Institut für Informatik
Prof. Dr. Stefan Conrad

Sommersemester 2021
Übung 1

Hardwarenahe Programmierung
Gruppe 17 (Florian)

In dieser Übung decken wir die Grundlagen der C-Programmierung ab, insbesondere Funktionen, Arrays, Pointer und Unit-Tests.

- *Um die praktische Übung zu bestehen müssen Sie:*
  - *Mindestens 11 von insgesamt 15 Pflichtaufgaben fehlerfrei lösen und abgeben.*
    - *Um besser auf das Abschlussprojekt vorbereitet zu sein, sollten Sie alle Aufgaben und auch die optionalen Vorbereitungsaufgaben lösen.*
  - *Das Abschlussprojekt bestehen.*
  - *Die 6 Tests im ILIAS bestehen.*
- *Hinweise:*
  - *Beginnen Sie frühzeitig mit den Übungsaufgaben! Wenn Sie zu spät anfangen, haben Sie nicht mehr die Möglichkeit, in Ihrer Übungsgruppe Fragen zu stellen.*
  - *Nutzen Sie das Lernmodul und beachten Sie die Code-Vorgaben!*
  - *Beachten Sie die Warnungen Ihres Compilers! In der Regel weisen diese auf schwerwiegende Programmierfehler hin.*
  - *Denken Sie daran, Ihre Variablen zu initialisieren! Nicht initialisierte Variablen können beliebige Werte annehmen, was sehr häufig zu Fehlern führt.*
  - *Wenn ein Test (z.B. `*.ts`- oder `test.sh`-Dateien) nicht fehlerfrei durchläuft, dann gilt die Aufgabe als nicht bestanden.*
  - *Selbst ein Programm, dass die Tests besteht, kann immer noch falsch sein. Wir behalten uns vor, zusätzliche Testfälle zu überprüfen.*
  - *Nutzen Sie `valgrind`, um leichter Speicherzugriffsfehler zu finden. Lesen von nicht initialisierten Variablen führt zum Nichtbestehen der Aufgabe.*
  - *Abschreiben und abschreiben lassen ist verboten. Aufgaben müssen selbstständig bearbeitet werden.*

Aufgabe 1 *Funktionen (Vorbereitungsaufgabe. Vorbereitungsaufgaben sind optional.)*

(a) Erstellen Sie eine Funktion `even(int a)`, die testet, ob a eine gerade Zahl ist. Falls a gerade ist, soll 1 als Rückgabewert zurückgegeben werden, andernfalls 0.

(b) Schreiben Sie ein Programm, das mit Ihrer Funktion aus (a) eine Zahl x testet und dem Benutzer am Bildschirm eine Meldung ausgibt, ob die Zahl gerade oder ungerade ist. Dabei soll der Benutzer nach Programmstart aufgefordert werden, die Zahl x einzugeben. Beispiel:

```
Geben Sie eine Zahl ein:      15
-------------------------------------
Die Zahl ist nicht gerade!
```

Aufgabe 2 *Pointer und Arrays (Vorbereitungsaufgabe)*

(a) Schreiben Sie ein Programm, das 12 Messwerte (jeweils vom Typ `double`) von der Konsole in ein Array einliest und den Inhalt des Arrays anschließend ausgibt.

(b) Falls Sie in Ihrer Lösung eckige Klammern, also `[]`, für die Lese- und Schreibzugriffe auf das Array verwendet haben, schreiben Sie Ihr Programm so um, dass es stattdessen Pointerzugriffe verwendet! Verwenden Sie eckige Klammern hier also **nur** noch zur Deklaration des Arrays.

Aufgabe 3 *Pointer und Funktionen (Vorbereitungsaufgabe)*

Schreiben Sie ein Programm, in dem zwei `double` Zahlen von der Tastatur eingelesen und in lokalen Variablen gespeichert werden. Die Werte der Variablen sollen anschließend in einer Funktion namens `swap` miteinander vertauscht werden. Geben Sie die beiden Variablen vor und nach dem Aufruf von `swap` auf dem Bildschirm aus.

Aufgabe 4 *Unittests (**Pflichtaufgabe. Pflichtaufgaben müssen gelöst werden.**)*

In dieser Aufgabe sollen Sie die Funktionalität von vorgegebenen C-Funktionen mittels des Unit-Test Frameworks *check* überprüfen. Schauen Sie sich dazu zunächst die vorgegebenen Unittests in der Datei `simple_string_tests.ts` zur Implementierung in `simple_string.c` an, welche ihnen zeigen, wie solche Tests aussehen können.

