E3FR 3R-IN1A
Initiation à la programmation C
2022-2023

ESIEE PARIS

Examen 3R-IN1A

5 Décembre 2022 - durée 2 heures

DOCUMENTS ET TÉLÉPHONES INTERDITS

Remarques :
— Veuillez lire attentivement les questions avant de répondre.
— Le barème donné est un barème indicatif qui pourra évoluer lors de la correction.
— Rendez une copie propre et faites des réponses claires (servez vous de vos brouillons).
— Si vous ne savez pas, n'inventez pas ! Un malus de 25% sur le barème sera appliqué en cas de mauvaise réponse à l'une des questions à choix multiples(QCM).

Le sujet est composé de 4 pages

QCM et questions de cours (5 points)

QCM

1. Quelle est la sortie de ce programme ?

```c
#include <stdio.h>
int main(){
        int a = -3;
        int b = 7;
        int res = b++ - a--;
        printf( "%d - %d - %d", a, b,
            res );
        return 0;
}
```

Réponse A → - 4 ; 8 ; 4
Réponse B → - 4 ; 8 ; 12
Réponse C → - 4 ; 8 ; 10
Réponse D → - 4 ; 8 ; 11

2. Quelle est la sortie de ce programme ?

```c
#include <stdio.h>
int main(){
double result = 2/3;
printf( "%f\n", result );
        return 0;
}
```

Réponse A → 0.000000
Réponse B → 0.666666
Réponse C → 2/3
Réponse C → Produit une erreur à l'exécution

3. Laquelle de ces lignes crée un tableau de 8 `float` ?
Réponse A → `float* tableau[8];`
Réponse B → `float tableau[8];`
Réponse C → `float tableau{8};`

4. Quelle est l'autre façon d'initialiser mon tableau avec ces valeurs ?

```c
int tableau[4];
tableau[0] = 10;
tableau[1] = 23;
tableau[2] = 505;
tableau[3] = 8;
```

Réponse A → `int tableau[4] = 10, 23, 505, 8;`
Réponse B → `int tableau[4] = {10, 23, 505, 8};`
Réponse C → `int tableau[4] = [10, 23, 505, 8];`
Réponse D → `int tableau[4] = (10, 23, 505, 8);`

5. Lequel de ces prototypes de fonction ne permet pas de faire passer un tableau en paramètre ?
   Réponse A → `void fonction(float tableau[], int taille);`
   Réponse B → `void fonction(float *tableau, int taille);`
   Réponse C → `void fonction(float tableau, int taille);`

6. On suppose que `nombre` se trouve à l'adresse 4000, et `pointeur` à l'adresse 5000. Quelle sera la valeur affichée après la ligne 5.

```
1 #include <stdio.h>
2 int main (){
3       int nombre = 3;
4       int *pointeur = &nombre;
5       printf("%d \n", *pointeur);
6       return 0;
7 }
```

   Réponse A → 3
   Réponse B → 5000
   Réponse C → 4000
   Réponse D → Impossible à prédire

7. On définit les constantes et les variables suivantes. Lesquelles de ces déclarations sont justes :

```
#define A 5.6
#define B 5
#define C 7
int d;
int f = 7 ;
```

   Réponse A → `int T1[10][B];`
   Réponse D → `int T1[1][d];`
   Réponse B → `int T1[A][B];`
   Réponse C → `float T1[C][B];`
   Réponse E → `int T1[C][f];`

8. Identifier les 3 erreurs de ce programme. Proposer une correction pour chacune des erreurs

```
1 #include <stdio.c>
2 int main() {
3       int m; n = 5;
4       printf("Saisir un entier  ? ");
5       scanf("%d", m);
6       printf("somme = %d\n", m+n);
7       return 0;
```

Exercices (15 points)

Exercice 1. Boucles et récursivité (5 points)

On appelle $pgcd(a, b)$, le plus grand nombre qui divise à la fois a et b. Dans cet exercice, on utilisera l'algorithme d'Euclide pour calculer le PGCD.

Principe : On divise par le plus petit des deux nombres jusqu'à obtenir un reste nul.
Exemple : pgcd(60,36) = 12 et pgcd (561,357) = 51

60 % 36	=	24	561 % 357	=	204
36 % 24	=	12	357 % 204	=	153
24 % 12	=	0	204 % 153	=	51
			153 % 51	=	0

Dans cet exercice on vous demande d'écrire plusieurs versions de la fonction *pgcd()* qui calcule et renvoie le PGCD de deux nombres. Cette fonction prendra en paramètre deux nombres entiers et renvoie un nombre entier.

1. (a) Écrire la fonction *pgcd()* à l'aide d'une boucle *while*.
   Indication : Supposons que l'on veuille calculer pgcd(A,B). On pose a := maximum(A,B) et b := minimum(A,B), puis on va répéter les opérations suivantes tant que $a \,\%\, b \neq 0$

$$\{c := a \,\%\, b \ ; \ a := b \ \ b := c \ ; \ \}.$$

(b) Écrire une version récursive (c'est à dire sans boucle *while, for, do while*).

