
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.

Barème. 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".


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

 1 pt
9 min

Correction. L'instruction `define` est une instruction préprocesseur elle sera donc exécutée au tout début de la compilation. Cette instruction permet de définir une constante symbolique (ici `TAILLE`) et un texte de substitution pour cette constante (ici `3`). Le préprocesseur remplacera chaque occurrence de la constante symbolique par le texte de substitution dans le source du programme juste avant qu'il soit compilé.

Barème. 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.

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.

```
11     int somme = 0; /* accumulateur pour la somme */
12     int i; /* var de boucle */
13
14     for (i = 0; i < TAILLE; i = i + 1) /* pour chaque case */
15     {
16         somme = somme + t[i]; /* ajout de la case a la somme */
17     }
18
```

Barème. 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.

Barème. Compter 0.25 pt si ça parle tout de suite de l'erreur du `scanf` (au lieu de donner la signification du message d'erreur), 0.5 pt si il est juste fait mention de la mémoire (sans réservation etc.).

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**).

Barème. Compter 0.5 pt par scanf correctement corrigé (et oui c'est un piège). Si d'autres corrections, hors sujet, apparaissent diminuer la note de l'exercice (jusqu'à zéro).


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.

Barème. Compter juste si il est fait mention de `-Wall`. Sinon zéro.

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

 1 pt
9 min

Correction. Nous avons besoins de deux variables supplémentaires :

```
15     int i; /* var de boucle */
16     int produit = 1; /* accumulateur pour le produit */
17
```

Et il suffit de répéter n fois la multiplication par x :


```
26     for (i = 0; i < n; i = i + 1) /* faire n fois */
27     {
28         produit = produit * x; /* multiplication par x */
29     }
30
31     /* affichage */
32     printf("%d a la puissance %d = %d\n", x, n, produit);
```

Barème. Compter 1,5 point si tout juste et dans ce cas veiller à ne pas dépasser le barème de la section (7 points). Autrement enlever un demi point par erreur parmi : oubli de déclaration d'une variable, pas de boucle, trop de boucles, pas d'affichage, calcul faux. Par exemple pas de boucle + calcul faux c'est un point en moins.

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

Correction.

```
1  #include <stdlib.h> /* EXIT_SUCCESS */
2  #include <stdio.h> /* printf, scanf */
3
4  /* déclaration constantes et types utilisateurs */
```

```


5  #define N 3
6  #define TRUE 1
7  #define FALSE 0
8
9  /* Fonction principale */
10 int main()
11 {
12     int t[N] = {10,10,15};
13     int unique = TRUE;
14     int i; /* var de boucle */
15
16     i = 1;
17     while (unique && (i < N)) /* tant que t[0] est unique */
18     {
19         if (t[i] == t[0]) /* t[0] n'est pas unique */
20         {
21             unique = FALSE;
22         }
23         i = i + 1; /* case suivante */
24     }
25
26     /* affichage du résultat */
27     if (unique)
28     {
29         printf("Vrai\n");
30     }
31     else
32     {
33         printf("Faux\n");
34     }
35     /* valeur fonction */
36     return EXIT_SUCCESS;
37 }

```

Barème. 2 pts si arrive au résultat avec une seule boucle (for ou while). L’affichage doit être correct mais on accepte les variantes : "machin n’est pas unique / "machin est unique" etc.

Sinon maximum 1,5 pts : 0,5 pts s’il y a une seule boucle ; 0,5 points de plus si c’est un while ; ajouter 0,5 points si utilise une variable booléenne (déclarée et initialisée, et les define TRUE et FALSE).

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

Correction.

```

1  #include <stdlib.h> /* EXIT_SUCCESS */
2  #include <stdio.h> /* printf, scanf */
3
4  /* déclaration constantes et types utilisateurs */
5  #define N 4
6  #define TRUE 1
7  #define FALSE 0

```

```

8
9  /* Fonction principale */
10 int main()
11 {
12     int t[N] = {10,12,15,12};
13     int unique = TRUE;
14     int i; /* var de boucle */
15     int j; /* var de boucle */
16
17     j = 0; /* premiere case */
18     while (unique && (j < N)) /* pour chaque case, verifier que la case
19     est unique */
20     {
21         i = j + 1; /* case suivant t[j] */
22         while (unique && (i < N)) /* tant que t[j] est unique */
23         {
24             if (t[i] == t[j]) /* t[j] n'est pas unique */
25             {
26                 unique = FALSE;
27             }
28             i = i + 1; /* case suivante */
29         }
30         j = j + 1; /* case suivante */
31     }
32     /* affichage du resultat */
33     if (unique)
34     {
35         printf("Vrai\n");
36     }
37     else
38     {
39         printf("Faux\n");
40     }
41     /* valeur fonction */
42     return EXIT_SUCCESS;
43 }

```

Barème. 2pt si arrive au résultat avec deux boucles (avec affichage mais variantes acceptée).

Sinon maximum 1,5 pts : 0,5 pts s'il y a deux boucles avec variables de boucles différentes (quelles que soient les boucles et les bornes); 0,5 point si elles sont toutes deux des while; 0,5 pts S'il a un algorithme correct en français en commentaire ou dans le texte; 0,5 points si utilise une variable booléenne ou deux variables booléennes.

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

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

Je viens de gagner une place de cinéma pour un 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.

