Datastructuren 2016

Programmeeropdracht 1: Stapel

Bouw vier versies van een stapel (ofwel *stack*) en een simpel testprogramma.

Deadline. Woensdag 21 september 23:59.

De ADT Stapel

De abstracte datastructuur Stapel heeft hier de volgende operaties:

- `void create()` — maak een lege stapel
- `bool isEmpty()` — bepaal of de stapel leeg is
- `void clear()` — maak de stapel leeg
- `bool push(newItem)` — voeg een `newItem` toe op de top van de stapel, en geef terug of de opdracht gelukt is
- `bool pop()` — verwijder het bovenste element van de stapel, en geef terug of de opdracht gelukt is
- `bool top(topItem)` — geef het bovenste stapelelement, zonder dit te verwijderen, en geef terug of de opdracht gelukt is

Implementeer de ADT Stapel in C++ op vier verschillende manieren:

1. met een 'klassiek' array,
2. met pointers, als gelinkte lijst,
3. met de STL `vector`,
4. rechtstreeks met de STL `stack`.

De eerste twee implementaties zijn al behandeld bij het college Programmeermethoden, hoofdstuk Datastructuren. Zie daar voor voorbeeld code.

Omdat in C++ de ADT functie `create` overeenkomt met de constructor hoeft `create` niet apart te worden geïmplementeerd.

Maak zinvol gebruik van een *class template* `<T>` zodat de stapel voor algemene type data beschikbaar is.

In theorie zou het mooi zijn als de specificaties (`.h`) en de implementaties (`.cpp`) gescheiden zouden kunnen blijven, maar dat laten we hier achterwege omdat de standaard compiler niet altijd goed kan omgaan met templates. *Dus*: maak vier `.h` files, één voor elke implementatie van de stapel, de header file bevat de code, niet alleen de headers van de functies.

Test: Undo

Test je code door een simpele 'backspace' simulatie, uit te voeren op een ingevoerde regel tekst. In de regel met letters staat de asterix voor het commando backspace, het verwijderen van de laatste letter. Achtereenvolgende asterixen verwijderen de eerdere letters op bekende LIFO manier.

Bijvoorbeeld, als de string `abcc*ddde***ef*fg` ingevoerd wordt, moet deze veranderd worden tot `abcdefg`.

Omdat dezelfde test op vier verschillende implementaties moet worden uitgevoerd is het handig [en verplicht] om met een *templated* test functie te werken, zie voorbeeld-code op de laatste bladzijde.

Instructies. Werk in tweetallen. Raadpleeg de *Werkcollege Intro* van Bart. Documenteer je code en vermeld de auteurs in elke file. Voor deze kleine opdracht is een verslag niet nodig. Stuur je bijdrage (vier implementaties plus main met test) in een zip-file naar `data.structuren.2016@gmail.com` met in het onderwerp `[STAPEL]` gevolgd door de achternamen van de inzenders. Stuur een `cc` naar je partner zodat een antwoord aan iedereen gegeven kan worden.

```
#include <iostream>
#include "arraystack.h"

using namespace std;

template<class T>
void backspace(string invoer)
{
    T invoerstack;

    // Nog te implementeren: "backspace simulatie"
    invoerstack.push(invoer[0]);
    invoerstack.push('b');
    invoerstack.pop();

    // Nog te implementeren: Resultaat uitvoeren
    cout << invoerstack.top();
    cout << endl;
}

int main()
{
    // Lees een "woord" in, tot de eerste spatie
    string invoer;
    cin >> invoer;

    // Voer vier keer dezelfde test uit, op verschillende implementaties.
    // Voor je datastructuur ArrayStack, bijvoorbeeld:
    backspace<ArrayStack<char> >(invoer);

    return 0;
}
```