4.4.A Datenstrukturen/Strings – Arbeitsblatt

Hinweis: Bei den folgenden Aufgaben soll auf die Verwendung von String-Funktionen aus der C-Standard-Bibliothek verzichtet werden.

Aufgabe 1: Ermitteln der Länge einer Zeichenkette

Gegeben ist die Zeichenkette mit der Vereinbarung:

```
1   char s[41]="Das ist das Haus vom Nikolaus.";
```

Die Länge dieser Zeichenkette soll ermittelt werden.

a) `zklaengesizeof.c`: Ermitteln Sie den Speicherbedarf der Zeichenkette in Byte mit Hilfe des Operators `sizeof`!

b) `zklaengetermpos.c`: Ermitteln Sie die Anzahl ausgegebener Bytes, indem Sie mit einer Schleife die Position (den Index) des Terminators ermitteln und ausgeben!

c) `zklaengestrlen.c`: Ermitteln Sie die Anzahl ausgegebener Bytes, indem Sie mit Hilfe der Funktion `strlen()` aus `<string.h>` ermitteln und ausgeben!

d) `zklaengesizeofohne.c`: Wenn man in der Vereinbarung die Zahl `41` wegläßt, wie groß ist dann der Speicherbedarf der Zeichenkette in Bytes?

Aufgabe 2: Bildschirm-Ausgabe einer Zeichenkette

Gegeben ist die Zeichenkette mit der Vereinbarung:

```
1   char s[41]="Programmierung ist 100% irrsinnig.\n";
```

Diese Zeichenkette soll auf dem Bildschirm ausgegeben werden.

a) `zkausgabe1.c`: Geben Sie die Zeichenkette Zeichen für Zeichen mit `putchar()` aus!

b) `zkausgabe2.c`: Geben Sie die Zeichenkette Zeichen für Zeichen mit `printf()` (Platzhalter: `%c`) aus!

c) `zkausgabe3.c`: Geben Sie die Zeichenkette auf einmal mit `printf()` (ohne Platzhalter) aus!

d) `zkausgabe4.c`: Geben Sie die Zeichenkette auf einmal mit `printf()` (Platzhalter: `%s`) aus!

e) `zkausgabe5.c`: Geben Sie die Zeichenkette Zeichen für Zeichen **in umgekehrter Reihenfolge** mit `putchar()` aus!

Aufgabe 3: Tastatur-Eingabe einer Zeichenkette

Gegeben ist die Zeichenkette mit der Vereinbarung:

```
1   char s[41]="-leer-";
```

Der Benutzer soll in diese Zeichenkette auf der Tastatur einen neuen Inhalt eingeben können. Anschließend soll dieser Inhalt wieder auf den Bildschirm ausgegeben werden:

Terminal

```
schueler@debian964:~$ ./a.out
Ihre Eingabe (max. 40 Zeichen): Carl Severing
Sie haben eingegeben: >>Carl Severing<<
```

Testen Sie Ihre Programme mit normalen Eingaben (ein oder mehrere Wörter), zu langen Eingaben, Leereingaben und Eingabe-Abbrüchen mit Strg + D (unter Wind.: Strg + Z)!

a) `zkeingabe1.c`: Die Eingabe soll durch `scanf()` mit dem Platzhalter `"%40s"` erfolgen.

b) `zkeingabe2.c`: Die Eingabe soll durch `scanf()` mit dem Platzhalter `"%40[^\n]"` erfolgen. Hinweis: Nach der Eingabe muss der Eingabepuffer durch folgende Zeile geleert werden:

```
1  while(getchar()!='\n'){}
```

c) `zkeingabe3.c`: Die Eingabe soll durch `fgets()` erfolgen. Hinweis: Nach der Eingabe muss ggf. der Zeilenumbruch aus der Zeichenkette entfernt werden mit der Zeile:

```
1  strtok(s, "\n");
```

d) Zusatzaufgabe `zkeingabe4.c`: Die Eingabe soll Zeichen für Zeichen durch `getchar()` erfolgen.

