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++.
    a. C++ grew out of C and has OO programming features. C++ is largely compatible with C source code. Many, not all, of the constructs of C99 are included in C++.
2. Programming languages are categorized by the programming paradigm they follow. What programming paradigms do C and C++ follow?
    a. C folllows the procedural and imperative programming paradigms.
    b. C++ follows Object Oriented and Generic programming paradigms but can also implement High Level Assembly, procedural, imperative and data abstraction paradigms.
3. Briefly describe the OO programming paradigm using 'ideas' and 'objects'.
    a. We think of the world in terms of things that implement ideas. In the OO paradigm, objects correspond to 'things' and classes correspond to 'ideas'. An object implements a class; similar to the way in which a 'thing' implements and 'idea'.

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.

The ancient analogue computer called the antikythera mechanism is regarded as the birth of programming. Invented in about 1 BC the ancient computer was used in Greece to predict astronomical position and eclipses for calendar and astrological purposes decades in advance.

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.

Class:
In general use, a class is a user-defined type with member functions. In C++, a class is a structure with private instance variables.

Data abstraction:
Data abstraction refers to providing only essential information to the outside world and hiding their background details, i.e., to represent the needed information in program without presenting the details.
Data abstraction is a programming (and design) technique that relies on the separation of interface and implementation.

Encapsulation:
Encapsulation means to divide a large complex program into components (like functions) and isolate the components from each other (for example, by using local variables).

information hiding:
Information hiding is the process of hiding details of an object or function. Information hiding is a powerful programming technique because it reduces complexity.

Inheritance:
Inheritance allows us to define "is-a" relationships, but it cannot be used to implement "has-a" relationships. In order for an inheritance relationship to occur, a class must have an "is-a" relationship with the inherited class. Inheritance types include Single inheritance, Multiple inheritance, Hierarchical inheritance, Multilevel inheritance and Hybrid inheritance. Single inheritance inherits properties and behaviors of only one base class. Multiple inheritance occurs when a derived class inherits properties and behaviors of more than one base class. Hierarchical inheritance occurs when the properties and behaviors are inherited by more than one derived class. Multilevel inheritance occurs when properties and behaviors of a derived class are inherited by another class, resulting in a chain of inheritance.

Polymorphism:
Polymorphism is the ability to take on more than one form. A class can be used through its parent interface while a child class may override some of the parent's methods the abstract methods. If a method is abstract in the parent class, then the child class can develop their own individual method that overrides the parent.

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:

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

   cout is used together with the insertion operator, which is written as << (i.e., two "less than" signs).The << operator inserts the data that follows it into the stream that precedes it. Multiple insertion operations (<<) may be chained in a single statement:

   cin is used together with the extraction operator, which is written as >> (i.e., two "greater than" signs). This operator is then followed by the variable where the extracted data is stored. The extraction operation on cin uses the type of the variable after the >> operator to determine how it interprets the characters read from the input; if it is an integer, the format expected is a series of digits, if a string a sequence of characters, etc.

2. How do you define a namespace?

   Namespaces are used to define a scope, and allow us to group global classes, functions, and/or objects into a single group. To use a namespace, you can either put the keyword namespace in front of any method definition:

   ```
   namespace myAwesomeNamespace {    int a=10, b;}
   ```

   and use the `namespace::` notation:`myAwesomeNamespace::a`to refer to the namespace, or you can use the keyword `using namespace` and the name of the namespace:`using namespace myAwesomeNamespace;`

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?
   a. 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?
   a. 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?
   1. A variable name may consist of letters, digits, and underscores.
   2. A variable name must begin with a letter.
   3. The length of a variable name cannot exceed 31 characters, but should be no more than eight.
   4. Variable names are case-sensitive; upper- and lowercase letters are different.
   5. Variable names may not be a C-reserved word.

2. What are the specific naming conventions for C++?
   1. Start a variable name with lowercase letters.
   2. Try to use meaningful identifiers
   3. Separate "words" within identifiers with mixed upper and lowercase (for example empCode) or underscores (for example emp_code).
   4. For symbolic constants use all uppercase letters (for example #define LENGTH 100, #define MRP 45).

3. What are the keywords and identifiers that are part of the C++ language?

1. Keywords:

alignas (since C++11)	delete(1)	reflexpr (reflection TS)
alignof (since C++11)	do	register(2)
and	double	reinterpret_cast
and_eq	dynamic_cast	requires (since C++20)
asm	else	return
atomic_cancel (TM TS)	enum	short
atomic_commit (TM TS)	explicit	signed
atomic_noexcept (TM TS)	export(1)	sizeof(1)
auto(1)	extern(1)	static
bitand	false	static_assert (since C++11)
bitor	float	static_cast
bool	for	struct(1)
break	friend	switch
case	goto	synchronized (TM TS)
catch	if	template
char	import (modules TS)	this
char8_t (since C++20)	inline(1)	thread_local (since C++11)
char16_t (since C++11)	int	throw
char32_t (since C++11)	long	true
class(1)	module (modules TS)	try
compl	mutable(1)	typedef

concept (since C++20)	namespace	typeid
const	new	typename
consteval (since C++20)	noexcept (since C++11)	union
constexpr (since C++11)	not	unsigned
const_cast	not_eq	using(1)
continue	nullptr (since C++11)	virtual
co_await (coroutines TS)	operator	void
co_return (coroutines TS)	or	volatile
co_yield (coroutines TS)	or_eq	wchar_t
decltype (since C++11)	private	while
default(1)	protected	xor
	public	xor_eq

