7.2.A Projekte/Prüfungstrainer – Arbeitsblatt

1 Aufgabe

Für Prüfungsvorbereitungen brauchen Sie ein Programm, welches Ihnen stichprobenartig Fragen stellt, die Sie beantworten müssen. Nach jeder Frage wird die richtige Antwort ausgegeben.

```

Terminal
schueler@debian964:~$ pruefungstrainer
Bitte waehlen Sie:
0 Ende
1 Quiz
Ihre Auswahl: 1
Frage: Welches ROM ist nur durch UV-Licht loeschbar?
Ihre Antwort: EPROM
Richtig!
Moechten Sie noch eine Frage (J/N)? J
Frage: Rechnen Sie 1101(2) um ins Oktalsystem!
Ihre Antwort: 12345
Falsch!
Richtige Loesung: 15
Moechten Sie noch eine Frage (J/N)? N
Anzahl der richtigen Antworten: 50%
Bitte waehlen Sie:
0 Ende
1 Quiz
Ihre Auswahl: 0
schueler@debian964:~$

```

- Entwerfen Sie die Variable `fragenfeld` so, dass die Fragen und Antworten in einem Feld aus maximal 200 Einträgen gespeichert sind! Jeder Eintrag soll eine Struktur aus zwei Zeichenketten (je maximal 80 Zeichen) sein. Ergebnis: Jackson-Diagramm von `fragenfeld`, Deklaration von `fragenfeld`.
- Zuerst ist es sinnvoll, dass das Feld zu Beginn quasi von Hand gefüllt wird; dazu empfiehlt sich eine funktion `feld_laden()`:

```

1  unsigned short int feld_laden(struct eintrag_t feld [])
2  {
3      strcpy(feld[0].frage, "Wie_heisst_der_Buergermeister_von_Wesel?");
4      strcpy(feld[0].antwort, "Esel");
5      ...
6      strcpy(feld[17].frage, "Kfz-Kennzeichen_von_Warendorf?");
7      strcpy(feld[17].antwort, "WAF");
8      return 18; /* Anzahl von Fragen */
9  }

```

- Zunächst sollte man eine Funktion `einefragestellen()` erstellen, die zu einem Eintrag erst die Frage stellt, dann die Antwort einliest, mit der im Eintrag gespeicherten Lösung vergleicht und dem Benutzer mitteilt, ob seine Antwort richtig war (und falls nicht, die richtige Antwort nennt). Falls ja, soll die Funktion den Wert 1 zurückgeben, sonst 0. Ergebnis: Funktion `einefragestellen()`
- Außerdem sollte das Programm zunächst so arbeiten, dass alle Fragen nacheinander abgearbeitet werden (Funktion `quiz1()`). Ergebnis: Struktogramm des Hauptprogramms, Hauptprogramm, Funktion `quiz1()`

- e) Nun sollen die Fragen zufällig ausgewählt werden. Dazu soll ein neuer Menüpunkt (2 – Quiz mit Zufallsauswahl) eingerichtet werden.

Ergebnis: Funktion `quiz2()`

- f) Jetzt soll das Programm die Fragen aus einer Textdatei laden. In dieser Datei sollen abwechselnd je eine Fragenzeile und die dazugehörige Antwortzeile stehen. Außerdem muss ein neuer Menüpunkt (3 – Fragen aus Datei laden) eingerichtet werden.

Ergebnis: Funktion `feld_laden_aus_datei()`

- g) — **Zusatzaufgabe**

Jetzt soll es möglich sein, zur Laufzeit des Programms neue Fragen und Antworten dynamisch aus einem C-Programm zu erzeugen. Als Beispiel soll folgendes C-Programm dienen (`src/p1.c`):

```

1  /* p1.c */
2  #include <stdio.h>
3  int main(void)
4  {
5      int x, y;
6      srand( getpid()+time(NULL));
7      x=random()%10;
8      y=random()%10;
9      printf("Wieviel ist %d+%d?\n", x,y);
10     printf("%d\n", x+y);
11     return 0;
12 }
```

Dieses Programm gibt bei jedem Aufruf eine Additionsaufgabe aus:

```

Terminal
schueler@debian964:~$ ./a.out
Wieviel ist 8 + 1?
9
schueler@debian964:~$
```

In der ersten Zeile ist die Frage aufgeführt, in der zweiten Zeile die durch das Programm berechnete Antwort.

Für solche dynamisch erzeugten Fragen und Antworten soll im Eintrag die Fragen-Komponente als erstes Zeichen ein # besitzen und danach den Pfadnamen. Die Antwort-Komponente des Eintrags darf beliebig sein.

```

1  feld[0].frage="#p1.c";
2  feld[0].antwort="";
```

Die Funktion `einefragestellen()` soll jetzt so erweitert werden, dass sie bei einem Eintrag, bei dem die Frage mit dem Zeichen # beginnt, nicht sofort die Frage stellt.

Stattdessen ruft sie die Funktion `einefrageerzeugen()` auf, die die Quelltextdatei kompiliert und ausführt und die Ausgabe (Frage und Antwort) zurückgibt. Auch diese Funktion ist zu erstellen.

Man sollte darauf achten, das so geänderte Programm nur in vertrauenswürdigen Umgebungen und niemals als Administrator auszuführen, da man durch entsprechend böse Frageprogramme jede erlaubte Funktion auf dem System ausführen kann.