Institut für Numerische und Angewandte Mathematik
FB Mathematik und Informatik der Universität Münster
Prof. Dr. Christian Engwer, Dipl. Math. Dipl. Inf. Sebastian Westerheide

11.04.2012

Übung zur Vorlesung

# Wissenschaftliches Rechnen

SS 2012 — Blatt 0

**Abgabe:** Keine Abgabe (Anwesenheitsaufgaben)

**Aufgabe 1** (Wertebereich von Basisdatentypen) (0 Punkte)

Schreiben Sie ein Programm, welches eine positive ganze Zahl einliest und in einem `short int` speichert. Addieren Sie anschließend 1 auf die Zahl und geben Sie das Ergebnis aus.

(a) Was passiert, wenn Sie 32.767 eingeben?

(b) Was passiert, wenn Sie 65.535 eingeben?

(c) Was passiert, wenn Sie statt eines `short int` einen `unsigned short int` verwenden und die Eingaben wiederholen?

(d) Was passiert, wenn Sie `long int` verwenden und die Eingaben wiederholen?

(e) Was passiert, wenn Sie stattdessen $-1$ addieren und Sie 0 eingeben?

**Aufgabe 2** (Primzahlen) (0 Punkte)

Schreiben Sie ein Programm das alle Primzahlen bis $n$ sucht und ausgibt, wobei $n$ eine vom Benutzer festzulegende natürliche Zahl ist.

- Überlegen Sie sich einen einfachen Algorithmus um zu testen, ob eine Zahl prim ist
- Führen Sie diesen Test für alle Zahlen bis $n$ durch

**Aufgabe 3** (Fakultät) (0 Punkte)

Schreiben Sie ein Programm, welches die Fakultät $n! = 1 \cdot 2 \cdot \ldots \cdot n$ für alle ganzen Zahlen $0 < n \leq 10$ berechnet und ausgibt.

- Verlagern Sie die Fakultätsberechnung in eine Prozedur `int fakultaet (int n)`
- Das Hauptprogramm soll diese Prozedur für die gewünschten Werte von $n$ aufrufen

Hinweis: Die Prozedur `fakultaet` kann sowohl iterativ als auch rekusiv implementiert werden.

**Aufgabe 4** (Fehlersuche) (0 Punkte)

```
#include <iostream>

// summiert alle Zahlen im Intervall [a,b]
int summieren (int a, int b)
{
  int summe;
  for (int i=a; i<=b; i++)
  {
    int summe = summe + i;
  }
  return 0;
}

int main ()
{
  std::cout << summieren(1,10) << std::endl;
  return 0;
}
```

Das obige Programm soll alle Zahlen von 1 bis 10 aufsummieren. Obwohl es syntaktisch korrekt ist, rechnet es falsch. Der Code enthält drei Fehler. Finden und reparieren Sie diese.

**Aufgabe 5** (Fehlersuche 2) (0 Punkte)

```
#include <iostream>

int main()
{
  std::cout << "Flaeche: " << flaeche(6.5, 5.5) << std::endl;
  return 0;
}

double flaeche (double laenge, double hoehe)
{
  double ergebnis;
  ergebnis = laenge * hoehe;
  return ergebnis;
}
```

Das obige Programm soll die Fläche eines Rechtecks berechnen, kann jedoch nicht fehlerfrei übersetzt werden. Finden und korrigieren Sie den Fehler.

Institut für Numerische und Angewandte Mathematik
FB Mathematik und Informatik der Universität Münster
Prof. Dr. Christian Engwer, Dipl. Math. Dipl. Inf. Sebastian Westerheide

18.04.2012

Übung zur Vorlesung

# Wissenschaftliches Rechnen

SS 2012 — Blatt 0

**Abgabe:** Keine Abgabe (Anwesenheitsaufgaben)

**Aufgabe 6** (Klassen und Objekte: Quadratische Funktionen) (0 Punkte)

Univariate quadratische Funktionen lassen sich allgemein durch folgende Normalform beschreiben:

$$f(x) = ax^2 + bx + c \quad \text{mit} \quad a, b, c, x \in \mathbb{R}.$$

(a) Schreiben Sie eine Klasse `QuadF` zur Repräsentation solcher quadratischer Funktionen. Dem Konstruktor der Klasse sollen dabei die drei Koeffizienten $a$, $b$ und $c$ als Parameter übergeben werden können, welche dann in Attributen eines von Ihnen sinnvoll zu wählenden Datentyps gespeichert werden. Zusätzlich zu dem Konstruktor soll die Klasse die drei Methoden `getA`, `getB` und `getC` zur Verfügung stellen, mit denen die Koeffizienten einer Funktion ausgelesen werden können. Achten Sie darauf, die Attribute der Klasse `QuadF` vor Zugriffen aus fremden Klassen zu schützen.

