Lab 8.1.1 Operators: non-natural way of using

Objectives

Familiarize the student with:

- writing operators to modify the content of classes;
- writing iostream operators;
- non-natural ways to use operators (should help the student think about the legitimacy of using overloaded operators).

Scenario

Write a simple container class that holds the elements of a stack (the type of element is up to you, it may be an integer or double, or whatever you want). A stack is a simple data structure with two operations: push, which adds an element to a collection, and pop, which removes the most recently added element that hasn't yet been removed. The order in which the elements come off a stack gives rise to its alternative name, LIFO (last in, first out). These two methods (push and pop) modify the stack content. Apart from the two methods, add a method to get access to the top without modifying the stack (name it top). If the stack is empty, add some exception throwing code (as in Chapter 7). Your main task in this lab is to write operator functions (i.e. << and >>) to push a value to and get (top and pop methods)a value from the stack. In the main code, add your code to handle these specific exceptions. Also, in the main code, add some code to input data to the stack. Ask the user how many items should be pushed onto the stack, ask the user about these items, and ask again how many items your program should print and pop from the stack. Then print the stack – if an exception occurs, print an adequate message. Use the operators defined in your program.

Example input

```
5
1
2
3
4
5
6
```

Example output

```
5
4
3
2
1
Exception: stack is empty.
```