Fakultät IV - Elektrotechnik und Informatik

Einführung in die Programmierung WS 2017/18

Feldmann / Foucard / Lichtblau / Streibelt / Krüger / Enghardt / Palmer

Aufgabenblatt 4

letzte Aktualisierung: 23. Oktober, 08:50 Uhr

(47167f2625ea1af4e9252f19263fdd337daaa0af)

Ausgabe: *Freitag, 20.10.2017*
Abgabe: *Dienstag, 24.10.2017, 21:59*

Thema: *Rekursion*

Abgabemodalitäten

1. Die Aufgaben des C-Kurses bauen aufeinander auf. Versuche daher bitte Deine Lösung noch am gleichen Tag zu bearbeiten und abzugeben.
2. Alle abzugebenden Quelltexte müssen ohne Warnungen und Fehler auf den Rechnern des tubIT/eecsIT mittels `gcc -std=c99 -Wall` kompilieren.
3. Die Abgabe erfolgt ausschließlich über unser SVN im Abgaben-Ordner. Nur wenn ein Test in Osiris angezeigt wird ist sichergestellt, dass die Abgabe erfolgt ist.
4. Du kannst bis zur Abgabefrist beliebig oft neue Versionen abgeben.
5. Die Abgabe erfolgt in folgendem Unterordner:
   `ckurs-wise1718/Studierende/<L>/<tubIT-Login>/Abgaben/Blatt0<X>`
   wobei `<L>` durch den ersten Buchstabe des TUBIT-Logins und `<X>` durch die Nummer des Aufgabenblattes zu ersetzen sind. Die Ordner werden automatisch angelegt sobald die Abgabe freigeschaltet wird.
6. Benutze für alle Abgaben soweit nicht anders angegeben das folgende Namensschema: `ckurs_blatt0<X>_aufgabe0<Y>.c` wobei `<X>` und `<Y>` entsprechend zu ersetzen sind. Gebe für jede Unteraufgabe genau eine Quellcodedatei ab.
7. Du darfst den Abgabeordner für das Blatt nicht selbst erstellen, das machen wir jeden Morgen kurz nach 8 Uhr!
8. Du musst aber den Befehl `svn up` auf der obersten Verzeichnisebene des Repositories (also in `ckurs-wise1718`) ausführen um alle Änderungen vom Server abzuholen.
9. Im Abgaben-Ordner gelten einige restriktive Regeln. Dort ist nur das Einchecken von Dateien mit den in der Aufgabe vorgegebenen Namen erlaubt, ausserdem werden die Abgabefristen vom Server überwacht. Beachte eventuelle Fehlermeldungen beim SVN-Commit. Lade nur Dateien hoch, die Du selbst bearbeiten sollst, insbesondere also keine Vorgaben.
10. Es gibt einen Ordner 'Arbeitsverzeichnis', in dem Du Dateien für Dich ablegen kannst.
11. Die Ergebnisse der automatischen Tests kannst Du auf OSIRIS einsehen:
    https://teaching.inet.tu-berlin.de/services/osiris-wise1718/

Die Fibonacci-Folge

Vielleicht kennst du die Fibonacci-Folge schon aus der Schule. Ansonsten empfehle ich dir den Eintrag auf Wikipedia[a] einmal anzuschauen. Jede Zahl dieser Folge ist als Summe der beiden vorhergehenden Zahlen definiert. Nur die ersten beiden Fibonaccizahlen sind fix auf 1 definiert. Die ersten fünf Fibonacci Zahlen sind dementsprechend: $1, 1, 2, 3, 5$. Die darauf folgende Zahl wäre $3 + 5$ also 8. Diese Folge hat kein Ende, sondern läuft unendlich weiter. Mathematisch kann man diese Reihe wie folgt ausgedrückt werden:

$$\text{fib}(x) = \begin{cases} x = 1: & 1 \\ x = 2: & 1 \\ x > 2: & \text{fib}(x-1) + \text{fib}(x-2) \end{cases}$$

Die Fibonacci-Folge kann mithilfe von Schleifen (also iterativ) gelöst werden. Ein Codebeispiel folgt.

```
int n = 10;
int zahla = 1;
int zahlb = 1;
int temp;

printf("Fib(1) = 1\n");
printf("Fib(2) = 1\n");
for (int i=3; i <=n; i++) {
    temp = zahla + zahlb;
    printf("Fib(%d) = %d + %d = %d\n", i, zahla, zahlb, temp);
    zahlb = zahla;
    zahla = temp;
}
```

Um den Code auszuführen muss der Code erst in eine Main Funktion kopiert werden.

[a] siehe https://de.wikipedia.org/wiki/Fibonacci-Folge

1. Aufgabe: Fibonacci (2 Punkte)

Schreibe ein Programm, das rekursiv die n'te Fibonacci-Zahl der Fibonacci-Folge ausgibt. Der Nutzer gibt über das Terminal die Zahl n ein und das Programm gibt dann die n'te Fibonaccizahl zurück. Ein beispielhafter Aufruf inklusive der Ausgabe wird in Listing 1 gezeigt.

Listing 1: Programmbeispiel

```
> gcc -std=c99 -Wall ckurs_blatt04_aufgabe01.c input.c
      -o ckurs_blatt04_aufgabe01
> ./ckurs_blatt04_aufgabe01
Bitte geben sie eine Nummer ein: 4
Fib(4) = 3

> ./ckurs_blatt04_aufgabe01
Bitte geben sie eine Nummer ein: 6
Fib(6) = 8
```

Wie Du sehen kannst kompilieren wir in Listing 1 die Aufgabe mit der Datei `input.c`. Wir benutzen `input.c`

als Bibliothek um die Funktion `lese_int()` in unser Programm einzubinden. Diese praktische Funktion ermöglicht es uns den Benutzer während der Ausführung des Programms um die Eingabe einer Zahl zu bitten. Zu diesem Zweck müssen alle drei Dateien im selben Ordner liegen.

Ihr wundert euch vielleicht, woher `C` weiß, dass es diese Funktion gibt? Das kommt daher, dass in beiden Dateien `ckurs_blatt04_aufgabe01.c` und `input.c` eine dritte Datei `input.h` eingebunden ist. Diese "Header"-Datei fungiert als Schnittstelle und definiert die Funktion `lese_int`.

Um die Hausaufgabe zu vereinfachen bitten wir Dich die vorgegebene Programmstruktur zu verwenden (siehe Listing 2). Die Abgabe muss folgenden Kriterien entsprechen:

- Die Zahl wird mithilfe von `lese_int` eingelesen.
- Die zusätzliche Ausgabe beschränkt sich auf eine Zeile.
- Es werden keine Schleifen verwendet.
- Die gesuchte Zahl wird wie in der Vorlage ausgegeben.

Achtung! Da wir den Datentyp **`int`** verwenden können nur die Fibonacci-Zahlen bis einschließlich `Fib(23)` (also 28657) berechnet werden.

Checke die Abgabe im SVN ein, wie unter "Abgabemodalitäten" beschrieben.

Listing 2: Mögliche Programmstruktur

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include "input.h" // Hier binden wir die Bibliothek ein
4
5  // Schreibe hier die Funktion "int fibonacci"
6
7  int main(){
8      int n = lese_int(); // Hier rufen wir die Funktion lese_int auf.
9      int f = fibonacci(n);
10     printf("Fib(%d) = %d\n" , n, f);
11
12     return 0; //Beende das Programm ohne Fehlermeldung
13 }
```