2.3.A Funktionen/Funktionen erstellen – Arbeitsblatt

Aufgabe 1: Programm für Systeminformationen

Das Programm `systeminf.c` soll wichtige Systeminformationen ermitteln und ausgeben. Jede dieser Informationen soll dabei aus Gründen der Übersicht in einer eigenen Funktion ermittelt werden.

a) Schreiben Sie eine Funktion `bsclear()`, die mit Hilfe des Befehls `system("clear")` den Bildschirm löscht! [1] Für die Benutzung von `system()` muss übrigens die Headerdatei `<stdlib.h>` mit `#include` eingebunden werden.

b) Schreiben Sie eine Funktion `printdatum()`, die mit Hilfe des Befehls `system("date")` das aktuelle Datum mit Uhrzeit auf dem Bildschirm ausgibt! [2]

c) Schreiben Sie eine Funktion `printbs()`, die mit Hilfe des Befehls `system("uname -a")` die aktuelle Betriebssystemversion auf dem Bildschirm ausgibt![3]

d) Schreiben Sie eine Funktion `printnutzer()`, die mit Hilfe des Befehls `system("who")` die Namen der momentan eingeloggten Nutzer auf dem Bildschirm ausgibt![4]

e) Schreiben Sie eine Funktion `printproz()`, die mit Hilfe des Befehls `system("ps -a")` die Informationen zu den aktuellen Prozessen auf dem Bildschirm ausgibt![5]

f) Schreiben Sie eine Funktion `printplatz()`, die mit Hilfe des Befehls `system("df")` die den momentan verfügbaren Massenspeicherplatz auf dem Bildschirm ausgibt![6]

g) Das Hauptprogramm (also die Funktion `main()`) soll die oben genannten Funktionen nacheinander ausführen (das Löschen des Bildschirm natürlich zuerst), eine Sekunde warten (mit `sleep(1)`) und dann wieder von vorne anfangen.

h) Binden Sie alle genannten Funktionen zu einem lauffähigen Programm zusammen!

Aufgabe 2: Bild mit Flaggen

Wenn man mit dem Programm `liner` Gegenstände zeichnet, ist es sinnvoll, das Zeichnen jedes Gegenstands in einer Funktion zusammenzufassen. Im Programm `flaggen2.c` soll eine Zeichnung mit zwei Flaggen entstehen.

a) Erstellen Sie ein Hauptprogramm, das zwei Funktionen mit den Namen `landeins()` und `landzwei()` nacheinander aufruft und dann für eine Minute wartet (`sleep(60)`), bevor es sich beendet.

b) Erstellen Sie (fast) leere Funktionen `landeins()` und `landzwei()`! Jede Funktion soll nur ihren Namen ausgeben (etwa so: `printf("Hier ist landeins()\n")`) und sonst noch nichts tun. Compilieren und testen Sie Ihr Programm!

c) Jetzt soll die Funktion `landeins()` das linke obere Viertel des Bildschirms mit einer Flagge füllen. Compilieren und testen Sie Ihr Programm wiederum!

d) Nun soll `landzwei()` das rechte untere Viertel mit einer anderen Flagge füllen. Compilieren und testen Sie Ihr Programm noch einmal!

[1] in Windows: `system("cls")`
[2] in Windows: `system("time")`
[3] in Windows: `system("ver")`
[4] in Windows so nicht möglich
[5] in Windows so nicht möglich
[6] in Windows: `system("chkdsk")`

Aufgabe 3: Hello, World als Banner

Das Programm `banner.c` soll den Schriftzug `HELLO` sehr groß in senkrechter Richtung auf den Bildschirm ausgeben:

Terminal

```
schueler@debian964:~$ banner
********
   *
   *
   *
********
********
*    *   *
*    *   *
*    *   *

********
*
*
*
*
********
*
*
*
*
  ****
 *    *
*      *
 *    *
  ****
```

Dazu sollen Sie C-Funktionen erstellen und gebrauchen.

a) Jeder Buchstabe soll eine eigene Funktion bekommen. Erstellen Sie leere Funktionen `h()`, `e()`, `l()` und `o()`!

b) Erstellen Sie die `main()`-Funktion, die nacheinander `h()`, `e()`, `l()` (zweimal) und `o()` aufruft! Compilieren und testen Sie Ihr Programm!

c) Bauen Sie jetzt die Funktionen `h()`, `e()`, `l()` und `o()`; compilieren und testen Sie Ihr Programm immer wieder! Falls der Schriftzug zu schnell durchläuft, fügen Sie eine `sleep(1)`-Anweisung zwischen den Buchstaben ein!

Aufgabe 4: Eingabepuffer löschen

Es ist sinnvoll, in einem C-Programm nach jedem Aufruf von `scanf()` den Eingabepuffer zu löschen. Dazu kann man folgende Zeilen verwenden:

```
1  int ch;
2  do{
3      ch=getchar();
4  }while(ch!=EOF && ch!='\n');
```

Weil diese Zeilen sehr oft aufgerufen werden, bietet es sich an, dafür einen Funktionsbaustein zu erstellen. Er soll den Namen `scanclear` bekommen.

a) Wie lautet der Prototyp von `scanclear`?

b) Erstellen Sie die Funktionsdefinition von `scanclear`!

c) Testen Sie `scanclear` mit dem folgenden Programm, einmal mit und einmal ohne `scanclear`:

scanclear.c

```c
#include <stdio.h>
/* hier kommt der Prototyp hin */
int main(void)
{
    double x=123.4, y=567.8;
    printf("Bitte x eingeben (danach irgendwas): ");
    scanf("%lf", &x);
/* scanclear(); */ /* Kommentarklammern entfernen */
    printf("Bitte y eingeben: ");
    scanf("%lf", &y);

    printf("\n");
    printf("x=%lg\n", x);
    printf("y=%lg\n", y);
    return 0;
}
/* und hier kommt die Funktionsdefinition hin */
```