Durée 3 h Documents autorisés (polycopiés de cours uniquement)

Sujet d'examen 1

Pratique du C

Février 2012

Introduction

Écrivez lisiblement et n'hésitez pas à commenter votre code en langage C. Vous ne pouvez utiliser que les fonctions C dont le prototype est donné dans l'énoncé et celles dont vous donnez la définition dans vos copies.

Les sections sont indépendantes ; lisez l'énoncé complet avant de commencer à le résoudre.

1 Quizz

1. Quel est le contenu du tableau `tab` défini ci-dessous après l'exécution des instructions suivantes :

```
#define N 10
int tab[N] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
int *p = &tab[0], *q = &tab[N-1], temp;
while (p < q) {
   temp = *p;
   *(p++) = *q;
   *(q--) = temp;
}
```

2. Traduisez en français les déclarations suivantes :

```
double **average;
int *a[10];
int *(*sqrt[5])(int *);
struct { int a, int *(*b)(char) } foo;
char *k(int);
int (*j)[10];
```

   Par exemple, la déclaration `int *x;` peut se traduire par "x est une variable, un pointeur pointant sur un entier".

3. Réécrivez ces déclarations en utilisant `typedef` et en décomposant les déclarations complexe en les plus simples possibles.

2 Matrice creuse

La taille d'une matrice carrée creuse n'est pas connue à la compilation mais seulement lors de sa création. Les coefficients de cette matrice sont des entiers machines signés.

On représente une matrice carrée creuse sous la forme d'une liste chaînée dont chaque élément correspond à une ligne de la matrice et contient comme information l'indice de ligne et la représentation d'une ligne. Cette liste est triée suivant la valeur de l'indice de ligne.

Une ligne est représentée par une liste chaînée stockant les éléments non nuls de cette ligne. Un élément d'une ligne contient comme information le couple (indice de colonne, coefficient). Chaque liste est triée suivant la valeur de l'indice de colonne.

Questions.

1. Donnez la déclaration des types `ligne_t` et `element_t` correspondant à une ligne et à un élément de la liste chaînée représentant une ligne.
2. Donnez la déclaration du type `MatriceCreuse_t` représentant une matrice carrée creuse comme décrit ci-dessus.
3. Donnez la définition d'une fonction de prototype `void freemat(MatriceCreuse_t)` qui libère l'espace mémoire associé à la matrice creuse passée en paramètre.
4. Donnez la définition d'une fonction `void multscal(int a, MatriceCreuse_t M)` qui réalise la multiplication de `M` par le scalaire `a`. Les éléments de `M` sont directement modifiés par cette procédure. La multiplication d'une matrice par un scalaire revient à multiplier chaque élément de la matrice par ce scalaire.
5. Donnez la définition d'une fonction `MatriceCreuse_t convert(int **M, int n)` qui convertit une matrice carrée pleine `M` de taille `n` et codée par un tableau bidimentionnel en une matrice creuse et renvoie le résultat. Cette fonction réalisera toutes les allocations dynamiques nécessaires.

Indication : Vous pouvez utiliser les fonctions classiques d'allocation et de désallocation :

```
void * malloc(int);
void free(void *) ;
```

3 Une implantation sommaire de la fonction scanf

Dans cette section, on se propose d'implanter la fonction `scanf` dans une architecture dans laquelle le passage des paramètres se fait par une pile (cf. fin de la section pour une description de l'architecture).

Par ailleurs, on suppose ne disposer que d'une seule fonction externe dont le prototype est `int getchar(void);` et qui lit depuis l'entrée standard un caractère dont le code ASCII est retourné après conversion en un entier ; en cas d'erreur ou si l'entrée est un fichier dont on a atteint la fin, cette fonction retourne l'entier défini par la macro `EOF`.

