Ada-Kompaktkurs (Wirtschaftsinformatiker), Übungsblatt 5

Claus/N.Weicker, WS 02/03

Bitte beachten Sie die allgemeinen Hinweise am Ende des Blattes!

Aufgabe 1: Quadratzahl (sehr leicht) 3 Punkte

Eine natürliche Zahl m heißt *Quadratzahl*, wenn sie das Quadrat einer anderen Zahl ist, d.h. eine ganze Zahl k existiert, so dass $k^2 = m$ ist. Schreiben Sie ein Programm, welches überprüft, ob eine eingegebene Zahl eine Quadratzahl ist. Ist dies der Fall, soll zusätzlich die Wurzel (d.h. k) ausgegeben werden.
Hinweis: Ihre Lösung muss nicht sehr effizient sein – Hauptsache, sie erfüllt die Aufgabe.

Aufgabe 2: Anagramme (leicht) 5+4 Punkte

Eine Zeichenkette t nennt man ein *Anagramm* einer Zeichenkette s, wenn man t erhalten kann, indem man die Zeichen von s in einer neuen Reihenfolge anordnet. Zum Beispiel ist `einbrecher` ein Anagramm von `bereichern` (und natürlich umgekehrt) und `internet` eines von `renitent`.
Man kann diesen Sachverhalt auch mathematisch präzise ausdrücken: Haben s und t die gleiche Länge n und sind von der Form $s = s_1 \dots s_n$ und $t = t_1 \dots t_n$, so sind s und t genau dann Anagramme voneinander, wenn es eine bijektive Funktion $f\colon \{1, \dots, n\} \to \{1, \dots, n\}$ (d.h. eine Permutation) gibt, so dass $s_i = t_{f(i)}$ gilt für alle $1 \le i \le n$.

1. Schreiben Sie ein Programm, welches von zwei eingegebenen Zeichenketten (beliebiger Länge) überprüft, ob sie Anagramme voneinander sind.
2. Erweitern Sie das Programm aus (a) wie folgt: Wenn es sich bei den Zeichenketten um Anagramme handelt, soll zusätzlich die entsprechende Permutation ausgegeben werden.

Hinweis: Falls Sie Aufgabe (b) lösen, brauchen Sie für (a) nichts abzugeben. Zum Einlesen von Zeichenketten siehe Aufgabe 3.3.

Aufgabe 3: Binomialkoeffizienten (mittel) 4+4 Punkte

In der Kombinatorik bezeichnet der *Binomialkoeffizient* $\binom{n}{k}$ (sprich: "n über k" oder "k aus n"), wobei k und n natürliche Zahlen sind, die Anzahl von Möglichkeiten, die es gibt, aus n verschiedenen Kugeln genau k Kugeln auszuwählen, wobei keine Kugel mehrmals ausgewählt wird. Zum Beispiel ist die Zahl $\binom{49}{6}$ die Anzahl der Möglichkeiten, die es gibt, sechs Kugeln aus 49 Kugeln auf diese Weise auszuwählen, d.h. die Anzahl der verschiedenen Möglichkeiten für den Hauptgewinn beim Lotto.
Mathematisch ist die Zahl wie folgt definiert, wobei das Ausrufezeichen für die Fakultätsfunktion steht:

$$\binom{n}{k} = \frac{n!}{k! \cdot (n-k)!} \quad \text{für } 0 \le k \le n \qquad \text{und} \qquad \binom{n}{k} = 0 \quad \text{für } 0 \le n < k$$

1. Schreiben Sie ein Programm, welches zwei natürliche Zahlen n und k vom Benutzer einliest und die Zahl $\binom{n}{k}$ ausgibt. Benutzen Sie hierbei ausschließlich den Datentyp `Natural`.
2. Sie werden feststellen, dass $\binom{n}{k}$ sehr schnell recht groß werden kann. Der Datentyp `Natural` kann in den vorliegenden Versionen von Ada nur Werte zwischen 0 und $2^{31} - 1$ annehmen.[1] Eine direkte Berechnung der obigen Formel ist daher nicht der beste Weg, um Binomialkoeffizienten zu erhalten, weil dabei Zwischenergebnisse entstehen, die größer als das Endergebnis sind. Für manche Werte von n und k schlägt dann die Berechnung fehl, weil die Zwischenergebnisse zu groß für den Datentyp `Natural` sind, obgleich sich das Endergebnis noch damit darstellen ließe.

   Implementieren Sie eine bessere Berechnungsmethode für Binomialkoeffizienten, mit der man möglichst große Ergebnisse berechnen kann (z.B. durch frühzeitiges Kürzen). Als Erfolgskriterium sei festgelegt, dass das Ergebnis von $\binom{49}{6} = 13983816$ korrekt berechnet wird.

[1] Letzteren Wert erhalten Sie in Ada mit dem Ausdruck `Natural'Last`.

