RHEINISCH-
WESTFÄLISCHE
TECHNISCHE
HOCHSCHULE
AACHEN

Lehrstuhl für Hochleistungsrechnen
RWTH Aachen · D-52074 Aachen · GERMANY
http://programmierung.informatik.rwth-aachen.de

Prof. Christian Bischof, Ph.D.
Andre Vehreschild, Jakob T. Valvoda

Übung *Programmierung WS 06/07* – Blatt 3

Die Übungsblätter sollen in **Gruppen von je 2 Studierenden** bearbeitet werden, wobei diese aus der selben Übungsgruppe kommen müssen. Lösungen müssen bis zum **13. November 2006, 17:00 Uhr** in den Kasten Ihrer Übungsgruppe eingeworfen werden. Sie finden die Kästen am Eingang Halifaxstr. des Informatikzentrums (Ahornstr. 55).

Bitte vergessen Sie nicht, die **Nummer Ihrer Übungsgruppe**, sowie die **Namen** und **Matrikelnummern** der Mitglieder Ihrer 2er-Gruppe auf jedes Lösungsblatt Ihrer Abgabe zu schreiben. Bitte heften Sie ihre Blätter zusammen.

Alle erstellten Java-Programme sind sowohl in gedruckter Form abzugeben, als auch per E-Mail an den jeweiligen Tutor zu senden. Vergessen Sie auch bei der elektronischen Abgabe per E-Mail nicht die Angabe der **Nummer Ihrer Übungsgruppe**, sowie die jeweiligen **Namen** und **Matrikelnummern**.

Aufgabe 1 (1 + 1 + 1 + 1 = 4 Punkte)

Werden Operanden verschiedenen Typs in einem Ausdruck miteinander verknüpft, so erfolgt nach bestimmten Konventionen eine implizite Konvertierung des Datentyps. In manchen Fällen ist diese Konvertierung nicht möglich, so dass eine entsprechende explizite Typumwandlung, so genanntes Casting erfolgen muß. Geben sie für die folgenden Zuweisungen an, ob eine implizite Konvertierung möglich ist und welcher Datentyp daraus resultiert. Falls keine implizite Anpassung an den Typ der Deklaration erlaubt ist, die Zuweisung also fehlerhaft ist, so korrigieren Sie den Ausdruck, so weit möglich, durch explizites Casting. Sollte auch ein explizites Casting nicht möglich sein, so ändern Sie gegebenenfalls den Typ der Deklaration. Verwenden Sie jeweils den minimalen Typ, der den Wert ohne Datenverlust darstellen kann.

a) `int var = 1.0 * 'a' + 20f;`

b) `float var = 1.0f / 2.5f == 0;`

c) `int num = (char)2.0 + 'b' | 'c';`

d) `boolean alpha = 'a' < 'b' && !( 'b' < 'c' );`

Aufgabe 2 (4 Punkte)

Erstellen Sie ein Java-Programm, welches ein gerahmtes, ausgefülltes Quadrat auf dem Bildschirm ausgibt. Das Quadrat sollte dabei durch Multiplikationszeichen (`*`), der Rahmen durch Rauten (`#`) dargestellt werden. Die Seitenlänge des Quadrates sowie die Rahmenstärke sind variabel und werden vom Benutzer eingegeben. Das Aussehen des Quadrates wird nur durch diese

beiden Werte bestimmt. Überprüfen Sie, ob valide Werte für Seitenlänge und Rahmenstärke, d.h. beide $\geq$ `0`, eingegeben wurden und ersetzen Sie fehlerhafte Benutzereingaben durch den Wert `0`.

Beispiele:

```
Seite  = 1        Seite  = 5        Seite  = 4
Rahmen = 1        Rahmen = 2        Rahmen = 0

###               #########        ****
#*#               #########        ****
###               ########         ****
                  ########         ****
                  ########
                  ########
                  ########
                  #########
                  #########
```

Wenn für die Seitenlänge `0` gewählt wird, so soll nur der Rahmen gezeichnet werden, sofern dessen Breite nicht `0` ist.

Hinweis: Verwenden Sie für die Eingabe der Werte die Methode `readInt()` der Klasse `AdvancedIO`, welche von der Vorlesungsseite heruntergeladen werden kann. Hiermit sind Zuweisungen der Form `int iteration = AdvancedIO.readInt()` möglich.

