Übungsblatt 5

Ralph Gauges Ursula Rost Katja Wegner

21.11.2007

Aufgabe 1:

Schreiben Sie eine Klasse *Image*, welche die Daten eines Bitmap-Bildes speichern soll. Die Größe des Bildes soll über den Konstruktor festlegbar sein und anschließend nicht mehr verändert werden können. Jeder Pixel der Bitmap soll einen Wert für die rote, grüne und blaue Komponente verfügen, wobei der Wertebereich für die einzelnen Farbkomponenten zwischen 0-255 liegen sollen, d.h. **unsigned char** genügt. Die Klasse soll ferner über Methoden verfügen mit denen man die Farbwerte der einzelnen Pixel setzen und auslesen kann. Da solche Bilder unter Umständen recht groß werden können, soll der Speicher für die Daten dynamisch verwaltet werden.
Wir werden diese Klasse in späteren Aufgaben erweitern und wenn Sie fertig ist in einem größeren Programm benutzen.

Aufgabe 2:

Ziel ist die Implementierung einer Listenklasse welche Integer Werte speichern kann. Das Einfügen und Löschen von Elementen soll stets am Ende erfolgen.
Um die Liste von Elementen mit Hilfe eines Feldes variabler Größe zu realisieren, sollen Sie dynamische Felder verwenden.
Deklarieren Sie das Feld als Pointer:

```
int *elements;
```

Dann können Sie mit Hilfe des **new** Operators zur Laufzeit dem Array eine frei wählbare Größe geben:

```
int k = 10;
elements = new int[k];
```

Schreiben Sie zunächst die Header Datei und vereinbaren Sie dort folgende Attribute und Methoden:

- ein privates **elements** Attribut, welches, wie oben beschrieben, ein Feld von int Werten über einen Pointer definiert
- ein privates**listLength** Attribut, welches zur Laufzeit jeweils die aktuelle Listenlänge enthalten soll
- ein privates **currentPosition** Attribut, welches zur Laufzeit jeweils die Position enthält, an der die nächste Einfügung stattfinden wird
- eine private **init** Methode, die einen **u**nsigned int Parameter hat und nichts zurückgibt
- eine public **insertElementAtEnd1** Methode mit einem **i**nt Parameter
- eine public **deleteElementFromEnd** Methode ohne Parameter
- eine public **printElementAt** Methode mit einem **i**nt Parameter
- eine public **printList** Methode ohne Parameter
- einen Konstruktor ohne Parameter (Standardkonstruktor)
- einen Konstruktor, der einen **u**nsigned int Parameter hat

Implementieren Sie nun die Methoden der Klasse in der zugehörigen Quelldatei mit der folgenden Funktionalität:

init soll dem Attribut **elements** ein Feld von int Werten zuweisen, wobei die Anzahl der Werte durch den Parameter der Methode bestimmt wird (siehe oben). Die Elemente sollen alle auf 0 gesetzt werden. Darüber hinaus müssen die Attribute **listLength** und **currentPosition** entsprechend initialisiert werden (Hinweis: angefangen wird bei Position 0)

insertElementAtEnd1 soll ein Element an der aktuellen Position einfügen und den Wert für die aktuelle Position anschließend entsprechend aktualisieren. Falls die Liste (d.h. das Array) bereits voll ist, soll eine passende Fehlermeldung ausgegeben werden.

deleteElementFromEnd soll das Element an der aktuellen Position löschen, sofern die Liste nicht leer ist

printElementAt falls eine gültige Position angegeben wurde (zwischen Anfang und Ende der Liste) soll dieses Element mit seiner Position zusammen formatiert ausgegeben werden (z.B. in der Form: 'element at position : 27')

printList soll die ganze Liste formatiert ausgegeben und dazu die vorige Methode benutzen

Standardkonstruktor soll die Liste mit 10 Elementen initialisieren

Konstruktor mit Parameter soll die Liste mit der Anzahl von Elementen initialisieren, die durch den Parameter vorgegeben ist

Schreiben Sie nun eine main Funktion, in der zunächst eine Listen-Variable angelegt wird. In das Listen Objekt sollen nun 10 Elemente eingefügt werden. Versuchen Sie nun ein 11. Element einzufügen und geben Sie die Liste dann aus.
Erweitern Sie die Klasse durch zwei Methoden

extend diese Methode soll das Feld, welches die Listenelemente enthält, um einen konstanten Faktor (z.B. um 10 Elemente) verlängern. Hierzu müssen Sie ein neues Feld von der aktuellen Länge + 10 erzeugen, die momentan im Feld enthaltenen Elemente kopieren und den Rest mit 0 auffüllen (und natürlich das Attribut, daß die Länge der Liste enthält, aktualisieren und den Speicher für das ursprüngliche Feld wieder freigeben). Definieren Sie das neue Feld wieder auf die oben beschriebene Art

insertElementAtEnd2 diese Methode soll prüfen, ob noch Platz in der Liste ist: wenn nicht, soll zuerst die Liste erweitert werden (mit der **extend**-Methode, bevor ein Element eingefügt wird.

Fügen Sie mit Hilfe der neuen Methode zwei weitere Elemente in die Liste ein.
Schreiben Sie nun eine weitere Methode **insertSorted**, die eine Element jeweils so einfügt, dass die Liste immer (aufsteigend) sortiert ist. Dazu müssen ab der Stelle, an der eingefügt werden muss, alle weiteren Elemente zunächst um eins nach hinten geschoben werden (Hinweis: fangen Sie hinten an).
Ergänzen Sie die main Funktion derart, dass 11 Elemente gelöscht werden (es sollte dann noch ein Element in der Liste sein) und fügen Sie dann mittels der neuen Methode 10 Elemente sortiert ein und geben Sie die Liste dann nochmals aus.