Stack Data Structure

An example of a C++ class and program

```
//--------------------------------------------------------------------
// stack.h
//--------------------------------------------------------------------
// Stack class
// Written by Charles A. Cusack, January 2000
//
// This is a class for the stack data structure.
// It is implemented with a simple array.
//
//--------------------------------------------------------------------
class Stack {

private:                                  // The data used for the stack

int *stack;       // The stack elements will be stored here
int size;         // The capacity of the stack
int top;          // The top of the stack

public:                                        // The stack operations

Stack(int s=100);                // The constructor initialize the stack
~Stack();                    // The destructor frees the allocated memory
int Push(int v);       // Place item on the stack. Return 1 if successful
int Pop();                // Remove top of the stack and return its value
                          // Return -1 if unsuccessful
int Empty();                              // Return 1 if empty, 0 otherwise
int Full();                                // Return 1 if full, 0 otherwise
};
//--------------------------------------------------------------------
```

```
//--------------------------------------------------------------------
// stack.cc
//--------------------------------------------------------------------
// Stack class
// Written by Charles A. Cusack, January 2000
//
// This is a class for the stack data structure.
//
//--------------------------------------------------------------------
#include "stack.h"
//--------------------------------------------------------------------
Stack::Stack(int s) {                             // Initialize the stack
      size = s;
      stack = new int[size]; // Allocate the necessary memory
      top = -1;              // Stack is empty
      }
Stack::~Stack() {            // The destructor frees the allocated memory
    delete []stack;
    }

int Stack::Push(int v) {                      // Place item on the stack
                             //Return 1 if successful, and 0 if not
     if( !Full() )  {                      // Make sure stack is not full
        top++;                                  // Move the top pointer
        stack[top] = v;                     // insert element into stack
        return 1;                      // regular termination, so return 1
       }
     else return 0;                           // Stack was full, so return 0
     }

int Stack::Pop() {          // Remove top from stack and return its value
    if( !Empty() ) {                     // Make sure stack is not empty
       return stack[top--];   // Return the value at the top of the stack
                              // and move the top counter
       }
    else return -1;                           // Empty stack, so return -1
    }

int Stack::Empty() {    // If the top=-1, stack is empty.
   return (top==-1);
   }

int Stack::Full() {  // If top=size, stack is full.
   return (top+1==size);
   }
//--------------------------------------------------------------------
```

```
//--------------------------------------------------------------------
// mystack.cc
//--------------------------------------------------------------------
// Written by Charles A. Cusack, August 1999
//
// This is a simple program that reads informtation from an input file,
// and performs one of several tasks.
// Each entry in the file is either 'u #', 'o', or 'p'.
// If the entry is 'u #', the number '#' is inserted onto the stack
// If the entry is 'o', the top of the stack is deleted
// If the entry is 'p', the remaining elements in the stack are printed
// and the program terminates.
// This program uses the class 'Stack' defined in the files
// stack.h and stack.cc
//--------------------------------------------------------------------
#include <iostream>
#include <fstream>
#include "stack.h"
using namespace std;
//--------------------------------------------------------------------
main(int argc, char *argv[]) {
if(argc!=2) {                    // Check for correct number of arguments.
  cout<<"USAGE: "<<argv[0]<<" filename\n";
  exit(1);
  }

fstream infile(argv[1],ios::in);                              // Open file

Stack L(100);                    // The stack which we will store stuff on
char operation=' ';             // The operation to be performed
int  value;                     // The item inserted/removed from the stack

while(operation!='p') {       // Continue until the termination condition

     infile>>operation;    // Read until non-whitespace character is read
     while(operation==' ' || operation=='\n')
         infile>>operation;
     if(operation=='u') {               // Read and push the next integer
          infile>> value;                  // Read in the integer to push

                            // Push the value.  If there is an error quit
          if (!L.Push(value)) {
             cout<<"The stack is full, and I was asked to push\n";
             cout<<"an element. I can't do that, so I am quitting.\n\n";
             exit(3);
             }
          }
    else if(operation=='o') {               // Pop the stack if possible
          if( !L.Empty() ) {         // Check that the stack is not empty
              value=L.Pop();                           // Pop the stack
              }
          else {                            // If the stack is empty, quit
             cout<<"The stack is empty, and I was asked to pop\n";
             cout<<"an element. I can't do that, so I am quitting.\n\n";
             exit(4);
             }
          }
    else if(operation=='p') {                          // Print and quit
          while( !L.Empty() ) {         // Pop and print each element
               cout<<L.Pop()<<" ";
               }
          cout<<"\n"; }
    else {            // Something is wrong with the input file, so quit
          cout<<"I was asked to do some unknown task.\n";
          cout<<"Since I don't know what to do, I am quitting.\n\n";
          exit(4);
         }
    }

infile.close();                                 // Close the input file
return 0;
}
//--------------------------------------------------------------------
```