4.12.A Datenstrukturen/String-Bibliothek – Arbeitsblatt

Aufgabe 1: Programm reparieren

Das folgende Programm `strreparier.c` kann nicht kompiliert werden. Verändern Sie jede Zeile so, dass das Programm so wie gewünscht funktioniert:

```

1 #include <stdio.h>
2 #include <string.h>
3 int main(void)
4 {
5     char s[100];
6     char t[100];
7     s="Hallo";
8     t="Willi";
9     s=s+",_";
10    s=s+t;
11    printf("%s", s);
12    return 0;
13 }
```

Aufgabe 2: System-Befehl (String zusammenbauen)

Das Programm `holewebsite1.c` soll eine bestimmte Website herunterladen.

Der Benutzer soll zuerst die URL eingeben (z. B. `www.beispiel.de`). Anschließend soll das Programm mit der Funktion `system()` das Programm `wget` aufrufen, das diese Aufgabe übernimmt.

Der Funktionsaufruf muss dazu lauten:

```
1 rc=system("wget_www.beispiel.de");
```

Mit dem Rückgabewert `rc` kann man erkennen, ob die Aktion funktioniert hat.

- Erstellen Sie `holewebsite.c`!
- Mit `holewebsite2.c` soll ein ganzer Webauftritt auf den eigenen Rechner geholt werden. Dazu dienen die Optionen `-nc -r -I/`. Mit ihnen kann man `wget` dazu bringen, auch verlinkte Seiten auf demselben Server herunterzuladen.

Aufgabe 3: Telefonnummer erstellen

Ein Programm `synth_tel.c` soll zuerst die Vorwahl und die Durchwahl einer Telefonnummer erfragen. Anschließend soll es die Gesamtnummer in der Form (Vorwahl) Durchwahl erstellen und zum Schluss ausgeben. Bei leerer Vorwahl soll "05231" als Vorwahl angenommen werden. Beispiel für einen Dialog (Benutzereingabe in Fettdruck):

```

Terminal
schueler@debian964:~$ synth_tel
Vorwahl..: 0521
Durchwahl: 68000
Ergebnis: (0521) 68000
schueler@debian964:~$ synth_tel
Vorwahl..: 
Durchwahl: 32032
Ergebnis: (05231) 32032
```

Aufgabe 4: Echo-Programm (Strings vergleichen)

Das Programm `zecho.c` soll jede eingelesene Zeile sofort wieder ausgeben. Ausgenommen ist nur eine Zeile mit dem Inhalt `QUIT`. Bei dieser Zeile soll das Programm sofort stoppen. Hier ein Beispiel für einen Dialog (Benutzereingaben in Fettdruck):

```
Terminal
schueler@debian964:~$ zecho
So
So
ein
ein
Quark
Quark
QUIT
Tschuess!
schueler@debian964:~$
```

Aufgabe 5: Zusammenbau einer Wortliste

Das Programm `wortliste1.c` soll jedes Wort (durch Leerzeichen, Tabulator usw. getrennt), das vom Benutzer eingegeben wird, an einen String mit maximal 80 sichtbaren Zeichen anhängen. Zu Beginn soll der String leer sein. Sobald das Wort `QUIT` eingegeben wird, soll die Liste beendet werden. Zwischen zwei Worten sollen jeweils ein Komma und ein Leerzeichen stehen. Zum Schluss soll die Wortliste ausgegeben werden. Hier ein Beispiel:

```
Terminal
schueler@debian964:~$ wortliste
Eingabe: Hund Katze Maus QUIT
Wortliste: Hund, Katze, Maus
```

- Erstellen Sie das Programm `wortliste1.c`! Zunächst soll nicht überprüft werden, ob die maximale Stringlänge überschritten wird ().
- Im Programm `wortliste2.c` soll sichergestellt sein, dass die maximale Stringlänge nicht überschritten wird. Es soll aber auch kein Wort abgeschnitten werden. Stattdessen soll dann, wenn ein Wort nicht mehr in den String passt, das Programm wie bei der Eingabe des Wortes `QUIT` beendet und vorher die Wortliste (natürlich ohne das überzählige Wort) ausgegeben werden.

