

Re: template compile errors

Source: http://coding.derkeiler.com/Archive/C_CPP/comp.lang.cpp/2004-10/1509.html

From: E. Robert Tisdale (*E.Robert.Tisdale_at_jpl.nasa.gov*)

Date: 10/13/04

Date: Tue, 12 Oct 2004 20:54:54 -0700

Thomi Richards wrote:

```
> I'm trying to create a simple stack class using C++ and templates.
> Everything works well and good if I amke the class totally inline.
> However, as soon as I try to seperate the class into a .h and a .cpp
> file, gcc throws some strange linking errors.
>
> Compiling like this:
>
> gcc test.cpp stackList.cpp -lstdc++ -o test
>
> produces these errors:
>
> /tmp/ccCJLrbl.o(.text+0x84): In function `main':
> : undefined reference to `Stack<int>::Stack[in-charge]()'
> /tmp/ccCJLrbl.o(.text+0x101): In function `main':
> : undefined reference to `Stack<int>::push(int)'
> /tmp/ccCJLrbl.o(.text+0x10c): In function `main':
> : undefined reference to `Stack<int>::size() const'
> /tmp/ccCJLrbl.o(.text+0x18b): In function `main':
> : undefined reference to `Stack<int>::empty() const'
> /tmp/ccCJLrbl.o(.text+0x19c): In function `main':
> : undefined reference to `Stack<int>::size() const'
> /tmp/ccCJLrbl.o(.text+0x1aa): In function `main':
> : undefined reference to `Stack<int>::pop()'
> /tmp/ccCJLrbl.o(.text+0x20f): In function `main':
> : undefined reference to `Stack<int>::~~Stack [in-charge]()'
> /tmp/ccCJLrbl.o(.text+0x229): In function `main':
> : undefined reference to `Stack<int>::~~Stack [in-charge]()'
> collect2: ld returned 1 exit status
>
>
> I'm tearing my hair out... why would it not work in seperate files?
> My class definition is something like this:
>
> template <class T>
> class Stack
> {
```

```
> private:
> StackNode<T> *stackHead;
> /// The number of elements on the stack. It's easier to keep count
> /// of them here than to count them every time we need them.
> unsigned int stackSize;
> protected:
> public:
> /// Default constructor: set the stackSize to be 0.
> Stack(void);
>
> /// Default destructor: clean up any memory allocated.
> ~Stack(void);
> /// push: push an item onto the stack, and increment the size.
> bool push(T item);
>
> /// pop: return the topmost element from the stack, removing it
> /// from the stack at the same time:
> T pop(void);
>
> /// size: returns how many items are on the stack:
> unsigned int size(void) const;
> /// empty: return true if the stack is empty:
> bool empty(void) const;
> /// full: return true if the stack is full: – for a linked list
> /// implementation, this can be safely ignored:
> bool full(void) const;
>
> };
>
>
> and the code for a single method for the above class looks like this:
>
> template <class T> bool Stack<T>::push(T item)
> {
> if (full())
> return false;
>
> StackNode<T> *node = new StackNode<T> (item);
> node->Head = stackHead;
> stackHead = node;
> stackSize++;
>
> return true;
> }
```

Since you didn't tell us, I'll suppose that you moved the function template definitions from your stackList.h header file to your stackList.cpp source file and that you include'd the stackList.h header file but **not** the stackList.cpp source file

in your test.cpp source file.

Please allow me to ask a simple question,

"How does the C++ compiler know
where to find the function template definitions
so that it can instantiate them
when it is compiling test.cpp?"

> *I've been following the C++ language tutorial,
> and a few books I own at home.*

> *info gcc 'C++ extensions' 'Template Instantiation'*

> *It looks like I'm doing everything properly...
> Can anyone shed some light on this?*

If you know all of the template functions that you need.
you can instantiate the template functions **explicitly**:

```
template Stack<int>::Stack(void);  
template std::bool Stack<int>::push(int);  
template size_t Stack<int>::size(void) const;  
template std::bool Stack<int>::empty(void) const;  
template int Stack<int>::pop(void);  
template void Stack<int>::~Stack(void);
```

right after the function template definitions
in your stackList.cpp source file.