Universität Augsburg
Institut für Informatik

Übung zur Vorlesung Informatik 1 WS 2017/18
Fakultät für Angewandte Informatik
Lehrprofessur für Informatik
Prof. Dr. Lorenz, Marius Brendle, Johannes Metzger, Lev Sorokin 22.11.2017

Übungsblatt 5

Abgabe: 29.11.2017, 12:00 Uhr (Postkasten der Veranstaltung und E-Mail an Tutor)

- Dieses Übungsblatt muss im Team abgegeben werden (Einzelabgaben sind nicht erlaubt).
- Bitte zur Angabe von Namen, Übungsgruppe und Teamnummer das **Deckblatt** verwenden!
- Die **Zeitangaben** geben zur Orientierung an, wie viel Zeit für eine Aufgabe später in der Klausur vorgesehen wäre; gehen Sie davon aus, dass Sie zum jetzigen Zeitpunkt wesentlich länger brauchen und die angegebene Zeit erst nach ausreichender Übung erreichen.

* leichte Aufgabe / ** mittelschwere Aufgabe / *** schwere Aufgabe

Aufgabe 17 * *(Wiederholte Auswahl von Programmoptionen, 23 Minuten)*

Betrachten Sie das folgende Programm:

```c
#include <stdio.h>

void print_uebersicht();

int main() {
        double rate_children = 5.00;
        double rate_adults = 9.50;
        double rate_students = 7.50;
        int status = 1;
        int option;
        while (status) {
                print_uebersicht();
                option = getchar();
                if (getchar() != '\n') {
                        printf("Zuviele Zeichen eingegeben\n");
                        while (getchar() != '\n') {}
                        continue;
                }
                switch (option) {
                case 'a':
                        printf("Preiskategorie Kind: %.2f\n", rate_children);
                        break;
                case 'b':
                        printf("Preiskategorie Erwachsener: %.2f\n", rate_adults);
                        break;
                case 'c':
                        printf("Preiskategorie Student: %.2f\n", rate_students);
                        break;
```

```
                case 'q':
                        status = 0;
                        printf("Auswahl beendet\n");
                        break;
                default:
                        printf("Unbekannte Kategorie\n");
                }
        }
        return 0;
}

void print_uebersicht()
{
        printf("<<<MENU>>>\n");
        printf("Bitte waehlen Sie eine Preiskategorie:\n");
        printf("a Kinder\n");
        printf("b Erwachsene\n");
        printf("c Studenten\n\n");
        printf("q Auswahl beenden\n");
}
```

a) (*, 5 Minuten)
Was leistet das Programm? (3-5 Sätze)

b) (*, 2 Minuten)
Schreiben Sie eine neue Version des Quellcodes, in der Sie für die `while`-Schleife in `main` die Schleifenbedingung `1` verwenden, und modifizieren Sie den restlichen Code, s.d. das Verhalten des Programms gleich bleibt.

c) (*, 2 Minuten)
Schreiben Sie eine neue Version des Quellcodes, in der Sie in `main` die `continue`-Anweisung weglassen, und modifizieren Sie den restlichen Code, s.d. das Verhalten des Programms gleich bleibt.

d) (*, 2 Minuten)
Schreiben Sie eine neue Version des Quellcodes, in der Sie die `while`-Schleife in `main` durch eine `for`-Schleife ersetzen, wobei das Verhalten des Programms natürlich wieder gleich bleiben soll.

e) (*, 2 Minuten)
Schreiben Sie eine neue Version des Quellcodes, in der Sie die `while`-Schleife in `main` durch eine `do-while`-Schleife ersetzen, wobei das Verhalten des Programms natürlich wieder gleich bleiben soll.

f) (*, 10 Minuten)
Verbessern Sie den Quellcode, indem Sie den Code zum Einlesen der Benutzereingabe in eine separate Funktion `int choose_option()` auslagern. Diese Funktion soll für die verschiedenen zur Verfügung stehenden Optionen und die verschiedenen möglichen Fehlerfälle bei der Eingabe (zu viele Zeichen, unbekannte Option) jeweils einen spezifischen Rückgabewert haben.

Bei Aufruf von `choose_option` in `main` soll abhängig vom Rückgabewert die passende Ausgabe auf Kommandozeile erfolgen.

Das Verhalten des Programms soll dabei natürlich wieder gleich bleiben.

Aufgabe 18 ** *(Eingabefunktionen, 16 Minuten)*

In jeder Teilaufgabe sollen Sie jeweils ein compilierbares Hauptprogramm implementieren, das eine bestimmte Benutzereingabe erwartet.

Das Programm soll jeweils

- eine Benutzereingabe einlesen,
- diese auf Gültigkeit überprüfen,
- in geeigneten Variablen speichern und
- schließlich auf Kommandozeile ausgeben, ob die Eingabe gültig war oder nicht.

Hinweise:

- Alle nachfolgenden Aufgaben sind unabhängig und bauen nicht aufeinander auf.
- Puffer-Fehler (EOF) müssen nicht berücksichtigt werden.

Das Einlesen soll dabei jeweils in einer separaten Funktion erfolgen, die bei einer ungültigen Eingabe den Eingabe-Puffer leert und einen Fehlerwert zurückgibt.

a) (*, 8 Minuten)
Schreiben Sie ein Hauptprogramm, das als Benutzereingabe einen Kleinbuchstaben erwartet.

