Ziele der Aufgabenstellung:

- Programm mittels strukturierter Programmierung entwerfen
- Aufteilung des Programms in Module
- Einbindung eines Assemblerprogramms in ein C-Programm

Vorzubereiten:

- strukturierte Programmierung
- Aufteilung der Unterprogramme in sinnvolle Module
- Umgang mit Feldern und Zeigern, Call-by-reference-Aufrufkonvention
- Sortieren von Feldern

Vorausgesetztes Wissen u. Praktikumstest:

Diese Themen werden Bestandteil des Praktikumstests sein.

- Wiederholung und Vertiefung von Assemblerwissen
  - Datenfelder (DCB, DCW, DCD), Byteswapping, ….
  - Unterprogrammaufrufe in Assembler (Linkregister, Framepointer, Stack)
- definieren von Funktionen in C (call-by-value, call-by-reference)
- Umgang mit Strings, Feldern und Zeigern in C

Abnahme:

- funktionierendes Programm
- sinnvoll kommentierter Code

Aufgabenstellung:

Gegeben sei ein Feld von Strings (*Stringliste*) sowie ein Feld von Zeigern darauf, z.B.

```
char *pMeineStrings[] =  {
     "Haller       25 EUR",
     "Kandinsky    13 EUR",
     "Brombach      5 EUR",
     "Zaluskowski 120 EUR",
     "Osman        17 EUR",
     ""
     };
```

Das Ende der Stringliste ist am Leerstring `""` zu erkennen, d.h. dieser String besteht nur aus dem 0-Terminator. Alle Strings sind in der Form *„Name Zahl EUR“* aufgebaut.

a) Realisierung als reines C-Programm

Das Programm soll die Strings entsprechend der Größe der Zahl im String sortieren und ausdrucken, also hier:

```
Brombach      5 EUR
Kandinsky    13 EUR
Osman        17 EUR
Haller       25 EUR
Zaluskowski 120 EUR
```

Folgende C-Unterprogramme sind zu erstellen:

```
/* Ausdrucken der Stringliste */
void PrintStringliste( ....... );

/* Sortieren einer Stringliste nach der Größe */
/* der Zahl im String.                        */
void SortiereStrings( ........ );

/* Zahl im String bestimmen und zurückgeben */
int  getNum( ........ );
```

Für die Sortierung kann das Sortierverfahren "Bubblesort" verwendet werden.

Die Ausgabe der Strings erfolgt mit *printf*. Als Terminal wird auf dem PC die Anwendung **TI_Terminal** (Icon auf dem Desktop) gestartet.

Die C-Unterprogramme (s.o.) sollen im Modul `stringsort.c` definiert sein.
Das startende Hauptprogramm (main) soll im Modul `main.c` definiert sein.
Die Funktionsdeklarationen sollen über eine Include-Datei (`stringsort.h`) bekannt gemacht werden.

b) **Realisierung als gemischtes C-/Assemblerprogramm**

Die C-Funktion `getNum()` soll jetzt ersetzt werden durch ein Assembler-Unterprogramm `getNum_asm`.

Hierfür wird dem Projekt eine Datei `getnum.s` zugefügt. Die Funktionsdeklaration für `getNum_asm` muss ebenfalls in `stringsort.h` eingetragen werden.

Das Programm (auch der Assemblerteil) soll
- strukturiert und eingerückt
- sinnvoll kommentiert und
- mit selbsterklärenden Variablennamen versehen sein.

Aufbau der Datei `getnum.s`:

```
     AREA MyCode, CODE, readonly, align = 2

     GLOBAL    getNum_asm

; ------------------------------------------
; Inputregister:  r0 (Stringadresse)
; Outputregister: r0 (Zahlenwert)
; Verwendete Register retten und restaurieren.
; ------------------------------------------

getNum_asm     PROC
; Code starts here ……


     bx   lr
; ------------------------------------------
     ENDP
     ALIGN  4
     END
```

Aufbau der Datei `main.c`:

```
#include "main.h"
#include <stdio.h>
#include "TI_Lib.h"  //für evtl. Ausgabe auf TFT-Bildschirm
#include "tft.h"

// … sonstige Headerdateien …

char *pMeineStrings[] = {
      // die Strings (s.o.) ….
      }

int main(void){
     Init_TI_Board();
     printf("\f\n");   // Lösche Terminal-Screen
     TFT_cls();        // Lösche TFT-Display des TI-Boards
     PrintStringliste(…);
     SortiereStrings(…) ;
     PrintStringliste(…);

     while(1){}        // Endlosschleife
}
```