

CISC181 Fall 2006 Project 2

Making your Linked List more generic

Submit electronically on Tuesday Dec 5th at midnight. Deliver paper copy in class on Wednesday. No projects will be accepted after Wednesday, so be sure to start early.

Description

In this project you will make your linked list class more generic. You will create an abstract container class (e.g. Item) to be put in a Node. The container class will be a super class (“base” class) for any item you wish to put in a list (I’m assuming you will use the lists you made in project two, unless you contact me this week to discuss another arrangement.)

Another method of making the list generic would be to make your Node class a template class, so that its data pointer can point to any type known at compile time.

0.1 Defined interface

We have seen that pure virtual functions define an interface for subclasses. Use this technique to allow your linked list to hold any of your item classes from project two, without having to recompile Node or LList. A list will hold only one subclass at a time, but because the interface declares a single comparison operator, the list can still be sorted. The comparison operator must be appropriately defined in each subclass. Implement this method of making your list more generic.

0.2 Template class

Declaring Node as a template class will allow its data pointer to point to any type. Of course, any type that it points to must implement the same comparison operator, so that the list can still be sorted. Do not implement this, but sketch the idea on paper so you can think about it.

0.3 Discussion

Write the advantages and disadvantages of these two approaches. You may use paragraph style or bulleted lists. Consider factors such as ease of use (the degree to which a programmer’s task is made clear and simple when interacting with your classes), compile time, run time, and safety (if the programmer uses your class incorrectly, at what point is this discovered? Think in terms of a makefile.)

0.4 User interface

Use a hierarchical menu to allow the user to control the behavior of the program (add, delete). User must be able to nicely terminate the program from any menu, and the program will return to main() and print the exit message.

Grading

- 30% testing code and makefile(s): completeness and correctness
- 50% correct operation, coding, style
- 20% discussion

Submission

Projects will not be accepted after Wednesday.

If your testing code is complete, you will not need to further demonstrate your project's abilities. If your testing code does not test all aspects of your code's function, then script examples to do so.

Answers to the discussion questions must be typed, and submitted both on paper and with your electronic submission.