Lab 5

A Dynamic Stack Class

Introduction

The goal of this exercise is to enhance your integer stack class to use dynamic memory management.

Suggested Time: 45 minutes.

Instructions

1. Build and run the starter code in the IntStack folder.

2. Modify the private data of **IntStack.h** to contain a pointer (type **long ***) for holding the stack data and a long integer **stacksize** for holding the stack size. You will keep the long integer **top** for the top of stack. The constant **STACKSIZE** now becomes a default stack size, with the stack size specified as a parameter in the constructor.

3. Modify the constructor in the file **IntStack.cpp** to dynamically allocate the array **stack** for storing the stack data and do the appropriate initializations. Do appropriate cleanup in the destructor, and in the function **IsFull** test against the variable **stacksize** instead of against a constant. Build and run. Notice that we have not changed the test program, verifying that our new implementation preserves the public interface of the class.

4. Next, modify the test program to allow dynamically creating and destroying a stack object. Begin by replacing the declaration of an **IntStack** object by a pointer to an **IntStack**, and make all the calls to methods of **IntStack** go through this pointer. Build and run.

5. Next, add new commands "create" and "destroy" that will dynamically create and destroy an **IntStack**. In creating a new stack, you should query for the size. In destroying a stack, set the **IntStack** pointer to 0. Build and run. Now if you try to invoke one of the commands other than "create" after "destroy" the program will crash.

6. As a final modification to your test program, place a test that the **IntStack** pointer is not 0 before carrying out any of the commands other than "destroy" or "create". Build and test thoroughly.