Familienname:

Vorname:

Matrikelnummer:

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

Gesamtpunktzahl:

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

24. Januar 2013

Aufgabe 1 (2 Punkte). Schreiben Sie einen Struktur-Datentyp `polynomial` zur Speicherung von Polynomen beliebigen Grades $n \in \mathbb{N}$, die bezüglich der Monombasis dargestellt sind, d.h.

$$p(x) = \sum_{j=0}^{n} a_j x^j.$$

In der Struktur sollen neben dem Grad n auch die Koeffizienten $(a_0, \ldots, a_n) \in \mathbb{R}^{n+1}$ gespeichert werden. **Diese Struktur soll auch in den nachfolgenden Aufgaben verwendet werden.**

Lösung zu Aufgabe 1.

Aufgabe 2 (3 Punkte). Schreiben Sie eine Funktion `newPoly`, die für gegebenes $n \in \mathbb{N}$ ein Polynom vom Typ `polynomial` allokiert und initialisiert.

Lösung zu Aufgabe 2.

Aufgabe 3 (2 Punkte). Schreiben Sie eine Funktion `delPoly`, die den Speicher eines mittels `newPoly` allokierten Polynoms freigibt und den `NULL`-Pointer zurückliefert.

Lösung zu Aufgabe 3.

Aufgabe 4 (2 Punkte). Schreiben Sie eine Funktion `getCoefficient`, die zu einem übergebenen Polynom p und einem Index i den Koeffizienten a_i zurückgibt. Achten Sie insbesondere darauf, den Fall $i > n$ ordentlich zu behandeln.

Lösung zu Aufgabe 4.

Aufgabe 5 (3 Punkte). Die folgende Funktion `swap` soll die Werte zweier Variablen vertauschen. Welchen Wert haben die Variablen `x` und `y` nach dem Aufruf von `swap(x,y)` in `main` und warum? Schreiben Sie eine Version von `swap`, die das Richtige tut, und geben sie auch den Aufruf in `main` an.

```
void swap(double x, double y)
{
   double tmp;
   tmp = x;
   x = y;
   y = tmp;
}
```

Lösung zu Aufgabe 5.

Aufgabe 6 (3 Punkte). Was versteht man unter Rekursion? Was sind die Vor- und Nachteile? Schreiben Sie exemplarisch einen rekursiven C-Code.

Lösung zu Aufgabe 6.

Aufgabe 7 (4 Punkte). Schreiben Sie die Klassendefinition zu einer Klasse `polynomial` die, wie die Struktur aus Aufgabe 2, zur Speicherung von Polynomen von beliebigem Grad $n \in \mathbb{N}$ verwendet werden soll. Entgegen der bisher verwendeten Struktur soll die `polynomial`-Klasse die Koeffizienten in Form eines `vector<double>` speichern. Der Grad n soll nicht explizit, sondern implizit durch die Länge des Koeffizientenvektors gespeichert werden. Außerdem soll die Klasse über folgende Funktionalität verfügen:

- Ein Konstruktor, der zu einem gegebenen Koeffizientenvektor $v \in \mathbb{R}^{n+1}$ ein Polynom vom Grad n mit diesen Koeffizienten erzeugt.
- Ein Standardkonstruktor, der das Nullpolynom erzeugt.
- eine Methode `getDegree`, welche den Grad n des Polynoms zurückliefert.
- `get`- und `set`-Methoden zum Zugriff auf die Koeffizienten.
- Die Möglichkeit, zwei Polynome p und q mit der Syntax $r = p + q$ addieren zu können, sowie die Möglichkeit eine Zahl $d \in \mathbb{R}$ mittels $r = p + d$ zu einem Polynom hinzuaddieren zu können.
- Eine Methode `computeDerivative`, welche die Ableitung $q = p'$ des Polynoms p berechnet und als Polynom vom Grad $n - 1$ zurückliefert.
- Eine Methode `computeZero` zur Bestimmung einer Nullstelle des Polynoms mit Hilfe des Newton-Verfahrens.
- Eine Methode `eval`, die das Polynom p an einer gegebenen Position $x \in \mathbb{R}$ auswertet, d.h. $p(x)$ zurückgibt.

Hinweis: An dieser Stelle ist nur die Klassendefinition gefragt. Es soll also noch keine Funktionalität implementiert werden. Denken Sie daran, entsprechende Zugriffsspezifizierer zu verwenden.

