Intro to Graphics Spring 2017 – Lab 1

Due Date: January 18th

Lab Objective

- Learn how to write basic C++ code.

This lab consists of completing some exercises with writing C++ code, in preparation for the rest of the work in this course.

About C++

The C++ programming language is an extension of the C programming language. These exercises will give you experience with some of the extensions the C++ language added to C.

The C++ language is often the topic of interview questions, so studying it seriously can have a big payoff for your career.

Submission Guidelines

Please answer the exercises below and upload your solution to Connex. Keep your answers short and simple, these are just warm-up exercises. Your code will not be compiled, so you don't need to submit any Makefiles or executables or similar. We just want to see the code.

Object-Orientation in C++

C++ enables object-oriented programming by adding features like member functions, virtual functions, and inheritance. These exercises will familiarize you with object-oriented programming in C++.

- Write a simple class that uses with a private variable and a public member function.
- Create an interface class with a pure virtual function. Create another class that inherits from it, and give a concrete implementation of the pure virtual function.
- Virtual functions in C++ are internally implemented using a virtual function pointer table. Summarize in your own words how a calling virtual functions work using a vtable.

Generic Types in C++

The C++ programming language allows you to create a "class template", a concept similar to generics in Java/C#. For example, you can create a class "std::list<T>", which can be used to create a list of any type, like a "std::list<int>" or a "std::list<float>" or a "std::list<std::string>". This exercise is about writing a simple class template.

- Write a simple class template "LinkedListNode<T>", which stores an object of type "T" and a pointer to the next node of type "LinkedListNode<T>*". It doesn't need to have any member functions.

Standard Containers in C++

C++ supplies built-in container classes, like std::vector (a self-resizing array), std::string (an easy way to manipulate strings), and std::map (an associative key-value pair container). Familiarity with these classes will save you the time of implementing your own data structures. These exercises will introduce you to some of the most useful standard containers.

- Concatenate two strings using the std::string class.
    - Note: #include <string>
- Create an empty std::vector<int> then push_back() the values 1,2,3 to it.
    - Note: #include <vector>
- Create a std::map<std::string, int> and insert into it the key-value pairs {“one”,1}, {“two”,2}, and {“three”,3}.
    - Note: #include <map>

Operator Overloading in C++

The C++ programming language allows you to define standard math operators for your own classes. This is usually used for vector and matrix math, so you’ll see it used a lot in such linear algebra libraries. This exercise will give you a chance to practice implementing operator overloading.

- Define operator+ and operator+= for the simple 3D vector class below:

```
struct Vector3 {

    float x, y, z;

};
```