Examen – CORRECTION - semestre 1
Initiation à la programmation 2013/2014
Électronique et Physique appliquée
Matériaux et chimie

Tous documents
non électroniques autorisés
durée 1h30

Nom, prénom :

Spécialité :

- **Répondre sur le document.**
- Écrire vos programmes en langage C en respectant la syntaxe et l'indentation usuelle ;
- Soignez votre présentation ;
- Les questions sont relativement indépendantes ;
- Chaque question est notée sur 2 points.

INDICATIONS A LA FIN

1. Analyse de code

Qu'affiche le programme ci-dessous ?

```
#include<stdio.h>

int main (void) {
    int a  ;
    int tab[5] ;
    int *p ;
    for (a=0 ; a<5 ; a++ )
        tab[a]=a ;
    p = tab   ;
    *p = 1 ;
    for (a=0 ; a<5 ; a++ )
        printf ("tab[%d] = %d ; ", a, tab[a] ) ;
    p = &a ;
    *p = 0 ;
    printf ("\n a = %d, *p= %d\n", a, *p) ;
}
```

tab[0] = 1 ; tab[1] = 1 ; tab[2] = 2 ; tab[3] = 3 ; tab[4] = 4 ;

a = 0, *p= 0

2. Tri

Soit le programme principal suivant :

```
#include<stdio.h>
int main (void) {
    int a=7, b=4, c=5 ;
    trier3Entiers (&a, &b, &c) ;
    afficher3entiers(a, b, c) ;
}
```

2.1 Écrire la fonction `void afficher3entiers(int a, int b, int c)` qui affiche à l'écran les valeurs des 3 nombres a, b, c.

```
void afficher3entiers (int a, int b, int c) {
    printf ("a = %d, b = %d, c = %d \n", a, b, c) ;
}
```

2.2 Écrire la fonction `void trier3Entiers(int *pa, int *pb, int *pc)` qui trie 3 nombres stockés dans trois variables dont les adresses *pa, pb* et *pc* sont passées en paramètre. A la fin de la fonction, la variable pointée par *pa* contiendra le plus petit et la variable pointée par pc contiendra le plus grand.

```
// on regarde a et b qu'on swap si besoin
    // on regarde b et c qu'on swap si besoin
    // on regarde de nouveau a et b qu'on swap si besoin
    int tmp ;
    if (*b < *a) {
        tmp = *a ;
        *a = *b ;
        *b = tmp ;
    }
    if (*c < *b) {
        tmp = *b ;
        *b = *c ;
        *c = tmp ;
    }
    if (*b < *a) {
        tmp = *a ;
        *a = *b ;
        *b = tmp ;
    }
}
```

2.3 Écrire la fonction `demander3Entiers( )` qui demande 3 entiers à l'utilisateur et qui retourne ces 3 nombres dans trois variables dont les adresses sont passées en paramètre. A vous de trouver le prototype adéquat.

```
void  demander3Entiers  (int *a, int *b, int *c) {
    printf ("a = ") ;
    scanf ("%d", a) ;
    printf ("b = ") ;
    scanf ("%d", b) ;
    printf ("c = ") ;
    scanf ("%d", c) ;
}
```

2.4 Écrire la fonction `main( )` permettant de tester la fonction `demander3Entiers( )`.

```
int main (void) {
    int a=7, b=4, c=5 ;
    demander3Entiers( &a, &b, &c );
    trier3Entiers (&a, &b, &c) ;
    afficher3entiers(a, b, c) ;
}
```

3. tableaux

Soit le programme principal suivant :

```
#include <stdio.h>
#define N 3 // lignes
#define M 2 // colonnes

int main (void) {
    int t1[N][M] = {{2, 3}, {2, 4}, {2, 5} } ;
    int t2[M] ;
    int i;
    calculColonne(t1, t2, N, M) ;
    for (i=0 ; i<M ; i++ )
    printf("%d ", t2[i]);
}
```

