Project 4 – Shape List

CS 251, Spring 2023

Collaboration Policy

By submitting this assignment, you are acknowledging you have read the collaboration policy in the syllabus for projects. This project should be done individually. You may not receive any assistance from anyone outside of the CS 251 teaching staff.

Late Policy

You are given the grace period between the deadline and the first lecture of the following day (9am). You do not need to let anyone know you are using this grace period. Beyond this period, no late submissions will be accepted.

What & Where to Submit

1. Gradescope - shape.h, canvaslist.h, and tests.cpp

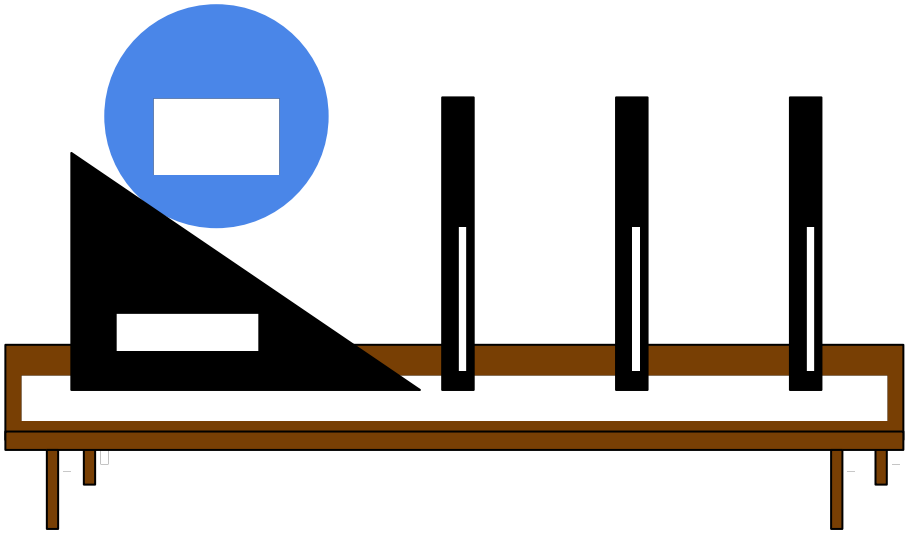
Do not submit any additional files.

Table of Contents

Copyright Notice

Project 4 – Shape List

CS 251, Spring 2023



Project Summary

Computers repeatedly refresh and redraw images to a screen. In this project we are going to implement a very simplified drawing canvas to mimic this storage of items to be drawn to the screen. We will implement a CanvasList class in C++ that maintains a linked list of shapes that will be "drawn" when the draw function is invoked. We will also implement a Shape class that has three derived classes Circle, RightTriangle, and Rectangle.

The project is designed to cover object oriented programming with test driven development. Throughout the implementation of the project you will craft and use multiple, develop test cases in either the Catch or Google Test framework, and work with and alter a provided main program that will help understand the usage of the objects you are creating.

Program and Coding Restrictions

The class definitions are provided and the public aspects of the class cannot be changed. You are encouraged to create private helper functions to help with implementation or function decomposition. Additionally, you may add private data members should you need additional private data members to increase algorithm efficiency.

In addition to passing the implementation harnesses, you must have a clean valgrind report. This means your program has zero memory leaks and zero memory errors.

Given Files (7)

We have provided two header files (`shape.h` and `canvaslist.h`), and the implementations for all the classes must be contained within these two files. You do not have to write the code within the class definition, you may write the class implementations in the space below the definitions. We provide a `main.cpp` file that has commented out calls for each function to demonstrate usage strategies. We provide a `tests.cpp` and `catch.hpp` file that can be used to develop tests within Catch framework style. Finally, we provide a `makefile` that can be used to build, execute, and test using the Catch framework or valgrind.

Project 4 – Shape List

CS 251, Spring 2023

Milestone 0 - Starter Files & Strategies

Completely read this entire project description and all the provided starter files.

