* if two values are not equal to each other, *false* otherwise
<	Returns *true* if the left variable is less than the right variable, *false* otherwise
>	Returns *true* if the left variable is greater than the right variable, *false* otherwise
<=	Returns *true* if the left variable is less than or equal to the right variable, *false* otherwise
>=	Returns *true* if the left variable is greater than or equal to the right variable, *false* otherwise

Learning the uses of different operators will become more apparent as you start needing them for different programs, but it's good to just get familiar with the different types out there.

CHECK YOUR LEARNING

1. **What is the object used to receive user input?**
2. **What variable type can hold a true or false value?**
3. **True or false, you need to redeclare a variable every time you change its value?**
4. **What arithmetic operator returns the remainder of division between two variables?**

THINGS STUDENTS MAY STRUGGLE WITH

1. Make sure you take some time to look up the different components used within C++. Actually understanding everything you put in a program is far more beneficial than just "trusting that it works".
2. It's easy to be lazy when programming and not comment and come up with "junk" variable names, but taking the time to properly take notes of what is happening in your programs and making meaningful variable names will do you a better service in the long run when your programs start to be a lot more complicated.
3. Depending on the type of variable used, the behavior for different arithmetic operators can behave certain ways or can even not exist (eg. string / string).
4. The modulo, %, operator can be slightly confusing for those who have never seen it before. It can only be used between non-floating point variables and will serve as the remainder of division between the numbers. An example of this is 17 % 6. Since 17 divided by 6 results in 2 remainder 5, then the answer of 17 % 6 is 5.
5. You can store the result of an arithmetic operation the same as when you store a number. For example, if you want to store the sum of variables numberOne and numberTwo in the variable result (assuming all three variables are int's and already declared) then you would just perform the following statement
    - result = numberOne + numberTwo;

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. cin
2. bool, or a boolean variable
3. False (Once a variable is declared, it will stay declared until the end of the program)
4. %, or the modulo operator