Familienname:

Vorname:

Matrikelnummer:

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

Gesamtpunktzahl:

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

22. Jänner 2015

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

Lösung zu Aufgabe 1.

Lösung zu Aufgabe 1.

Aufgabe 2 (2 Punkte). Schreiben Sie einen C-Strukturdatentyp `Polynomial` zur Speicherung von Polynomen $p(x) := \sum_{j=0}^{N} a_j x^j$. In der Struktur soll neben dem dynamischen Vektor der Koeffizienten auch der Grad $N \in \mathbb{N}_0$ des Polynoms gespeichert werden.
Hinweis. Diese Struktur sowie die zugehörigen Funktionen

- `delPolynomial`
- `getPolynomialDegree`
- `setPolynomialCoeff`

sollen in den folgenden Aufgaben 3–7 verwendet werden, wobei Sie die Existenz dieser Funktionen voraussetzen dürfen.

Lösung zu Aufgabe 2.

Aufgabe 3 (2 Punkte). Schreiben Sie die Funktion `newPolynomial`, die ein Polynom $p(x) = \sum_{j=0}^{N} a_j x^j$ vom Grad N allokiert und den Koeffizientenvektor mit Null initialisiert.
Hinweis. Verwenden Sie den Strukturdatentyp aus Aufgabe 2.

Lösung zu Aufgabe 3.

Aufgabe 4 (1 Punkt). Schreiben Sie eine Funktion `getPolynomialCoeff`, die den j-ten Koeffizienten a_j eines Polynoms $p(x) := \sum_{j=0}^{N} a_j x^j$ zurückgibt.
Hinweis. Verwenden Sie den Strukturdatentyp aus Aufgabe 2.

Lösung zu Aufgabe 4.

Aufgabe 5 (3 Punkte). Schreiben Sie eine Funktion `evalPolynomial`, die ein Polynom $p(x)$ an einer gegebenen Stelle t auswertet und den Funktionswert $p(t)$ zurückgibt.
Hinweis. Verwenden Sie den Strukturdatentyp aus Aufgabe 2. Die Funktion `pow` darf nicht verwendet werden.

Lösung zu Aufgabe 5.

Aufgabe 6 (4 Punkte). Die Ableitung $p'(x)$ eines Polynoms $p(x)$ ist wieder ein Polynom. Schreiben Sie eine Funktion `diffPolynomial`, die die Ableitung $p'(x)$ berechnet und als Polynom zurückgibt.
Hinweis. Verwenden Sie den Strukturdatentyp aus Aufgabe 2.

Lösung zu Aufgabe 6.

Aufgabe 7 (5 Punkte). Eine Möglichkeit, eine Nullstelle eines Polynoms $p(x)$ zu berechnen, ist das Newton-Verfahren. Ausgehend von einem Startwert x_0 definiert man induktiv eine Folge (x_n) durch

$$x_{k+1} = x_k - p(x_k)/p'(x_k).$$

Schreiben Sie eine Funktion `getPolynomialZero`, die zu gegebenen Polynom $p(x)$, Startwert x_0 und Toleranz τ das Newton-Verfahren durchführt, wobei die Iteration endet, falls

$$|p(x_n)| \leq \tau \quad \text{und} \quad |x_n - x_{n-1}| \leq \tau$$

gelten. In diesem Fall werde der letzte Wert x_n als Approximation der Nullstelle zurückgegeben.
Hinweis. Verwenden Sie den Strukturdatentyp aus Aufgabe 2. Vergessen Sie nicht, Hilfsspeicher wieder freizugeben.

Lösung zu Aufgabe 7.

Aufgabe 8 (3 Punkte). Erklären Sie den Begriff *Vererbung* in C++ anhand eines selbstgewählten Beispiels. Erklären Sie in diesem Zusammenhang auch `public`, `protected`, `private` als Qualifier für die Members bei `public`-Vererbung.

Lösung zu Aufgabe 8.

Aufgabe 9 (4 Punkte). Schreiben Sie eine C++ Klasse `Polynomial` zur Speicherung von Polynomen $p(x) := \sum_{j=0}^{N} a_j x^j$. In der Klasse soll neben dem (dynamischen) Vektor der Koeffizienten auch der Grad $N \in \mathbb{N}_0$ gespeichert werden. Ferner soll die Klasse die folgenden Methoden enthalten:

- Destruktor,
- Konstruktor zum Allokieren von $p(x) = \sum_{j=0}^{N} a_j x^j$ mit $N \in \mathbb{N}_0$ und $a = 0 \in \mathbb{R}^{N+1}$,
- Kopierkonstruktor,
- Zuweisungsoperator,
- `getDegree`-Methode, um den Grad N des Polynoms auszulesen,
- Zugriff mittels `[ ]`, um auf die Koeffizienten a_j lesend oder schreibend zuzugreifen,
- Auswertung des Polynoms an einer Stelle $t \in \mathbb{R}$ mittels `( )`.

Hinweis. An dieser Stelle sollen nur die Signaturen implementiert werden, nicht die Funktionalität der Methoden.

Lösung zu Aufgabe 9.

Lösung zu Aufgabe 9.

Aufgabe 10 (2 Punkte). Schreiben Sie den Konstruktor der Klasse `Polynomial` aus Aufgabe 9.

Lösung zu Aufgabe 10.

Aufgabe 11 (3 Punkte). Schreiben Sie den Zuweisungsoperator der Klasse `Polynomial` aus Aufgabe 9.

Lösung zu Aufgabe 11.

Aufgabe 12 (4 Punkte). Überladen Sie den `+` Operator so, dass er die Summe $r(x) = p(x) + q(x)$ zweier Polynome $p(x) := \sum_{j=0}^{M} a_j x^j$ und $q(x) := \sum_{k=0}^{N} b_k x^k$ berechnet.
Hinweis. Verwenden Sie die Klasse `Polynomial` aus Aufgabe 9. Beachten Sie, dass $p(x)$ und $q(x)$ unterschiedlichen Grad haben können und legen Sie $r(x)$ mit Grad $\max\{M, N\}$ an.

Lösung zu Aufgabe 12.

Aufgabe 13 (3 Punkte). Was sind die Bestandteile M, $e_{\min}$, $e_{\max}$ des Gleitkommazahlsystems $\mathbb{F}(2, M, e_{\min}, e_{\max})$? Wie lässt sich jede Gleitkommazahl $x \in \mathbb{F}(2, M, e_{\min}, e_{\max})$ darstellen? Welchen Wert haben die größte und die kleinste positive normalisierte Gleitkommazahl im `double`-Gleitkommazahlsystem $\mathbb{F}(2, 53, -1021, 1024)$?

Lösung zu Aufgabe 13.