3ETI, Examen [CSC2]
Developpement Logiciel en C
CPE Lyon

2014-2015 (1ere session)
durée 3h

Tous documents et calculatrices autorisés.

Répondez aux questions sur une copie séparée

Le sujet comporte 8 pages

Le temps approximatif ainsi que le barème sont indiqués pour les grandes parties. Notez que le barème est donné à titre purement indicatif et pourra être adapté par la suite.

En cas de doute sur la compréhension de l'énoncé, explicitez ce que vous comprenez et poursuivez l'exercice dans cette logique.

Note préliminaire:

- Toutes les questions concernent le language de programmation C dans le cadre du developpement de logiciels sur architecture PC standard.
- Nous vous invitons à commenter le code et les réponses. En particulier, en cas d'ambiguité de compréhension d'une question, ajoutez toutes remarques et illustrations supplémentaires permettant d'expliquer votre démarche.
- Sauf mention contraire explicite, on supposera que l'on dispose d'un système Linux standard fonctionnant correctement sur un PC récent 64 bits (identiques aux conditions de TP des PC de CPE: *int* étant encodé sur 4 octets, pointeurs étant encodés respectivement sur 8 octets sur 64 bits).
- On supposera que le code C est compilé avec une version récente de gcc sous la norme C99 (ou ultérieure) identique aux conditions de TP des PC de CPE.
- On supposera dans chaque cas que les `#include` des en-tête standards nécessaires à la bonne compilation et exécution des programmes décrits sont correctement installés et appelés (ex. stdio, stdlib, string, math, etc).

1 Méthodologie et bonne pratiques de programmation

[50min max] 5 points

On considère les structures suivantes

```
#define NBR_PLANETE 8
//les noms des 8 planetes du systeme solaire.
enum nom_planete {mercure,venus,terre,mars,jupiter,saturne,uranus,neptune};

struct info_planete
{
  enum nom_planete nom; //la designation du nom d'une planete
  float rayon;          //le rayon moyen de la planete (float >0)
  int distance_soleil;  //la distance moyenne au soleil (entier >0)
  int nbr_satellites;   //le nombre de satellites de la planete
                                                  (entier >=0)
};

//structure stockant les informations des 8 planetes du systeme solaire
struct info_systeme_solaire
{
  struct info_planete planete[NBR_PLANETE];
};
```

On suppose également que l'on dispose de la fonction `systeme_solaire_initialiser` qui vient initialiser une structure de type `info_systeme_solaire` avec les bons paramètres pour chaque planète.
Notons que les planètes sont référencées dans l'ordre du tableau (mercure, venus, terre, mars, jupiter, saturne, uranus, neptune).

On souhaite réaliser la fonction `obtenir_info_planete` qui prend en paramètre un pointeur vers un système solaire et un indice, et renvoie un pointeur vers la planète correspondant à cet indice. Par exemple, l'indice 0 correspondra à la planète mercure, et l'indice 3 à la planète mars.

Le contrat de la fonction est le suivant:

```
/* La fonction obtenir_info_planete permet de recuperer un pointeur vers
 *  la planete du systeme solaire designee par son indice dans l'ordre
 *  des planetes.
 *
 * Necessite:
 *  - Pointeur vers un systeme_solaire non modifiable. Pointeur non NULL.
 *  - Un indice entier designant la position de la planete dans l'ordre du
 *     systeme solaire. L'indice doit etre compris entre 0 et NBR_PLANETE-1
     (inclus).
 * Garantie:
 *  - Renvoie un pointeur vers une structure d'info_planete non modifiable.
 *    Le pointeur est non NULL.
 *
 */
```

Question 1 *Écrivez l'en-tête de la fonction correspondant à ce contrat.*

Question 2 *Écrivez l'implémentation du corps de cette fonction correspondant à ce contrat et à votre en-tête.*