2. Identifiers:

override (C++11)
final (C++11)
audit (C++20)
axiom (C++20)
transaction_safe (TM TS)
transaction_safe_dynamic (TM TS

4. What are the four types of constants and what are the rules for the construction of integer and String constants?
    1. Types of constants are: Integer Numerals, Floating-Point Numerals, Characters, Strings and Boolean Values.
    2.
        i. Integer Literals: An integer literal can be a decimal, octal, or hexadecimal constant. A prefix specifies the base or radix: 0x or 0X for hexadecimal, 0 for octal, and nothing for decimal. An integer literal can also have a suffix that is a combination of U and L, for unsigned and long, respectively. The suffix can be uppercase or lowercase and can be in any order.
        ii. String Literals: String literals are enclosed in double quotes. A string contains characters that are similar to character literals: plain characters, escape sequences, and universal characters.You can break a long line into multiple lines using string literals and separate them using whitespaces.

5. What is the difference between a String and a char variable?
    i. A char is a fundamental datatype and a char variable holds a single character. In C++ Strings are objects that represent sequences of characters. Since Strings are, in fact, sequences of characters, we can represent them also as plain arrays of elements of a character type.

6. What is the difference between a float and a double?
    1. float is a 32 bit IEEE 754 single precision Floating Point Number1 bit for the sign, (8 bits for the exponent, and 23* for the value), i.e. float has 7 decimal digits of precision.
    2. double is a 64 bit IEEE 754 double precision Floating Point Number (1 bit for the sign, 11 bits for the exponent, and 52* bits for the value), i.e. double has 15 decimal digits of precision.

7. What are the four classes of data types supported by C?
    1. primary, user-defined, derived, and the empty data set
8. How are String variables handled differently than char arrays?
    1. A char array is just that - an array of characters:

       If allocated on the stack (like in your example), it will always occupy eg. 256 bytes no matter how long the text it contains is

       If allocated on the heap (using malloc() or new char[]) you're responsible for releasing the memory afterwards and you will always have the overhead of a heap allocation.

       If you copy a text of more than 256 chars into the array, it might crash, produce ugly assertion messages or cause unexplainable (mis-)behavior somewhere else in your program.

       To determine the text's length, the array has to be scanned, character by character, for a \0 character.

    2. A string is a class that contains a char array, but automatically manages it for you. Most string implementations have a built-in array of 16 characters (so short strings don't fragment the heap) and use the heap for longer strings

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++?
    a. Conditionals use two kinds of special operators: relational and logical. These are used to determine whether some condition is true or false.
        i. relational operators are used to test a relation between two expressions: > Greater than >= Greater than or equal to < Less than <= Less than or equal to == Equal to != Not equal to
        ii. logical operators are often used to combine relational expressions into more complicated Boolean expressions: && and || or ! not

2. What are the results of basic truth tables?

The operators return `true` or `false`, according to the rules of logic:

a	b	a && b
true	true	true
true	false	false
false	true	false
false	false	false

a	b	a || b
true	true	true
true	false	true
false	true	true
false	false	false

The `!` operator is a unary operator, taking only one argument and negating its value:

a	!a
true	false
false	true

3. How do you write a switch statement?
    a. switch(expression) { case constant1: statementA1 statementA2 ... break; case constant2: statementB1 statementB2 ... break; ... default: statementZ1 statementZ2 ... }

4. What are the three different types of loops used in C++ and how are they used?
    a. C++ has three kinds of loops: while, do-while, and for.
        i. The while loop has a form similar to the if conditional: while(condition) { statement1 statement2 … } As long as condition holds, the block of statements will be repeatedly executed.
        ii. The do-while loop is a variation that guarantees the block of statements will be executed at least once.
        iii. The for loop works like the while loop but with some change in syntax: for(initialization; condition; incrementation) { statement1 statement2 …

} The for loop is designed to allow a counter variable that is initialized at the beginning of the loop and incremented (or decremented) on each iteration of the loop.

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?
    a. Before you can use any of the math functions, you have to include the math header file. As a rule of the thumb, you should write using namespace std; whenever you use iostream. Similarly, the math header file contains information about the math functions. You can include it at the beginning of your program along with iostream: #include <cmath>.
2. How do you write a function definition and then use those functions?
    a. void NAME (LIST OF PARAMETERS) { STATEMENTS }
    b. In main call it using NAME (LIST OF PARAMETERS);

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.

    c. Pass by value - By default, non-pointer arguments in C++ are passed by value. When an argument is passed by value, the argument's value is copied into the value of the corresponding function parameter.
    d. Pass-by-reference means to pass the reference of an argument in the calling function to the corresponding formal parameter of the called function. The called function can modify the value of the argument by using its reference passed in. Must use the & operator before the parameters in the function definition.

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