Universität Bonn
Institut für Informatik III
Jun.-Prof. Dr. Janis Voigtländer
Dipl.-Math. Daniel Seidel[1]

31.10.2011

Lösungseinreichung bis
06.11.2011, 24:00 Uhr

# Übungen
# Algorithmisches Denken und imperative Programmierung
# WS 11/12

## Blatt 3

Mit diesem Übungsblatt werden Sie zum ersten Mal (zumindest innerhalb der Lehrveranstaltung) programmieren. Die Aufgaben sind innerhalb Ihrer Kleingruppe zu bearbeiten und nur **einmal pro Kleingruppe** abzugeben. Dies kann über den eCampus-Account eines beliebigen Mitgliedes Ihrer Kleingruppe geschehen.
Obwohl nur eine Lösung eingereicht wird, sollte **jedes Gruppenmitglied die Lösung verstanden haben und erklären können**. In der Kleingruppe wird Ihr Tutor sich Ihre Einreichung erklären lassen — und je nachdem, wie gut Sie Ihre Lösung erklären, erhalten Sie entsprechend viele Punkte. Das heißt, werden Sie in der Übung von Ihrem Tutor aufgefordert, die eingereichte Lösung zu erklären und können dies nicht, so erhalten Sie auch keine Punkte, selbst wenn die Lösung richtig ist. Wer die Aufgabe erklären kann, oder nicht gefragt wird (aber anwesend ist), erhält natürlich die mit der Einreichung möglichen Punkte.
Reichen Sie Ihre Lösung, sofern zumindest eines Ihrer Kleingruppenmitglieder angemeldet ist, über eCampus (ILIAS) ein. Wie Sie bei eCampus einreichen, wurde schon im 1. Übungsblatt beschrieben. **Anderweitige Einreichungen werden bei diesem, und allen folgenden, Übungsblättern nicht akzeptiert.** Wenn Sie Ihre Uni-ID jetzt wissen, aber noch nicht gemeldet haben, teilen Sie diese bitte umgehend mit (Email an ds@iai.uni-bonn.de).
Achten Sie bei der Abgabe auf die Deadline. Einreichungen nach der Deadline werden nicht mehr akzeptiert. Achten Sie auch auf die Dateiformate Ihrer Abgaben. Erlaubt sind nur **pdf, Quelltextdateien (Endung: .c) und zip-Archive mit mehreren solchen Dateien**. **Zu entwickelnde Programme sind als einzelne Quelltextdateien mit der Endung .c einzureichen**. Für weitere Hinweise schauen Sie bitte auf dem 1. Übungsblatt nach.

### Warnung!

Ihre Lösung **muss**, um als korrekt gewertet zu werden, auf den Rechnern im Pool **mit `gcc` (ohne irgendwelche speziellen Parameterangaben) kompilierbar sein**. Das heißt insbesondere, dass hier auch **kein C++** kompiliert wird (denn das würde `g++` machen). Testen Sie daher, um sicher zu gehen, Ihre Abgabe mit einem `gcc`. Sollten Sie keinen installiert haben, dann erfahren Sie im 0. Übungsblatt, wie Sie dies ändern können.

[1] Bei Fragen wenden Sie sich bitte via E-Mail an Daniel Seidel (ds@iai.uni-bonn.de).

**Aufgabe 8** (C-Programm zur Kaninchenpopulation, [**7+2P**])**.**
Sie haben in der 1. Übung (Aufgabe 6) die Entwicklung einer Kaninchenpopulation als Folge dargestellt. Dabei starben Kaninchenpaare im 4. Monat ihres Lebens und warfen in diesem Monat auch kein neues Kaninchenpaar. Wie bei der ursprünglichen Fibonacci-Folge aus der Vorlesung (1. Vorlesung ADIP WiSe 2011/12, Folie 18) bekommen auch einjährige Kaninchen noch keinen Nachwuchs. Nun soll es um die computergestützte Berechnung der Kaninchenpopulation nach $n$ ($n \in \mathbb{N}_0$) Monaten gehen.

(a) [**5P**] Schreiben Sie ein **effizientes** C-Programm, das als Eingabe einen **int**-Wert $n$ nimmt und als Ausgabe die Anzahl der lebenden Kaninchenpaare nach $n$ Monaten liefert. Wird für $n$ eine negative Zahl eingegeben, so soll das Programm mit der Ausgabe „`Bitte geben Sie nur nichtnegative, ganze Zahlen ein.`“ terminieren.

(b) [**2P**] Berechnen Sie mit Ihrem Programm die Anzahl der lebenden Kaninchenpaare nach 5, 10, 20 und 80 Monaten. Sollte Ihr Programm für eine Berechnung länger als 5 Sekunden benötigen, dann ist es nicht effizient genug. Versuchen Sie dann, einen effizienteren Algorithmus als Programm umzusetzen.

**Achtung: Ihr Ergebnis wird Sie vielleicht überraschen!**

(c) [**2 Extrapunkte**] Wenn Ihr Algorithmus „richtig“ funktioniert hat, haben Sie nach 80 Monaten eine Kaninchenpopulation von $-23081$, von $-1916623401$, oder von $6673311191$ Kaninchenpaaren. Erklären Sie, wie es zu diesen Ergebnissen kommen kann.

Zur Lösung können Sie folgendes Programmgerüst nutzen, das Ihnen mit dem Übungsblatt auch als Quelltextdatei `fibonacci_mit_tod.c` zur Verfügung gestellt wird.

