

Übung 1

Aufgabe 1:

Programmieren Sie eine Methode, die folgendes auf dem Bildschirm ausgibt:

```
  *
 ***
*****
 ***
  *
```

Dabei soll die Höhe durch einen Parameter n angegeben werden.

Aufgabe 2:

Programmieren Sie eine Methode, die eine Zahl n übergeben bekommt und alle Primzahlen $< n$ auf dem Bildschirm ausgibt.

Aufgabe 3:

Programmieren Sie eine Methode, die zwei `int` Zahlen $i1$ und $i2$ übergeben bekommt und diejenige Zahl zurückliefert, die mehr Einsen in ihrer Binärdarstellung enthält.

Hinweis: Zerlegen Sie Ihre Lösungen immer in sinnvolle Hilfsmethoden.

Übung 2

Aufgabe 1:

Zwei Zahlen a und b können dadurch multipliziert werden, dass eine Schleife von 0 bis $b-1$ läuft und immer wieder zum Ergebnis a hinzuaddiert.

Implementieren Sie eine Methode `mult_iter`, die dieses Verfahren iterativ implementiert und eine Methode `mult_rec`, die dieses Verfahren rekursiv implementiert. Beachten Sie, dass a und b auch jeweils negativ sein können.

Aufgabe 2:

Implementieren Sie in die Türme von Hanoi (ohne Klassen, nur mit Text auf dem Konsolenfenster wie in Prog I „Hanoi2“ Lösung, bei der die Türme als Blockgrafik ausgegeben werden).

Übung 3

Aufgabe 1:

Schreiben Sie eine Methode, die solange Zeichen von der Tastatur einliest, bis ein Zeichen ungleich der Ziffern eingegeben wird. Das erste Zeichen darf aber auch das ‚-‘ Minus sein. Die Methode soll die Zahl errechnen, die aus der Reihenfolge der Ziffern besteht. Beispieleingabe: - 2 3 4 2 5, Ausgabe: die Zahl -23425.

Aufgabe 2:

Schreiben Sie eine Methode, die alle Lösung des 8-Dame Problems ausgibt. Das 8-Dame Problem besteht darin, auf einem Schachbrett (8x8 Feld) 8 Damen derart zu platzieren, dass keine Dame eine andere Dame schlagen kann, d.h. in keiner Zeile und in keiner Spalte und in keiner Diagonalen dürfen 2 oder mehr Damen gleichzeitig stehen. (Tipp: arbeiten Sie rekursiv)

Übung 4

Aufgabe 1:

Schreiben Sie die Funktion `cat`, die zwei Strings (`char-Pointer`) übergeben bekommt und einen neuen String (`char Pointer`) zurückliefert, der die Konkatenation der beiden enthält. Vermeiden Sie vordefinierten Funktionen wie `strlen`, `cpy`..

Aufgabe 2:

Schreiben Sie die Funktion `rev`, die einen String (`char-Pointer`) übergeben bekommt und einen neuen String (`char-Pointer`) zurückliefert, der das gespiegelte Wort enthält. Vermeiden Sie vordefinierten Funktionen wie `strlen`, `cpy`.

Aufgabe 3:

Schreiben Sie die Funktion `pal`, die einen String (`char-Pointer`) übergeben bekommt und einen booleschen Wert zurückliefert. Der Wert ist `true`, genau dann wenn das übergebene Wort ein Palindrom ist. Vermeiden Sie vordefinierten Funktionen wie `strlen`, `cpy`.

Übung 5

Aufgabe 1:

Implementieren Sie den Shell- und Quicksort für int-Arrays. Testen Sie Ihre Programme mit Arrays der Größen 10, 100, 1000 und 10000. Dabei sollen die Arrays bereits sortiert, invers sortiert und mit zufälligen Zahlen belegt sein.

Übung 6

Aufgabe 1:

Implementieren Sie den Heap- und Mergesort für int-Arrays. Testen Sie Ihre Programme mit Arrays der Größen 10, 100, 1000 und 10000. Dabei sollen die Arrays bereits sortiert, invers sortiert und mit zufälligen Zahlen belegt sein.

Übung 7

Aufgabe 1:

Implementieren Sie eine Vektorklasse für double Zahlen. Fügen Sie der Implementierung eine `push_front` Methode hinzu, die am Anfang des Vektors ein neues Element hinzufügt.

Aufgabe 2:

Modifizieren Sie Ihre `push_front` Methode derart, dass sie analog wie die `push_back` Methode auch am Anfang mehr Elemente alloziert als notwendig ist.

Übung 8

Aufgabe 1:

Implementieren Sie eine Klasse A, die einen int Wert speichert. Implementieren Sie eine einfach verkettete Liste, die A Objekte speichert. Legen Sie eine Liste an, speichern mehrere A Objekte ab und geben die Liste aus. Die Liste soll zusätzlich die Methoden front (erstes Element zurückliefern) und back (letztes Element zurückliefern) besitzen. Denken Sie an den Destruktor, Copykonstruktor, Zuweisungsoperator und Ausgabeoperator.

