CSE30: Lab #13 – Templates

Overview

Templates are useful to make data structures available for different data types. Instead of implementing three separate linked lists for integers, floats, and characters, we can create a linked list as a template class. In this lab, we will explore the use of templates by converting the **LinkedList** and **Stack** created in Lab #8 and Lab #9 into template classes.

Getting started

Create a new directory in your main development directory (probably on Desktop/CSE30) called Lab_13. Try to use the terminal on your own, without getting help from the TA to setup the new directories (try to learn/remember the terminal commands).

The g++ syntax to compile classes is slightly different than for a single program comprised of a main (and potential functions):

g++ *class1.h class1.cpp class2.h class2.cpp mainSource.cpp* -o *executable*

where:

- g++ is the compiler (installed on your Linux system) of C++ source files,
- ***mainSource.cpp*** is the source file for your main program (main function),
- ***class1.h*** is the class declaration for your 1st class,
- ***class1.cpp*** is your 1st class definition,
- ***class2.h*** is the class declaration for your 2nd class,
- ***class2.cpp*** is your 2nd class definition (and so on...),
- -o tells the compiler that you want to give the executable its own name, and
- ***executable*** is the name you want to give your program.

As an example, if we have a main source file called ***Exercise1.cpp***, the class declaration called ***LinkedList.h***, the class definition called ***LinkedList.cpp***, and want to create an executable called ***aTestProgram***, you would type:

g++ *LinkedList.h LinkedList.cpp Exercise1.cpp* -o *aTestProgram*

Assuming that your program compiled successfully (i.e. no errors were found), you can run your program as you normally would by typing "***./aTestProgram***" in the terminal/console.

Good coding practices (worth 2 points!)

Writing code that is understandable by humans is as important as being correct for compilers. Writing good code will help you complete the code, debug it and … get good grades. It is very important to learn as soon as possible, because bad habits are hard to get rid of and good habits become effortless. Someone (guess who) reads your code will be in a better mood if it is easy to understand … leading to better grades! This lab will include 2 points (10% for code quality):

- Explanations with comments

- Meaningful names
- Indenting of blocks { } and nesting …
- Proper use of spaces, parentheses, etc. to
- Visible, clear logic
- One / simple statements per line
- Anything that keeps your style consistent

(Exercise 1) LinkedList

In this part of the lab, you will be implementing a template version of the LinkedList class you had created in Lab #8. The declaration of template class is provided in LinkedList.h (on CatCourses). Since templates are declared in the header files, you will need to implement the functions in the **same .h file**. The specifications of the functions are exactly the same as in Lab #8; therefore, you can copy your implementations for Lab #8 and make the changes accordingly.

Test your LinkedList with Exercise1.cpp (on CatCourses) and compare the output with the sample bellow:

Example1:

```
The first list is empty!
The second list is empty!
The size of the first list is: 0
The size of the second list is: 0
Here is the first list: [5.7,4.6,3.2,2.4,1.5]
Here is the second list: []
Here is the first list: [5.7,4.6,3.2,2.4,1.5]
Here is the second list: [A]
Here is the first list: [4.6,3.2,2.4,1.5]
Here is the second list: []
Here is the first list: []
Here is the second list: [g,e,f,C]
The size of the first list is: 0
The size of the second list is: 4
The first list is empty!
The second list is NOT empty...
Here is the second list: [g,e,f,C]
Successfully removed an item from the list...
Here is the second list: [e,f,C]
Successfully removed an item from the list...
Here is the second list: [f,C]
Successfully removed an item from the list...
Here is the second list: [C]
Successfully removed an item from the list...
Here is the second list: []
COULD NOT remove an item from the list!
Here is the second list: []
COULD NOT remove an item from the list!
```

```
The size of the first list is: 0
The size of the second list is: 0
The first list is empty!
The second list is empty!
```

(Exercise 2) Stack

In this part of the lab, you will convert the Stack.h created in Lab #9 and use it with the LinkedList created from Exercise 1. Follow the example from the LinkedList class and implement the Stack as a template class. You can copy your implementations of Stack from Lab #9 and make the changes accordingly. Again, you must implement all the functions within the **same Stack.h**.

Test your Stack with Exercise2.cpp (on CatCourses) and compare the output with the sample bellow:

Example2:

```
Testing Stack #1:
isEmpty(): true
push(A) Stack contents:A
push(Y) Stack contents:Y,A
Size(): 2 Stack contents: Y,A
Pop(): Y Stack contents: A
isEmpty(): false
push(D) Stack contents:D,A
Top(): D Stack contents: D,A
push(T) Stack contents:T,D,A
Pop(): T Stack contents: D,A

Testing Stack #2:
isEmpty(): true
push(1) Stack contents:1
push(5) Stack contents:5,1
Size(): 2 Stack contents: 5,1
Pop(): 5 Stack contents: 1
isEmpty(): false
push(10) Stack contents:10,1
Top(): 10 Stack contents: 10,1
push(-5) Stack contents:-5,10,1
Pop(): -5 Stack contents: 10,1
```

What to hand in

When you are done with this lab assignment, you are ready to submit your work. Make sure you have included the following ***before*** you press Submit:

- Your **LinkedList.h**, **Stack.h**, and a list of Collaborators.
- Documentation (in a text file) of code you used from the internet. You may want to cut-and-paste the original code as well.