
Éléments d'informatique : partiel de mi-semestre

Durée : 3 heures.

Documents autorisés : Aucun.

Recommandations : Un barème vous est donné à titre indicatif afin de vous permettre de gérer votre temps. La notation prendra en compte à la fois la syntaxe et la sémantique de vos programmes, c'est-à-dire qu'ils doivent compiler correctement. Une fois votre programme écrit, il est recommandé de le faire tourner à la main sur un exemple pour s'assurer de sa correction.


1 Étude de programmes et questions de cours

1.1 Somme des éléments d'un tableau (*3,5 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.


 1 pt
9 min

Correction. Il manque la déclaration (et l’initialisation à zéro) de la variable `somme`. La compilation échoue à l’analyse sémantique.

Question B. Expliquer le fonctionnement de l’instruction `#define`.

 1 pt
9 min

Question C. 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

Correction.


1. 0.5 pt pour “déclaration de la variable `somme`” + 0.5 pt pour “analyse sémantique” ou 0.25 pt si “avant la génération du code objet” et/ou “avant l’édition de lien”.
2. 1 pt si juste. 0.5 pt si seulement notion de constantes et pas de mise en relation avec la substitution au moment de la compilation.
3. 1.5pt pour le code juste (compilation correction quel que soit `TAILLE`). Sinon, maxi 1 pt :
 - pas de boucle `-> 0`
 - 0.5pt une (et une seule) boucle (`for`, `while` accepté)
 - 0,5pt la boucle parcourt effectivement un tableau de taille `TAILLE`
 - pas de points en moins ou en plus pour la déclaration de la variable de boucle (redite sous une autre forme de la première question) ou pour l’initialisation de `somme` à zéro.

1.2 Une erreur classique (3 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 D. Expliquer brièvement ce que signifie ce message d’erreur (dernière ligne).

 1 pt
9 min

Correction. Le message d’erreur `Segmentation fault` signifie que le programme a tenté de lire ou d’écrire dans un espace mémoire qui ne lui était pas réservé, ce que le système a détecté et refusé, provoquant la terminaison prématurée du programme et l’affichage du message d’erreur.


Voici quelques lignes choisies du programme.

```

10  int main()
11  {
12      /* Déclaration et initialisation des variables */
13      double x;
14      int n;
15
16      :
17
22      printf("Entrer un nombre reel : ");
23      scanf("%lg", x);
24      printf("Entrer son exposant (entier positif) : ");
25      scanf("%d", n);

```

Question E. Que faut-il corriger ?

 1 pt
9 min

Correction. Il faut mettre une esperluette devant le nom de la variable dans les deux `scanf`. Comme ceci :


```

22      printf("Entrer un nombre reel : ");
23      scanf("%lg", &x); /* <-- ici */
24      printf("Entrer son exposant (entier positif) : ");
25      scanf("%d", &n); /* <-- ici */

```


Complément de correction. Le rôle de l'esperluette est de donner à `scanf` l'adresse de la variable dans laquelle écrire le résultat de la saisie utilisateur. Sans esperluette `scanf` récupère la valeur de la variable et l'utilise comme une adresse, ce qui provoque en général (si on a de la chance) une erreur de segmentation (**Segmentation fault**).

Question F. Quelle option de compilation aurait dû utiliser Pippo pour obtenir de l'aide ?

 0.5 pt
4 min

Correction. En utilisant l'option `-Wall` (afficher tous les avertissements) de la commande `gcc`, Pippo aurait eut à la compilation un message d'avertissement le prévenant d'un erreur probable à la ligne 23 et de même pour la ligne 25.


Question G. Compléter le programme pour que, lorsque l'entier n saisi est positif, il calcule et affiche x^n .

 1 pt
9 min


2 Unicité des éléments d'un tableau (4 points)

Nous disposons d'un tableau t de N entiers. Utiliser une constante symbolique pour N . Nous souhaitons savoir si chaque entier apparaissant dans le tableau n'y apparaît qu'une seule fois, autrement dit on veut savoir si chaque entier est unique.

Question H. Écrire un programme qui, étant donné un tableau initialisé t , teste si le premier élément du tableau est unique et affiche **Vrai** si c'est le cas, **Faux** sinon.

 2 pt
18 min

Question I. Écrire un programme qui étant donné un tableau initialisé t , teste si tous les éléments sont uniques et affiche **Vrai** si c'est le cas, **Faux** sinon.

 2 pt
18 min

Tout le traitement sera effectué dans le `main`, sans faire appel à des fonctions utilitaires.

3 Faut-il aller au cinéma ? (4 points)

Je viens de gagner une place de cinéma pour une séance ce soir. Je dispose des informations suivantes, codées dans des variables entières, pour décider si je vais me rendre à cette séance ou rester chez moi :

- **critique** une critique du film lui attribuant une appréciation parmi MAUVAIS, MOYEN, BON.
- **acteurs** mon appréciation personnelle du choix des acteurs : MOYEN ou BON.
- **distance** la distance approximative (en kilomètres) qui me sépare du cinéma.

