Familienname:

Vorname:

Matrikelnummer:

Aufgabe 1 (1 Punkte):
Aufgabe 2 (2 Punkte):
Aufgabe 3 (2 Punkte):
Aufgabe 4 (1 Punkte):
Aufgabe 5 (2 Punkte):
Aufgabe 6 (3 Punkte):
Aufgabe 7 (3 Punkte):
Aufgabe 8 (3 Punkte):
Aufgabe 9 (3 Punkte):
Aufgabe 10 (1 Punkte):
Aufgabe 11 (3 Punkte):
Aufgabe 12 (2 Punkte):
Aufgabe 13 (3 Punkte):
Aufgabe 14 (5 Punkte):
Aufgabe 15 (2 Punkte):
Aufgabe 16 (4 Punkte):

Gesamtpunktzahl:

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

28. Juni 2013

Aufgabe 1 (1 Punkt). Schreiben Sie einen Strukturdatentyp `Complex` zur Speicherung komplexer Zahlen. Eine komplexe Zahl $z = x + iy \in \mathbb{C}$ wird durch ihren Realteil $x \in \mathbb{R}$ und Imaginärteil $y \in \mathbb{R}$ beschrieben. **Diese Struktur soll auch in den nachfolgenden Aufgaben verwendet werden.**

Aufgabe 2 (2 Punkte). Implementieren Sie `get`- und `set`-Funktionen für den Realteil des Datentyps `Complex`.

Aufgabe 3 (2 Punkte). Schreiben Sie eine Funktion mit der Signatur `Complex* newComplex(double x, double y)` zum Anlegen einer komplexen Zahl $z = x + iy$. Verwenden Sie in dieser Aufgabe die Funktion `malloc`.

Aufgabe 4 (1 Punkt). Schreiben Sie eine Funktion `delComplex(Complex* z)` zum Freigeben einer zuvor mittels `newComplex` angelegten komplexen Zahl. Die Funktion soll den `NULL`-Pointer zurückliefern.

Aufgabe 5 (2 Punkte). Schreiben Sie eine Funktion `mult`, welche die Multiplikation zweier komplexer Zahlen realisiert und das Ergebnis als `Complex*` zurückliefert. Für zwei

komplexe Zahlen $w = u + iv, z = x + iy$ gilt

$$(u + iv)(x + iy) = ux - vy + i(vx + uy).$$

Sie können annehmen, dass die `get/set`-Funktionen für den Imaginärteil vorhanden sind.

Aufgabe 6 (3 Punkte). Geben Sie eine C-Funktion an, welche ein dynamisches Array der Länge N von komplexen Zahlen anlegt und jedes Element mit 0 initialisiert. Benutzen Sie für dieses Beispiel die Funktion `newComplex` aus Aufgabe 3.

Aufgabe 7 (3 Punkte). Was versteht man unter einer rekursiven Funktion? Was sind Vor- und Nachteile der rekursiven Programmierung? Schreiben Sie exemplarisch eine rekursive Funktion `factorial` welche bei Eingabe einer Zahl $n \in \mathbb{N}$ die Faktorielle $n! = n \cdot (n-1)!$ zurückliefert.

Aufgabe 8 (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 = x;
  x = y;
  y = tmp;
}

int main() {
  double x = 1;
  double y = 0;
  swap(x,y);
  // Wert von x,y an dieser Stelle
}
```

Aufgabe 9 (3 Punkte). Schreiben Sie die Definition einer Klasse `Complex`, welche eine komplexe Zahl $z = x + iy \in \mathbb{C}$ mit $x, y \in \mathbb{R}$ wie in Aufgabe 1 speichert. Diese Klasse soll folgende Methoden enthalten, ohne dass deren Funktionalität implementiert werden soll.

- Den Standardkonstruktor, der Real- und Imaginärteil auf 0 setzt
- Einen Konstruktor, der eine reelle Zahl (`double`) übernimmt und als komplexe Zahl abspeichert.
- `get`-Funktionen für Real- und Imaginärteil

Hinweis: Denken Sie daran, entsprechende Zugriffsspezifizierer zu verwenden. Für nachfolgende Aufgaben können Sie davon ausgehen, dass die `get`-Funktionen implementiert sind.

Aufgabe 10 (1 Punkt). Implementieren Sie die Konstruktoren der Klasse `Complex` aus Aufgabe 8.

Aufgabe 11 (3 Punkte). Überladen Sie den Operator `>` für Instanzen der Klasse `Complex`, so dass dieser Operator die unten definierte Relation realisiert. Der Rückgabewert des Operators `>` soll dabei `bool` sein.
Wir definieren die Relation $>$ auf $\mathbb{C}$ folgendermaßen: Für zwei Zahlen $w, z \in \mathbb{C}$ gilt $w > z$, falls der Realteil von w größer als der Realteil von z ist oder falls die Realteile von w, z übereinstimmen und der Imaginärteil von w größer als der Imaginärteil von z ist.

Aufgabe 12 (2 Punkte). Überladen Sie den Operator `==` für Instanzen der Klasse `Complex`. Zwei komplexe Zahlen w, z stimmen überein falls die Real- und Imaginärteile übereinstimmen.

Aufgabe 13 (3 Punkte). Welche Funktionalität hat die Funktion `wasMacheIch`? Wie sieht die Bildschirmausgabe aus?

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

vector<double> wasMacheIch(vector<double>& vec) {
  vector<double> tmp;

  tmp.push_back(vec[0]);
  int k=0;

  for(int j=1; j<vec.size(); ++j) {
    if(vec[j] != tmp[k]) {
      tmp.push_back(vec[j]);
      k++;
    }
    cout << "j = " << j << ", vec[" << j << "] = " << vec[j]
      << ", k = " << k << endl;
    for(int m=0; m<tmp.size(); ++m)
      cout << "tmp[" << m << "] = " << tmp[m] << " ";
    cout << endl;
  }
  return tmp;
}

int main() {
  vector<double> vec(5);
  vec[0] = 1; vec[1] = 1; vec[2] = 3; vec[3] = 1; vec[4] = 2;
  vector<double> t = wasMacheIch(vec);
}
```

