Fakultät IV - Elektrotechnik und Informatik
Einführung in die Programmierung WS 2017/18
Feldmann / Foucard / Lichtblau / Streibelt / Krüger / Enghardt / Palmer

Aufgabenblatt 5

letzte Aktualisierung: 23. Oktober, 08:53 Uhr

(47167f2625ea1af4e9252f19263fdd337daaa0af)

Ausgabe: *Montag, 23.10.2017*
Abgabe: *Mittwoch, 25.10.2017, 21:59*

Thema: *Arrays, Call-by-Reference, Call-by-Value*

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/

1. Aufgabe: Array Aggregation (2 Punkte)

In dieser Aufgabe sollst Du Funktionen implementieren, die aus einer Zahlenreihe Werte berechnen. Wie in der Vorgabe zu dieser Aufgabe zu sehen, ist die Zahlenreihe in einem Array gespeichert.

Ausgabe eines Array
Implementiere als erstes eine Funktion, die die Zahlen des Arrays jeweils durch Komma getrennt ausgibt. Nenne sie `print_array`. Verwende eine Schleife um alle Elemente des Arrays zu erreichen. Die Ausgabe erfolgt via `printf` auf der Kommandozeile. (Genauso wie letzte Woche auch.) Pass auf, dass Du nicht versuchst auf mehr Elemente zuzugreifen als im Array sind. Als Parameter erhält diese als erstes Argument das Array und als zweites Argument die Länge des Arrays. Diese Funktion hat keinen Rückgabewert, sondern gibt mittels `printf` eine Zeile aus, die mit `Array:` beginnt.

Min & Max
Implementiere zwei Funktionen `min` und `max`. Die erste soll das kleinste (`min`) und die zweite das größte (`max`) Element des Arrays finden. Die Parameter kannst Du dir von der Funktion `print_array` abschauen, dabei soll die Reihenfolge und Anzahl der Parameter nicht verändert werden. Beide Funktionen benutzen kein `printf`, sondern geben einen **`int`** Wert zurück. Erweitere die Funktion `main`, sodass Minimum und Maximum unter Verwendung deiner Funktionen ausgegeben werden. Beachte, dass die ausgegebenen Zeilen mit `Minimum:` bzw. `Maximum:` beginnen müssen.

Summe
Schreibe eine Funktion `sum`, die die Summe des Arrays berechnet. Dafür wird der Funktion `sum` als erstes Argument das Array, als zweites Argument die Länge des Arrays und als letztes Argument ein Pointer auf einen Integer (Call-by-Reference) mitgegeben. Diese Funktion soll ebenfalls kein `printf` verwenden und hat keinen Rückgabewert. Erweitere anschließend deine `main` Funktion, sodass die Summe unter Verwendung der soeben geschriebenen Funktion ausgegeben wird. Die Ausgabe muss mit `Summe:` beginnen.

Nutze die Funktionen um nacheinander folgendes auszugeben:

1. das Array
2. das Minimum
3. das Maximum
4. die Summe

Ganz wie in Listing 1 gezeigt wird.

Listing 1: Programmbeispiel

```
1 > gcc -std=c99 -Wall ckurs_blatt05_aufgabe01.c -o ckurs_blatt05_aufgabe01
2 > ./ckurs_blatt05_aufgabe01
3 Array: 9, 4, 7, 8, 10, 5, 1, 6, 3, 2
4 Minimum: 1
5 Maximum: 10
6 Summe: 55
```

Um die Hausaufgabe zu vereinfachen bitten wir Dich die vorgegebene Programmstruktur zu verwenden (siehe Listing 2). Der Fall, in dem das Array leer ist, muss nicht behandelt werden. Die Abgabe muss folgenden Kriterien entsprechen:

- Das Array **`int`** `array[];` wird zum Einlesen verwendet.
- Die Länge des Array ist in der Variable **`int`** `len;` gespeichert.

- Die Variablennamen `array` und `len` werden nicht in anderen Funktionen verwendet. (Sonst können wir die Aufgabe nicht testen.)
- Die Reihenfolge und Anzahl der Parameter entspricht der Beschreibung auf diesem Aufgabenblatt.
- Es werden vier Zeilen ausgegeben, wie in Listing 1 dargestellt. Vor den Ergebnissen wird der entsprechende String ausgegeben (”Array:”,“Minimum:”, “Maximum:”, “Summe:”).
- Es werden nur ganze Zahlen ausgegeben.
- Es dürfen keine weiteren Leerzeilen ausgegeben werden.

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

Listing 2: Mögliche Programmstruktur

```c
#include <stdio.h>
#include <stdlib.h>

void print_array(int array[], int len) {
    // HIER Code einfügen
}

// Schreibe die Funktion "sum", "min" und "max"

int main() {
    int array[] = {9, 4, 7, 8, 10, 5, 1, 6, 3, 2};
    int len = 10;
    print_array(array, len);
    // Gebe hier nacheinander das Minimum, Maximum und die Summe
    // aus. Trenne die Werte durch einen einzelnen Zeilenumbruch.
}
```