For each milestone you should be building tests in your tests.cpp file allowing you to test your class implementations independently before moving on to other portions of the class or other classes or files. You should also be running valgrind on your program when you are acquiring or releasing memory (new/delete) to make sure your program maintains zero memory errors and zero memory leaks, rather than discovering these errors at the end of your development.

For all your implementations you should be utilizing your classes' functions rather than reimplementing the same code. For example, if you are implementing a member function and you need to put a node into the list then you should be using the appropriate member function to place that item within the linked list, not repeating code.

Milestone 1 - Basic `Shape` Class

Implement the Shape class, this will be the parent or base class for all the other shapes, but we will not worry about implementing those until later. As you are implementing, remember to create tests in your framework of choice.

Milestone 2 - The `CanvasList` Class

The CanvasList class is a drawing class that maintains a linked list of shapes to be drawn. For this program, drawing will simply mean outputting the details about the shape. See example execution later in this project description.

Milestone 3 - The `Rectangle` Class

Implement the derived Rectangle class as well as tests for the various implementations.

Milestone 4 - The `Circle` Class

Implement the derived Circle class as well as tests for the various implementations.

Milestone 5 - The `RightTriangle` Class

Implement the derived RightTriangle class as well as tests for the various implementations.

Milestone 6 - Additional Overall Testing

Implement any additional tests that may be missing as they will test multiple classes at a time whereas many of the tests you have developed to this point are independent and specific to a single class.

Project 4 – Shape List

CS 251, Spring 2023

Class Descriptions

shape.h: Shape, Rectangle, RightTriangle, Circle

Each of these classes has constructors, destructors (empty but needed), an object copier that returns a new heap allocation of the same pointer type, a print function (see example for output), and accessors and setters for private data members. The copy function uses dynamic memory allocation, none of the other functions do. None of the functions for these classes deallocate memory.

canvaslist.h: ShapeNode, CanvasList

The ShapeNode class is utilized as the node or element of the linked list. It contains two public data members, both pointers.

The CanvasList class is an implementation of a linked list that works with our Shapes classes to build and maintain a list of Shapes that can be printed in two ways. The draw function prints the details of the shapes within the linked list, the printAddresses function prints out the memory addresses of the ShapeNode as well as the address of internal Shape (the address in value).

Several functions within the CanvasList class allocate or deallocate memory where appropriate when either adding or removing nodes from the linked list.

Additional descriptions of functions that are non-obvious based on name:

Shape:

- `copy()` returns a dynamic memory allocation of a constructor call with the dereference implicit object passed as a parameter to return a memory address of the same type

CanvasList:

- `clear()` releases all allocations of memory (all Nodes and their internal Shapes)
- `front()` returns the private data member.
- `find()` discovers and returns the index of the first shape with the given x, y values.
- `shapeAt()` discovers and returns a pointer to the shape at the given linked list index.
- `pop_front()` and `pop_back()` return the pointer to the shape and release the node's memory but not the shape's memory.
- All functions that take a pointer to a Shape are taking a Shape that already exists. You can see the main.cpp for examples.
- `printAddresses()` has a tab character separating the two printouts

Requirements Reminders

1. You cannot change the provided public class definitions.

2. Your code file must have a header comment with your name and a program overview. Each function must have a header comment above the function, explaining the function’s purpose, parameters, and return value (if any). Inline comments should be supplied as appropriate; comments like “declares variable” or “increments counter” are useless. Comments that explain non-obvious assumptions or behavior are appropriate.