Aufgabe 14 (5 Punkte). Schreiben Sie eine Funktion

```
void sort(vector<Complex>& vec)
```

welche den Vektor `vec` mit komplexen Einträgen sortiert. Dabei soll als Sortierverfahren der *Bubble-Sort-Algorithmus* benutzt werden. Folgendes Prinzip liegt bei diesem Algorithmus zu Grunde: Gegeben sei ein Vektor $\mathbf{v} = (v_0, \ldots, v_{N-1})$. Im ersten Durchlauf vergleicht man für $j = 0, \ldots, N-2$ jedes Element v_j mit seinem Nachfolger v_{j+1}. Gilt $v_j > v_{j+1}$, werden die beiden Elemente vertauscht. Nach dem ersten Durchlauf, weiß man mit Sicherheit, dass das größte Element an der letzten Stelle im Vektor steht. Im zweiten

Durchlauf vergleicht man daher für $j = 0, \ldots, N-3$ das Element v_j mit seinem Nachfolger v_{j+1}. Gilt $v_j > v_{j+1}$, werden die beiden Elemente vertauscht. Nach dem zweiten Durchlauf, weiß man dann, dass das zweitgrößte Element im gesamten Vektor an der vorletzten Stelle steht. Es werden insgesamt so viele Durchläufe gemacht, bis das zweitkleinste Element an zweiter Stelle steht (dann weiß man auch, dass das kleinste Element an der ersten Stelle im Vektor ist).
Hinweis: Überlegen Sie sich zunächst wie viele Durchläufe man benötigt. Danach überlege man sich, bis zu welchem Index in jedem Durchlauf gegangen wird.

Aufgabe 15 (2 Punkte). Was versteht man unter Vererbung, und worin besteht der Vorteil? Erklären Sie in diesem Zusammenhang auch den Begriff Polymorphie.

Aufgabe 16 (4 Punkte). Die Exponentialfunktion kann auch als folgende Reihe

$$\exp(t) = \sum_{j=0}^{\infty} \frac{t^j}{j!} \qquad \text{für } t \in \mathbb{R}$$

dargestellt werden. Diese Reihendarstellung dient auch dazu, die Exponentialfunktion für andere mathematische Objekte, z.B. komplexe Zahlen oder quadratische Matrizen, zu definieren. Schreiben Sie eine Template-Funktion `exp`, welche eine Approximation der Exponentialreihe für beliebige Datentypen zurückliefert. Dabei sei `T` der Templateparameter und `N` eine natürliche Zahl. Die Funktion soll also den Wert der Reihe

$$\sum_{j=0}^{\mathtt{N}} \frac{\mathtt{t}^j}{j!} \approx \exp(\mathtt{t})$$

zurückliefern, wobei `t` eine Instanz von `T` ist. Der Rückgabewert ist dabei wieder vom Typ `T`. Achten Sie auch darauf den Term $\frac{\mathtt{t}^j}{j!} = \frac{\mathtt{t}*\mathtt{t}^{j-1}}{j(j-1)!}$ geeignet und rechenökonomisch zu realisieren.
Hinweis: Sie können davon ausgehen dass für den Typ `T` sowohl die innere Addition und Multiplikation als auch die Multiplikation mit reellen Zahlen definiert sind (nicht aber die Division mit reellen Zahlen). Den Wert $\mathtt{t}^0$ können Sie mit `T sum = 1` erzeugen.