Logan is out this week and next,
I'm your sub TA today
My name is Martin

CptS 122 – Data Structures

Lab 12: Inheritance and Containers in C++

Assigned: Week of April 8, 2024
Due: At the end of the lab session

I. Learner Objectives:

At the conclusion of this programming assignment, participants should be able to:

- Design, implement and test classes in C++ which apply inheritance
- Apply and implement private inheritance to container classes
- Compare and contrast inheritance ("is-a") relationships versus composition ("has-a") relationships
- Apply and implement *overloaded* functions and operators

II. Prerequisites:

Before starting this programming assignment, participants should be able to:

- Analyze a basic set of requirements for a problem
- Create test cases for a program
- Design, implement and test classes in C++
- Declare and define *constructors*
- Declare and define *destructors*
- Compare and contrast *public* and *private* access specifiers in C++
- Describe what is an *attribute* or data member of a class
- Describe what is a *method* of a class
- Apply and implement *overloaded* functions
- Distinguish between pass-by-*value* and pass-by-*reference*
- Discuss *classes* versus *objects*
- Describe and define *inheritance*

III. Overview & Requirements:

This lab, along with your TA, will help you navigate through designing, implementing, and testing inheritance with container classes in C++. It will also, once again, help you with understanding how to apply *inheritance* to an application.

Labs are held in a "closed" environment such that you may ask your TA questions. Please use your TAs knowledge to your advantage. You are required to move at the pace set forth by your TA. Please help other students in need when you are finished with a task. However, I encourage you to compose your own solution to each problem. Have a great time! Labs are a vital part to your education in CptS 122 so work diligently.

Tasks:

NOTE: Parts of this lab are courtesy of Jack Hagemeister.

One of the powers of inheritance is that it facilitates large amounts of code reuse. In this lab, you will redesign your queue class by inheriting from a base list class.

Task 1. List

Implement a templated class List and ListNode. Note: that you'll have the opportunity to work with the C++ keyword `friend` in this lab. The keyword `friend` allows for a function or class direct access to the private and protected members of a class. You may add methods/functions as you see fit. Test these classes. I have left all of the implementation as an exercise for you.

```
template< class NODETYPE > class List;  // forward declaration

template<class NODETYPE>
class ListNode
{
   friend class List< NODETYPE >; // make List a friend
public:
   ListNode( const NODETYPE &newData);  // copy constructor
   NODETYPE getData() const;      // return data in the node
private:
   NODETYPE data;                 // data
   ListNode< NODETYPE > *nextPtr; // next node in the list
};


template< class NODETYPE >
class List
{
public:
   List();       // constructor
   ~List();      // destructor
   void insertAtFront( const NODETYPE &newData );
   void insertAtBack( const NODETYPE &newData );
   bool removeFromFront( NODETYPE &removedData );
   bool removeFromBack( NODETYPE &removedData );
   bool isEmpty() const;
   void print() const;
private:
   ListNode< NODETYPE > *firstPtr;  // pointer to first node
   ListNode< NODETYPE > *lastPtr;   // pointer to last node

   // Utility function to allocate a new node
   ListNode< NODETYPE > *getNewNode( const NODETYPE &newData );
};
```

Task 2. Queue

Create a Queue class template that *privately* inherits from a List class. You should define enqueue ( ) and dequeue ( ) operations in terms of the inherited list operations.

Task 3. Network Traffic Application

Write an application that simulates network traffic. The traffic is represented by packets (of information) moving through the network. These packets must be represented by a class. Each packet must include an integer length field (in bytes) and a std::string field for data, where the length field is the number of characters in the std::string. You must represent one device in the network, which is represented by a Queue object. The application must randomly assign the arrival time of the first packet and the time that it takes to process the packet at the device. As a new packet arrives to the device, the arrival time for the next packet should be generated. Every time a packet leaves the device or a new packet arrives, print out the packet information for the one at the front and the back of the Queue.

IV. Submitting Labs:

- You are not required to submit your lab solutions, unless you are unable to attend them synchronously. You should keep them in a folder that you may continue to access throughout the semester.

V. Grading Guidelines:

- This lab is worth 10 points. Your lab grade is assigned based on completeness and effort. To receive full credit for the lab you must show up on time, work in a team, continue to work on the problems until the TA has dismissed you, and complete at least 2/3 of the problems.