Aufgabe 4: Erstellen einer Zeichenkette

Die Zeichenkette `zk` soll den Inhalt einer 40 Zeichen langen LCD-Zeile aufnehmen können:

```
1  char zk[41];
```

Ihr Programm soll diese Zeichenkette erstellen und zu Testzwecken ausgeben.

a) Warum hat `zk` 41 statt 40 Elemente?

b) `zkerstellen1.c`: Der Inhalt von `zk` soll aus lauter Leerzeichen bestehen. Ausnahme: Die vorderste und die hinterste Stelle sollen jeweils ein Pluszeichen enthalten. Hinweis: Ein Leerzeichen (ASCII Nr. 32) ist zu unterschieden vom Terminator (ASCII Nr. 0).

Aufgabe 5: Analyse einer Zeichenkette

Die Zeichenkette `fm` soll bis zu 80 Zeichen für eine Fehlermeldung aufnehmen können:

```
1  char fm[81]="Zur␣Zeit␣kein␣Fehler␣(wenn␣das␣mal␣so␣bleibt...)";
```

Ihr Programm soll die Zeichenkette untersuchen und eine entsprechende Meldung auf dem Bildschirm ausgeben.

a) `zkuntersuchen1.c`: Wenn die Zeichenkette leer ist (Element Nr. 0 ist schon der Terminator), soll `"leer"` ausgegeben werden.

b) `zkuntersuchen2.c`: Wenn die Zeichenkette zu Beginn ein Prozentzeichen enthält, soll die Meldung `"beginnt mit %"` ausgegeben werden.

c) `zkuntersuchen3.c`: Wenn die Zeichenkette *irgendwo* ein Prozentzeichen enthält, soll die Meldung `"enthaelt %"` ausgegeben werden. Hinweis: Zeichen hinter dem Terminator dürfen in dieser und den folgenden Teilaufgaben nicht berücksichtigt werden!

d) `zkuntersuchen4.c`: Wenn die Zeichenkette *irgendwo* eine Ziffer enthält, soll die Meldung `"enthaelt Ziffer"` ausgegeben werden. Zur Abfrage, ob ein Element eine Ziffer ist, sollen Sie hier den ASCII-Wert des Elements benutzen. Eine Ziffer hat einen ASCII-Wert zwischen 48 (Ziffer 0) und 57 (Ziffer 9).

e) `zkuntersuchen5.c`: Wenn die Zeichenkette *irgendwo* eine Ziffer enthält, soll die Meldung `"enthaelt Ziffer"` ausgegeben werden. Zur Abfrage, ob ein Element eine Ziffer ist, sollen Sie hier die Funktion `isdigit()` aus `<ctype.h>` verwenden. `isdigit(x)` liefert als Rückgabewert null, falls `x` keine Ziffer ist, andernfalls einen von null verschiedenen Wert.

f) `zkuntersuchen6.c`: Es soll ausgegeben werden, wie viele Ziffern die Zeichenkette enthält. Zur Abfrage, ob ein Element eine Ziffer ist, sollen Sie hier den ASCII-Wert des Elements benutzen.

g) `zkuntersuchen7.c`: Es soll ausgegeben werden, wie viele Ziffern die Zeichenkette enthält. Zur Abfrage, ob ein Element eine Ziffer ist, sollen Sie hier die Funktion `isdigit()` verwenden.

Aufgabe 6: Substitution (Ersetzung) von Zeichen in einer Zeichenkette

In dieser Aufgabe sollen in einer Zeichenkette einzelne Zeichen durch andere Zeichen ersetzt werden. Anschließend soll die Zeichenkette ausgegeben werden. Die Zeichenkette ist folgende:

```
1  char zk[]="Hier steht ein Text.";
```

a) `zksubst1.c`: Jedes `'t'` soll durch ein `'T'` ersetzt werden.

b) `zksubst2.c`: Jeder Vokal (a, e, i, o oder u) soll durch ein `'e'` ersetzt werden.

c) `zksubst3.c`: Jeder Kleinbuchstabe (zu ermitteln durch die Funktion `islower()` aus `<ctype.h>`) soll durch ein `'a'` ersetzt werden.

