Zürcher Hochschule für Angewandte Wissenschaften

Arbeitsblatt: INF1

Name: Nathaniel Walser Kurznamen: walnat01

Arrays, Zeiger, Datenstrukturen 2

Aufgabe 1: Datenstrukturen, strukt

Vereinbaren Sie eine Datenstruktur *Zeit* wie folgt:

```
typedef struct Zeit {
    int h, min, sec;
} Zeit;
```

Verwenden Sie die typedef Schreibweise. Die Deklaration der Datenstruktur befindet sich in der Programmdatei oberhalb der Funktionsdefinitionen. Vergessen Sie das abschliessende Semikolon nicht. Eine *zeit* besteht also aus drei int-Werten: h, min und sec. Eine Funktion zum Einlesen einer Uhrzeit soll so implementiert werden:

```
Zeit leseZeit (void) {
    Zeit t;
    printf("Zeit (h:min:sec): ");
    scanf(" %d:%d:%d", &(t.h), &(t.min), &(t.sec));
    return t;
}
```

Die Zeit muss übrigens genau im Format h:min:sec eingegeben werden, damit es funktioniert. Das Blank in der Formatangabe bei scanf sorgt dafür, dass führende Leerzeichen überlesen werden. Es ist klar, dass diese Implementierung nicht sehr robust gegen Fehleingaben ist, aber für dieses Beispiel soll es genügen.

Im Gegensatz zu Arrays lassen sich Datenstrukturen einfach als Werte zuweisen. Es werden dann die einzelnen Komponenten der Datenstruktur in die entsprechenden Komponenten der Variablen auf der linken Seite der Zuweisung **kopiert**:

```
Zeit t1;
t1 = leseZeit();
```

Aufgaben

- Schreiben Sie eine Funktion ***`void schreibeZeit(Zeit t)`***, die eine Zeit ausgibt. Hinweis: Mit `%02d` können Sie ein int zweistellig ausgeben, bei Bedarf wird vorne eine 0 eingefügt, Bsp: 02:00:10
- Schreiben Sie eine *main*-Funktion, die eine Zeit einliest und wieder ausgibt und dazu die beiden bereits definierten Funktionen verwendet.
- Schreiben Sie eine Funktion ***`Zeit zeitDifferenz(Zeit t1, Zeit t2)`***, die zwei Zeiten bekommt und die Zeitdauer zwischen den beiden Zeiten berechnet(Betrag > 0). Die Rückgabe soll ebenfalls in Form einer *Zeit* erfolgen.

- Passen Sie die *main*-Funktion so an, dass **zwei Zeiten** eingelesen werden. Dann wird die Differenz bestimmt und ausgegeben. Ein Dialog mit dem Programm könnte folgendermassen aussehen:

```
Bitte erste Zeit eingeben:
Uhrzeit (h:min:sec): 12:15:45
Bitte zweite Zeit eingeben:
Uhrzeit (h:min:sec): 15:10:50
Zeitdauer: 2:55:05
```

Hinweis

- Eine elegante Lösung ergibt sich, wenn man die Zeiten zuerst in Sekunden umwandelt
- Die Unix Zeit ist die Sekunden seit Donnerstag, dem 1. Januar 1970, 00:00 Uhr UTC https://de.wikipedia.org/wiki/Unixzeit; dies wird im Jahr 2038 zu einem Problem führen.

Abgabe

Praktikum: INF8.1

Filename: zeiten.c

Aufgabe 2: Beste Zeit bestimmen

Sie möchten eine Folge von Zeiten, wie z.B. Zielzeiten bei einem Wettkampf eingeben können. Dafür speichern Sie die Zeiten in einem Array von "genügender" Grösse (die Maximalgrösse soll in einer Konstanten angegeben werden). Es soll eine Schleife programmiert werden, in der diese Zeiten eingegeben werden können. Soll die Eingabe beendet werden, dann soll einfach die Zeit 0:0:0 eingegeben werden können. Aus diesem Array soll nun mit der Funktion `Zeit bestZeit (int maxT, Zeit zeiten[])` die beste Zeit bestimmt werden. maxT gibt die Grösse des Arrays an. Die einzelnen Zeiten soll in einer Funktion `int vergleiche(Zeit t1, Zeit t2)` durchgeführt werden, die wiefolgt definiert ist:

- t1 < t2 → return Wert <0
- t1 == t2 → return Wert =0
- t1 > t2 → return Wert >0

Hinweis

- Eine elegante Lösung ergibt sich, wenn man die Zeiten zuerst in Sekunden umwandelt.

Aufgabe

Schreiben Sie ein Programm, mittels dem Sie **beliebig viele Zeiten** eingeben können und das Ihnen die Bestzeit daraus bestimmt und diese ausgibt (Ende der Eingabe bei Zeit 0:0.0)

Abgabe

Praktikum: INF8.2

Filename: zeiten.c

Aufgabe 3: Übersetzung mittels Make (optional)

Schreiben Sie ein Makefile, mittels dem sie das Programm "zeiten.c" auf der Konsole *übersetzen* und *starten* können.