Systèmes Embarqués 1 & 2

Classes T-2/I-2 // 2018-2019

a.05 – C - Structures complexes

Exercices

Exercice 1

Soit la structure *"personne"* contenant les champs suivants

- Caractère *"classe"*
- Valeur entière (16bits) non signée *"ident"*
- Valeur entière (16 bits) non signée *"age"*

(a) Déclarer les variables *pers_a* et *pers_d* de type *"personne"*

(b) Initialiser les variables de type *"personne"* avec

- A, 12453, 45 ans
- D, 987, 67 ans

Exercice 2

Un message est composé de 10 octets de donnée *(unsigned char)* suivit d'un octet de parité verticale paire suivant la structure suivante caractères. A partir de ce code, écrire en C la fonction de déchiffrement.

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

Développer une fonction en C capable de calculer et de compléter la structure message suivant le prototypage suivant

```
/**
 * fonction de calcul de la parité
 * effectue le calcul de la parité verticale
 * @param msg message
 * @return    parité verticale
 */
unsigned char calc_parity (struct message msg);
```

Exercice 3

Ci-dessous 3 fichiers écrits en langage C d'un programme à compiler. Ce programme défini et utilise des fonctions d'addition et de soustraction de nombre complexes. Les nombre complexes sont représenté dans une structure *"struct complex"* contenant deux variables *"real"* et *"imag"* de un octet chacun *(char)*.

Ecrire les deux fichiers *"nombre.h" et "opp.h" nécessaire à la compilation des 3 fichiers de code sources "hello.c", "nombre.c" et "opp.c".*

(a) 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;
}
```

(b) opp.c

```c
#include "nombre.h"
#include "opp.h"
// realise l'addition de a et de b et le sauve dans c
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;
}

// realise la soustraction de a et de b et le sauve dans c
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;
}
```

(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;
}
```

Exercice 4

Déclarer un type de données permettant l'échange de 5 messages de type et de taille différents entre un client et un serveur. Les messages et leur taille doivent clairement pouvoir être identifié. Le résultat du traitement de la requête par le serveur doit être rapporté au client.

```
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;}
```

Exercice 5

Développer un petit programme permettant d'indiquer l'offset des attributs attr<x> des structures ci-dessous. Utiliser la macro offsetof(type, member) de <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;
};
```

Exercice 6

Sur la base de la figure ci-dessous décrivant un périphérique matériel, développer un programme permettant de

(a) décrire la structure périphérique

(b) initialiser la structure par algorithme

(c) initialiser la structure avec des valeurs par défaut de manière classique

(d) initialiser la structure avec des valeurs par défaut avec le standard C11

	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