On souhaite réaliser la fonction `calculer_rayon_moyen` qui, étant donné l'information d'une structure `info_systeme_solaire`, calcule le rayon moyen de l'ensemble des planètes. Le contrat et l'en-tête de cette fonction sont les suivants:

```
/* Fonction calculer_rayon_moyen calcule le rayon moyen sur
 *  l'ensemble des planetes decrites dans ce systeme.
 *
 * Necessite:
 *  - Pointeur non modifiable vers un systeme_solaire. Pointeur non null.
 *  - Un systeme pour lequel l'ensemble des planetes possedent un rayon >
    0.
 * Garantie:
 *  - Renvoie un flottant positif (>0) correspondant au rayon moyen de l'
    ensemble des planetes.
 *
 */
float calculer_rayon_moyen(const struct info_systeme_solaire *systeme);
```

Question 3 *Écrivez le corps de cette fonction. Prenez soin de bien respecter les bons critères de programmation, et n'oubliez pas d'utiliser la fonction déjà existante.*

On souhaite finalement coder une fonction qui permet d'afficher sur la ligne de commande le nom (en toutes lettres) des planètes dont le rayon est compris entre deux bornes min et max.

L'en-tête de cette fonction est la suivante:

```
void afficher_planete_avec_rayon_specifique(const struct
   info_systeme_solaire *systeme,float rayon_min,float rayon_max)
```

Question 4 *Écrivez le contrat ainsi que le corps de cette fonction. Prenez soin de bien réspecter les critères de bonnes programmation. Notez qu'il est possible de créer des sous fonctions si cela permet de simplifier la lecture du code ou de factoriser celui-ci.*

2 Connaissances générales du développement logiciel

[35min max] 4 points

Question 5 *Pour chacune des propositions suivantes, dites si celle-ci est vraie ou fausse en justifiant ou en expliquant votre réponse. (Une réponse sans explication n'apporte pas de points). Remarque: un apport d'informations supplémentaires pertinentes peut dans certains cas apporter un bonus.*

1. Si on supprime les fichiers objets (.o), il n'est plus possible de lancer l'executable qui a été réalisé à partir de ceux-ci.
2. SVN est un compilateur C.
3. Un code sous licence GPL peut être modifié librement puis redistribué.
4. L'appel à la fonction `sizeof` sur un pointeur renvoie une taille qui dépend du type de la variable pointée.
5. Il est possible d'avoir des erreurs de gestion de mémoire lors de l'execution d'un programme alors que celui-ci termine normalement sans avoir affiché d'erreur de segmentation (ou Segmentation Fault en version anglaise).
6. Les commentaires (code après `\\`, ou entre `\*` et `*\`) sont parfois analysés par gcc pour optimiser le code binaire généré.
7. L'étape de préprocesseur lors de la compilation permet de générer le code assembleur associé au code C.
8. Pour comparer l'égalité de deux nombres flottants `x1` et `x2`, il est préférable d'utiliser la comparaison suivante: `if( x1-x2<epsilon )`, avec `epsilon` initialisé à un nombre petit tel que 10^{-6}.
9. Dans un grand code déjà existant, il est généralement préférable de modifier l'interface des fonctions que leurs implémentations.
10. Soit une chaîne de caractères contenue dans un tableau de 4 cases (`char T[4]`). Les 4 cases contiennent respectivement les lettres suivantes: `abcd`. L'affichage de ce tableau par la commande `printf(``%s'',T)` affiche de manière certaine `abcd` et uniquement cela.
11. La mémoire dite de `cache` permet de rendre inaccessible certaine variables au programme qui ne peut alors plus accéder à leurs valeurs respectives. On dit que cette mémoire vient les *cacher* ou encore les *dissimuler*.