d) `zksubst4.c`: Jeder Kleinbuchstabe soll ersetzt werden durch das Zeichen, das im ASCII direkt dahinter liegt, also `'a'` durch `'b'` und `'b'` durch `'c'` usw. Der Kleinbuchstabe `'z'` soll durch ein `'a'` ersetzt werden (Cäsar-Code).

e) `zksubst5.c`: Die ersten 13 Kleinbuchstaben sollen ersetzt werden durch das Zeichen, das im ASCII 13 Zeichen dahinter liegt. Die anderen 13 Kleinbuchstaben sollen ersetzt werden durch das Zeichen, das im ASCII 13 Zeichen weiter vorn liegt (ROT13-Codierung).

Aufgabe 7: Kopieren von Zeichenketten I

Es sind zwei Arrays vorhanden:

```
1  char quelle[51]="Erste Hilfe";
2  char ziel[101] ="Hier steht nur Unsinn drin.";
```

Der Inhalt des ersten Arrays soll nun in das zweite Array übetragen werden. Zum Schluss soll das zweite Array auf den Bildschirm ausgegeben werden.

a) `zkkopieren1.c`: Die Quelle soll unverändert ins Ziel kopiert werden.

b) `zkkopieren2.c`: Die Quelle soll um drei Zeichen verschoben ins Ziel kopiert werden. Das Ziel soll zu Beginn drei Leerzeichen haben:
`"   Erste Hilfe"`

c) `zkkopieren3.c`: Die Quelle soll verdoppelt ins Ziel kopiert werden:
`"Erste HilfeErste Hilfe"`

d) `zkkopieren4.c`: Die Quelle soll Zeichen für Zeichen verdoppelt ins Ziel kopiert werden:
`"EErrssttee  HHiillffee"`

e) `zkkopieren5.c`: Die Quelle soll so kopiert werden, dass im Ziel so genannte Sperrschrift vorliegt. Dabei wird jedes Zeichen durch ein Leerzeichen vom nächsten Zeichen getrennt:
`"E r s t e  H i l f e "`

f) `zkkopieren6.c`: Die Quelle soll so kopiert werden, dass ihr Inhalt an den alten Inhalt des Ziels *angehängt* wird:
`"Hier steht nur Unsinn drin.Erste Hilfe"`

Aufgabe 8: Kopieren von Zeichenketten II

Es sind wieder zwei Arrays vorhanden:

```
1  char quelle[51]="1 Schulstunde kommt selten allein.";
2  char ziel[101] ="Hier steht nur Unsinn drin, nichts Wichtiges.";
```

Der Inhalt des ersten Arrays soll nun in das zweite Array übetragen werden. Zum Schluss soll das zweite Array auf den Bildschirm ausgegeben werden.

a) `zkkopieren7.c`: Im Ziel soll jede Ziffer `1` durch den Text `Eine` ersetzt sein:
`"Eine Schulstunde kommt selten allein."`

b) `zkkopieren8.c`: Im Ziel soll das vorderste Zeichen fehlen:
`" Schulstunde kommt selten allein."`

c) `zkkopieren9.c`: Im Ziel sollen alle Leerzeichen (ASCII Nr. 32) fehlen:
`"1Schulstundekommtseltenallein."`

d) `zkkopierena.c`: Im Ziel sollen die Elemente Nr. 0 und Nr. 1 vertauscht sein:
`" 1Schulstunde kommt selten allein."`

e) `zkkopierenb.c`: Im Ziel sollen die Elemente Nr. 0 und Nr. 1, Nr. 2 und Nr. 3 usw. vertauscht sein:
`" 1cSuhslutdn eokmm testlnea llie.n"`

f) `zkkopierenc.c`: Der Zielstring soll umgekehrt sein:
`.niella netles tmmok ednutsluhcS 1`

Aufgabe 9: Kopieren von Zeichenketten III

Es sind diesmal drei Arrays vorhanden:

```
1  char aquelle[51]="Katze";
2  char bquelle[51]="Maus";
3  char ziel[101] ="Hier steht nur Unsinn drin, nichts Wichtiges.";
```

