Familienname:

Vorname:

Matrikelnummer:

Aufgabe 1 (2 Punkte):
Aufgabe 2 (4 Punkte):
Aufgabe 3 (2 Punkte):
Aufgabe 4 (2 Punkte):
Aufgabe 5 (1 Punkte):
Aufgabe 6 (4 Punkte):
Aufgabe 7 (3 Punkte):
Aufgabe 8 (4 Punkte):
Aufgabe 9 (4 Punkte):
Aufgabe 10 (3 Punkte):
Aufgabe 11 (1 Punkte):
Aufgabe 12 (4 Punkte):
Aufgabe 13 (3 Punkte):
Aufgabe 14 (3 Punkte):

Gesamtpunktzahl:

Schriftlicher Test (120 Minuten)
VU Einführung ins Programmieren für TM

20. Juni 2014

Aufgabe 1 (2 Punkte). Was sind Lifetime und Scope einer Variable?

Lösung zu Aufgabe 1.

Aufgabe 2 (4 Punkte). Was ist der Output des folgenden Programms? Zeichnen Sie einen Zeitstrahl, wo Sie Lifetime und Scope der Variablen `x`, `y` und `z` auftragen. Kennzeichnen Sie am Zeitstrahl die einzelnen Blöcke und Funktionen.

```
1  #include <stdio.h>
2
3  int max(int,int);
4
5  main() {
6    int x = 1;
7    int y = 2;
8    int z = 3;
9
10   printf("(x,y,z) = (%d,%d,%d)\n",x,y,z);
11
12   {
13     int x = 100;
14     y = 2;
15     z = max(x,y);
16     printf("(x,y,z) = (%d,%d,%d)\n",x,y,z);
17
18     {
19       int z = y;
20       y = 200;
21
22       printf("(x,y,z) = (%d,%d,%d)\n",x,y,z);
23     }
24     printf("(x,y,z) = (%d,%d,%d)\n",x,y,z);
25   }
26   printf("(x,y,z) = (%d,%d,%d)\n",x,y,z);
27 }
28
29 int max(int x, int y) {
30   if(x>=y) {
31     return x;
32   }
33   else {
34     return y;
35   }
36 }
```

Lösung zu Aufgabe 2.

Aufgabe 3 (2 Punkte). Schreiben Sie einen C-Strukturdatentyp `vector` zur Speicherung von Vektoren $x \in \mathbb{R}^N$. In der Struktur soll neben dem dynamischen Vektor der Koeffizienten auch die Dimension $N \in \mathbb{N}$ gespeichert werden.

Lösung zu Aufgabe 3.

Aufgabe 4 (2 Punkte). Schreiben Sie eine Funktion `newVector`, die einen Vektor $x \in \mathbb{R}^N$ allokiert und mit Null initialisiert. **Hinweis.** Verwenden Sie den Strukturdatentyp aus Aufgabe 3.

Lösung zu Aufgabe 4.

Aufgabe 5 (1 Punkt). Schreiben Sie eine Funktion `getVectorCoeff`, die den Koeffizienten x_j eines Vektors $x \in \mathbb{R}^N$ zurückgibt. **Hinweis.** Verwenden Sie den Strukturdatentyp aus Aufgabe 3.

Lösung zu Aufgabe 5.

Aufgabe 6 (4 Punkte). Schreiben Sie eine Funktion `normVector`, die für einen Vektor $x \in \mathbb{R}^N$ die Norm

$$\|x\| := \max_{j=1,\dots,N} \Big| \sum_{k=1}^{j} x_k \Big|$$

berechnet und zurückgibt. **Hinweis.** Verwenden Sie den Strukturdatentyp aus Aufgabe 3.

Lösung zu Aufgabe 6.

Aufgabe 7 (3 Punkte). Schreiben Sie eine Funktion `maxabsVector`, die für einen Vektor $x \in \mathbb{R}^N$ den ersten Index $k \in \{1, \ldots, N\}$ mit

$$|x_k| = \max_{j=1,\ldots,N} |x_j|$$

bestimmt und zurückgibt. **Hinweis.** Verwenden Sie den Strukturdatentyp aus Aufgabe 3.

Lösung zu Aufgabe 7.