Et voici une trace d'exécution :

```
6 12
```

Écrire la fonction `void calculColonne(int t1[][M], int t2[], int n, int m)` qui calcul les sommes des colonnes du tableau t1 et les stocke dans le tableau t2. Le tableau t1 est un tableau de « n » lignes et « m » colonnes. Le tableau t2 est un tableau de « m » cases.

```
void calculColonne(int t1[][M], int t2[], int n, int m) {
    int i, j ;
    for (i=0 ; i<m ; i++ ) {
        t2[i] = 0 ;
        for (j=0 ; j<n ; j++ )
            t2[i] = t2[i] + t1[j][i] ;
    }
}
```

4. Chaînes de caractères

Soit le programme principal suivant :

```
int main (void) {
    char phrase[]="Le langage C est un exemple de langage impératif" ;
    int nb ;
    char c = 'l' ;
    nb = nbMots (phrase, c) ;
    printf("Nb de mots commençant par %c : %d\n", c, nb ) ;
}
```

Et voici une trace d'exécution :

```
Nb de mots commençant par l : 2
```

La fonction nbMots compte le nombre de mots commençant par un caractère particulier. Dans l'exemple ci-dessus, la phrase « `Le langage C est un exemple de langage impératif` » contient 2 fois le mot « langage » qui commence par '`l`'. Dans ce cas la fonction retournera 2. On ne tiendra compte que des mots séparés par un espace, et donc on ne tiendra pas compte des apostrophes comme dans « l'été ».

Écrire le code de la fonction nbMots. A vous d'écrire correctement le prototype.

```
int nbMots (char * ph, char c) {
    int cpt = 0 ;
    int i ;
    if (ph == NULL) return 0 ;
    if (ph[0] == '\0') return 0 ;
    if (ph[0] == c) cpt++ ;
    i=0 ;
    while (1)  {
        if (ph[i+1] == '\0') return cpt ;
        if (ph[i] == ' ' && ph[i+1] == c ) cpt++ ;
        i++ ;
    }
}
```

5. fichiers, structures et allocation mémoire

Soit le programme principal suivant :

```
typedef struct sTab {
    int nb ;
    int tab[] ;
} Vecteur ;

int main (void) {
    Vecteur v ;
    v = lireVecteurLigne ("fic.txt") ;
    afficheColonne (v) ;
}
```

Et voici une trace d'exécution :

```
11
12
13
14
15
16
17
18
```

Le fichier texte « fic.txt » contient 2 lignes. Sur la première ligne se trouve le nombre de valeurs entières de la deuxième ligne. Dans l'exemple, la deuxième ligne contient 8 valeurs entières.

```
  8
  11 12 13 14 15 16 17 18
```

Le type Vecteur contient un champ de type entier « nb » qui représente la dimension du vecteur, et un champ « tab » qui est un tableau contenant les valeurs du vecteur.

5.1 Écrire la fonction `Vecteur newVecteur(int n)` qui retourne un vecteur de dimension « n ». La fonction alloue l'espace mémoire nécessaire au champ « tab » du type Vecteur. Un tableau de « n » cases est ainsi créé.

```
Vecteur newVecteur(int n) {
    Vecteur v ;
    v.nb = n ;
    v.tab = (int *) calloc (n, sizeof(int)) ;
    return (v) ;
}
```

5.2 Écrire la fonction `void afficheColonne (Vecteur v)` qui affiche en colonne toutes les coordonnées du vecteur « v ».

```
void afficheColonne (Vecteur v) {
     int i ;
     for (i=0 ; i< v.nb ; i++ )
          printf ("%d \n", v.tab[i]) ;
}
```

5.3 Écrire la fonction `Vecteur lireVecteurLigne (char *nomFic)` qui retourne un vecteur dont les coordonnées auront été lues dans un fichier texte. Le fichier aura un format similaire à celui de « fic.txt ».

```
Vecteur lireVecteurLigne (char *nomFic ) {
     Vecteur v ;
     v.nb = 0 ;
     int i ;
     FILE *fp ;
     fp = fopen( nomFic, "r" ) ;
     if (fp == NULL ) {
          printf("Pb ouverture %s\n", nomFic) ;
          return(v) ; // retourne un Vecteur vide
     }
     fscanf (fp, "%d", &v.nb) ;
     v = newVecteur (v.nb) ;
     for (i=0 ; i< v.nb ; i++ )
          fscanf (fp, "%d", &(v.tab[i])) ;
     fclose (fp) ;
     return v ;
}
```