Das Ziel soll nun in Abhängigkeit der beiden Quellen gefüllt und anschließend ausgegeben werden.

a) `zkkopierend.c`: Im Ziel soll zuerst der Inhalt von `aquelle`, dann der Inhalt von `bquelle` stehen:
`"KatzeMaus"`

b) `zkkopierene.c`: Im Ziel soll wieder zuerst der Inhalt von `aquelle`, dann der Inhalt von `bquelle` stehen. Diesmal sollen die beiden Inhalte durch ein Komma und ein anschließende Leerzeichen getrennt sein:
`"Katze, Maus"`

c) Zusatzaufgabe `zkkopierenf.c`: Im Ziel sollen der Inhalt von `aquelle` und der Inhalt von `bquelle` stehen, und zwar so, dass immer ein Zeichen von `aquelle` und ein Zeichen von `bquelle` aufeinander folgen:
`"KMaatuzse"`
Hinweis: Falls `aquelle` und `bquelle` unterschiedlich lang sind, müssen die bei der kürzeren Zeichenkette „fehlenden“ Zeichen beim Kopieren durch Leerzeichen aufgefüllt werden.

Aufgabe 10: Ersetzen von Zeichen – Verteilen eines Textes auf mehrere Zeilen

Das Programm soll eine Zeichenkette so verändern, dass jedes Wort eine eigene Zeile bekommt. Leerzeilen sollen dabei kein Problem sein. Man kann das ganz einfach erreichen, indem man in der Zeichenkette jedes Zeichen, das kein Buchstabe und keine Ziffer ist, durch ein `\n` ersetzt. Anschließend soll die veränderte Zeichenkette auf dem Bildschirm ausgegeben werden. Hier der Dialog:

Terminal
```
schueler@debian964:~$ ./a.out
Eingabe: So ist es.
Ausgabe:
So
ist
es
```

a) `verteil1.c`: Erstellen Sie das Programm mit Benutzereingabe, Kopie in das Ziel und Ausgabe des Ziels!

Aufgabe 11: Verschlüsseln durch Sortierung

Betriebssysteme speichern ihre Passworte nicht im Klartext, sondern mit Hilfe einer Einwegverschlüsselung. Gibt man ein Passwort ein, wird es in ein Schlüsselwort umgerechnet. Nur das Schlüsselwort wird gespeichert, und das Passwort bleibt geheim — weil man aus dem Schlüssel nicht mehr auf das Passwort zurückkommen kann. Will nun jemand in das System einloggen, gibt er *sein* Passwort ein, und es wird mit derselben Einwegverschlüsselung in ein Schlüsselwort umgerechnet. Sind die beiden Schlüsselworte gleich, kann man davon ausgehen, dass die beiden Passworte auch gleich waren, und das Einloggen wird erlaubt.

Eine einfache Einwegverschlüsselung eines Textes besteht darin, alle Buchstaben der alphabetischen Reihenfolge nach zu sortieren. Man kann daraus nicht mehr den ursprünglichen Text rekonstruieren, aber man kann jeden Text darauf überprüfen, ob er denselben Schlüssel ergibt.

Als Sortierung kann das Verfahren *bubblesort* verwendet werden:

- Nacheinander werden das erste und zweite, das zweite und dritte, das dritte und vierte Element usw. verglichen.
- Sind zwei Elemente in der falschen Reihenfolge ($a > b$), werden sie sofort vertauscht. Dazu kann `swap()` benutzt werden.
- Ist man beim letzten Element angekommen, wird wieder von vorn begonnen.
- Das Ganze wird so lange fortgesetzt, bis bei einem ganzen Durchlauf durch das Feld keine Vertauschung mehr nötig war.

a) `cryptsort.c`: Ihr Programm soll die Einwegverschlüsselung bewirken, indem es die Zeichen einer Zeichenkette mit *bubblesort* sortiert. Hier eine Ausgabe des Programms:

Terminal
```
schueler@debian964:~$ ./a.out
Vorher :Dies ist nur irgendein sinnloser Satz.
Nachher:     .DSadeeeegiiiiilnnnnnorrrsssttuz
```