Aufgabe 8 (4 Punkte). Schreiben Sie eine Funktion `subVector`, die für einen Vektor $x \in \mathbb{R}^N$ und eine Schranke $C > 0$ den Vektor $y \in \mathbb{R}^n$ mit $n \leq N$ zurückgibt, bei dem alle Einträge x_j mit $|x_j| > C$ gestrichen werden. Für $x = (1, 4, -2, -3, -4)$ und $C = 3$ gilt beispielsweise $y = (1, -2, -3)$. Für $n = 0$ werde `NULL` zurückgegeben. **Hinweis.** Verwenden Sie den Strukturdatentyp aus Aufgabe 3. Sie dürfen die Existenz und kanonische Funktionalität einer Funktion `setVectorCoeff` voraussetzen. Bestimmen Sie in Ihrer Implementierung zunächst die Länge n, und legen Sie dann den Vektor $y \in \mathbb{R}^n$ mittels `newVector` an.

Lösung zu Aufgabe 8.

Aufgabe 9 (4 Punkte). Schreiben Sie eine C++ Klasse `Vector` zur Speicherung von Vektoren $x \in \mathbb{R}^N$. In der Klasse soll neben dem dynamischen Vektor der Koeffizienten (`double *`) auch die Dimension $N \in \mathbb{N}$ gespeichert werden. Ferner soll die Klasse die folgenden Methoden enthalten:

- Destruktor,
- Konstruktor zum Allokieren des Null-Vektors $x \in \mathbb{R}^N$ mit $x_j = 0$,
- Konstruktor zum Allokieren eines konstanten Vektors $x \in \mathbb{R}^N$ mit $x_j = C \in \mathbb{R}$,
- `getSize`-Methode, um die Länge N des Vektors auszulesen,
- `getEntry` und `setEntry`, um einen Koeffizienten x_j zu lesen oder zu schreiben.

Hinweis. An dieser Stelle sollen nur die Signaturen implementiert, nicht die Funktionalität der Methoden. Der Einfachheit halber verzichten wir in dieser Aufgabe auf die Dreierregel, d.h. Sie brauchen keinen Kopierkonstruktor bzw. Zuweisungsoperator zu implementieren.

Lösung zu Aufgabe 9.

Aufgabe 10 (3 Punkte). Schreiben Sie die beiden Konstruktoren der Klasse `Vector` aus Aufgabe 9. **Hinweis.** Verwenden Sie `new`.

Lösung zu Aufgabe 10.

Aufgabe 11 (1 Punkt). Schreiben Sie den Destruktur der Klasse `Vector` aus Aufgabe 9. **Hinweis.** Verwenden Sie `delete`.

Lösung zu Aufgabe 11.

Aufgabe 12 (4 Punkte). Die folgende C++ Funktion soll einen Vektor $x \in \mathbb{R}^N$ aufsteigend sortieren (Bubblesort), d.h. $x_j \leq x_{j+1}$ für alle Indizes j. Korrigieren Sie allfällige syntaktische und logische Fehler und begründen Sie kurz, was falsch ist. **Hinweis.** Der Input der Funktion soll vom Typ `Vector` sein.

```
1 void bubblesort(Vector &x) {
2   int N = getSize(x);
3   for (int i = N-1; i >= 1; i--) {
4     for (int j = 0; j < i; j++) {
5       if ( x[j] < x[j+1] ) {
6         // vertauschen
7         x[j+1] = x[j];
8         x[j] = x[j+1];
9       }
10    }
11  }
12 }
```

Lösung zu Aufgabe 12.

Aufgabe 13 (3 Punkte). Was versteht man unter Aufwand? Welchen Aufwand hat die Implementierung von Bubblesort aus Aufgabe 12? Erklären Sie in diesem Zusammenhang auch die Landau-$\mathcal{O}(\cdot)$-Notation.

Lösung zu Aufgabe 13.

Aufgabe 14 (3 Punkte). Überladen Sie den `+` Operator so, dass er die Summe $x + y$ zweier Vektoren $x, y \in \mathbb{R}^N$ gleicher Länge berechnet. **Hinweis.** Verwenden Sie die Klasse `Vector` aus Aufgabe 9.

Lösung zu Aufgabe 14.