**C/C++ STACK OF int DATA LAB REPORT**

**1) Enter your name, student ID, platform (Mac or PC) and date**

Name and Student ID: Samuel Indurkar, 0888068

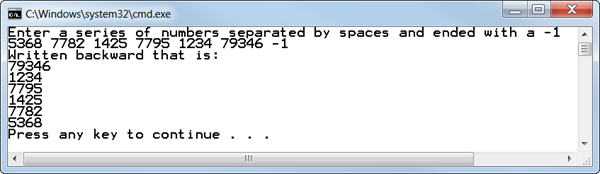
Class: CIS054 C/C++ Programming

Platform (Mac or PC): MAC and Eclipse  
Date: 7/25/2017

**DESCRIPTION:**

The textbook provides code to implement and test a stack that holds **char** data. The test program in the text asks the user to input a series of characters that are then **pushed** into the stack and then **popped** back out of the stack in reverse order. The code for the stack is provided on Canvas in two files:  
 **stack.h** - provides the header file for the definition of the stack holding **char** data  
 **stack.cpp** – provides the implementation of the stack for **char** data

A third file is provided on Canvas that is used to test a stack that holds **int** data. It is your job to modify the code for the stack of **char** data so that it will hold **int** data and test the modified stack code using the **TestStackLab.cpp** file.

When the program is run, it asks the user to input several integers, each separated by a space and ended with a -1. The program is then to display each integer in reverse order on a separate line. The only thing that needs to be done is to modify the **stack.h** and **stack.cpp** files to process **int** data instead of **char** data. The program output should be similar to:  
  


**LAB REPORT:  
2) Determine the Inputs, Processing and Outputs before creating the program**

|  |  |  |
| --- | --- | --- |
| **INPUTS** | **PROCESSING** | **OUTPUTS** |
| Ask the user for numeric input separated by spaces till -1 | Take the input from the user and using the push() member function of stack, store it into a linked-list and modify the top to always point to the last number inputted by the user. Then keep pop’ing till the stack is empty and display on screen, so it will display the numbers in the reverse order. | previously entered numbers in reverse order. |

**DISCUSSION:**

**3) Complete the DISCUSSION section. It does not need to be long, but it needs to be complete.**3a) What did you do to develop the program? ("Followed the Directions" is not a complete description)

Ask the user for numeric input separated by spaces till -1. Take the input from the user and using the push() member function of stack, store it into a linked-list and modify the top to always point to the last number inputted by the user. Then keep pop’ing till the stack is empty and display on screen, so it will display the numbers in the reverse order.

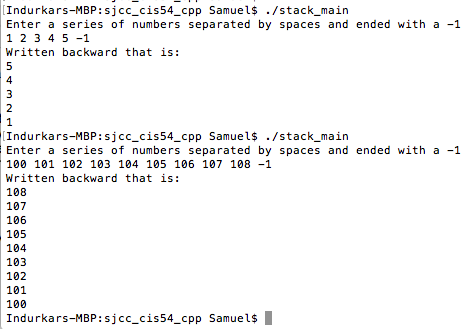
3b) What problems did you have and how did you overcome the problems?

No problems (because I did a search and replace to search char and replace it with int)

**PROGRAM OUTPUT:**

**4) Show one or more screen shots showing that the program successfully processed int data**

Refer to previous lab assignments for instructions on how to capture a screen or portions of a screen for either the PC or a Mac

****

**PROGRAM LISTING:**

**5) Copy and paste the code that YOU typed to make the program work. Your program should include a comment block at the top that shows the name of the program, date, version and your name.**

/\*

\* stack.h

\*

\* Created on: Jul 25, 2017

\* Author: Samuel

\*/

//DISPLAY 13.17 Interface File for a Stack Class

//This is the header file stack.h. This is the interface for the class Stack,

//which is a class for a stack of symbols.

#ifndef STACK\_H // prevent the Stack.h file from being compiled more than one time

#define STACK\_H

struct StackFrame

{

int data;

StackFrame \*link;

};

typedef StackFrame\* StackFramePtr;

class Stack

{

public:

Stack( );

//Initializes the object to an empty stack.

Stack(const Stack& a\_stack);

//Copy constructor.

~Stack( );

//Destroys the stack and returns all the memory to the freestore.

void push(int the\_symbol);

//Postcondition: the\_symbol has been added to the stack.

int pop( );

//Precondition: The stack is not empty.

//Returns the top symbol on the stack and removes that

//top symbol from the stack.

bool empty( ) const;

//Returns true if the stack is empty. Returns false otherwise.

private:

StackFramePtr top;

};

#endif //STACK\_H

/\*

\* stack.cpp

\*

\* Created on: Jul 25, 2017

\* Author: Samuel

\*/

//DISPLAY 13.19 Implementation of the Stack Class

//This is the implementation file stack.cpp.

//This is the implementation of the class Stack.

//The interface for the class Stack is in the header file stack.h.

#include <iostream>

#include <cstddef>

#include "stack.h"

using namespace std;

//Uses cstddef:

Stack::Stack( ) : top(NULL)

{

//Body intentionally empty.

}

Stack::Stack(const Stack& a\_stack)

{

if (a\_stack.top == NULL)

top = NULL;

else

{

StackFramePtr temp = a\_stack.top; // moves from top to bottom of a\_stack

StackFramePtr end; // point to end of new stack

end = new StackFrame;

end->data = temp->data;

top = end;

temp = temp->link;

while (temp != NULL)

{

end->link = new StackFrame;

end = end->link;

end->data = temp->data;

temp = temp->link;

}

end->link = NULL;

}

}

Stack::~Stack( )

{

int next;

while (! empty( ))

next = pop( );//pop calls delete.

}

//Uses cstddef:

bool Stack::empty( ) const

{

return (top == NULL);

}

void Stack::push(int the\_symbol)

{

StackFramePtr temp\_ptr;

temp\_ptr = new StackFrame;

temp\_ptr->data = the\_symbol;

temp\_ptr->link = top;

top = temp\_ptr;

}

int Stack::pop( )

{

if (empty( ))

{

cout << "Error: popping an empty stack.\n";

exit(1);

}

int result = top->data;

StackFramePtr temp\_ptr;

temp\_ptr = top;

top = top->link;

delete temp\_ptr;

return result;

}

/\*

\* stack\_main.cpp

\*

\* Created on: Jul 25, 2017

\* Author: Samuel

\*/

//Program to demonstrate use of the Stack class.

#include <iostream>

#include "stack.h"

using namespace std;

int main( )

{

Stack s;

int number;

cout << "Enter a series of numbers separated by spaces and ended with a -1" << endl;

cin >> number;

while (number != -1)

{

s.push(number);

cin >> number;

}

cout << "Written backward that is: " << endl;

while ( ! s.empty( ) )

cout << s.pop( ) << endl;

return 0;

}