Die Einlesefunktion soll den Prototyp `int read_alpha()` haben, bei gültigen Eingaben den ASCII-Code des eingelesenen Zeichens und bei ungültigen Eingaben den Wert `0` zurückgeben.

b) (**, 8 Minuten)
Schreiben Sie ein Hauptprogramm, das als Benutzereingabe eine Zahl zwischen 0.0 und 1.0 erwartet (inklusive). Die Einlesefunktion soll den Prototyp `int read_probability(double input)` haben, bei gültigen Eingaben `1` zurückgeben, und bei ungültigen Eingaben den Wert `0` zurückgeben.

Aufgabe 19 ** *(Schleifen programmieren, 23 Minuten)*

In den folgenden Aufgaben sollen jeweils eine Funktion selbst implementieren. Es wird diesmal kein Hauptprogramm zum Testen Ihrer Funktion verlangt, Sie sollten sich aber jeweils eines schreiben, damit Sie sicher sind, dass Ihre Abgabe auch stimmt. Außerdem ist das eine gute Übung und führt dazu, dass Sie lernen, korrekte Programme ohne viel Nachzudenken schreiben zu können.
Die Funktionen erwarten jeweils die Übergabe von positiven ganzen Zahlen. Das Verhalten bei anderen Eingaben bleibt undefiniert, d.h. darum müssen Sie sich nicht kümmern.

a) (*, 3 Minuten)
Schreiben Sie eine Funktion `void print_stars(int n)`, die bei Übergabe einer positiven ganzen Zahl n auf der Kommandozeile n neue Zeilen mit einem Stern ausgibt.

b) (**, 5 Minuten)
Schreiben Sie eine Funktion `int mod(int a, int b)`, die bei Übergabe von zwei positiven ganzen Zahl a und b den Rest der ganzzahligen Division von a und b berechnet und zurückgibt, wobei zur Berechnung ausschließlich die Operationen `=,==,<,>,<=,>=,!=,+,-,++,--` benutzt werden.

Beispiel: Ausgabe bei Aufruf `mod(7,2)` ist `1`.

c) (**, 6 Minuten)
Schreiben Sie eine Funktion `void print_sq(int n)`, die bei Übergabe einer ganzen, positiven Zahl n, alle Quadratzahlen zwischen 1 und n zeilenweise ausgibt.
Beispiel: Ausgabe bei Aufruf `print_sq(17)`
1
4
9
16

d) (**, 4 Minuten)
Schreiben Sie eine Funktion `int sumq(int n)`, die bei Übergabe einer positiven ganzen Zahl n die Summe der Quadrate der Zahlen von 1 bis n berechnet und zurückgibt.

Beispiel: Ausgabe bei Aufruf `sum(3)`
`14`

e) (**, 5 Minuten)
Schreiben Sie eine Funktion `void print_triangle(int n)`, die bei Übergabe einer positiven ganzen Zahl n auf Kommandozeile linksseitig ein Dreieck aus `O`-Zeichen mit der Breite und Höhe n ausgibt.

Beispiel: Ausgabe bei Aufruf `print_triangle(4)`
```
OOOO
OOO
OO
O
```

Aufgabe 20 * *(Funktionen für Felder, 22 Minuten)*

In jeder Teilaufgabe sollen Sie zur Verwaltung von Zahlenfolgen eine Funktion implementieren oder einen Algorithmus in Pseudocode angeben.

Es wird jeweils kein Hauptprogramm zum Testen Ihrer Implementierung verlangt, Sie sollten sich aber jeweils eines schreiben, damit Sie sicher sind, dass Ihre Abgabe auch stimmt. Außerdem ist das eine gute Übung und führt dazu, dass Sie lernen, korrekte Programme ohne viel Nachzudenken schreiben zu können.

a) (*, 4 Minuten)
Implementieren Sie eine Funktion mit dem Prototyp `void print_array(int w[], int size, int n)`, die die *letzten* `n` Komponenten des Feldes `w` mit der Größe `size` jeweils durch ein Komma getrennt auf Kommandozeile ausgibt.

b) (*, 5 Minuten)
Geben Sie einen Algorithmus in Pseudocode an, der bei Eingabe einer Zahlenfolge überprüft, ob diese (auf- oder absteigend) sortiert ist, im positiven Fall `1` ausgibt, und sonst `0` ausgibt.

c) (*, 3 Minuten)
Die Vektoraddition auf zwei Vektoren $v = (v_1, \ldots, v_n), w = (w_1, \ldots, w_n) \in \mathbb{Z}^n$ ist definiert als

$$v + w := (v_1 + w_1, \ldots, v_n + w_n)$$

Implementieren Sie eine Funktion mit dem Prototyp `void vsum(int w[], int v[], int n)`, die die Vektoraddition von `v` mit `w` realisiert und dabei die Komponenten von `v` überschreibt.

e) (*, 10 Minuten)
Betrachten Sie den folgenden in Pseudocode angegebenen Algorithmus:

Eingabe : $a_1, \ldots, a_n \in \mathbb{N}$ aufsteigend sortiert, $s \in \mathbb{N}$
$c \leftarrow 0$;
$i \leftarrow 1$;
solange $i \leq n$ **tue**
  wenn $a_i > s$ **dann**
    Ausgabe : c
  sonst
    wenn $a_i = s$ **dann**
      $c \leftarrow c + 1$;
    $i \leftarrow i + 1$;
Ausgabe : c

- Implementieren Sie den Algorithmus als C-Funktion
- Was ist die Ausgabe des Algrithmus für die Eingabe $0, 3, 3, 3, 13, 13, 15$ als sortierte Folge und 13 für s?
- Was leistet der Algorithmus?