Dr. U. Köthe *Heidelberg, 30. Januar 2017*

Bonusübungsblatt 13

Alle Punkte dieses Blatts werden als **Bonuspunkte** gewertet. Die Aufgaben ähneln den Klausuraufgaben. Benutzen Sie *keinen* Computer und geben Sie die Lösungen *auf Papier* ab.

Aufgabe 13.1 Effekt einer Funktion [6 Punkte]

Gegeben sei folgender Quellcode:

```
 1:    unsigned int foo(unsigned int a, unsigned int b)
 2:    {
 3:        unsigned int c = 0;
 4:        while(a >= b)
 5:        {
 6:            a = a - b;
 7:            c += 1;
 8:        }
 9:        return c;
10:    }
```

(a) Erstellen Sie eine Tabelle mit den sukzessiven Variablenbelegungen und Schleifenbedingungen für den Aufruf `foo(12, 5)`. Machen Sie dabei kenntlich, wo Sie sich bei der Ausführung gerade befinden (z.B. mit Zeilennummern).

(b) Welche bekannte Funktionalität realisiert `foo()`?

(c) Geben Sie eine mathematische Formel an, die diese Funktionalität auf ganze Zahlen verallgemeinert (das heißt, `a` und `b` dürfen jetzt auch negativ sein).

(d) Implementieren Sie die Version von `foo()` für ganze Zahlen nach Ihrer Formel.

Aufgabe 13.2 Erstellen einer Klasse [6 Punkte]

Gegeben seien folgende Tests für eine 3-dimensionale Punktklasse:

```
int main()
{
    Point3D a(3, 1, 2), b(2, -3, 1);

    assert(a.get(0) == 3 && a.get(1) == 1 && a.get(2) == 2);
    assert(to_string(a) == "[3, 1, 2]");
    assert(dot(a, b) == 5);
    assert(cross(a, b) == Point3D(7, 1, -11));
    assert(a + 1 == Point3D(4, 2, 3));
    assert(2 + a == Point3D(5, 3, 4));
    std::cout << "Alle Tests erfolgreich.\n";
}
```

wobei die üblichen Definitionen für Skalarprodukt und Kreuzprodukt verwendet werden:

$$\mathrm{dot}(\mathbf{a},\mathbf{b}) = \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} \cdot \begin{pmatrix} b_0 \\ b_1 \\ b_2 \end{pmatrix} = a_0b_0 + a_1b_1 + a_2b_2$$

$$\mathrm{cross}(\mathbf{a},\mathbf{b}) = \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} \times \begin{pmatrix} b_0 \\ b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} a_1b_2 - a_2b_1 \\ a_2b_0 - a_0b_2 \\ a_0b_1 - a_1b_0 \end{pmatrix}$$

Implementieren Sie die Klasse `Point3D` so, dass alle Tests durchlaufen.

Aufgabe 13.3 Sortieren [6 Punkte]

Das folgende Programm soll den Vektor `v` so sortieren, dass zuerst alle geraden Zahlen aufsteigend und danach alle ungeraden Zahlen absteigend angeordnet sind.

```
#include <algorithm>

bool order(int a, b)
{
    // IHR CODE HIER
}

void main()
{
    std::vector<int> v = [ 2, 3, 5, 6, 4, 1];

    std::sort(v, order);
}
```

(a) Der gegebene Code enthält fünf Fehler (dass die Funktion `order()` unvollständig ist, zählt dabei nicht). Finden und korrigieren Sie diese Fehler.

(b) Vervollständigen Sie die Funktion `order()`, so dass die geforderte Sortierung realisiert wird.

Aufgabe 13.4 Komplexität [6 Punkte]

Zwei Algorithmen benötigen

$$f_1(n) = 4(n \log_2 n + \sqrt{n}) \qquad \text{bzw.}$$
$$f_2(n) = n + n^2$$

Schritte, um eine Aufgabe der Größe $n \geq 1$ zu erledigen.

(a) Leiten Sie mittels O-Notation vereinfachte Formeln $g_1(n)$ bzw. $g_2(n)$ her (mit Begründung).

(b) Beweisen Sie (z.B. durch Bildung des Grenzwerts $n \to \infty$), welcher Algorithmus asymptotisch für große n schneller ist.

(c) Bestimmen Sie einen Bereich für n, wo sich die Geschwindigkeit der Algorithmen umgekehrt zum asymptotischen Fall verhält. (Tipp: Für bestimmte n kann man die Funktionen $f_1(n)$ und $f_2(n)$ leicht im Kopf ausrechnen.)

Bitte laden Sie Ihre Lösung spätestens bis 6. Februar 2017, 9:00 Uhr in Moodle hoch.