Aufgabe 3 (3 + 3 = 6 Punkte)

Der natürliche Logarithmus $\ln(x)$ kann für $0 < x < 2$ mit der Newton-Mercator-Reihe berechnet werden. Diese ist gegeben als

$$\ln(x) = \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k} (x-1)^k$$

a) Das Ergebnis kann approximiert werden, indem die Newton-Mercator-Reihe nach der n-ten Summation abgebrochen wird, d. h.

$$\ln(x) \approx \sum_{k=1}^{n} \frac{(-1)^{k+1}}{k} (x-1)^k = x - \frac{1}{2}(x-1)^2 + \cdots + (-1)^{n+1}\frac{1}{n}(x-1)^n$$

Schreiben Sie ein Programm, welches den natürlichen Logarithmus mit der Newton-Mercator-Reihe approximiert. Das Programm sollte den Benutzer auffordern, den x-Wert und die Anzahl der Summen n einzugeben. Der Benutzer wird solange zur Eingabe von x und n aufgefordert, bis beide Werte innerhalb ihrer Wertebereiche $0 < x < 2$ und $n > 0$ liegen. Anschliessend wird das Ergebnis berechnet und ausgeben.

b) Im angegebenen Wertebereich für x gilt für die Reihenglieder, dass diese mit steigendem k alternierend gegen Null streben. Insbesondere heisst dies, dass jede Iteration dem Betrag nach stetig kleiner werdenden Einfluss auf das Ergebnis hat.

Modifizieren Sie Ihr Programm dermaßen, dass statt der Anzahl an Iterationen n vom Benutzer ein Schwellwert $\varepsilon > 0.0$ (Gültigkeit prüfen) eingegeben wird, der bei Unterschreitung zum Ende der Berechnung führt, d.h. die Berechnung wird beendet, falls in

der k-ten Iteration gilt

$$\left| \frac{(-1)^{k+1}}{k} (x-1)^k \right| < \varepsilon$$

Hinweis: Verwenden Sie für die Eingabe der Werte die Methoden `readDouble()` und `readInt()` der Klasse `AdvancedIO` (s. auch Hinweis zur Aufgabe 1). Analog zu `readInt()` erlaubt die Methode `readDoulbe()` Zuweisungen der Form `double argument = AdvancedIO.readDouble()`.

Aufgabe 4 (4 Punkte)

In der letzten Übung sollte ein Programm entwickelt werden, welches das Volumen einer Pyramide berechnet. Das Volumen weiterer geometrischer Primitive wird mit den folgenden Formeln ermittelt

$$V_{\text{Kegel}} = \frac{1}{3} \pi r^2 h$$
$$V_{\text{Kugel}} = \frac{4}{3} \pi r^3$$
$$V_{\text{Pyramide}} = \frac{1}{3} lbh$$
$$V_{\text{Quader}} = lbh$$

wobei r den Radius, l die Länge, b die Breite und h die Höhe des Primitivs bezeichnen.

Enwickeln Sie ein Programm, welches das Volumen für die geometrischen Körper *Kegel, Kugel, Pyramide* und *Quader* berechnet. Beim Start des Programms soll der Benutzer zur Auswahl des Körpers aufgefordert werden. Danach werden die entsprechend benötigten Parameter abgefragt. Verwenden Sie zur Kodierung der geometrischen Körper einen Integer und ermöglichen Sie dem Benutzer die Auswahl durch ein Menü der Form

```
Bitte waehlen Sie den geometrischen Koerper aus
  1. Kegel
  2. Kugel
  3. Pyramide
  4. Quader
Geben Sie nun die entsprechende Zahl ein:
```

Verwenden Sie für die Verzweigungen in Ihrem Programm ausschliesslich die `switch`-Anweisung. Versuchen Sie hierbei Gemeinsamkeiten in den Formeln auszunutzen. Überprüfen Sie, ob die Auswahl gültig ist, und ob es sich bei den Parametern selbst um nicht negative Fließkommawerte handelt (hierzu dürfen Sie `if` verwenden). Sollte eine dieser Bedingungen nicht zutreffen, so soll sich Ihr Programm an dieser Stelle mit einer entsprechenden Fehlermeldung beenden.

Hinweis: Die Klasse `Math` stellt die Kreiszahl π als Konstante bereit. Diese Konstante kann als `Math.PI` in Ausdrücken verwendet werden. Ein Programm kann mit der Anweisung `System.exit( 0 );` abgebrochen werden.