Lab 9: Inheritance, Polymorphism & Templates

CPSC 1021

Lab Objective

- Use an abstract base class with a template
- Use inheritance to create class templates from existing class templates
- Use virtual functions and function overriding to achieve runtime polymorphism
- Use function overloading on templates
- Practice solving programming problems using OOP concepts and techniques

Collaboration Policy

Work with your assigned team members from Lab 0. ***Please make sure to integrate your team members who – because of COVID requirements – have to participate remotely.*** You can use the MS Teams video conferencing feature to collaborate with a remote team member. Use the search bar on the top of the MS Teams app to search for your team member with their CU user name, then click the video icon in the top-right corner of the screen to start a video call.

You and your partners can work on the assignments together.

Submission Policy

Code solutions to all challenges **can be submitted as teams** this week. This means only one member from each team needs to submit the code files on Gradescope (this person must then add all their other team members to the submission). However, **you are required to work on all problems together**.

Introduction

In this lab, we will again revisit the area calculation problem from last week, this time turning our solution from last week into templates. This lab also includes a second programming challenge for which you will write a function template to calculate the sum of the elements of a vector, with the template function overloaded for the STD string class.

Area Calculation using Inheritance and Polymorphism

Start with your solution from Lab 8. Change the Shape class and all derived classes to templates. Then write a main.cpp program that calculates the area of each shape, first using double values for all member variables, then using integer values. Use the following values to initialize your objects:

Double values:

- Circle: 2.0
- Square: 2.0
- Rectangle: 2.5, 3.0
- Trapezoid: 2.5, 3.5, 4.0

Integer values:

- Circle: 2
- Square: 2
- Rectangle: 2, 3
- Trapezoid: 2, 6, 5

Your program must produce the following output:

```
Area of Circle is: 12.6
Area of Square is: 4.0
Area of Rectangle is: 7.5
Area of Trapezoid is: 12.0
Area of Circle is: 12
Area of Square is: 4
Area of Rectangle is: 6
Area of Trapezoid is: 20
```

Submission Instructions

- Submit your complete and correct program to Gradescope. Your submission must include the following 8 files:

  1. main.cpp
  2. Shape.h
  3. Circle.h
  4. Square.h
  5. Rectangle.h
  6. Trapezoid.h
  7. Makefile

- Make sure your program compiles and runs on the SoC Linux machines; the autograder automatically assigns 0 points to programs that do not compile.
- Code style (proper indentation, consistency, readability, use of comments, etc.) will be considered when grading your submission.

Sequence Accumulation

Write a function `T accum(vector <T> v)` that forms and returns the sum of all items in the vector `v` passed to it. If `T` is a numeric type such as int or double, the numeric sum will be returned. Overload the function for the STL string type so that the function returns a single string with the elements concatenated and separated by commas (see the below sample run for an example of the expected output).

Hint: For any type `T`, the expression `T()` yields the value or object created by the default constructor. For example, `T()` yields the empty string object if `T` is the string class. If `T` represents a numeric type such as `int`, then `T()` yields 0. Use this fact to initialize your "accumulator."

Test your function with a driver program that asks the user to enter four integers, uses `accum` to compute the sum, and prints out the sum. The program than asks the user to enter four strings, uses accum to concatenate the strings and add commas, and prints the result. Format your output exactly as in the following sample run.

Sample run:

```
./main.out

Enter four numbers: 1 2 3 4

The sum of the numbers is 10

Enter four strings: john paul george ringo

The sum of the strings is john, paul, george, ringo
```

Submission Instructions

- Submit your complete and correct program to Gradescope. Your submission must include the following 3 files:
  1. main.cpp
  2. accum.h
  3. Makefile
- Make sure your program compiles and runs on the SoC Linux machines; the autograder automatically assigns 0 points to programs that do not compile.
- Code style (proper indentation, consistency, readability, use of comments, etc.) will be considered when grading your submission.