3 Structs, adressage et portée des variables en C

[20min max] 2 points

Soit les structures, fonctions et code suivants (pour information, le programme compile et 5 lignes sont affichées à son exécution).

```c
struct bouteille
{
  int volume;
  int prix;
};
struct collection
{
  struct bouteille B[2];
};

void fonction_1(struct collection c)
{
  c.B[0].prix=20;
}
void fonction_2(struct collection *c)
{
  struct bouteille b=c->B[1];
  b.volume=4;
}
void fonction_3(struct collection *c)
{
  c->B[1].prix=6;
}
void fonction_4(struct collection c)
{
  struct bouteille *b=&(c.B[1]);
  b->prix=9;
}
```

```c
int main()
{
  struct collection c0;
  struct collection c1;

  c0.B[0].volume=1;
  c1=c0;
  c1.B[0].volume=2;
  printf("%d\n",c0.B[0].volume);

  c0.B[0].prix=10;
  fonction_1(c0);
  printf("%d\n",c0.B[0].prix);

  c0.B[1].volume=3;
  fonction_2(&c0);
  printf("%d\n",c0.B[1].volume);

  c0.B[1].prix=5;
  fonction_3(&c0);
  printf("%d\n",c0.B[1].prix);

  c0.B[1].prix=8;
  fonction_4(c0);
  printf("%d\n",c0.B[1].prix);

  return 0;
}
```

Question 6 *Qu'affiche ce programme (écrivez votre réponse sur votre copie double)? Expliquez rapidement votre raisonnement pour chaque affichage (remarque: il est inutile de passer du temps à recopier le code du programme).*

4 Utilisation du mot clé const

[20min max] 2 points

Soit la structure et les 8 fonctions suivantes appelées depuis un programme (qui n'est pas décrit).

```
1  struct fleur_rose
2  {
3    int prix;
4    int avoir_piquants;
5  };
6  struct fleur_jasmin
7  {
8    int prix;
9    int nbr_petales;
10 };
11 struct bouquet
12 {
13   struct fleur_rose rose;
14   struct fleur_jasmin jasmin;
15 };
16
17 void initialize(struct bouquet* b)
18 {
19   b->rose.prix=2;
20   b->rose.avoir_piquants=0;
21   b->jasmin.prix=3;
22   b->jasmin.nbr_petales=9;
23 }
24
25 void fonction_1()
26 {
27   struct bouquet b; initialize(&b);
28   const struct bouquet b2=b;
29 }
30
31 void fonction_2()
32 {
33   struct bouquet b; initialize(&b);
34   const struct bouquet b2=b;
35   const struct bouquet b3=b2;
36 }
37
38 void fonction_3()
39 {
40   struct bouquet *b; initialize(b);
41   struct bouquet b2=*b;
42 }
```

```
1
2
3
4
5
6
7
8  void fonction_4()
9  {
10   struct bouquet b; initialize(&b);
11   const struct fleur_rose *r=&(b.
       rose);
12 }
13
14 void fonction_5()
15 {
16   struct bouquet b; initialize(&b);
17   const struct bouquet *b2=&b;
18   const struct fleur_rose *r=&(b2->
       rose);
19 }
20
21 void fonction_6()
22 {
23   struct bouquet b; initialize(&b);
24   const struct bouquet b2=b;
25   struct fleur_rose *r=&(b2.rose);
26 }
27
28 void fonction_7()
29 {
30   struct bouquet b; initialize(&b);
31   const struct bouquet *b2=&b;
32   struct fleur_rose *r=&(b2->rose);
33 }
34
35 void fonction_8()
36 {
37   struct bouquet b; initialize(&b);
38   const struct bouquet *b2=&b;
39   struct fleur_rose r=b2->rose;
40 }
```

Question 7 *Pour chacune des 8 fonctions, indiquez si celle-ci est correcte: c'est à dire ne présentant aucun warning ni erreur à la compilation et à l'exécution (Écrivez votre réponse sur votre copie. Il est inutile de passer du temps à recopier tout le code du programme).*
Si la fonction est incorrecte, expliquez pourquoi.

5 Bonnes pratiques de programmation et fichiers

[30min max] 4 points

Vous utilisez un code écrit en C qui gère les employés d'une entreprise. La structure entreprise contient au maximum `NMAX` employés.

Ce code contient la fonction suivante dans son API:

```
int calculer_cout_total_salaire(const struct employes T[NMAX]);
```

Question 8 *Que pensez vous de ce prototype de fonction? Vous semble t-il satisfaire aux contraintes de bonnes pratiques? Si non, quel serait votre proposition de changement?*

Un employé de cette entreprise est enregistré en mémoire de manière à ce que l'on puisse stocker les informations suivantes:
Son nom, age, fonction, salaire, montant de ses primes, ancienneté dans la société, numéro de sécurité sociale, nom de son superieur direct, nom des personnes sous sa responsabilité (maximum 8 personnes), nombre d'arrêts maladie dans l'année, nombre de jours de congés restants, sa photo, la couleur de ses yeux (marron, vert, ou bleu).
On suppose que la photo est stockée dans un fichier présent sur le disque dur.

