

Examen de Langage C

Durée : 1h30

Aucun document autorisé – Mobiles interdits

Notez que les affirmations (antécédents, conséquents, rôles, et invariants) dans vos codes C entreront pour partie dans la note finale.

- 1. Combien de mode(s) transmission des paramètres existe(nt) en C ? Cochez une seule réponse !

- ☐ un seul, la transmission par référence.
☐ un seul, la transmission par valeur.
☐ deux, la transmission par valeur et référence.
☐ zéro.

Question sur 1 pt Réponse : **un seul** mode de transmission **par valeur**.

- 2. Dans l'appel de procédure `lire(x)`, `x` est

- ☐ un paramètre *donné* ;
☐ un paramètre *résultat* ;
☐ un paramètre *donné* et *résultat*.
☐ zéro.

Question sur 1 pt Réponse : `x` est paramètre *résultat*.

- 3. Écrivez en C la fonction `toFahrenheit` qui convertit en degré *fahrenheit* un degré *celsius* passé en paramètre. Rappel : $F = \frac{9 \times C}{5} + 32$

Question sur 2 pts

```
/*  
 * antécédent : c un degré celsius  
 * rôle : renvoie la conversion en degré fahrenheit de c  
 */  
double toFahrenheit(const double c) {  
    return (9.0*c)/5.0+32.0;  
}
```

- 4. Écrivez en C une fonction `somme` qui lit sur l'entrée standard `n` entiers (`int`) et renvoie leur somme. Cette fonction possède un seul paramètre, le nombre `n` d'entiers.

Question sur 4 pts

```
/*  
 * antécédent : n>0  
 * rôle : lit n entiers sur l'E/S et renvoie leur somme  
 */  
int somme(const int n) {  
    assert(n>0);  
    int som=0;  
    for (int i=0; i<n; i++) {  
        int x;  
        scanf("%d", &x);  
        som+=x;  
        // ∀k ∈ [0,i], som = ∑k=0i xk  
    }  
    // ∀k ∈ [0,n[, som = ∑k=1i=n-1 xk  
    return som;  
}
```

- 5. Écrivez en C la procédure `init` qui prend en paramètre un tableau d'entiers (`int`) `t` et son nombre d'éléments `n` et qui initialise le tableau de façon aléatoire avec la fonction `rand`.

Question sur 2 pts

```
/*  
 * antécédent : n>0 et t tableau de n entiers  
 * conséquent : t est initialisé de façon aléatoire  
 */  
void init(const int n, int t[]) {  
    assert(n>0);  
    for (int i=0; i<n; i++)  
        t[i]=rand()  
    //  
}
```

- 6. Écrivez en C la fonction `nbPairs` qui prend en paramètre un tableau d'entiers (`int`) `t` et son nombre d'éléments `n` et qui renvoie le nombre d'entiers pairs contenus dans le tableau.

Question sur 2 pts

```
/*  
 * antécédent : n>0 et t tableau de n entiers  
 * rôle : renvoie le nb d'entiers pairs contenus dans t  
 */
```

```

/*
int nbPairs(const int n, const int t[]) {
    assert(n>0);
    int nbP = 0;
    for (int i=0; i<n; i++)
        if ((t[i]&1)==0)
            // t[i] est un entier pair
            nbP++;
    //
    return nbP;
}

```

- 7. En utilisant la procédure `init` et la fonction `nbPairs`, écrivez en C la fonction `main` qui :
- lit sur l'entrée standard un entier `n` (> 0);
 - déclare un tableau d'entiers `tab` de `n` éléments;
 - écrit sur la sortie standard le nombre d'entiers pairs contenus dans `t`.

Question sur 3 pts

```

int main(void) {
    int n;
    scanf("%d", &n);
    assert(n>0);

    int tab[n];
    init(n, tab);
    printf("%d\n", nbPairs(n, tab));

    return EXIT_SUCCESS;
}

```

- 8. Écrivez en C la procédure `initAléa` qui initialise de façon aléatoire une matrice `mat` $m \times n$ de réels (**double**). Le nombre de lignes `m` et de colonnes `n` sont passés en paramètres de la procédure. Vous utiliserez la fonction `drand48` vue en cours.

Question sur 3 pts

```

/*
* antécédent : m et n>0 et mat matrice m x n de double
* conséquent : mat est initialisée de façon aléatoire
*/
void initAléa(const int m, const int n, double mat[][n]) {
    assert(m>0 && n>0);
    for (int i=0; i<m; i++)
        for (int j=0; j<n; j++)
            mat[i][j]=drand48();
}

```

- 9. Soit une matrice carrée `mat` $n \times n$ d'entiers (**int**). Écrivez la fonction `sommeDiags` qui renvoie la

somme des entiers des deux diagonales de la matrice. **Attention** : vous ne devez accéder qu'**une seule fois** à chaque entier des deux diagonales de la matrice.

Question sur 3 pts

```

/*
* antécédent : n>0 mat matrice carrée n x n
* rôle : renvoie la somme des 2 diagonales de la matrice mat
*/
int sommeDiags(const int n, const int mat[][n]) {
    assert(n>0);
    int som = 0;
    // le 1ère diagonale
    for (int i=0; i<n; i++)
        som+=mat[i][i];
    // le 2ème diagonale
    int nDiv2 = n/2;
    for (int i=0; i<nDiv2; i++)
        som+=mat[i][n-i-1]+mat[n-i-1][i];
    //
    return som;
}

```