Fun with Lists and Templates

9-11-2003

Opening Discussion

- What did we talk about last class? Who wants to come up to the board and write a vector based stack?

Memory Management in a Linked List

- What needs to go in the constructors and destructors of a linked list class? What about the nodes?
- Making a node recursively delete can be dangerous. The list itself should oversee that.
- Also, this is a case where deep copying is required. If you don't want to write it you have to explicitly disallow it.

Sentinels with Linked Lists

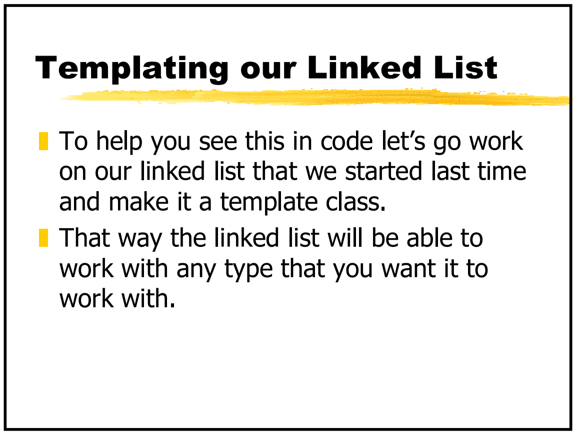
- We can simplify our code for the doubly linked list some if we make it a circular linked list with a sentinel.
- The sentinel is a special node that represents the end of the list. For a doubly linked list we make it circular and it is both head and tail. This takes out some of the special case logic for adding, removing, etc.

Templates in C++

- Data structures should generally work with any type. In Java we did this with inclusion polymorphism by making them work with Object and all classes were subtypes of Object. In C++ that won't work.
- Instead we can use templates which give us parametric polymorphism.
- Templates are typically used when the same code can work for any type.

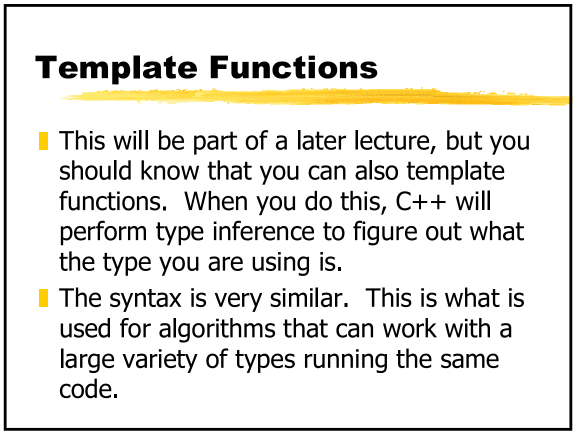
Syntax of Template Classes

- To have a class be a template class we precede it with template<class TypeName> where TypeName can be anything you want and it just holds the place of what the real type will be.
- You can use that type in any part of the class as you write it.
- When you declare an object of that type you tell it what the true template type is.



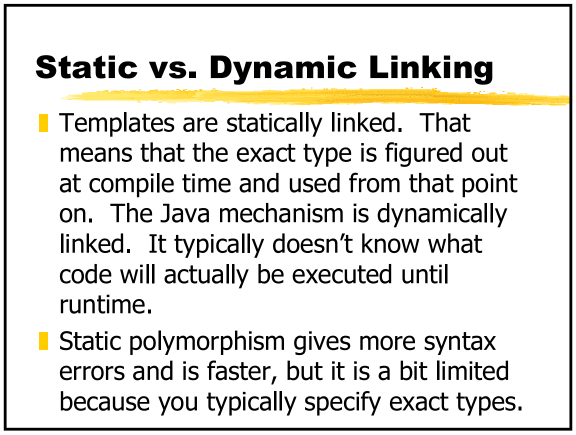
Templating our Linked List

- To help you see this in code let’s go work on our linked list that we started last time and make it a template class.
- That way the linked list will be able to work with any type that you want it to work with.



