Week 10 Exercises : Reuse via containment

April 10th 2012

1 Approach

This week there was no lecture as Monday was a public holiday. As a result the exercises this week focus on extending your results from last week in new ways. The first task is to be completed by everyone on the course. The second task is optional for students who want to progress further.

Task 1 : Implement a Stack class

A stack is a container with a very simple interface. New items can be pushed onto the top of the stack. The item on the top can be pulled off from the stack. The easiest way to visualise this is as a stack (or pile) of boxes. We cannot put new boxes in the middle or pull existing boxes out. We can only add a box to the top to grow the pile, or remove the box from the top to shrink the pile. Adding (pushing) a box replaces writing to the container, and removing (pulling) a box replaces reading from the container.

You will implement a Stack class that holds an arbitrary number of integers. The interface to the class will tell the programmer the size of the stack, push a new integer onto the stack and pull an integer from the stack. Internally the storage will be a single object of your Vector class from last week.

Task 2* : Undo operations on text

In this task, you will use the stack datastructure to keep a record of some basic editing operations applied to a single line of text, so they can be reversed in a similar fashion to a real text editor. You will write a program which reads one single line of text from the user, and displays a prompt to the user to modify such text. The program will accept four operations, and modify the text accordingly. After each modification, the program will display the new version of the text. Following the description of the operations:

i position newtext Insert in position *ith* the text *newtext*.

r start end newtext Replace text from *start* to *end*, with *newtext*.

d start end Delete text from *start* to *end*.

u number Undo the last number of operations applied to the text.

Example of execution:

```
DOS_CMD> program.exe
This is the original line of text.
Cmd> i 8 not
This is not the original line of text.
Cmd> r 30 36 entered
This is not the original line entered.
Cmd> d 16 24
This is not the line entered.
Cmd> u 2
This is not the original line of text.
```