REVIEW

The C++ Programming Language

Computer programming courses generally concentrate on program design that can be applied to any number of programming languages on the market. It is imperative, however, to apply that design to a particular language. This course uses C++, a popular object-oriented language, for that purpose.

For now, we can think of a C++ program as consisting of two general divisions: header and main. The **header**, or **global section**, gives preliminary instructions to the compiler. It consists of comments that describe the purpose of the program, as well as information on which library routines will be used by the program.

```
// This program prints to the screen the words:
// PI = 3.14
// Radius = 4
// Circumference = 25.12

#include <iostream>
using namespace std;
const double PI = 3.14;

int main()
{
	float radius;
	radius = 4.0;
	cout << "PI = " << PI << endl;
	cout << "Radius = " << radius << endl;
	cout << "Circumference = " << 2 * PI * radius << endl;

	return 0;
}
```

Everything in bold (everything above the `int main()` statement) is considered the header or global section. Everything else is the main section.

Comments are included in every program to document what a program does and how it operates. These statements are ignored by the computer but are most valuable to the programmers who must update or fix the program. In C++, comments begin with `//` which is an indication to the compiler to ignore everything from the `//` to the end of the line. Comments can also cross line boundaries by beginning with `/*` and ending with `*/`. Notice that the first three lines of the also have been written as the following:

```
/*
PI = 3.14
Radius = 4
Circumference = 25.12
*/
```

The next statement, the `#include` statement, indicates which library will be needed by the program.

```
#include <iostream>
```

Recall from Lesson Set 1, that every program needs other modules attached so that needed for each particular programming assignment; however, in time you will learn this task for yourself.

Every C++ program has a **main** function which indicates the start of the executable instructions. Every `main` must begin with a left brace `{` and end with a right brace `}`. The statements inside those braces will be explained as we progress through this lesson.

Memory

Memory storage is the collection of locations where instructions and data that are used by the program are temporarily stored. Recall from Lesson Set 1 that a computer only understands a sequence of 1s and 0s. These are binary digits or **bits (BInary digiTs)**. Eight of these brought together are called a **byte**, which is the most common unit of storage. These chunks of memory can be thought of as hotel mailboxes at the registration desk. The size of each of those boxes indicates the type of mail that can be stored there. A very small mailbox can only hold notes or postcards. Larger mailboxes can hold letters, while even larger ones can hold packages. Each mailbox is identified by a number or name of an occupant. We have identified two very important attributes of these mailboxes: the name or number, which indicates the mailbox that is being referenced, and the size, which indicates what type of "data" can be placed there.

Example: **postcards Jim** is an indication that the mailbox called Jim can only hold postcards, while the statement **packages Mary** indicates that the mailbox called Mary can hold large packages. Memory locations in a computer are identified by the same two attributes: data type and name.

Much of programming is getting data to and from memory locations and thus it is imperative that the programmer tell the computer the name and data type of each memory location that he or she intends to use. In the sample program the statement **float radius** does just that. `float` is a data type that indicates what kind of data can be stored and `radius` is the name for that particular memory location.

Identifiers in C++

Identifiers are used to name variables, constants and many other components of a program. They consist exclusively of letters, digits and the underscore _ character. They cannot begin with a digit and cannot duplicate reserved words used in C++ such as **int** or **if**. All characters in C++ are case sensitive; thus memory locations called `simple`, `Simple`, and `SIMPLE` are three distinct locations. It has become standard practice among programmers to make constants all uppercase and variables predominantly lowercase characters.

The statement **float radius;** in the sample program is contained in the main section. It defines a variable memory location called `radius` that holds a floating point data type (type discussed shortly) which can be changed during the execution of the program.

Both of these statements are called **definitions**. They reserve by name enough memory to hold the data type specified.

Variables, like constants, can be given an initial value when they are defined, but that value is not permanent and can be altered. For example:

```
int count = 7;          // Defines a variable memory location called count that
                        // initially has the value of 7
count = count + 1;      // count is now altered
```

Data Types

As noted earlier, computer memory is composed of locations identified by a data type and a name (like the room number of a hotel mailbox). The data type indicates what kind of data can be stored, thus setting the size of that location.

Integer Data Type

Integers are real numbers that do not contain any fractional component. They take up less memory than numbers with fractional components. C++ has three data types that are integers: **short**, **int** and **long**. The difference is strictly in the amount of memory (bytes) they reserve: `short` reserving the least and `long` reserving the most. Larger integers may need the `long` data type.

The following three statements define integer variables in C++:

```
short count;
int sum;
long total;
```

Floating Point Data Type

In computer science 3 = 3.0 is not a true statement. The number on the left is an integer and the number on the right is a real, or floating point, number (a number that has a fractional component). Although mathematically the two are equal, the computer stores them as different data types. C++ uses both **float** and **double** to indicate floating point numbers, with `double` using more memory than `float`.

The following two statements define floating point variables in C++.

```
float average;
double nationaldebt;
```

Character Data Type

Character data includes the letters of the alphabet (upper and lower cases), the digits 0–9 and special characters such as ! ? . , *. All these symbols combined are called **alphanumeric**. Each character data is enclosed with single quotes to distinguish it from other data types. Thus `'8'` is different than `8`. The first is a character while the second is an integer. The following statement defines a C++ character variable initialized to `'a'`.

```
char letter = 'a';
```

Boolean Data Type

The Boolean data type, named after the mathematician George Boole, allows only two values: true or false, which are internally represented as 0 and non-zero, respectively. The following statement defines a Boolean variable initialized to false.

```
bool found = false;
```

Assignment Operator

The = symbol in C++ is called the **assignment operator** and is read “is assigned the value of.” It assigns the variable on its left the value on its right. Although this symbol looks like an equal sign, do not confuse it with equality. The left hand side must always be a variable. For example, `count = 8;` is a legitimate statement in C++, however `8 = count;` generates a syntax error.

Arithmetic Operators

Programming has the traditional arithmetic operators:

Operation	**C++ Symbol**
addition	+
subtraction	-
multiplication	*
division	/
modulus	%

Integer division occurs when both the numerator and denominator of a divide operation are integers (or numbers stored in variables defined to be integers). The result is always an integer because the decimal part is truncated or "chopped" from the number. Thus 9/2 will give the answer 4 not 4.5! For this reason there are two division operations for integer numbers. The modulus operator, (%) used only with integers, gives the remainder of a division operation. 9/2 gives 4 while 9 % 2 gives 1 (the remainder of the division).

Example:

```
int count = 9;
int div = 2;
int remainder;
int quotient;
quotient = count / div;    // quotient is assigned a 4
remainder = count % div;   // remainder is assigned a 1
```

Review #2 CS 151 SEC# _____ Name _____

Fill-in-the-Blank Questions

1. _____ is a value that does not change during the program's execution.

2. _____ is a data type that only holds numbers with no fractional component.

3. _____ is a data type that holds numbers with fractional components.

4._____ is an arithmetic operator that gives the remainder of a division problem.

5. `cout <<` is an example of the _____ fundamental instruction.

6._____ data types only have two values: true and false.

7. One byte consists of _____ bits.

8. // or /* in C++ indicates the start of a _____.

9. A _____ is a memory location whose value can change during the execution of the program.

10. A _____ can hold a sequence of characters such as a name.