(b) Ergänzen Sie die Klasse `QuadF` um eine Methode `double evaluate (double x)`, die den Wert der Funktion für den übergebenen Parameter $x$ berechnet und das Ergebnis zurückgibt.

(c) Jede der oben beschriebenen Funktionen (mit $a \neq 0$) besitzt eine Stelle, an der die Funktion einen Extremwert (Minimum oder Maximum) annimmt. Bestimmen Sie allgemein die $x$-Koordinate der Extremstelle einer solchen Funktion und fügen Sie der Klasse `QuadF` eine Methode `double getExtremePos ()` hinzu, welche die $x$-Koordinate der Extremstelle zurückgibt.

(d) Schreiben Sie ein Hauptprogramm `TestQuadF`, dessem Prozedur `main` zwei unterschiedliche Instanzen der Klasse `QuadF` angelegt und die Koeffizienten, die Extremstelle sowie den zugehörigen Extremwert der Funktionen auf der Konsole ausgibt.

**Aufgabe 7** (Abstrakte Klassen, Vererbung, Virtuelle Methoden) (0 Punkte)

(a) Schreiben Sie ein eine abstrakte Klasse `Expression` zur Repräsentation arithmetischer Ausdrücke in C++. Die Klasse soll eine abstrakte Methode `evaluate ()` definieren, die zur Berechnung des Wertes eines arithmetischen Ausdrucks dient und als Ergebnis eine Zahl von Typ double zurückgibt. Außerdem soll die Klasse eine Methode `int compareTo (const Expression& exp2)` enthalten, die den Wert des Ausdrucks mit dem des übergebenen Ausdrucks vergleicht und $-1$, 0 oder 1 zurückgibt, wenn der Wert kleiner, gleich oder größer als der Wert von `exp2` ist.

(b) Implementieren Sie die konkreten Klassen `Constant`, `Sum` und `Product` als Unterklassen der Klasse Expression. Die Klasse `Constant` dient dabei der Darstellung konstanter Zahlen vom Typ `double`, die vom Konstruktor in ein Attribut `value` gespeichert werden. Die Konstruktoren der Klassen `Sum` und `Product` zur Realisierung von Summen bzw. Produkten sollen jeweils zwei Objekte vom Typ `const Expression&` entgegennehmen und in geeigneten, vor Zugriffen von außen gekapselten Attributen speichern. Natürlich ist in allen drei Klassen die Methode `evaluate ()` auf sinnvolle Weise zu implementieren.

(c) Schreiben Sie ein Testprogramm `ExpressionTest`, welches verschiedene arithmetische Ausdrücke erzeugt und sämtliche Methoden der Klassen auf sinnvolle Weise überprüft. Bei den Tests sollen mindestens die folgenden Ausdrücke verwendet werden:

$$((5 * 8) + 2), \quad (3 * (2 + 9)) \quad \text{und} \quad ((2 + 4) * (6 + 1))$$

(d) Jeder arithmetische Ausdruck soll mit Hilfe einer Methode

```
std::string toString ()
```

in einer lesbaren Weise dargestellt werden können. Überlegen Sie sich, wie solch eine Methode sinnvollerweise implementiert werden kann (welche Rolle spielt insbesondere die Klasse `Expression`), implementieren und testen Sie die Methode. Achten sie auf eine korrekte Klammerung der Ausdrücke.

**Aufgabe 8** (Klassen und Objekte: Zahlentripel) (0 Punkte)

Die Klasse `Triple` soll in der Lage sein, drei ganze Zahlen in der beim Anlegen einer Instanz der Klasse festgelegten Reihenfolge in Attributen a, b und c zu speichern.

(a) Realisieren Sie die Klasse `Triple` in C++.

(b) Erweitern Sie die Klasse um eine Methode `bool containsZero ()`, die genau dann `true` zurückgibt, wenn mindestens einer der gespeicherten Werte in einem `Triple`-Objekt null ist.

(c) Implementieren Sie innerhalb der Klasse `Triple` eine Methode `getMin ()`, die den kleinsten der drei in einem Objekt gespeicherten Werte zurückgibt.

(d) Um große Mengen von Zahlentripeln effizienter Verwalten zu können, wird ein Kriterium benötigt, nach dem die Tripel in eine gewisse Ordnung gebracht werden können. Schreiben Sie zu diesem Zweck eine Methode `bool isGreater (const Triple& t2)` die überprüft, ob das Zahlentripel, dessen isGreater-Methode aufgerufen wird, elementweise größer ist als das als Parameter übergebene Tripel.