Aufgabe 6: Inventurhilfe-Programm

Ein Programm `ihilfe.c` soll es erlauben, nacheinander gezählte Bauteile einzugeben. Nach Ende der Eingabe (durch Eingabe des Wortes `Quit`) soll zu jedem möglichen Bauteil die Anzahl ausgegeben werden. Hier ist ein Beispiel für einen Dialog:

```
Terminal
schueler@debian964:~$ ihilfe
Willkommen beim Inventurhilfe-Programm.
Eingabe: BC548
Eingabe: LED
Eingabe: 8051
Eingabe: 8051
Eingabe: Quit
Ergebnis:
Anzahl BC548: 1
Anzahl LED: 1
```

```
Anzahl 8051: 2
Anzahl LCD: 0
```

Als Datenstruktur für die erlaubten Bauteile kann man ein Array aus Strings verwenden:

```
char bauteile[4][20]={"BC548", "LED", "8051", "LCD"};
```

Als Datenstruktur für die Anzahl der Bauteile kann man ein weiteres Array benutzen:

```
unsigned int anzahl[4]={0,0,0,0};
```

bauteile	anzahl
BC548	0
LED	0
8051	0
LCD	0

Aufgabe 7: Kennzeichen zerteilen

Das Program `kfz.c` soll nach Eingabe eines Kfz-Kennzeichens den Landkreis (bzw. die kreisfreie Stadt) ermitteln und ausgeben:

```
Terminal
schueler@debian964:~$ kfz
Kennzeichen eingeben.....: PB-XY 1234
Landkreis oder kreisfreie Stadt: Paderborn
```

Aufgabe 8: Trennen einer Liste

Im Programm `trennliste.c` gibt der Benutzer eine durch Doppelpunkt getrennte Liste von Dingen ein (maximal 800 sichtbare Zeichen). Das Programm soll die Liste zerteilen und die Elemente einzeln ausgeben. Hier ein Beispiel:

```
Terminal
schueler@debian964:~$ trennliste
Ihre Eingabe: Pfeffer:Zwiebeln:Gelbe Paprika:Reis:Erbsen
Elemente:
Pfeffer
Zwiebeln
Gelbe Paprika
Reis
Erbsen
schueler@debian964:~$
```

Aufgabe 9: Suche in Assoziativem Array

Im folgenden Programm `getinhalt.c` enthält die Variable `arr` eine Reihe von Zeigern auf konstante Zeichenketten der Form `Name=Inhalt`. Man spricht hier von einem Assoziativen Speicher, denn zu jedem Inhalt (nach dem Gleichheitszeichen) ist die zugehörige Bedeutung (vor dem Gleichheitszeichen) vermerkt.

In diesem Fall ist im Namen ein Ländercode nach ISO 3166 (allerdings in Kleinbuchstaben) angegeben. Im Inhalt ist die Bezeichnung des zugehörigen Landes. Die Funktion `get_inhalt()` soll nun zu einem als Parameter übergebenen Namen den entsprechenden Inhalt ausgeben. Falls nichts gefunden wurde oder falls ein oder mehrere `NULL`-Zeiger als Parameter übergeben wurden, soll der Wert `NULL` zurückgegeben werden.

- a) Erstellen Sie diese Funktion!
- b) Der Fragezeichen-Operator in Zeile 10 hat nichts mit Zeichenketten zu tun. Was bewirkt er?
- c) Falls das Array nach Ländercodes sortiert und die Anzahl der Elemente vorher bekannt ist — wie kann man dann die Suche beschleunigen?

```
1 /* 1 */ #include <stdio.h>
2 /* 2 */ #include <string.h>
3 /* 3 */ char *get_inhalt(const char *arr [], const char *such);
4 /* 4 */ int main(void)
5 /* 5 */ {
6 /* 6 */     const char *arr []={ "dk=Daenemark", "it=Italien",
7 /* 7 */                               "lu=Luxemburg", "nl=Niederlande", NULL};
8 /* 8 */     char *p;
9 /* 9 */     p=get_inhalt(arr, "nl");
10 /* 10 */    printf(">>>%s<<\n", p?p: "nicht_gefunden");
11 /* 11 */    return 0;
12 /* 12 */ }
```