Exercice 2. Tableaux (5 points)

```c
#include <stdio.h>
#include <stdlib.h>
#define TAILLE 10

... affiche_tab ( ... , ...);
... plus_grd_ecart(..., ...);
... nb_occurence (..., ... , ...);

int main(){
        int tab[TAILLE] = {10,-4,2,4,5,7};
        int n = 6, m = -4;
        affiche(tab,n);
        printf("Le plus grand ecart est %d\n", plus_grd_ecart(tab,n));
        printf("Le nb %d apparait %d fois \n", m, nb_occurence(tab,n,m));
        return 0;

}
```

1. Écrire la procédure *affiche_tab*( ) qui affiche le contenu d'un tableau d'entier.
   Indication : la procédure prendra deux paramètres en entrée : un tableau d'entier, et un nombre entier (le nombre d'éléments du tableau à afficher).
2. Écrire la fonction *plus_grd_ecart*( ) qui calcule et renvoie le plus grand écart entre deux éléments consécutifs du tableau. L'écart est la valeur absolue de la différence de deux éléments du tableau. On pourra utiliser la fonction *abs()* pour calculer la valeur absolue d'un nombre.
   Indication : la fonction prendra deux paramètres en entrée : un tableau d'entier, et un nombre entier (le nombre d'éléments du tableau à afficher) et un paramètre de sortie.
3. Écrire la fonction *nb_occurence*() qui calcule le nombre d'occurrence du nombre x dans le tableau.
   Indication : la fonction prendra trois paramètres en entrée : un tableau d'entier et deux nombres entiers (le nombre d'éléments du tableau à parcourir et la variable x).

Exercice 3. Tableaux et structures (5 points)

```c
#include <stdio.h>
#include <stdlib.h>
#define NB_PATIENT_MAX 100

.... patient;

patient saisir_patient();
.... calcul_IMC(....);
.... calcul_IMG(....);
.... calcul_IMC_moyen(...., ...., ....);
.... calcul_stat_IMC(...., ...., ...., ....);

int main(){
    int i,nb_patient;
    patient patient_i;
    float tab_IMC[NB_PATIENT_MAX];
    float tab_IMG[NB_PATIENT_MAX];
    float IMG_moyen, percent_30, percent;
    printf("Saisir le nombre de patient participant a l'etude \n");
```

```
    scanf("%d", &nb_patient);

    for (i=0;i<nb_patient;i++)
    {
        patient_i = saisir_patient();
        tab_IMC[i] = calcul_imc(patient_i);
        tab_IMG[i] = calcul_img(patient_i);
    }
    IMG_moyen = calcul_img_moyen(tab_IMG,nb_patient);
    percent = 30;
    percent_30 = calcul_stat_imc(tab_IMC,nb_patient, percent);
    printf("Pourcentage de patient avec un IMC > %.2f = %.2f \n ",percent,
        percent_30);

}
patient saisir_patient(){
    patient p;
    printf("Saisir la taille (en m) du patient \n") ;
    scanf("%f",&p.taille);
    printf("Saisir le poids (en kg) du patient \n") ;
    scanf("%f",&p.poids);
    printf("Saisir l'age du patient \n") ;
    scanf("%f",&p.age);
    printf("Si le patient est une femme saisir 0 sinon 1 \n");
    scanf("%d",&p.sexe);
    return p;
}
.... calcul_imc(...){
   ....
}
.... calcul_img(...){
    ...
}
... calcul_img_moyen(...,  ...){
    ....
}
... calcul_stat_imc(...,  ..., ...){
 ....
}
```

1. Définir un type *patient* qui permet de stocker les informations d'individu participant à une étude. Pour chaque patient on souhaite accéder aux informations suivantes

age	(un nombre entier)
taille (en m)	(un nombre réel)
poids	(un nombre réel)
sexe	(un nombre entier)

2. Écrire la fonction *calcul_imc*() qui prend en paramètre une variable de type *patient* et renvoie l'indice IMC (Indice de Masse Corporelle) de l'individu.
   Pour rappel
   $$IMC = \frac{\text{poids}}{\text{taille}^2}$$
   (Indication : *l'indice IMC est un nombre réel*)
3. Écrire la fonction *calcul_img*() qui prend en paramètre une variable de type *patient* et renvoie l'indice IMG (Indice de Masse Grasse) de l'individu.
   Pour rappel
   $$IMG = (1.2 \times IMC) + (0.23 \times \text{age}) - (10.8 \times \text{sexe}) - 5.4$$
   (Indication : *l'indice IMG est un nombre réel*)

4. (a) Écrire la fonction *calcul_img_moyen*() qui calcule et renvoie l'indice IMG moyen de tous les patients.
   Indication : la fonction prend deux paramètres en entrée et un paramètre en sortie.

   (b) Écrire la fonction *calcul_stat_imc*() qui calcule le pourcentage d'individu avec un indice IMC supérieur à une valeur donnée v.
   Indication : la fonction prend trois paramètres en entrée et un paramètre de sortie. Les paramètres d'entrée sont un tableau de nombre réel, un nombre entier (le nombre de patient) et un nombre réel (la valeur v).