Haute école d'ingénierie et d'architecture Fribourg
Hochschule für Technik und Architektur Freiburg

Embedded Systeme 1 & 2

Klassen T-2/I-2 // 2018-2019

a.05 - C - Komplexe Strukturen

Übung 1

Gegeben sei die Struktur "*personne*" mit den folgenden Feldern:

- Zeichen "*classe*"
- Ganzzahlwert (16 Bit) ohne Vorzeichen "*Ident*"
- Ganzzahlwert (16 Bit) ohne Vorzeichen "*age*"

1. Deklarieren Sie die Variablen "*pers_a*" und "*pers_b*" vom Typ "*personne*"
2. Initialisieren Sie die Variablen mit:
    a. 'A', 12453, 45 Jahre
    b. 'D', 987, 67 Jahre

Übung 2

Eine Meldung setzt sich aus 10 Datenbytes zusammen (`unsigned char`) gefolgt von einem Byte mit gerader vertikaler Parität, das der nachstehenden Struktur folgt:

```
struct message {
   unsigned char info[10] ;
   unsigned char parity ;
}
```

Entwickeln Sie eine Funktion in C, die in der Lage ist, die Struktur **`message`** nach folgendem Prototyp zu berechnen und zu vervollständigen:

```
// Funktion zur Berechnung der Parität
// Parameter: Meldung
// Rückgabe: Parität
// Beschreibung: Berechnung der vertikalen Parität

unsigned char calc_parity (struct message msg);
```

Übung 3

Nachstehend folgen 3 in der Sprache C geschriebene Dateien eines zu kompilierenden Programms. Dieses Programm definiert und benutzt Additions- und Subtrationsfunktionen komplexer Zahlen. Die komplexen Zahlen sind in einer Struktur "**`struct complex`**" dargestellt, die zwei Variablen "**`real`**" und "**`imag`**" von je einem Byte (char) enthält.

Schreiben Sie die beiden Dateien "`nombre.h`" und "`opp.h`", die für das Kompilieren der 3 Quellcodedateien "`hello.c`", "`nombre.c`" und "`opp.c`" notwendig sind.

hello.c

```c
#include "nombre.h"
#include "opp.h"
int main() {
  struct complex s1 = {20,65};
  struct complex s2 = {56,12};
  struct complex s3 = addi(&s1,&s2);
  struct complex s4 = subs(&s1,&s2);
  return 0;
}
```

opp.c

```c
#include "nombre.h"
#include "opp.h"
// führt die Addition von a und b durch
struct complex addi(const struct complex *a, const struct complex *b) {
   struct complex c = {
      .real = real_part(a) + real_part(b),
      .imag = imag_part(a) + imag_part(b),
   };
   return c;
}

// führt die Subtraktion von a und b durch
struct complex subs(const struct complex *a, const struct complex *b) {
   struct complex c = {
      .real = real_part(a) - real_part(b),
      .imag = imag_part(a) - imag_part(b),
   };
   return c;
}
```

nombre.c

```c
#include "nombre.h"
long real_part(const struct complex *nbre) {
  long real = nbre->real;
  if (nbre->real > REAL_MAX)
    real = REAL_MAX;
  else if (nbre->real < REAL_MIN)
    real = REAL_MIN;
  return real;
}

long imag_part(const struct complex *nbre) {
  long imag = nbre->imag;
  if (nbre->imag > IMAG_MAX)
    imag = IMAG_MAX;
  else if (nbre->imag < IMAG_MIN)
    imag = IMAG_MIN;
  return imag;
}
```

Übung 4

Deklarieren Sie einen Datentyp, der den Austausch von 5 Meldungen von unterschiedlicher Grösse und unterschiedlichem Typ zwischen einem Client und einem Server erlaubt. Die Meldungen und ihre Grösse müssen klar festgestellt werden können. Das Behandlungsergebnis der Anfrage durch den Server muss dem Client mitgeteilt werden können.

```c
struct request1 {int a; int b;};
struct request2 {float a; int b; int c;};
struct request3 {};
struct request4 {int a[10]; int b;};
struct request5 {long a; char b;};

struct response1 {int a; int b;};
struct response2 {};
struct response3 {int a; int b; int c; int d;};
struct response4 {int a;};
struct response5 {long a;}
```

Übung 5

Entwickeln Sie ein kleines Programm, das die Anzeige des Offsets der Attribute attr<x> der nachstehenden Strukturen erlaubt. Verwenden Sie das Makro offsetof(type, member) von <stddef.h>.

```
struct struct1 {
   char    attr1;
   short   attr2;
   char    attr3;
   long    attr4;
   short   attr5;
   long    attr6;
};

struct struct2 {
   char    attr1;
   short attr2;
   short attr3;
   long    attr4;
   short   attr5;
   long    attr6;
} __attribute__((__packed__));

struct struct2 {
   char    attr1;
   char  dummy1;
   short   attr2;
   char    attr3;
   char  dummy2[3];
   long    attr4;
   short   attr5;
   char    dummy3[2];
   long    attr6;
};
```

Übung 6

Entwickeln Sie basierend auf der unten stehenden Abbildung, die ein Hardware-Peripheriegerät beschreibt, ein Programm, das Folgendes erlaubt:

1. die Struktur des Peripheriegeräts zu beschreiben
2. die Struktur durch den Algorithmus zu initialisieren
3. die Struktur mit den Vorgabewerten auf die herkömmliche Art zu initialisieren
4. die Struktur mit den Vorgabewerten mit dem Standard C11 zu initialisieren

	7	6	5	4	3	2	1	0	
cmd	X	X	X	X	X	GO	STP	RST	0
status	X	X	OK	X	X	X	ERR	IRQ	1
i/o	X	I/O	I/O	I/O	I/O	I/O	I/O	I/O	2
	X	X	X	X	X	X	X	X	3
counter1	b7							b0	4
	b15							b8	5
counter2	b7							b0	6
	b15							b8	7