Vous utiliserez des constantes symboliques pour coder les appréciations et vous initialiserez les trois variables, **critique**, **acteurs**, **distance** à des valeurs de votre choix. Pour prendre ma décision je dispose de l'arbre de décision suivant.

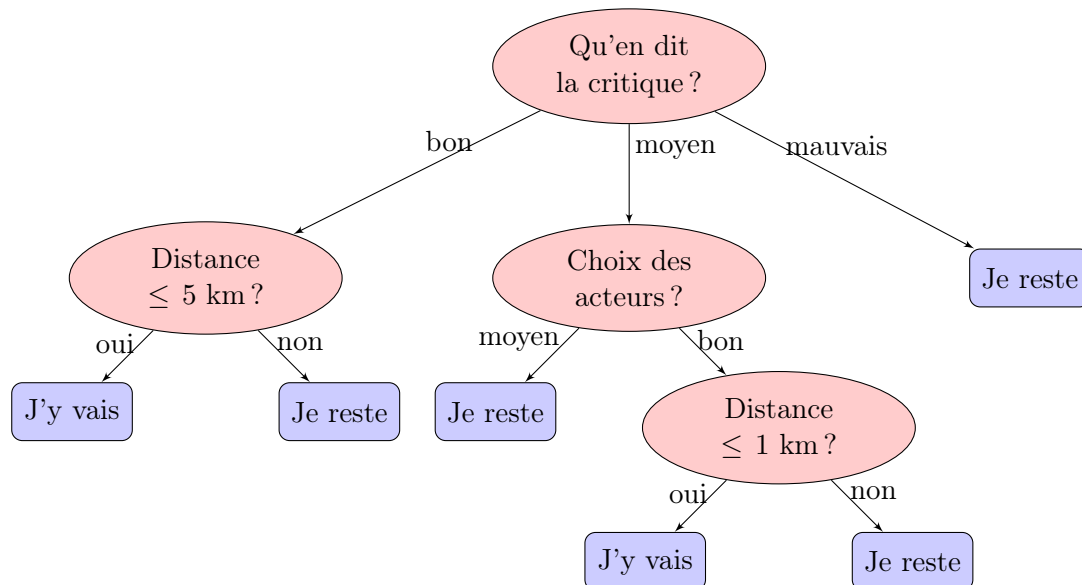


FIGURE 1 – Décider si je vais au cinéma

Question J. Écrire un programme implantant cet arbre de décision et affichant la décision à prendre (quel que soit le choix d'initialisation des variables).

4 pt
36 min

Correction.

```
1  /* Declaration de fonctionnalités supplémentaires */
2  #include <stdlib.h> /* pour EXIT_SUCCESS */
3  #include <stdio.h> /* pour printf() */
4
5  /* Declaration des constantes et types utilisateur */
6  #define BON 2
7  #define MOYEN 1
8  #define MAUVAIS 0
9
10 /* Declaration des fonctions utilisateur */
11
12 /* Fonction principale */
13 int main()
14 {
15     /* Declaration et initialisation des variables */
```


```

16     int critique = MOYEN;
17     int acteurs = BON;
18     int distance = 2; /* accepter la version distance réelle (double) */
19
20     /* Il y a trois possibilités exclusives les unes des autres */
21     if (critique == BON) /* 1) c'est un bon film */
22     {
23         if (distance <= 5) /* c'est à moins de 5 km */
24         {
25             printf("J'y vais\n");
26         }
27         else /* c'est à plus de 5 km */
28         {
29             printf("Je reste\n");
30         }
31     }
32     if (critique == MOYEN) /* 2) le film est moyen */
33     {
34         if (acteurs == MOYEN) /* j'apprécie moyennement les acteurs */
35         {
36             printf("Je reste\n");
37         }
38         else /* il y a un bon choix d'acteurs */
39         {
40             if (distance <= 1) /* c'est à moins de 1 km */
41             {
42                 printf("J'y vais\n");
43             }
44             else /* c'est à plus de 1 km */
45             {
46                 printf("Je reste\n");
47             }
48         }
49     }
50     if (critique == MAUVAIS) /* 3) le film est mauvais */
51     {
52         printf("Je reste\n");
53     }
54
55
56     /* Valeur fonction */
57     return EXIT_SUCCESS;
58 }
59
60 /* Définition des fonctions utilisateur */

```

4 Histogrammes (5 points)

Question K. Soit un entier n initialisé à une valeur positive de votre choix. Écrire un programme qui affiche une ligne contenant n fois le caractère '#' et se terminant par un saut de ligne. Pour cette question, vous pouvez répondre en ne donnant que le code de la fonction principale du programme (**main**).

