# CS107: C++ PROGRAMMING



Back to 'Study Guides and Review Exercises'

# Unit 1 Study Guide and Review: Introduction and Setup

**1a. Describe the basic history of C++**

1. Describe the role of C in the development of C++.
2. Programming languages are categorized by the programming paradigm they follow. What programming paradigms do C and C++ follow?
3. Briefly describe the OO programming paradigm using 'ideas' and 'objects'.

The history of a programming language helps us understand why it was developed, the new features that it introduced, and how it relates to earlier languages.

To review for the final, watch Jay's "History of Programming". Make sure you know which finding was regarded as the birth of programming. Note the differences in high-level versus low-level programming languages. A high level programming language has heavy abstraction, which means that many of the details of an action are hidden. The more details required of a programmer to provide, the lower the level of the language. In addition, make sure you understand the difference between compilers and interpreters. A compiler analyzes code in detail at the onset and ensures that all code is in order and then is able to execute the code quickly. An interpreter translates the code into an intermediate code and then interprets the code on the fly, as it is being executed.

You should be able to explain object-oriented programming. OOP involves the use of classes, data abstraction, encapsulation, information hiding, inheritance and polymorphism. Make sure that you are familiar with each of these terms. C, which is not object-oriented, provided a language that was portable, flexible, effective, reliable and interactive. The addition of C++ allowed for the object-oriented capabilities. C is largely upward-compatible with C++. A C program will likely compile by a C++ compiler.

**1b. Set up and create a simple C++ project using Eclipse CDT and Ubuntu Linux**

Since Eclipse was originally designed as a Java IDE, some plug-ins are required to adapt its use for C++. In order to use Eclipse for C++, you must install the following:

1. Eclipse should be installed first. This can be downloaded for free.

2. You must set up Eclipse to work with Java before you can adapt it to other languages, so you must install the Java Runtime Environment.
3. Now you are ready to install the Eclipse C/C++ Development toolkit, which will provide you the capability to write code in the C++ language.
4. If you are using a Mac or Linux, you are done. If you are using Windows, you need to install Cygwin- which will provide a Linux-like environment on Windows. This provides the g++, make, and GDB files needed.

**1c. Explain the meaning of simple C++ commands**

Semicolons are a vitally important syntax in C++, and are likely to be the item most overlooked when programming. Also, the use of a return statement is important if your function expects something to be returned. To explore basic C++ commands in more detail, revisit this introduction to C++.

1. How do you use basic input, output commands: cin and cout?
2. How do you define a namespace?
3. What is the purpose of a main function? Creating a main function is vital to C++ operations, because when a program runs, it looks for the main function first, to tell it how to begin processing.

**1d. Use cout and cin objects effectively**

Cout and cin objects are part of the iostream. This video explores the use of cin and cout objects.

1. What is the difference in cin and cout and how are the data flow arrows used to represent input and output? The key to success with cin and cout is to ensure that the data flow arrows are going in the correct direction. The arrows represent the movement of data with cin pointing to the right and cout pointing to the left.
2. When outputting data to the screen, how should the output should be presented? Make sure that literal values are placed in quotes and separated by double arrows from variable data.

**1e. Declare and use variables**

A basic requirement in C++ is that all variables must be declared before they can be used. This page defines variables and how to declare them.

1. What are the rules to construct a valid variable name? What are the specific naming conventions for C++? What are the keywords and identifiers that are part of the C++ language?
2. What are the four types of constants and what are the rules for the construction of integer and String constants?
3. What is the difference between a String and a char variable?
4. What is the difference between a float and a double?
5. What are the four classes of data types supported by C?
    1. primary, user-defined, derived, and the empty data set
6. How are String variables handled differently than char arrays?

This video also explores data types, but includes data types that are not part of C, particularly the boolean variable. In addition, this video demonstrates the specific details on how to declare variables and use them.

**1f. Use conditional and iteration structures in C++**

Conditional structures rely on a set of operators and truth table rules to make decision. These notes explore the use of conditional operators and truth tables. You should be sure to understand each of these fully.

1. What are the conditional operators used in C++?
2. What are the results of basic truth tables?
3. How do you write a switch statement?
4. What are the three different types of loops used in C++ and how are they used?

**1g. Define and use simple functions**

It is important that you understand how to define and use functions that you have created, as well as the use of functions defined in other classes. The Math class is an example of this. This chapter demonstrates the use of methods and how to define them.

1. How do you use predefined functions, such as in the Math class?
2. How do you write a function definition and then use those functions?
3. How do you pass parameters into and out of functions? The resource listed above introduces this concept of passing variables, but a more detailed explanation can be found on this page.

**Unit 1 Vocabulary**

This vocabulary list includes terms that might help you with the review items above and some terms you should be familiar with to be successful in completing the final exam for the course.

Try to think of the reason why each term is included.

- Arguments
- Boolean
- Character
- Classes
- Compiler
- Conditional operator
- Data abstraction
- Data type
- Double
- Float
- Function
- High-level programming
- Information hiding
- Inheritance
- Integer
- Interpreter
- Low-level programming
- Namespace
- Parameters
- Pass by Reference
- Pass by Value

- Polymorphism
- String
- Switch statement
- Truth table
- Variable

Last modified: Wednesday, December 7, 2016, 2:21 PM