Übungsblatt 5

(18. Mai 2007)

(2 Punkte)

Aufgabe 1 Ergänzen Sie die obige Listenklasse um weitere Listenoperatoren.

- `A operator[](int i);` soll das i-te Element zurückgeben.
- `Li<A>* operator-(int i);` soll eine neue Liste erzeugen, indem von der Liste die ersten i Elemente weggelassen werden.
- `bool operator<(Li<A>* that);` soll die beiden Listen bezüglich ihrer Länge vergleichen.
- `bool operator==(Li<A>* that);` soll die Gleichheit auf Listen implementieren.

Testen Sie ihre Implementierung an einigen Beispielen.

Aufgabe 2 Schreiben Sie eine Funktion zur Funktionskomposition mit folgender Signatur:

```
1 template <typename a, typename b, typename c>
2 c composition(c f2(b),b f1(a),a x);
```

Sie sei spezifiziert durch die Gelichung: $composition(f_2, f_1, x) = f_2(f_1(x))$. Testen Sie Ihre Implementierung für verschiedene Typen.

Aufgabe 3 Schreiben Sie eine Funktion `fold` mit folgender Signatur:

Fold.h

```
1 template <typename a,typename b>
2 b fold(a xs [],int length,b op(b,a),b startV);
```

Die Spezifikation der Funktion sei durch folgende Gleichung gegeben:

fold(xs,l,op,st) = op(...op(op(op(st,xs[0]),xs[1]),xs[2])...,xs[l-1])

Oder bei Infixschreibweise der Funktion `op` gleichbedeutend über folgende Gleichung:

$$\text{fold}(\{x_0, x_1,,\ldots,x_{l-1}\},\text{l},\text{op},\text{st}) = st \text{ op } x_0 \text{ op } x_1 \text{ op } x_2 \text{ op } \ldots \text{op } x_{l-1}$$

Testen Sie Ihre Methode `fold` mit folgenden Programm:

FoldTest.cpp

```
#include <iostream>
#include <string>
#include "Fold.h"

int add(int x,int y){return x+y;}
int mult(int x,int y){return x*y;}
int gr(int x,int y){return x>y?x:y;}
int strLaenger(int x,std::string y){
  return x>y.length()?x:y.length();}

int main(){
  int xs [] = {1,2,3,4,5,4,3,2,1};
  std::cout << fold(xs,9,add,0) << std::endl;
  std::cout << fold(xs,9,mult,1)<< std::endl;
  std::cout << fold(xs,9,gr,0)  << std::endl;
  std::string ys [] = {"john","paul","george","ringo","stu"};
  std::cout << fold(ys,5,strLaenger,0)  << std::endl;
}
```

Schreiben Sie ein paar zusätzliche Beispielaufrufe von `fold`.

Aufgabe 4 Sie sollen in dieser Aufgabe die Sortiermethode des Bubble-Sort, wie sie im ersten Semester und in ADS vorgestellt wurde, möglichst generisch umsetzen:

a) Implementieren Sie eine Funktion
`void bubbleSort(int xs [],int length )`,
die Elemente mit dem Bubblesort-Algorithmus der Größe nach sortiert. Schreiben Sie ein paar Tests für Ihre Funktion.

b) Schreiben Sie jetzt eine verallgemeinerte Sortierfunktion, in der die Sortierrelation als Parameter mitgegeben wird. Die Sortierfunktion soll also eine Funktion höherer Ordnung mit folgender Signatur sein:
`void bubbleSortBy(int xs [],int length, bool smaller(int,int) )`

c) Verallgemeinern Sie jetzt die Funktion `bubbleSortBy`, dass Sie generisch für Reihungen mit verschiedenen Elementtypen aufgerufen werden kann.