Par exemple, pour extraire de l'entrée standard un entier et le stocker dans une variable de type entier, on peut utiliser le code suivant :

```
#include<stdio.h>

int main(){
  unsigned char c ;
  c = (unsigned char) getchar() ;
  return 0 ;
}
```

L'entrée standard est considérée comme une suite fini ou vide de n octets $o_1, \ldots, o_n$; si elle est vide la fonction `getchar` attend qu'un octet lui soit transmis, sinon après un appel à `getchar`, l'entrée standard est constituée de la suite fini de $n-1$ octets $o_2, \ldots, o_n$.

Objectif. L'objectif est d'implanter la fonction de prototype : `int mscanf(const char *format, ...)` permettant de saisir depuis l'entrée standard :

- des caractères ASCII classique de type `char`.
- des entiers machines positifs dans les bases décimale et binaire.

Cette fonction retourne 0 en cas de problème (mauvaise directive ou problème lors de la saisie i.e. le codage de la suite de caractères de l'entrée standard ne correspond pas aux conventions, cf. la suite) et 1 sinon.

Caractéristique de la fonction mscanf. Cette fonction a un paramètre obligatoire et un nombre variable de paramètres complémentaires.

Le paramètre obligatoire est constitué par une chaîne de caractères composée d'une ou plusieurs directives. Une directive commence par le caractère % et peut être de plusieurs formes :

- la directive %c indique que l'on souhaite saisir un caractère ;
- la directive %s indique que l'on souhaite saisir une chaîne de caractères ;
- la directive %d indique que l'on souhaite saisir un entier positif en base décimale ;
- la directive %b indique que l'on souhaite saisir un entier positif en base binaire.

Convention pour la saisie par mscanf. On convient qu' :

- un entier se termine par tout caractère non numérique ;
- une chaîne de caractères (de type `char *`) se termine par un caractère de code 0 ou `EOF`.

Questions

1. De quels types peuvent être les paramètres complémentaires de la fonction `mscanf` ?
2. Donnez la définition d'une fonction `int ScanString(char *s,unsigned int stringsize)` qui saisit depuis l'entrée standard une chaîne de caractères d'au plus `stringsize` caractères et les places dans l'espace mémoire pointé par `s` et retourne 1 en cas de problème et 0 sinon. On convient que :
   - les opérations d'allocation et de désallocation de `s` incombent à l'utilisateur et non pas à la fonction `ScanString`.

- la saisie d'une chaîne de caractères s'interrompt par la saisie du caractère de code ASCII 0 ou `EOF` (ce dernier est remplacé par 0 dans la chaîne de caractères).
- un problème survient lorsque la fonction saisie `stringsize` caractères sans rencontrer de caractère de code ASCII 0 ou `EOF` et dans ce cas le dernier caractère de l'espace mémoire associé à `s` doit être de code ASCII 0.

3. Donnez la définition d'une fonction `int ScanInt(int b, unsigned int *res)` qui saisit depuis l'entrée sortie standard une chaîne de caractères représentant un entier positif exprimé dans la base $b \in \{2, 10\}$ et le transmet par l'intermédiaire de son second paramètre `res`. Cette fonction arrête la saisie au premier caractère de l'entrée standard n'étant pas un chiffre de la base `b` ; ce caractère est converti en entier et retourné.
4. Donnez la définition de la fonction `mscanf`.

Modèle de passage de paramètres aux fonctions. Rappelons que les paramètres d'une fonction sont passés par la pile. Cette pile est composée de cellules dont la taille en octet correspond au type `int`, elle croit vers les adresses décroissantes et elle a la structure suivante :

0000	$\cdots$
	seconde variable locale
	première variable locale
	ancien pointeur de contexte
	adresse de retour
	premier paramètre
	second paramètre
FFFF	$\cdots$

Ainsi si `int foo` est la première variable locale automatique définie dans la fonction appelée et que `ptr` est un pointeur sur la cellule correspondante, `ptr+1` pointe sur l'ancien pointeur de contexte.

De plus, lors de l'appel d'une fonction, ses paramètres sont empilés du dernier au premier et on note qu'un pointeur occupe une cellule dans tous les cas.