Aufgabe 2:

Erweitern Sie die Implementierungen von Vektoren und Listen um eine pop_front und pop_back Methode, die das erste bzw. letzte Element entfernt.

Aufgabe 3:

Implementieren Sie eine doppelt verkettete Liste für unsigned int, die als Methoden push_front (Element am Anfang hinzufügen), push_back (Element am Ende hinzufügen), pop_front (erstes Element entfernen), pop_back (letztes Element entfernen), front (erstes Element zurückliefern) und back (letztes Element zurückliefern) besitzt. Denken Sie an den Destruktor, Copykonstruktor, Zuweisungsoperator und Ausgabeoperator.

Übung 9

Aufgabe 1:

Implementieren Sie eine Klasse Rational zur Speicherung und Darstellung rationaler Zahlen. Implementieren Sie sinnvolle Konstruktoren und Operatoren (+, -, *, /, <, <=, >, >=, ==, !=, <<, =). Die Speicherung soll immer in gekürzter Form erfolgen.

Aufgabe 2:

Wiederholen Sie die erste Aufgabe jedoch für komplexe Zahlen.

Aufgabe 3:

Implementieren Sie den Bubblesort für Rationale Zahlen.

Aufgabe 4:

Implementieren Sie eine Klasse, die gleichzeitig einen Stack und eine Queue für unsigned int realisiert. Diese Klasse soll von der Klasse der doppelt verketteten Liste abgeleitet sein. Welche der beiden Funktionalitäten im tatsächlichen Objekt realisiert werden, soll durch ein boolesches Flag im Konstruktor entschieden werden.

Übung 10

Aufgabe 1:

Implementieren Sie einen binären Suchbaum über unsigned int. Denken Sie an den Destruktor, Copykonstruktor, Zuweisungsoperator und Ausgabeoperator.

Aufgabe 2:

Implementieren Sie für den binären Suchbaum über unsigned int eine rekursive Suchmethode und eine Methode, die die Anzahl der Knoten zurückliefert.

Übung 11

Aufgabe 1:

Implementieren Sie eine Hashtabelle, die Strings auf float Werte abbildet. Denken Sie an den Destruktor, Copykonstruktor und Zuweisungsoperator. Implementieren Sie für die Hashtabelle auch eine remove Methode, die Elemente aus der Hashtabelle löscht. Die remove Methode bekommt einen Schlüssel (sprich String) übergeben. Testen Sie Ihre Methode und prüfen Sie, ob Sie nach dem Löschen noch immer alle anderen Elemente finden können. Wann ist es sinnvoll, ein resize durchzuführen?

Aufgabe 2:

Testen Sie Ihre Hashtabelle, indem Sie viele Werte eintragen. Protokollieren Sie die Häufigkeit von Kollisionen. Wie viele Kollisionen treten im Durchschnitt auf, wenn ein neues Element eingetragen wird und die Tabelle zu 40% (respektive 50%, 60%, 70%, 80%, 90% 95%) gefüllt ist?

Wichtig: Überlegen Sie sich zunächst, wie ein solcher Testaufbau stattfinden kann. Würden Sie bei einer Füllung von 40% einen Eintrag vornehmen und die Kollisionen zählen, wäre für den nächsten Eintrag die Füllung nicht mehr 40% sondern 40% + 1 Eintrag. **Dies soll nicht passieren!** Führen Sie die Messungen mit unterschiedlichen Hashfunktionen durch.

Übung 12

Aufgabe 1:

Fügen Sie der Implementierung von Vektoren und Listen Exceptions hinzu, wenn mittels des []-Operators oder der front(), back(), pop_front() bzw. pop_back() Methoden auf illegale Elemente bzw. leere Vektoren / Listen zugegriffen wird.

Übung 13

Aufgabe 1:

Implementieren Sie eine Klasse Student, die einen Namen, Vornamen (beides als char*), eine Matrikelnummer und das Fachsemester (beides unsigned int) enthält. Implementieren Sie eine weitere Klasse Verwaltung, die ein Array von Studenten enthält. Diese Klasse soll neue Studenten anlegen und bestehende Studenten löschen können. Sie soll das Fachsemester aller Studenten erhöhen können, Studenten bestimmter oder auch aller Fachsemester ausgeben können. Implementieren Sie noch eine Suchfunktion, so dass nach Namen oder Matrikelnummer gesucht werden kann. Wird nach Namen gesucht, müssen u.U. mehrere Studenten zurückgegeben werden.

Achten Sie darauf, dass im Destruktor alle Daten wieder freigegeben werden.

Aufgabe 2:

Implementieren Sie noch einmal die Türme von Hanoi, jedoch mit Klassen. Achten Sie darauf, dass die Destruktoren die Daten am Ende wieder freigeben.

Übung 14

Aufgabe 1:

Stellen Sie die Implementierung von Vektoren, Listen und Bäumen auf Templates um.