Thema, Ziele: Listen

Aufgabe 1: Realisierung einer einfach verketteten Liste

Implementieren Sie eine einfach verkettete Liste für `double`-Werte in C++ mit der Entwicklungsumgebung Eclipse. Die Listenklasse erhält den Namen `SList`. `SList` soll die nachfolgenden Methoden anbieten, wobei die geeigneten Parameter noch zu definieren sind. Achten Sie auf den korrekten Einsatz von `const`.

Implementieren Sie auch ein einfaches Testprogramm auf der Console.

Konstruktor (Ctor)
Initialisiert eine leere Liste.

Destruktor (Dtor)
Löscht die gesamte Liste und gibt den allozierten Speicher wieder frei.

`insertAt`
Fügt ein Element an (nach) der bezeichneten Position ein. Einfügen am Anfang der Liste kann mittels Position `0` bewerkstelligt werden.

`deleteAt`
Löscht das Element an der bezeichneten Position.

`search`
Sucht das erste Element in der Liste mit dem definierten Wert und gibt die Position zurück. Falls das Element nicht in der Liste vorhanden ist, wird als Position der Wert `0` zurückgegeben.

`isEmpty`
Gibt `true` zurück, falls die Liste leer ist.

`getNumber`
Gibt die Anzahl der Elemente der Liste zurück.

`getNext`
Gibt die Position des nächsten Elementes in der Liste zurück. Das erste Element hat die Position `1`, wenn kein weiteres Element mehr vorhanden ist, wird die Position `0` zurückgegeben.

`getValue`
Gibt den Wert des Elementes an der bezeichneten Position zurück.

`setValue`
Setzt den Wert des Elementes an der bezeichneten Position.

`print`
Schreibt den Inhalt der Liste in die Console.

Aufgabe 2: Realisierung einer doppelt verketteten Liste

Implementieren Sie eine doppelt verkettete Liste für `double`-Werte in C++. Die Liste erhält den Namen `DList`. `DList` soll dieselben Methoden anbieten wie bereits `SList` in Aufgabe 1, zusätzlich soll noch folgende Methode implementiert werden:

`getPrev`
Gibt die Position des vorherigen Elementes in der Liste zurück. Das erste Element hat die Position `1`, wenn kein weiteres Element mehr vorhanden ist, wird die Position `0` zurückgegeben.

Aufgabe 3: Gemeinsame Basisklasse

Die beiden Listen `SList` und `DList` haben verschiedene Gemeinsamkeiten, eine gemeinsame Basisklasse `List` bietet sich offensichtlich an. Implementieren Sie eine gemeinsame Basisklasse `List`. Achten Sie darauf, dass die Sichtbarkeiten (`public`, `private`, `protected`) so gewählt werden, dass die Unterklassen Zugriff auf die von ihnen benötigten Daten haben. Es ist jedoch keine Lösung, einfach alles als `public` zu deklarieren.

Aufgabe 4: Listen für beliebige Typen

Erweitern Sie die Listenklassen so, dass beliebige Typen abgespeichert werden können.

Zur Erinnerung: Sie erreichen dies durch Verwendung von Templates.