 1 pt
9 min

Correction.


```
int main()
{
    int n = 6;
    int i; /* var. de boucle */

    for (i = 0; i < n; i = i + 1) /* repeter n fois */
    {
        printf("#"); /* afficher # */
    }
    /* fin de ligne */
    printf("\n");

    return EXIT_SUCCESS;
}
```

Soit un tableau d'entiers, initialisé à des valeurs de votre choix, toutes positives ou nulles, et dont la taille N sera définie par une constante symbolique. Dans les exemples suivants la taille du tableau est 10 et les valeurs contenues dans le tableau sont 3, 0, 2, 8, 9, 10, 3, 5, 5, 2.

On veut écrire un programme qui affiche sous forme d'histogramme (graphique en bâtons, ou graphique barres) les données du tableau : chaque barre de l'histogramme aura une longueur égale à la donnée représentée.


Question L. Écrire un programme qui affiche sous forme d'histogramme les données du tableau. Les barres seront dessinées horizontalement, à l'aide du caractère '#'.  3 pt
27 min

Exemple d'exécution :


Graphique barre 1 :

```
###
##
#####
#####
#####
###
#####
#####
##
```

Correction. Voir le programme complet plus loin.

Question M. Indiquer comment modifier le programme de la question précédente, de manière à demander à l'utilisateur le caractère qui sera utilisé pour dessiner les barres (à la place du #).  1 pt
9 min

Correction. Voir le programme complet plus loin.

Question bonus (plus difficile). Ajouter à votre programme un nouvel affichage en histogramme des données dont les barres progressent cette fois verticalement. Indication : avant cela, votre programme devra trouver le maximum parmi les données du tableau.  2 pt
18 min

Exemple d'exécution :

Graphique barre 2 :

```

      ##
      ## ##
    ## ## ##
    ## ## ##
    ## ## ##
    ## ## ##  ## ##
    ## ## ##  ## ##
##      ## ## ## ## ## ##
##      ## ## ## ## ## ## ## ##
##      ## ## ## ## ## ## ## ##
```

Correction.

```

1  /* Declaration de fonctionnalites supplementaires */
2  #include <stdlib.h> /* pour EXIT_SUCCESS */
3  #include <stdio.h> /* pour printf() */
4
5  /* Declaration des constantes et types utilisateur */
6  #define N 10
7  #define LEGENDE 1
8  #define CHOIXCAR 1
9
10 /* Declaration des fonctions utilisateur */
11
12 /* Fonction principale */
13 int main()
14 {
15     /* Declaration et initialisation des variables */
16     int donnees[N] = {3,0,2,8,9,10,3,5,5,2}; /* donnees a afficher */
17     int maximum = 0; /* maximum dans les donnees */
18     int i; /* var de boucle */
19     int j; /* var de boucle */
20     char c = '#'; /* caractère d'affichage */
21
22     #if CHOIXCAR
23         /* choix du caractère */
24         printf("Entrer le caractere a utiliser pour l'affichage (par exemple #)\n");
25         scanf(" %c", &c);
26     #endif
27
28     /* Affichage des donnees sous la forme d'un graphique barre gauche-droite */
29     printf("Graphique barre 1 :\n\n");
30     for (i = 0; i < N; i = i + 1) /* pour chaque donnee */
31     {
32         #if LEGENDE
33             /* legende */
34             printf("%2d ", donnees[i]);
35         #endif
36         /* afficher une ligne d'etoiles de longueur la valeur de la donnee */
37         for (j = 0; j < donnees[i]; j = j + 1)
```

```

38         {
39     #if CHOIXCAR
40         printf("%c", c);
41     #else
42         printf("#");
43     #endif
44     }
45     /* fin de ligne */
46     printf("\n");
47 }
48
49 /* Trouver le maximum */
50 for (i = 0; i < N; i = i + 1) /* pour chaque donnee */
51 {
52     if (maximum < donnees[i])/* nouveau maximum */
53     {
54         maximum = donnees[i];
55     }
56 }
57
58 /* Affichage d'une graphe barre vertical */
59 printf("\nGraphique barre 2 :\n\n");
60 for (j = maximum; j > 0; j = j - 1) /* pour chaque hauteur de donnee */
61 {
62     for (i = 0; i < N; i = i + 1) /* pour chaque donnee */
63     {
64         if (donnees[i] >= j) /* si la barre de donnee atteint cette hauteur */
65         {
66     #if CHOIXCAR
67         printf(" %c%c", c, c); /* dessiner la barre */
68     #else
69         printf(" ##"); /* dessiner la barre */
70     #endif
71         }
72         else /* sinon laisser blanc */
73         {
74             printf("   ");
75         }
76     }
77     /* fin de la ligne */
78     printf("\n");
79 }
80 #if LEGENDE
81 /* Ajout d'une legende */
82 for (i = 0; i < N; i = i + 1) /* pour chaque donnee */
83 {
84     printf(" %2d", donnees[i]);
85 }
86 /* fin de la ligne */
87 printf("\n");
88 #endif
89 /* Valeur fonction */

```



```
90     return EXIT_SUCCESS;
91 }
92
93 /* Definition des fonctions utilisateur */
```