Institut für Informatik
Prof. Dr. Stefan Conrad

Sommersemester 2020
Übung 2

Hardwarenahe Programmierung
Gruppe 14 (Lars)

In dieser Übung liegt der Fokus auf dem Umgang mit Strings in C-Programmen.

Wichtig:

- *Denken Sie daran, Strings zu terminieren und Variablen zu initialisieren!*
- *Denken Sie daran, ihre Abgabe hochzuladen und die Tests im ILIAS zu absolvieren!*

Aufgabe 1 *String Konkatenierung (**Pflichtaufgabe**)*

(a) Schreiben Sie eine Funktion `strings_verbinden`, welche zwei Strings entgegen nimmt, diese konkateniert und in einem Ausgabestring speichert. Implementieren Sie die Funktion in einer Datei mit Namen `strings_verbinden.c`. Wählen Sie eine der beiden folgenden Funktionssignaturen:

entweder `void strings_verbinden(char zusammen[], char string1[], char string2[])`
oder `void strings_verbinden(char *zusammen, char *string1, char *string2)`

Sie dürfen hierbei annehmen, dass das `char`-Array mit Namen `zusammen`, in dem Sie den String speichern, ausreichend groß ist.

Verwenden Sie bei dieser Aufgabe keine Funktionen aus `string.h`! Zum Beispiel ist die Verwendung von `strlen` oder `strcat` hier nicht erlaubt. Bei anderen Aufgaben dürfen weiterhin Funktionen aus `string.h` verwendet werden.

(b) Verwenden Sie die bereitgestellten Tests, um zu demonstrieren, dass Ihre Funktion wie gefordert funktioniert. Verwenden Sie hierzu den Kommandozeilenbefehl `make run`, der die Tests baut, kompiliert und ausführt.

Aufgabe 2 *Caesar-Verschlüsselung (**Pflichtaufgabe**)*

Schreiben Sie ein Programm, das folgende Funktionen enthält. Vereinfachend dürfen Sie in der gesamten Aufgabe davon ausgehen, dass Eingabestrings ausschließlich aus Kleinbuchstaben bestehen.

(a) eine Funktion `void caesar(char *klartext)`, die einen String mit der Caesar-Chiffre codiert: Jeder Buchstabe soll um 2 Stellen im Alphabet nach rechts verschoben werden (so wird z.B. “x” zu “z” und “y” zu “a”). Überlegen Sie sich, wie der C-Datentyp `char` intern gespeichert ist, und wie Sie dies nutzen können, um diese Verschiebung elegant zu lösen.

Input:	a	s	s	e	m	b	l	y	\0
Output:	c	u	u	g	o	d	n	a	\0

(b) eine Funktion `int encode_and_compare(char *encoded_string, char *string_to_encode)`, welche die beiden eingegebenen Strings vergleicht und 1 zurück gibt, falls `encoded_string` gleich der verschlüsselten Fassung von `string_to_encode` ist, und 0 sonst.

(c) Wir haben Ihnen Unit-Tests und ein Makefile zur Verfügung gestellt. Sie können die Tests mit dem Befehl '`make run`' bauen und ausführen. Testen Sie Ihre beiden Funktionen aus (a) und (b) mit den bereitgestellten Unit-Tests und stellen Sie sicher, dass alle erfüllt sind. Sie sollten die Tests nicht verändern! Schauen Sie sich außerdem den Aufbau des Makefiles an, damit Sie in Zukunft selbst Makefiles erstellen können, um Ihren Kompilier- und Testprozess zu vereinfachen.

Aufgabe 3 *Tiere zählen (**Pflichtaufgabe**)*

Fünf Tierarten haben sich in einem String versteckt. In dieser Aufgabe sollen Sie ein Programm schreiben, dass diese Tiere findet, in ein Array schreibt und dabei zählt. Die Tierarten sind:

```
Ente
Gans
Maus
Nasenaffe
Saigaantilope
```

Zum Beispiel enthält der folgende String drei Tiere - `Ente`, `Maus` und nochmal `Ente`:

```
Ente.123 Maus_   456Banane?Ente!
```

- Implementieren Sie die Funktion `tiere_zaehlen`, die die Tiere aus einem `string` in das Array `tiere` schreibt und die Anzahl der gefundenen Tiere zurück gibt.

  ```
  int tiere_zaehlen(char tiere[10][16], char *string);
  ```

- Es darf angenommen werden, dass es höchstens 10 Tiere gibt und Tiernamen oder andere Buchstabenfolgen im String nicht länger als 15 Zeichen sind.
- Tiere verstecken sich nicht zwischen Buchstaben, sondern nur zwischen anderen Zeichen.
- Das Array `tiere` ist nicht initialisiert; es kann also irgendwelche Werte beinhalten.
- Führen Sie die Unit-Tests mit dem Befehl `make run` aus.