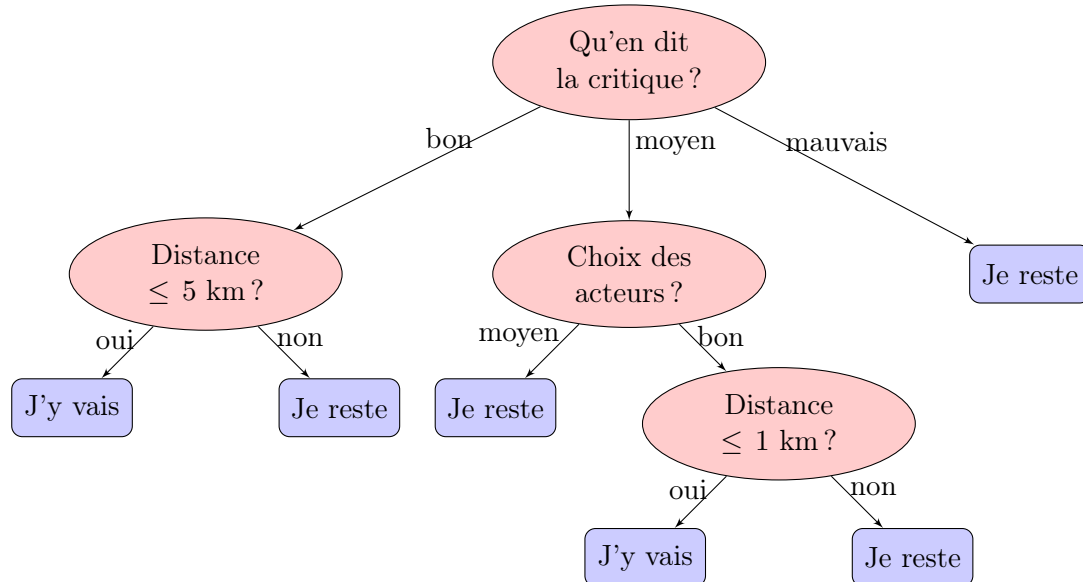


FIGURE 1 – Décider si je vais au cinéma

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 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 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 reelle (double) */
19
20     /* Il y a trois possibilites exclusives les unes des autres */

```

```

21     if (critique == BON) /* 1) c'est un bon film */
22     {
23         if (distance <= 5) /* c'est a moins de 5 km */
24         {
25             printf("J'y vais\n");
26         }
27         else /* c'est a 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'apprecie 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 a moins de 1 km */
41             {
42                 printf("J'y vais\n");
43             }
44             else /* c'est a 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 /* Definition des fonctions utilisateur */

```

Barème. On accepte le fait que la distance soit une variable réelle, sans pénalité.

Compter :

- 0,5 points pour le codage BON, MOYEN, MAUVAIS à l'aide de define s'il est correct (pas d'erreur de syntaxe, comme un signe égal), retirer une pénalité de 0.25 s'il a trop de constantes symboliques. S'il y a par exemple une constante symbolique pour définir la valeur d'initialisation de la distance.
- 0,5 points pour des déclarations avec initialisations correctes.
- 0,5 points pour l'indentation (si celle-ci est absente le sanctionner).
- Une pénalité de 0,25 points par oubli du stdlib, du stdio ou du return final (on ne sanctionne pas l'utilisation d'un 0 à la place du EXIT_SUCCESS).
- On admet que les cas exclusifs soient traités dans des if en cascade (comme dans

le corrigé à la racine de l'arbre) plutôt qu'avec des if else emboîtés. On admet les else if même si ça n'a pas été vu en cours.

- Pour le reste, dessiner l'arbre sous-jacent au programme de l'étudiant et calculer la distance d'édition entre cet arbre est celui demandé par les opérations suivantes (il ne s'agit pas d'un arbre ordonné) :

- étiquette à modifier sur un sommet ou un arc
- inversion père-fils
- sommet absent

On part d'un total de 2.5 et on compte 0.5 point de pénalité par opération d'édition nécessaire.

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;
}
```

Barème. On part de 1 point et on enlève 0.25 pt par erreur : oubli de déclarer la variable de boucle, absence du retour à la ligne etc. Zéro s'il n'y a pas de for.

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 '#'.
Exemple d'exécution :

3 pt
27 min

Graphique barre 1 :

###


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

Correction. Voir le programme complet plus loin.

Barème. On compte entre un et trois points si il y a une double boucle imbriquée : on part de 3 on enlève des pénalités jusqu'à 1 (ne pas descendre en dessous). Pénalité faible : il manque un include, le N n'est pas dans un define et non utilisé dans la boucle, le tableau n'est pas déclaré, pas initialisé, il manque un saut de ligne, variable de boucle non déclarée. Cas particulier : on ne compte que 1 point si les deux boucles ont la même variable, sans regarder plus loin.

S'il n'y a pas de double boucle on ne corrige pas plus on met zéro.


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.

Barème. On ne compte pas de pénalité si le scanf est fait dans un format "%c" au lieu de " %c" ou si c'est de type entier au lieu d'être un char, mais un bonus de 0.25pt si utilise un char. On compte un demi point pour le scanf correct (avec \mathcal{E} , sinon 0) et un demi point pour le printf correct. Pénalité de 0.25 si la variable n'est pas correctement déclarée.

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 */

```

Barème. Deux points si c'est vraiment parfait, uniquement.

On donne plus d'un point uniquement lorsque la réponse est très proche d'une solution correcte. On admet une réponse à base de tableau bidimensionnel.

On peut aller jusqu'à un point si des parties du programme ne fonctionnent pas (recherche du maximum par exemple), mais qu'il y a de bonnes idées (par exemple, penser à balayer la ligne courante).

S'il n'y a que la recherche du maximum on ne compte qu'un quart de point.