
Éléments d'informatique : partiel de fin de semestre

Durée : 3 heures.

Documents autorisés : Aucun.

Recommandations : Un barème vous est donné à titre indicatif, afin de vous permettre de gérer votre temps. Ne dépassez pas les temps indiqués.

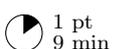
1 Étude de programmes et questions de cours

1.1 Somme des éléments d'un tableau (2,5 points)

Nous voulons écrire un programme qui calcule la somme des éléments d'un tableau d'entiers. Une partie du programme est déjà écrite, et Pippo pense qu'il ne reste plus qu'à écrire la partie qui calcule effectivement la somme. Voici son programme :

```
1  #include <stdlib.h> /* EXIT_SUCCESS */
2  #include <stdio.h> /* printf, scanf */
3
4  /* déclaration constantes et types utilisateurs */
5  #define TAILLE 3
6
7  /* Fonction principale */
8  int main()
9  {
10     int t[TAILLE] = {12,10,15}; /* tableau à sommer */
11
12
13
14     /* calcul de la somme (À FAIRE) */
15
16
17
18
19
20     /* affichage du résultat */
21     printf("La somme est %d\n", somme);
22
23     /* valeur fonction */
24     return EXIT_SUCCESS;
25 }
```

Question A. Avant d'aller plus loin Pippo veut tester son programme, mais la compilation échoue. Expliquer pourquoi, quelle étape de la compilation échoue précisément et ce qu'il manque pour que la compilation réussisse.



Question B. Compléter le programme pour qu'il calcule effectivement la somme des éléments du tableau (pour une taille de tableau arbitraire).

1.5 pt
13 min

Un exemple de sortie du programme pour le tableau donné dans le code est :

La somme est 37

1.2 Une erreur classique (1 points)

Pippo a écrit un programme C. Celui-ci compile, mais une erreur survient à l'exécution, qu'il ne comprend pas.

```
$ gcc puissance.c -o puissance.exe
$ puissance.exe
Entrer un nombre reel : 2.3
Entrer son exposant (entier positif) : 2
Segmentation fault
```

Question C. Expliquer brièvement ce que signifie ce message d'erreur (dernière ligne).

1 pt
9 min

2 Trace d'un programme avec fonctions (4 points)

Question D. Simulez l'exécution du programme figure 1 page 4, en réalisant sa **trace**, comme cela a été vu en TD et en cours.

4 pt
36 min

3 For ou while? (4,5 points)

Il est demandé de résoudre les questions suivantes sans définir de fonctions utilisateurs, directement dans le `main`, et en faisant le meilleur choix entre `for` et `while`.

Question E. Écrire un programme qui étant donné deux variables entières `largeur` et `hauteur`, initialisées à des valeurs de votre choix, affiche un rectangle d'étoiles de dimension `largeur` par `hauteur`. Dans l'exemple d'affichage suivant `largeur` vaut 6 et `hauteur` vaut 4.

1,5 pt
13 min

```
*****
*****
*****
*****
```

Question F. Expliquer comment modifier le programme précédent pour qu'il affiche le contour du rectangle avec des étoiles et l'intérieur avec des espaces.

1 pt
9 min

```
*****
*   *
*   *
*****
```

Question G. Écrire un programme qui, étant donné un tableau d'entier initialisé à des valeurs de votre choix, demande à l'utilisateur de saisir un entier puis si l'entier saisi est dans le tableau affiche son indice et sinon affiche à l'utilisateur `entier absent du tableau`.

2 pt
18 min

4 Fractions (5 points)

Question H. Une fraction $\frac{p}{q}$ est définie par deux entiers p et q . Le nombre q appelé dénominateur est nécessairement non nul et sera toujours positif, le nombre p , appelé numérateur, peut être négatif ou nul. Définir un type utilisateur pour les fractions (sans tenir compte des questions de signe qui n'ont d'importance que pour l'affichage).

1 pt
9 min

Question I. Déclarer et définir une fonction `multiplier_fractions` qui prend deux fractions en argument et renvoie la fraction obtenue par multiplication des deux fractions (ne pas chercher à simplifier la fraction obtenue). Répondre en faisant bien apparaître d'une part la déclaration, d'autre part la définition.

1.5 pt
13 min

Question J. Même question pour la somme de deux fractions `additionner_fractions`.

1 pt
9 min

Question K. Déclarer et définir une procédure affichant une fraction passée en argument exactement comme dans l'exemple suivant où les fractions $\frac{34}{26}$, $-\frac{34}{26}$, $\frac{34}{1}$, $\frac{0}{1}$ sont affichées tour à tour :

34/26

-34/26

34

0

Attention à bien respecter les deux derniers affichages.

1.5 pt
13 min

5 Fonctions (3 points)

Question L. Déclarer et définir :

1. une fonction `valeur_absolue` qui prend en entrée un argument réel et retourne sa valeur absolue ;
2. Une procédure `afficher_ligne` qui prend en entrée un entier `n` et un caractère `c` et affiche une ligne contenant `n` fois la caractère `c` ;
3. Une fonction `neper` qui prend en entrée un entier `n` et retourne la valeur de la somme suivante :

1 pt
9 min

1 pt
9 min

$$1 + \sum_{k=1}^{k=n} \frac{1}{k!}.$$

Vous pouvez faire appel à une fonction `int factorielle(int n)` calculant la factorielle de son argument.

1 pt
9 min

Bonus

Question bonus. Au choix. Déclarer et définir une fonction qui calcule le pgcd de deux entiers positifs ou nuls. Ou bien, déclarer et définir une procédure `afficher_disque` qui prend en paramètre un entier `rayon` et affiche un disque d'étoiles de ce rayon.

2 pt
18 min

```

1  #include <stdlib.h> /* EXIT_SUCCESS */
2  #include <stdio.h> /* printf, scanf */
3
4  /* declarations constantes et types utilisateurs */
5
6  /* declarations de fonctions utilisateurs */
7  int foo(int a, int b);
8  int bar(int n);
9
10 int main()
11 {
12     int x = 2;
13     int y = 6;
14     int res;
15     res = foo(x, y);
16     printf("foo(%d, %d) = %d\n", x, y, res);
17     res = bar(x);
18     printf("bar(%d) = %d\n", x, res);
19
20     return EXIT_SUCCESS;
21 }
22
23 /* definitions de fonctions utilisateurs */
24 int foo(int a, int b)
25 {
26     if (a < b)
27     {
28         return a;
29     }
30     return b;
31 }
32
33 int bar(int n)
34 {
35     if (n > 1)
36     {
37         return n * bar(n - 1);
38     }
39     return 1;
40 }

```

FIGURE 1 – Programme pour la trace