Universität Karlsruhe (TH)
Forschungsuniversität • gegründet 1825

Institut für Angewandte und Numerische Mathematik

Programmieren: Einstieg in die Informatik mit Java

WS 2006/2007

Dr. G. Bohlender
Dipl.–Math. techn. M. Richter

13.11.2006

Aufgabenblatt 3

Bearbeitungszeitraum: 16.11.2006 – 29.11.2006

Aufgabe 7 (Pflichtaufgabe): *Binomialkoeffizienten*

Für zwei natürliche Zahlen n und k ist der Binomialkoeffizient $\binom{n}{k}$ (lies: n über k) wie folgt definiert:

$$\binom{n}{k} := \begin{cases} \prod_{i=1}^{k} \frac{n-i+1}{i} & \text{falls } 0 \leq k \leq n \\ 0 & \text{falls } 0 \leq n < k \end{cases}$$

Dabei ist der Produktoperator $\prod$ analog zum Summenoperator $\sum$ definiert, d.h. für eine Funktion $i \mapsto f(i)$ ist $\prod_{i=1}^{k} f(i) = f(1) \cdot f(2) \cdot f(3) \cdot \cdots \cdot f(k)$. Zusätzlich definiert man $\prod_{i=1}^{0} f(i) := 1$.

(a) Erstellen Sie ein Java-Programm mit dem Namen `Binomialkoeffizient`, welches n und k als `int`-Werte von der Konsole einliest.

(b) Berechnen Sie anschließend den Wert des Binomialkoeffizienten $\binom{n}{k}$. Unterscheiden Sie dabei zwischen den beiden Fällen $k \leq n$ und $n < k$ mittels `if-else`-Anweisung. Verwenden Sie zur Berechnung des Produktes eine `for`-Schleife, in der Sie in jedem Schritt zunächst das Zwischenergebnis mit dem Zähler des Bruches multiplizieren und es danach durch den Nenner des Bruches dividieren. Die Zuweisungsoperatoren `*=` und `/=` sind hierbei hilfreich. Geben Sie den berechneten Wert des Binomialkoeffizienten auf dem Bildschirm aus.

(c) Prinzipiell ist es möglich für die `int`-Variablen n und k auch negative, ganze Zahlen einzugeben. Für solche Zahlen ist der Binomialkoeffizient allerdings nicht definiert. Erweitern Sie Ihr Programm dahingehend, dass bei Eingabe eines negativen Wertes für n oder k der Binomialkoeffizient nicht berechnet, sondern stattdessen die Meldung n `über` k `ist nicht definiert` ausgegeben wird.

(d) Für Binomialkoeffizienten gilt die Formel

$$\binom{n}{k} = \binom{n}{n-k}.$$

Da der k-Wert bei der Berechnung eines Binomialkoeffizienten $\binom{n}{k}$ die Anzahl der Multiplikationen bestimmt, kann diese Formel verwendet werden, um Binomialkoeffizienten mit $k > \frac{n}{2}$ effizient zu berechnen. Verändern Sie Ihr Programm so, dass es für $k > \frac{n}{2}$ nicht $\binom{n}{k}$ sondern $\binom{n}{n-k}$ berechnet.

Aufgabe 8 (Pflichtaufgabe): *Taschenrechner*

Erstellen Sie ein Java-Programm namens `Taschenrechner`, das einen einfachen Taschenrechner mit den Operationen Addition, Subtraktion, Multiplikation, Division und Prozentsatz simuliert. Gehen Sie dabei wie folgt vor:

(a) Definieren Sie zwei Variablen `speicher` und `operand` vom Typ `double` sowie eine Variable `operation` vom Typ `char`. Initialisieren Sie die Variable `operation` mit dem Leerzeichen `' '`. Lesen Sie anschließend einen Wert für `speicher` von der Konsole ein.