Hinweis: Wenn Ihr Programm Teilaufgabe (b) erfüllt, müssen Sie nur das Programm für (b) abgeben. Testen Sie Ihr Programm für eine Reihe von Eingaben, mindestens aber $\binom{4}{0}$, $\binom{4}{2}$, $\binom{4}{4}$, $\binom{4}{5}$, $\binom{50}{3}$, $\binom{50}{5}$, $\binom{50}{6}$.

Aufgabe 4: Game of Life (Zusatzaufgabe, mittel) 10 Punkte

Bei dem von dem Mathematiker Conway erfundenen 'Game of Life' handelt es sich nicht um ein Spiel im klassischen Sinne, sondern um eine Simulation. Die Simulation spielt auf einer Matrix von $m \times n$ 'Zellen'; jede Zelle ist entweder 'bewohnt' oder 'unbewohnt'. Ausgehend von einem Anfangszustand ändert sich der Zustand der Zellen von Generation zu Generation, und zwar nach den folgenden Regeln:

- Ist eine Zelle in einer Generation bewohnt und hat nur eine oder gar keine bewohnte Nachbarzelle, so ist sie in der nächsten Generation unbewohnt (man sagt: sie stirbt an Einsamkeit).
- Ist eine Zelle in einer Generation bewohnt und hat mehr als drei bewohnte Nachbarzellen, so ist sie in der nächsten Generation ebenfalls unbewohnt (sie stirbt an Überbevölkerung).
- Ist eine Zelle in einer Generation unbewohnt und hat genau drei bewohnte Nachbarn, so ist sie in der nächsten Generation bewohnt.
- In allen anderen Fällen ändert sich der Zustand einer Zelle von einer Generation zur anderen nicht.

Als Nachbarn einer Zelle an Position (x, y) zählen genau die acht Zellen, deren Positionen sich in keiner Dimension um mehr als 1 von x bzw. y unterscheiden (außer der Zelle selbst). Zellen am Rand der Matrix haben nur fünf Nachbarn, und Zellen in Ecken drei.
Schreiben Sie ein Programm, welches diese Simulation implementiert. Die Benutzer sollen m und n sowie einen reellen Schwellenwert p zwischen 0 und 1 angeben, wobei letzterer zur Erzeugung des Anfangszustandes benutzt wird (siehe unten). Sodann soll das Programm eine Generation nach der anderen am Bildschirm ausgeben, wobei Sie die Matrix sinnvollerweise als Tabelle ausgeben sollten, etwa mit Leerzeichen für unbewohnte und Sternchen (`*`) für bewohnte Zellen.
Hinweis 1: Den Anfangszustand erzeugen Sie mit Hilfe des Zufallsgenerators (siehe Hinweis 2 in Aufgabe 4.5) wie folgt: Erzeugen Sie für jede Zelle einen Zufallswert zwischen 0 und 1; liegt dieser unter dem Schwellenwert p, sei die Zelle bewohnt, sonst unbewohnt.
Hinweis 2: Ohne weiteres Zutun liefert der Zufallsgenerator immer dieselbe 'zufällige' Zahlenfolge. Wenn Sie bei jedem Ablauf unterschiedliche Folgen haben möchten, beginnen Sie Ihr Programm mit dem Befehl `Reset(seed)`, sofern `seed` der Name ihrer Zufallsgenerator-Variable ist.

Hinweise

- Programme sind in Ada95 zu schreiben und müssen fehlerfrei übersetzbar sein!
- Schreiben Sie Ihre Programme so, dass ihre Wirkungsweise möglichst leicht nachvollzogen werden kann (insbesondere vom Korrektor). Dazu gehören:
  - die Verwendung sinnvoller Variablennamen (oder zumindest die Erklärung, was der Zweck jeder Variablen ist, falls dies aus dem Namen nicht hervorgeht);
  - Kommentare dort, wo der Zweck von Anweisungen nicht unmittelbar offensichtlich ist, insbesondere, wenn mathematische Überlegungen eine Rolle spielen;
  - ein übersichtlicher Programmierstil, beispielsweise sollten die Inhalte von Schleifen oder die Zweige von if-Anweisungen eingerückt werden, damit die Struktur des Programms offenbar wird;
  - möglichst nur 80 Zeichen pro Zeile; wenn Sie längere Zeilen umbrechen, ist das Programm leichter zu lesen.
- Pro Aufgabenblatt werden maximal 20 Punkte auf den Übungsschein angerechnet.
- Falls Sie Fragen irgendwelcher Art haben, sei es zu den Aufgaben, zum Ablauf, oder falls Sie mit dem Stoff Probleme haben, wenden Sie sich bitte an Ihren Tutor oder an die Übungsleitung: `Nicole.Weicker@informatik.uni-stuttgart.de`, Raum 1.101, oder Tel. 7816-412