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

2012-2013 (2eme session)
durée 2h

Tous documents et calculatrices autorisés.

Le sujet comporte 6 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 32 ou 64 bits (identiques aux conditions de TP des PC de CPE: *int* étant encodé sur 4 octets, pointeurs étant encodés respectivement sur 4(/8) octets sur 32(/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 Concepts de programmation

1.1 Généralités

[20min max] 3 points

Question 1 *Pour les questions suivantes, répondez par vrai ou faux, puis justifiez en une ligne ou deux votre réponse:*

1. *Un programme* **portable** *est un programme qui est très léger sur le disque dur.*
2. Makefile *est un synonyme de* gcc.
3. *La* GPL *est une licence logiciel libre.*
4. *Dans le langage C, il existe une fonction permettant d'indiquer si un pointeur non NULL contient l'adresse d'une case valide ou non.*
5. *Fermer un descripteur de fichier (*fid*) avec la commande suivante:*
   assert(fclose(fid)==0);
   est une bonne idée.

1.2 Méthode de debug

[15min max] 3 points

Question 2

Quelles démarches pouvez-vous réaliser pour débugger votre code lorsque votre programme s'interrompt en indiquant une erreur de segmentation (ou Seg Fault*). Donnez un maximum de détails sur les procédures à suivre.*

2 Manipulation et structuration de code C

Dans l'ensemble des questions de l'exercice, on prendra soin de bien se placer dans le cadre d'un codage de type **défensif**.

Soit les structures de données définissant l'astraction d'une *forêt* constituée elle même d'un ensemble d'*abres*, formés eux même d'un *tronc* et d'un type de *feuille* qui sont eux même définis par des caractéristiques propres.

Le code de ces structures est le suivant:

```
//une structure de tronc d'arbre
struct structure_tronc
{
  float hauteur; //hauteur du tronc
  float rayon;   //rayon du tronc
};

//differentes couleurs de feuilles
enum type_couleur {vert_clair,vert_sombre,vert_rouge,rouge};

//une structure de donnees de feuilles
struct structure_feuille
{
  enum type_couleur couleur; //couleur de la feuille
  int nombre_de_veinure;     //nombre de veinure de la feuille
};

//une structre d'une entitee arbre
struct structure_arbre
{
  struct structure_tronc tronc;
  struct structure_feuille feuille;
};

//une structure modelisant une foret comme un
// ensemble d'arbres (ici stocke en tableau)
#define N_arbre 50
struct structure_foret
{
  struct structure_arbre arbre[N_arbre];
};
```

La fonction principale *main()* consiste dans un premier temps à déclarer une variable de type *structure foret*.

Voici le code correspondant:

```
int main()
{
  struct structure_foret foret;
  return 0;
}
```

2.1 Mémoire

[5min max] 1 points

Question 3 *Dans la fonction* `main()`*, l'espace mémoire permettant de stocker le contenu de la variable* `foret` *est-il alloué sur la pile ou sur le tas?*

- *Si vous avez répondu* pile *: Quelle commande aurait permis d'allouer ce contenu sur le tas?*
- *Si vous avez répondu* tas *: Quelle commande aurait permis d'allouer ce contenu sur la pile?*

2.2 Initialisation

[20min max] 3 points

Par défaut, un arbre doit posséder les propriétés suivantes:

- hauteur de tronc de 2 (m).
- rayon de tronc de 0.5 (m).
- couleur de feuille *vert clair*.
- nombre de veinures de feuille à 3.

La (nouvelle) fonction *main()* permettant d'initialiser l'ensemble de la forêt à des valeurs par défauts est désormais la suivante:

```
int main()
{
  struct structure_foret foret;
  foret_init(&foret);
  return 0;
}
```

Chaque entité est initialisée spécifiquement par sa fonction d'initialisation propre. On définit ainsi les 4 fonctions suivantes:

```
  void foret_init(struct structure_foret* foret);
  void arbre_init(struct structure_arbre* arbre);
  void tronc_init(struct structure_tronc* tronc);
  void feuille_init(struct structure_feuille* feuille);
```

Question 4 *Ecrivez le corps des 4 fonctions d'initialisations. Notez que la fonction* `foret_init()` *devra faire appel à* `arbre_init()`*, qui devra lui même faire appel à* `tronc_init()` *et* `feuille_init()`*.*

2.3 Manipulation d'adresse

[20min max] 3 points

On propose la fonction `arbre_recupere()` qui prend en entrée un **pointeur constant non NULL** sur une `structure_foret` ainsi qu'un nombre entier `k_arbre` compris entre 0 et `N_arbre`.
Cette fonction doit retourner une variable de type pointeur dont la valeur est égale à l'adresse de l'arbre numéro `k_arbre` (dans le tableau d'abres de la structure forêt).

On vous propose la signature suivante:

```
struct structure_arbre* arbre_recupere(const struct structure_foret*
    foret,int k_arbre);
```

Question 5 *Expliquez pourquoi l'implémentation de cette fonction avec cette signature ne pourra pas compiler? Que faut-il changer dans la signature pour avoir une chance de pouvoir mettre en place la fonctionnalité demandée.*

Question 6 *Ecrivez l'implémentation de la fonction* `arbre_recupere` *avec la signature corrigée.*

2.4 Bonne pratique et algorithmie

[35min max] 5 points

On cherche à écrire une fonction `arbre_le_plus_haut` qui prend en entrée un pointeur constant sur une `structure_foret` et retourne l'adresse de l'arbre dont la taille est maximale.

Question 7 *Ecrivez la signature de cette fonction.*

Question 8 *Ecrivez le contract de cette fonction.*

Question 9 *Ecrivez l'implémentation de cette fonction.*

2.5 Variables

[5min max] 2 points

On cherche à implémenter une fonction `arbre_ecrit_hauteur` dont la signature est la suivante:

```
void arbre_ecrit_hauteur(struct structure_arbre* arbre,float hauteur);
```

qui viens assigner la valeur `hauteur` au tronc de l'arbre désigné.

Deux implémentations de cette fonction vous sont proposées:

```
void arbre_ecrit_hauteur(struct structure_arbre* arbre,
                                          float hauteur)
{
  arbre->tronc.hauteur=hauteur;
}
```

```
void arbre_ecrit_hauteur(struct structure_arbre* arbre,
                                          float hauteur)
{
  struct structure_tronc tronc=arbre->tronc;
  tronc.hauteur=hauteur;
}
```

Question 10 *Il y a t-il une différence de résultat entre ces deux implémentations? Justifiez votre réponse.*