Lösung zu Aufgabe 7.

Aufgabe 8 (3 Punkte). Implementieren Sie `computeDerivative`, und geben Sie das Resultat wieder als Polynom zurück.
Hinweis: Sie können die `get`- und `set`-Methoden ohne Implementierung verwenden.

Lösung zu Aufgabe 8.

Aufgabe 9 (4 Punkte). Die Summe zweier Polynome ist wieder ein Polynom. Implementieren Sie die nötige Funktionalität, um zwei Polynome p und q mittels

$$r = p + q$$

addieren zu können. Eine Zahl vom Typ `double` ist auch ein Polynom (vom Grad 0). Implementieren Sie die Möglichkeit eine Zahl zu einem Polynom hinzuaddieren zu können. Hierbei soll für ein Polynom p und eine Zahl $a \in \mathbb{R}$

$$r = p + a$$

möglich sein.
Hinweis: Sie können die `get`- und `set`-Methoden ohne Implementierung verwenden.

Lösung zu Aufgabe 9.

Aufgabe 10 (4 Punkte). Implementieren Sie die Methode `computeZero`. Dieser wird eine Zahl $x_0 \in \mathbb{R}$ übergeben, mit Hilfe derer dann durch das Newton-Verfahren eine Nullstelle des Polynoms berechnet und zurückgegeben werden soll. Hierbei gehen Sie wie folgt vor: Ausgehend vom Startwert x_0 definieren Sie induktiv die Folge (x_n) durch

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

wobei p' die Ableitung von p bezeichnet. Die Folge (x_n) konvergiert dann gegen eine Nullstelle von p. Für eine gegebene Toleranz τ soll die Iteration abgebrochen werden, falls entweder

$$|p'(x_n)| \leq \tau$$

oder

$$|p(x_n)| \leq \tau$$

gilt. Im ersten Fall soll ein Hinweis ausgegeben werden, dass das numerische Ergebnis wahrscheinlich falsch ist. Außerdem soll das Programm in diesem Fall kontrolliert beendet werden (`assert`). Im zweiten Fall werde x_n als Approximation der gesuchten Nullstelle zurückgegeben.
Hinweis: Sie können (und sollen) die Methode `eval` ohne Implementierung verwenden.

Lösung zu Aufgabe 10.

Aufgabe 11 (3 Punkte). Was versteht man unter Vererbung, und wo ist der Vorteil? Erklären Sie in diesem Zusammenhang auch den Begriff Polymorphie. Was versteht man unter dem Begriff *Zugriffsspezifizierer*?

Lösung zu Aufgabe 11.

Aufgabe 12 (4 Punkte). Was tut der folgende C++ Code? Welche Funktionalität haben die Funktionen `func1` und `func2`? Wie sieht die Bildschirmausgabe aus?

```
#include <iostream>
#include <vector>
using namespace std;

void func2(vector<double> &dp, int mp);

void func1(vector<double> &dp)
{
    int mp = dp.size();
    func2(dp,mp);

    for (int i = mp-1; i>=1; i--){
        for(int j = 0; j<i; j++){
            if (dp[j] > dp[j+1])
                swap(dp[j], dp[j+1]); //Vertausche dp[j] und dp[j+1]
        }
        func2(dp,mp);
    }
}

void func2(vector<double> &dp, int mp){
    for (int k = 0; k<= mp-1; k++){
      cout << dp[k] << endl;
    }
}

int main(){
    vector<double> a(4,0);
    a[0] = 14;
    a[1] = 12;
    a[2] = 7;
    a[3] = 4;
    func1(a);
}
```

Lösung zu Aufgabe 12.

Aufgabe 13 (3 Punkte). Was versteht man unter *Aufwand*? Was besagt die Landau-Notation $\mathcal{O}(N^\alpha)$? Welchen Aufwand hat die Funktion `func1` aus der letzten Aufgabe in Landau-Notation? Begründen Sie Ihre Antwort. Was bedeutet dieser Aufwand für die Laufzeit der Funktion in Abhängigkeit von der Länge N des Vektors, d.h. welche Auswirkungen hat beispielsweise eine Verdopplung der Länge?
Hinweis: Der Aufwand von `swap` ist konstant, d.h. $\mathcal{O}(1)$.

Lösung zu Aufgabe 13.