3. No global variables; use parameter passing and function returns.

4. The cyclomatic complexity (CCN) of any function should be minimized.

Project 4 – Shape List

CS 251, Spring 2023

Example Execution - Given main.cpp

Same as the output.txt file contents in the provided files

```
List size: 0
Front: 0


Adding Shape to the front
List size: 1
It's a Shape at x:1, y: 3


Adding Circle to the front
List size: 2
It's a Cirlce at x: 2, y: 4, radius: 3
It's a Shape at x:1, y: 3


Adding Rectangle to the back
List size: 3
It's a Cirlce at x: 2, y: 4, radius: 3
It's a Shape at x:1, y: 3
It's a Rectangle at x: 0, y: 0 with width: 0 and height: 10


Adding Right Triangle to the front
List size: 4
It's a Right Triangle at x: 1, y: 2 with base: 3 and height: 4
It's a Cirlce at x: 2, y: 4, radius: 3
It's a Shape at x:1, y: 3
It's a Rectangle at x: 0, y: 0 with width: 0 and height: 10


Deleting last element
List size: 3
It's a Right Triangle at x: 1, y: 2 with base: 3 and height: 4
It's a Cirlce at x: 2, y: 4, radius: 3
It's a Shape at x:1, y: 3


Deleting first element
List size: 2
It's a Cirlce at x: 2, y: 4, radius: 3
It's a Shape at x:1, y: 3


Finding the shape at 2, 4
It's a Cirlce at x: 2, y: 4, radius: 3
It's a Shape at x:1, y: 3
Found location: 0
It's a Cirlce at x: 2, y: 4, radius: 3
List size: 2


Inserting Rectangle after index 1
Original:
It's a Cirlce at x: 2, y: 4, radius: 3
It's a Shape at x:1, y: 3
Updated Original:
It's a Cirlce at x: 2, y: 4, radius: 3
It's a Rectangle at x: 3, y: 4 with width: 5 and height: 5
It's a Shape at x:1, y: 3


Removing index 0
Original:
It's a Cirlce at x: 2, y: 4, radius: 3
It's a Rectangle at x: 3, y: 4 with width: 5 and height: 5
It's a Shape at x:1, y: 3
Updated Original:
It's a Rectangle at x: 3, y: 4 with width: 5 and height: 5
It's a Shape at x:1, y: 3


Removing index 45
Original:
It's a Rectangle at x: 3, y: 4 with width: 5 and height: 5
It's a Shape at x:1, y: 3
Updated Original:
It's a Rectangle at x: 3, y: 4 with width: 5 and height: 5
It's a Shape at x:1, y: 3


Creating a copy of the canvas list
Original:
It's a Rectangle at x: 3, y: 4 with width: 5 and height: 5
It's a Shape at x:1, y: 3
Copy:
It's a Rectangle at x: 3, y: 4 with width: 5 and height: 5
It's a Shape at x:1, y: 3
Original:
Node Address: 0x56526cefafa0    Shape Address: 0x56526cefaf80
Node Address: 0x56526cefaee0    Shape Address: 0x56526cefaec0
Copy:
Node Address: 0x56526cefaf00    Shape Address: 0x56526cefaf20
Node Address: 0x56526cefaf60    Shape Address: 0x56526cefaf40


Clearing all elements and internal shapes
List size: 0
Original:
Copy:
It's a Rectangle at x: 3, y: 4 with width: 5 and height: 5
It's a Shape at x:1, y: 3


Using the assignment operator to overright copy.
List size: 1
Original:
It's a Shape at x:4, y: 2
Copy:
It's a Rectangle at x: 3, y: 4 with width: 5 and height: 5
It's a Shape at x:1, y: 3
Original:
Node Address: 0x56526cefaee0    Shape Address: 0x56526cefaec0
Copy:
Node Address: 0x56526cefaf00    Shape Address: 0x56526cefaf20
Node Address: 0x56526cefaf60    Shape Address: 0x56526cefaf40
Original:
It's a Shape at x:4, y: 2
Copy:
It's a Shape at x:4, y: 2
Original:
Node Address: 0x56526cefaee0    Shape Address: 0x56526cefaec0
Copy:
Node Address: 0x56526cefaf60    Shape Address: 0x56526cefaf40
```