(b) Der Wert im Speicher soll nun durch verschiedene Operationen verändert werden können. Lesen Sie dazu **ein einzelnes Zeichen** als Wert für `operation` ein, indem Sie den Befehl

```
operation = sc.next().charAt(0);
```

verwenden. Manipulieren Sie anschließend den Wert von `speicher` in Abhängigkeit des eingelesenen Zeichens. Falls eines der Zeichen `'+'` (Plus), `'-'` (Minus), `'*'` (Stern) oder `'/'` (Schrägstrich) eingelesen wurde, so lesen Sie einen Wert für `operand` ein und verändern Sie den Wert in `speicher`, indem Sie den Wert in `operand` addieren, subtrahieren, multiplizieren bzw. abdividieren. Wurde das Zeichen `'%'` (Prozent) eingelesen, so teilen Sie den Wert in `speicher` durch `100`. Wurde das Zeichen `'='` (Istgleich) eingegeben, so belassen Sie den Wert in `speicher` unverändert. Wurde irgendein anderes Zeichen eingegeben, so belassen Sie den Wert in `speicher` unverändert und geben Sie die Meldung `Unbekannte Operation` aus. **Implementieren Sie die beschriebene Funktionalität unter Verwendung einer `switch`-Anweisung**. Geben Sie anschließend den veränderten Wert in `speicher` auf dem Bildschirm aus.

(c) Erweitern Sie ihr Programm dahingehend, dass der im Aufgabenteil (b) beschriebene Ablauf solange wiederholt wird, wie das in `operation` gespeicherte Zeichen nicht `'='` ist. Verwenden Sie eine `do-while`-Schleife.

Aufgabe 9: *Maschinengenauigkeit*

Zur Speicherung reeller Zahlen werden in einem Rechner entweder 32 Bits (für `float`-Variablen) oder 64 Bits (für `double`-Variablen) verwendet. Der Rechner kann also maximal 2^{32} bzw. 2^{64} verschiedene reelle Zahlen darstellen. Gibt man daher eine reelle Zahl x in einen Rechner ein, so wird diese auf die nächstgelegene Maschinenzahl $\boxed{x}$ gerundet. Den kleinsten Abstand zwischen zwei verschiedenen Maschinenzahlen bezeichnet man dabei als Maschinengenauigkeit ε. Die Maschinengenauigkeit ist abhängig vom verwendeten Datentyp `float` oder `double`. Schreiben Sie ein Java-Programm namens `Maschinengenauigkeit`, welches die Werte von $\varepsilon_{\texttt{float}}$ und $\varepsilon_{\texttt{double}}$ berechnet. Gehen Sie dabei wie folgt vor:

(a) Berechnen Sie zunächst die Maschinengenauigkeit $\varepsilon_{\texttt{double}}$ für den Datentyp `double`. Definieren Sie eine `double`-Variable `eps_double` und initialisieren Sie diese mit dem Wert `1.0`. Dividieren Sie anschließend den Wert in `eps_double` solange durch `2.0`, wie die Bedingung

```
(1.0 + eps_double/2.0) != 1.0
```

erfüllt ist. Verwenden Sie dazu eine `while`-Schleife.

(b) Die `while`-Schleife bricht tatsächlich nach einer bestimmten Anzahl von Iterationen ab. Dies geschieht genau dann, wenn der Wert δ in `eps_double` so klein ist, dass der Wert $x = 1 + \frac{\delta}{2}$ auf die Maschinenzahl $\boxed{x} = 1$ gerundet wird, d.h. wenn $\frac{\delta}{2}$ kleiner als die Hälfte von $\varepsilon_{\texttt{double}}$. Man kann daher $\varepsilon_{\texttt{double}}$ durch δ abschätzen. Geben Sie daher den Wert von `eps_double` als genäherte Maschinengenauigkeit auf dem Bildschirm aus.

(c) Berechnen Sie nun die Maschinengenauigkeit $\varepsilon_{\texttt{float}}$ für den Datentyp `float`. Definieren Sie dazu eine `float`-Variable `eps_float` und Verfahren Sie wie in den Aufgabenteilen (a) und (b). Wichtig hierbei ist, dass alle Rechenoperationen mit `float`-Werten durchgeführt werden müssen! Java konvertiert standardmäßig alle Gleitkommazahlen zu `double`-Werten. Das kann verhindert werden, indem man das Suffix `f` an jede Gleitkommazahl anhängt, d.h. es muss die Bedingung

```
(1.0f + eps_double/2.0f) != 1.0f
```

überprüft werden. Dividieren Sie außerdem die Zahl `eps_float` bei jedem Schleifendurchlauf nicht durch `2.0`, sondern durch `2.0f`. Geben Sie den berechneten Wert für $\varepsilon_{\texttt{float}}$ auf dem Bildschirm aus.