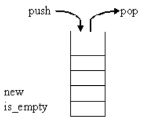
Class Templates

Templates

- Last time we talked about function templates
  - What are they?
- Templates provide a wonderful opportunity for software reusability
- Describe the notion of a class generically, and only specify the data type of data members when the class is instantiated
- To illustrate this, we will use the *stack* data structure

Stack

- A stack is a structure into which we insert items at the top
- Items are retrieved in the reverse order they were placed in
  - LIFO - Last In First Out



Class Template Header

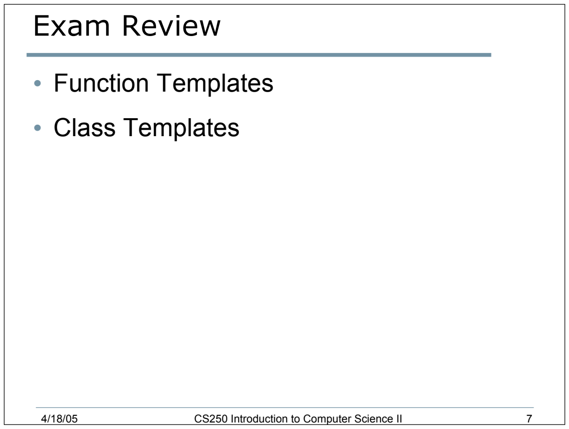
```
#ifndef TSTACK1_H
#define TSTACK1_H
template< class T >
class Stack {
public:
   Stack( int = 10 );
   ~Stack();
   bool push( const T& );
   bool pop( T& );
   bool isEmpty() const;
   bool isFull() const;
private:
   int size;
   int top;
   T *stackPtr;
};
#endif
```

Instantiating the Class

- Since the class was created using templates, it’s possible to instantiate it using any data type
  - Create a stack of doubles, ints, characters
- Syntax:
  - `Stack< double > doubleStack( 5 );`
  - `Stack< int > intStack( 5 );`
  - `Stack< char > charStack( 5 );`

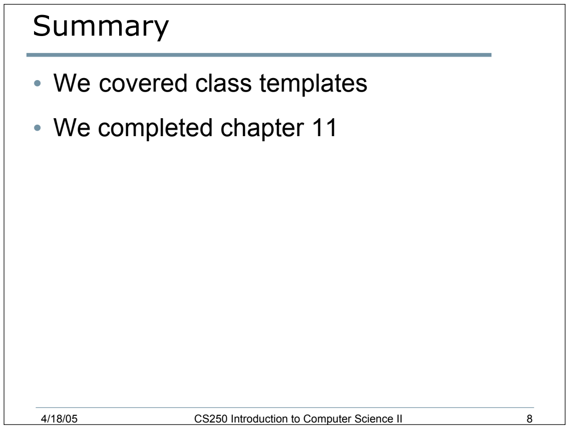
Exam Review

- Midterm will be on Friday, April 22
- Midterm will be on Chapters 8 - 11
- Topics are:
  - Operator overloading
  - Inheritance
  - Polymorphism
  - Virtual Functions
  - Abstract Classes



Exam Review

- Function Templates
- Class Templates



Summary

- We covered class templates
- We completed chapter 11