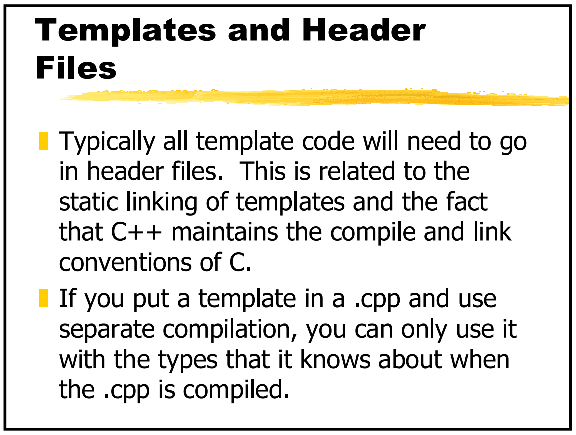
Template Functions

- This will be part of a later lecture, but you should know that you can also template functions. When you do this, C++ will perform type inference to figure out what the type you are using is.
- The syntax is very similar. This is what is used for algorithms that can work with a large variety of types running the same code.



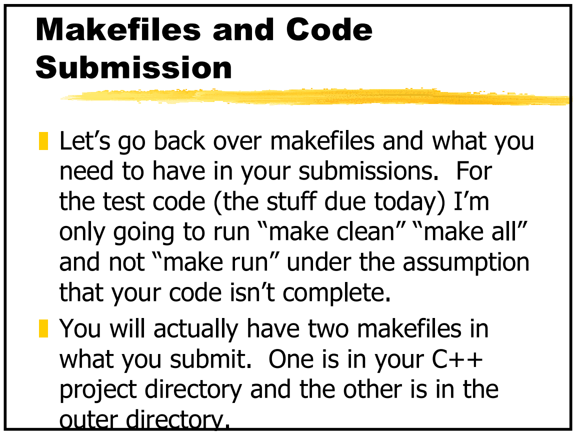
Static vs. Dynamic Linking

- Templates are statically linked. That means that the exact type is figured out at compile time and used from that point on. The Java mechanism is dynamically linked. It typically doesn’t know what code will actually be executed until runtime.
- Static polymorphism gives more syntax errors and is faster, but it is a bit limited because you typically specify exact types.



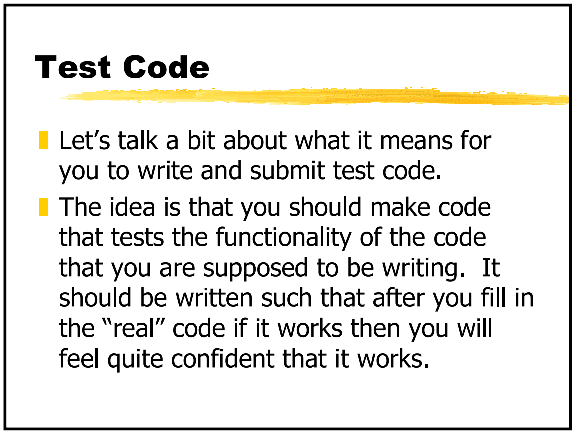
Templates and Header Files

- Typically all template code will need to go in header files. This is related to the static linking of templates and the fact that C++ maintains the compile and link conventions of C.
- If you put a template in a .cpp and use separate compilation, you can only use it with the types that it knows about when the .cpp is compiled.



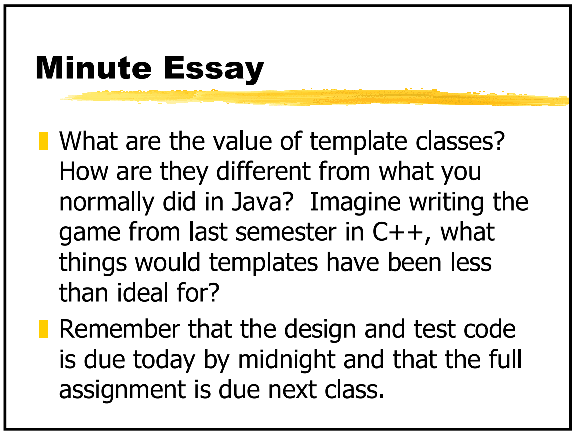
Makefiles and Code Submission

- Let’s go back over makefiles and what you need to have in your submissions. For the test code (the stuff due today) I’m only going to run “make clean” “make all” and not “make run” under the assumption that your code isn’t complete.
- You will actually have two makefiles in what you submit. One is in your C++ project directory and the other is in the outer directory.



Test Code

- Let’s talk a bit about what it means for you to write and submit test code.
- The idea is that you should make code that tests the functionality of the code that you are supposed to be writing. It should be written such that after you fill in the “real” code if it works then you will feel quite confident that it works.



Minute Essay

- What are the value of template classes? How are they different from what you normally did in Java? Imagine writing the game from last semester in C++, what things would templates have been less than ideal for?
- Remember that the design and test code is due today by midnight and that the full assignment is due next class.