CSI 1430
Week 3 Resource

Colin Burdine

1/31/21

Major Topics:

1. Basic Input and Output (I/O) in C++
2. Variable Types
3. Arithmetic Operations
4. Conditional Branching

Keywords: *I/O, variables, types, arithmetic, operators, branching*

---

0 Author's Note

Hello, students! My name is Colin Burdine and I will be your guide this semester as we embark on the fantastic journey that is CSI 1430! Throughout this course, you will be learning the basics of programming with the C++ language. Every week of semester, I will be posting resources like this one to help provide a concise review of some of the key topics you will be covering each week in class. My intent is that these resources will serve as a quick reference or review, not as a substitute for reading the textbook and attending lecture. I will do my best to address some frequently asked questions in these resource, but do understand that your professor will be able to provide more directed answers to questions than I can on a single sheet of paper.

This semester I will also be leading the group tutoring sessions for this course, which will be held online every Monday from 7:00pm to 8:00pm CST. For more information on how to sign up for these sessions, go to: https://www.baylor.edu/tutoring.

1 Notes on Major Topics:

1.1 Basic Input and Output (I/O) in C++

When learning a programming language for the first time, there will be many idioms and standards associated with that language that you will need to become accustomed to. One of the first things that your book covers is how basic input and output (I/O) works in C++. Below, we have a basic "Hello World" program that simply prints the text `Hello World!` to the screen:

```
1  // This tells the compiler (the program that interprets your code)
2  // to include the cin and cout objects:
3  #include <iostream>
4
5  // This tells the compiler that we will be referencing cin/cout
6  // by name, since cin and cout are contained the "std" namespace:
7  using namespace std;
8
9  // This is the main function; it is always executed first:
10 int main(){
11
12     // This prints "Hello World" to the console:
13     cout << "Hello World!" << endl;
14
15     // This exits from the main() function:
16     return 0;
17 }
```

Note the lines that begin with "`//`". These lines are called *comments*, and they have no effect on the execution of a program. They are inserted by the programmer to explain what a program does and how it works. When programming, is is always a good idea to use lots of comments.

1.2 Variable Types

In C++, a *variable* is a named entity that is capable of holding a single value. By using the name of a variable, we can access the value it holds. In C++, we can initialize variables in the following way:

```
// This sets aside memory to store an integer "x", but the value
// will be some junk value:
int x;

// This sets aside memory to store an integer "x" and sets its value to be
    42:
int x = 42;

// This assigns the value of 43 to "x".
x = 43;
```

Note that above we used the type specifier "`int`" to indicate that we are creating a variable that can only store integer values. There are some other types that we can declare as well:

Type Specifier	Purpose
`bool`	Stores a value that is either `true` or `false`
`int`	Stores an integer value
`double`	Stores a decimal value (e.g. `3.14159` or `2.998e+08`)
`char`	Stores a single character value (e.g. `'a'` or `'$'`)
`string`	Stores a text value (e.g. `"This is a message"`)

1.3 Arithmetic Operations

The purpose of most computer programs is to perform some kind of data processing. The most common kind of data processing you will encounter is arithmetic (mathematical operations). For example, suppose we want to write a program to divide 23 cookies fairly among five children determine how many are left over. We can do this with the following:

```

// initialize the number of cookies and children:
int numCookies = 23;
int numChildren = 5;

// we want to find the number of cookies per child and the
// number of cookies left over:
int cookiesPerChild;
int cookiesLeftOver;

// use arithmetic operators to determine the quantities we want:
cookiesPerChild = numCookies / numChildren;
cookiesLeftOver = numCookies % numChildren;

// cookiesPerChild is now 4; cookiesLeftOver is now 3
```

Notice the use of `%` and `/` operators. The definition and usage of these operators (and other arithmetic operators) are described in the table below:

Operator	Operand Type	Effect
`+`	(any numeric type)	Adds two numeric variables
`-`	(any numeric type)	Subtracts two numeric variables
`*`	(any numeric type)	Multiplies two numeric variables
`/`	`double`	Divides two floating-point (double) values
`/`	`int`	Returns the quotient of two integer values rounded down to the nearest integer
`%`	`int`	Returns the remainder of the division of two integer values
`==`	(any numeric type)	Returns `true` if the two operands are equal, `false` otherwise
`!=`	(any numeric type)	Returns `true` if the two operands are *not* equal, `false` otherwise
`<`	(any numeric type)	Returns `true` if the left operand is less than the right operand, `false` otherwise
`<=`	(any numeric type)	Returns `true` if the left operand is less than or equal to the right operand, `false` otherwise

1.4 Conditional Branching

In the table above, you might notice that there are some logical operations that return a "yes" or "no" value, which is encoded in C++ as a `bool`, which is short for a *boolean* value. We can use these values to perform *conditional branching*, where we can execute different segments of code depending on the value of some boolean value. Consider the following example where we ask the user to input a number and print whether it is divisible by 7. To perform the conditional branching we use the if-else conditional branch, which executes either the first branch if the condition following the "if" statement is met, or the second branch if it is not met:

```cpp
#include<iostream>

using namespace std;

int main(){
    int number;

    // ask the user for a number:
    cout << "Enter a number: ";

    // read the number from the console:
    cin >> number;

    // Here, we print whether or not the number is divisible by 7.
    // To do this, we check if the remainder of the number divided
    // by 7 (obtained through the % operator) is equal to 0:
    int remainder = number % 7;
    if(remainder == 0){
        cout << "Your number is divisible by seven." << endl;
    } else {
        cout << "Your number is not divisible by seven." << endl;
    }
    return 0;
}
```

Additional References:

(I would highly recommend looking into the following resources:)

1. C++ documentation (cplusplus.com): http://www.cplusplus.com/reference/
2. Draw.io (a great design tool to help with designing a flowchart of your program before you start writing code): https://app.diagrams.net/
3. Booth's Survival Guide (some sage advice for writing programs; requires ECS classnotes login): https://classnotes.ecs.baylor.edu/wiki/Booth%27s_Survival_Rules
4. Baylor's C++ Documentation Standard (how to format your comments on programming assignments; requires ECS classnotes login):

   https://classnotes.ecs.baylor.edu/wiki/Documentation_Standard

---