On souhaite définir une ou plusieurs struct permettant de stocker et manipuler aisément l'enregistrement d'un employé.

Question 9 *Écrivez la définition de la (ou les) struct(s) C que vous mettriez en place afin de réaliser cet enregistrement. Faites en sorte de bien respecter les bonnes pratiques de programmation.*

Question 10 *Quelle fonction souhaiteriez vous écrire en lien avec cette struct sans même avoir à connaitre l'utilisation de cet enregistrement dans le code ? Ecrivez le prototype de cette fonction (inutile d'écrire l'implémentation de celle(s)-ci). Notez que vous n'êtes pas restreint à une seule fonction: Il est possible d'en définir plusieurs.*

Vous souhaitez désormais écrire une fonction qui permet d'écrire dans un fichier texte sur le disque dur des informations d'identification d'une personne et de sa position dans l'entreprise. Voici un exemple du contenu d'un tel fichier pour une personne ayant sous sa responsabilité 3 autres employés:

```
nom: Holtz Francois
fonction: responsable
age: 42
superieur: Botlz Michel
responsable de :
  - Heitz René
  - Ruts Boris
  - Vickers Rolf
```

La fonction d'écriture doit prendre comme paramètre le nom du fichier dans lequel elle doit écrire.

Question 11 *En prenant particulièrement soin de respecter les critères de bonnes pratiques vues en cours et en projet, écrivez la signature et l'implémentation correspondante de la fonction réalisant cette écriture dans un fichier pour une personne quelconque de l'entreprise. N'oubliez pas de donner un contrat à votre fonction. Réfléchissez aux paramètres de la fonction. Notez que vous avez le droit de faire des suppositions sur des valeurs standards des paramètres. Indiquez clairement les hypothèses que vous prenez.*

6 Compilation, Makefile et préprocesseur

[25min max] 3 points

On suppose que l'on dispose dans un même répertoire de 5 fichiers: `Makefile`, `pgm_2.c`, `pgm_2.h`, `pgm_3.c`, `pgm_4.c` dont les contenus sont décrits ci-dessous.

`pgm_2.h`

```
#ifndef PGM_2_H
#define PGM_2_H

#ifdef A
#define VALEUR 6
#else
#define VALEUR 9
#endif

void fonction_2();

#endif
```

`pgm_3.c`

```
#include <stdio.h>

int main()
{
  printf("Bonjour\n");
  return 0;
}
```

`pgm_2.c`

```
#include "pgm_2.h"

#include <stdio.h>

void fonction_2()
{
  printf("%d\n",VALEUR);
}
```

`pgm_4.c`

```
#define A
#include "pgm_2.h"

#include <stdio.h>

int main()
{
  fonction_2();
  printf("%d\n",VALEUR);
  return 0;
}
```

`Makefile`

```
CFLAGS=-Wall -Wextra -g
all: pgm_1

pgm_1: pgm_1.o pgm_2.o
pgm_1.o: pgm_1.c pgm_2.h
pgm_2.o: pgm_2.c pgm_2.h
pgm_1.c: pgm_4.c
        cp pgm_4.c pgm_1.c
clean:
        rm -f *.o pgm_1 pgm_1.c pgm_3 *~
```

On suppose que l'on dispose d'une ligne de commande dans le répertoire où se situe l'ensemble des fichiers.

Question 12 *On tape* `make pgm_3`. *Que se passe t-il? Quels sont les fichiers désormais présents dans le répertoire après exécution de cette commande ?*

Question 13 *On tape* `make`. *Que se passe t-il ? Quels sont les fichiers désormais présents dans le répertoire après exécution de cette commande ?*

Question 14 *On tape* `./pgm_1`. *Que se passe t-il ? Que voit-on affiché sur la ligne de commande ?*