(a) Bauen Sie die Tests und führen Sie sie aus (entsprechend der Anleitung im ILIAS-Lernmodul). Die fehlschlagenden Tests helfen Ihnen, Fehler in der von uns bereitgestellten Implementierung zu erkennen. Sie sollten verstehen, was die Ausgabe der Tests bedeutet.

(b) Korrigieren Sie im Anschluss die Fehler. Die Tests selbst dürfen Sie nicht verändern! Alle Tests müssen nun erfolgreich durchlaufen. Dies heißt allerdings noch nicht, dass Ihr Code auch fehlerfrei ist. Achten Sie deshalb immer darauf, dass Sie ihre Strings terminieren, nicht aus uninitialisierten Speicherbereichen lesen und auch sonst keine Speicherzugriffsfehler begehen.

Folgende C-Funktionen sind gestellt:

- eine Funktion `int zeichen_zahelen(char zeichen[])`, die die Länge des übergebenen Strings zurückgeben soll.
- eine Funktion `void string_anhaengen(char string1[], char string2[])`, die `string2` an `string1` anhängen soll.

Bei dieser Aufgabe dürfen Sie keine eigene `main`-Funktion implementieren, da diese Funktion bereits durch das Check-Framework aus der Datei `simple_string_tests.ts` generiert wird.

Denken Sie daran, dass die Dateien nach jeder Code-Änderung neu kompiliert werden müssen, da die Änderungen sonst keine Wirkung zeigen.

Aufgabe 5 *Pointer und Funktionen (**Pflichtaufgabe**)*

In dieser Aufgabe sollen Sie in der Datei `automat.c` ein Programm implementieren, das einen Brot-Automaten repräsentiert. In der Datei finden Sie einige vorgegebene Funktionsprototypen, die sinnvoll verwendet werden müssen. Mit Ausnahme der `brot_ausgeben`-Funktion sind diese noch nicht implementiert.

Der Automat verfügt zu Beginn über 30 Mischbrote (Sorte 1) und 10 Vollkornbrote (Sorte 2).

Das Programm soll wie folgt ablaufen:

1. Der Automat nimmt als Tastatureingabe zwei Zahlen entgegen. Die erste eingegebene Zahl entspricht der Anzahl der geforderten Brote und die zweite Zahl entspricht der Brotsorte. Der Einfachheit halber darf angenommen werden, dass nur gültige Zahlen eingegeben werden.
2. Nach Eingabe einer Zahl soll der Automat die Anzahl der Brote der angefragten Sorte mit Hilfe der Funktion `brote_kaufen` reduzieren und entsprechend viele Brote auf der Standardausgabe ausgeben. Verwenden Sie zur Ausgabe der einzelnen Brote jeweils die Funktion `brot_ausgeben`. Es werden höchstens so viele Brote ausgegeben, wie sich noch im Automaten befinden.
3. Im Anschluss gibt der Automat die Anzahl der noch verbleibenden Brote im Automaten gefolgt von einem Zeilenumbruch aus. Hierbei muss das Format „`Noch <Anzahl> Brote der Sorte <Sorte> im Automaten.`“ verwendet werden, wobei „`<Anzahl>`“ und „`<Sorte>`“ durch den entsprechenden Zahlenwert zu ersetzen sind.
4. Wenn *alle* Brotsorten ausverkauft sind, soll sich der Automat beenden und die Nachricht „`Der Automat ist leer.`“, gefolgt von einem Zeilenumbruch, ausgeben.

Überprüfen Sie ihr Programm, indem sie die Script-Datei `test.sh` ausführen. Dabei wird dem Programm über die Kommandozeile die Datei `eingabe.txt` über die Standardeingabe zugeführt und eine Ausgabe produziert. Mit dem Programm `diff` wird dann überprüft, ob die Ausgabe mit der erwarteten Ausgabe in der Datei `erwartete_ausgabe.txt` identisch ist. Hierbei sind auch Leerzeichen und Zeilenumbrüche relevant! Eventuell müssen Sie die Test-Datei erst mit dem Befehl `chmod +x test.sh` ausführbar machen.

Bei dieser Aufgabe müssen Sie eine eigene `main`-Funktion implementieren, da Ihr Programm sonst nicht starten kann.