---

```c
#include <stdio.h>

int main () {
  int n = 0;

  // TODO: deklarieren Sie ggfs. weitere Variablen

  /* Programmbeschreibung */

  printf ("Berechnung der Kaninchenpopulation nach n Monaten\n");
  printf ("================================================\n\n");
  printf ("Die Population entwickelt sich wie folgt:\n");
  printf (" - Am Anfang (0. Monat) existiert ein neugeborenes ");
  printf ("Kaninchenpaar.\n");
  printf (" - Kaninchenpaare werfen im 2. und 3. Monat ihres Lebens ");
  printf ("je ein Paar Junge.\n");
  printf (" - Kaninchenpaare sterben nach 4 Monaten.\n\n");

  /* Eingabe */

  printf ("Nach dem wievielten Monat moechten Sie die ");
  printf ("Kaninchenpopulation wissen?\n");
  printf ("  Monat = ");
  scanf  ("%d", &n);
  printf ("\n\n");
```

```
  ////////////////////////////////////////////////////
  //                                                //
  // TODO: Fuegen Sie hier Ihren Code zum Test der  //
  //       Eingabe und zur Berechnung der           //
  //       Kaninchenpopulation ein.                 //
  //       Wenn Sie den unten angegebenen Code zur  //
  //       Ausgabe unveraendert benutzen moechten,  //
  //       muss Ihr Ergebnis zum Schluss in k liegen.//
  //                                                //
  ////////////////////////////////////////////////////

  printf ("Nach %d Monaten leben %d Kaninchenpaare.\n\n", n, k);
  return 0;
}
```

---

**Aufgabe 9** (Euklidischer Algorithmus, [**8P**])**.**

*„Wenn CD aber AB nicht misst, und man nimmt bei AB, CD abwechselnd immer das kleinere vom größeren weg, dann muss (schließlich) eine Zahl übrig bleiben, die die vorangehende misst.“*

Euklid, Die Elemente, Herausgegeben von Clemens Thaer, Wissenschaftliche Buchgesellschaft Darmstadt, VII Buch, §2

Der bekannteste Algorithmus zur Berechnung des größten gemeinsamen Teilers (ggT) zweier natürlicher Zahlen ist der Algorithmus von Euklid. Das obige Zitat beschreibt ihn informell in einer etwas anderen Anwendungssituation. Die moderne Fassung des Algorithmus ersetzt Folgen von Subtraktionen durch eine einzige Modulo-Operation (Bestimmung des Restes beim Teilen einer natürlichen Zahl durch eine andere).
Die moderne Fassung des euklidischen Algorithmus ließe sich z.B. wie folgt formulieren: Seien zwei nichtnegative, ganze Zahlen $a$ und $b$ gegeben, sowie $c$ als Hilfsvariable zur Verfügung.

(1) Tue:

  (1.1) Setze $c$ gleich $a$ modulo $b$

  (1.2) Setze $a$ gleich $b$

  (1.3) Setze $b$ gleich $c$

  solange $b$ ungleich 0.

(2) Gebe $a$ aus.

(a) [**7P**] Drücken Sie den euklidischen Algorithmus als C-Programm aus.

(b) [**1P**] Berechnen Sie mit Ihrem C-Programm den ggT von 444 und 228.

**Aufgabe 10** (Summe aller eingegebenen Zahlen, [**5P**])**.**
Schreiben Sie ein C-Programm, das so lange natürliche Zahlen ($\mathbb{N}_0$) einliest und diese aufaddiert, bis eine negative ganze Zahl gelesen wird. Sind alle Zahlen eingelesen, soll das Programm die Summe der Zahlen (ohne die negative zum Schluss) ausgeben und dann erfolgreich terminieren.

**Aufgabe 11** (Worte finden, [**4 Extrapunkte**]).
Diese Aufgabe wird Ihnen online durch eine Anwendung namens „Autotool“ gestellt. Sie erhalten Mittwoch oder Donnerstag (2. oder 3.11.2011) eine Email von no-reply-autotool@iai.uni-bonn.de. In dieser Email sind Ihr Login-Name und Ihr Passwort für den Zugang zu Autotool, sowie ein Link zur Login-Seite angegeben.
Haben Sie sich erfolgreich bei Autotool eingeloggt, dann erscheint unter dem Punkt *„Vorlesung“* ein Button mit der Aufschrift *„Algorithmisches Denken und imperative Programmierung“*. Klicken Sie auf diesen Button. Sie können nun auf *„Aufgaben“* klicken und es erscheint (unter Umständen auch schon vorher sichtbar) eine Übersicht mit Aufgaben. Darin befinden sich im Moment die Aufgaben *„Syntaxdiagramm-Übung“* und *„Syntaxdiagramm-Abgabe“*. Beide Aufgaben sind vom Typ her gleich. Bei *„Syntaxdiagramm-Übung“* erhalten Sie mehr Feedback über die Korrektheit Ihrer Lösung als bei *„Syntaxdiagramm-Abgabe“*. Nutzen Sie daher erstere Aufgabe zum Üben. Sie wird nicht bepunktet. Die Aufgabe *„Syntaxdiagramm-Abgabe“* wird bepunktet. Es zählt Ihre letzte dort eingegebene Lösung. Autotool liefert Ihnen bei *„Syntaxdiagramm-Abgabe“* — auch wenn es Anderes behaupten sollte — keine Aussage über die (inhaltliche) Korrektheit Ihrer Einreichung.

Warnung!

Löschen Sie die Email mit Ihren Anmeldedaten nicht, sondern heben Sie sie gut auf. Die Daten können nicht geändert werden, und Sie brauchen Ihren Zugang auch später noch für Übungsaufgaben (es sei denn, der Autotool-Server gibt vorher den Geist auf . . . ).

**Hinweis:** Wie viele Punkte Sie auf Ihre Autotool-Einreichung erhalten haben, erfahren Sie von Ihrem Tutor. Bitte ignorieren Sie Punkteangaben in Autotool.

Viel Erfolg!