
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 (7 points)

1.1 Trace d'un programme (4.5 points)

```
1  /* Déclaration de fonctionnalités supplémentaires */
2  #include <stdlib.h> /* EXIT_SUCCESS */
3  #include <stdio.h> /* printf */
4
5  /* Déclaration constantes et types utilisateurs */
6  #define N 2
7
8  /* Déclaration de fonctions utilisateurs */
9
10 /* Fonction principale */
11 int main()
12 {
13     /* Déclaration et initialisation variables */
14     int t[N] = {3,0}; /* donnees */
15     int s; /* commentaire supprimé */
16     int i; /* var. de boucle */
17     int j; /* var. de boucle */
18     for(i = 0; i < N; i = i + 1) /* commentaire supprimé */
19     {
20         /* commentaire supprimé */
21         s = 0;
22         for(j = 1; j <= t[i]; j = j + 1) /* commentaire supprimé */
23         {
24             /* commentaire supprimé */
25             s = s + j;
26         }
27         /* commentaire supprimé */
28         printf("%d\n", s);
29     }
30     /* commentaire supprimé */
31     return EXIT_SUCCESS;
32 }
33
34 /* implantation de fonctions utilisateurs */
```

Question A. À quoi sert la ligne 6 et à quelle étape de la compilation sera t-elle traitée ?

0.5 pt
4 min

Correction. La ligne 6 contient une directive préprocesseur qui définit une constante symbolique N comme étant équivalente à 2. Le préprocesseur du compilateur C traite ces directives avant la compilation proprement dite. Le traitement consistera ici en remplacer partout dans le programme C le texte N par le texte 2.

Compter tout juste ou tout faux.

Question B. Simulez l'exécution du programme, en réalisant sa **trace** : l'exécution du programme est représentée par un tableau à $n+2$ colonnes ; la première colonne étant le numéro de la ligne exécutée, les n colonnes suivantes, les colonnes des n variables du programme (n à déterminer) et la dernière colonne, la colonne servant à l'affichage à l'écran du programme. Seules les lignes modifiant l'état mémoire du programme sont à reporter dans le tableau.

3 pt
27 min

Correction. La trace est donnée dans le tableau 1.

ligne	t[0]	t[1]	s	i	j	Affichage (sortie écran)
initialisation	3	0	?	?	?	
18				0		
21			0			
22					1	
25			1			
26					2	
25			3			
26					3	
25			6			
26					4	
28						6\n
29				1		
21			0			
22					1	
28						0\n
29				2		
31	renvoie EXIT_SUCCESS					

TABLE 1 – Trace du programme du premier exercice.

Compter

- 0.5 pt pour une ligne d'en-tête juste, autrement dit pour les colonnes (0 sinon)
- 0.5 pt pour une ligne d'initialisation juste (0 sinon)
- aucune pénalité si la dernière ligne est absente (renvoie).
- 2 point pour le reste avec 0.5 point de pénalité par ligne erronée.
- Si des valeurs de variables non modifiées apparaissent sur certaines lignes compter également une pénalité de 0.5 pt.

Question C. Que calcule et affiche ce programme ?

1 pt
9 min

Correction. Pour chaque élément du tableau ce programme calcule et affiche la somme des $t[i]$ premiers entiers :

$$1 + \dots + t[i] = \sum_{k=1}^{t[i]} k.$$


Compter 0.5 pt si il est question de cette somme sur la première case du tableau et 0.5 point si il est question de faire l’affichage d’un calcul (attention : toujours le même) pour chaque case du tableau.

1.2 Une erreur classique (2.5 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.

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**).

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.

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

2 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.

Question G. 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 */
```

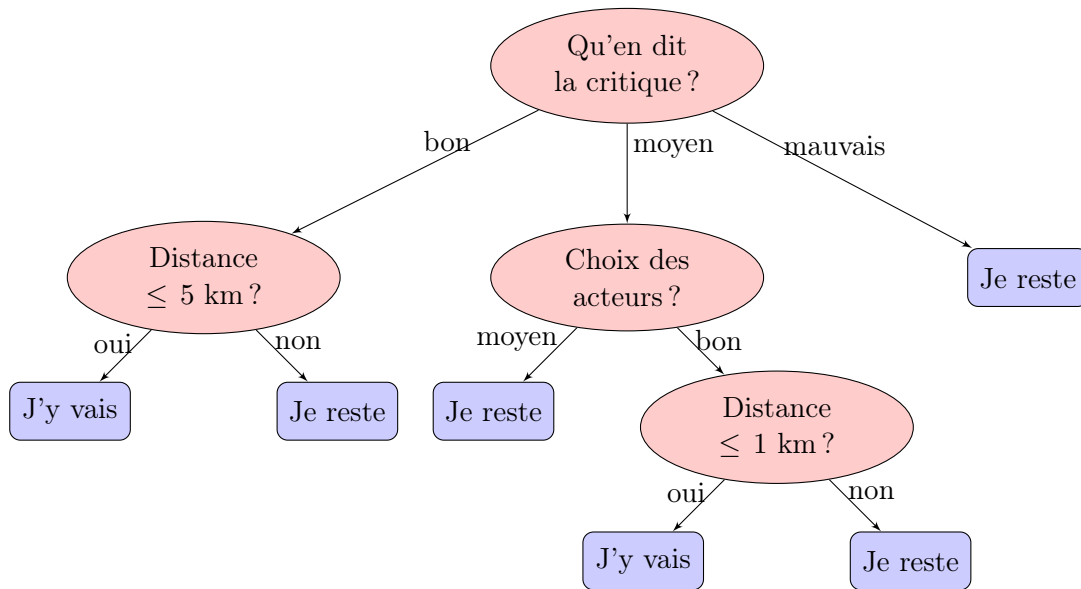


FIGURE 1 – Décider si je vais au cinéma

```

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 */

```


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.

3 Puissance (4 points)

 2 pt
18 min

Question H. Écrire un programme qui :

- demande à l'utilisateur d'entrer un nombre réel x ;
- calcule et affiche la valeur de x^2 .

Exemple d'exécution :

```

Entrer un nombre reel : 7.5e9
7.5e+09 exposant 2 = 5.625e+19

```

Correction.

```

1  /* Declaration de fonctionnalites supplementaires */
2  #include <stdlib.h> /* pour EXIT_SUCCESS */
3  #include <stdio.h> /* pour printf() et scanf() */

```

```

4
5  /* Declaration des constantes et types utilisateur */
6
7  /* Declaration des fonctions utilisateur */
8
9  /* Fonction principale */
10 int main()
11 {
12     /* Declaration et initialisation des variables */
13     double x;
14     double res;
15     int i; /* var de boucle */
16
17     /* Demander a l'utilisateur de saisir un nombre reel */
18     printf("Entrer un nombre reel : ");
19     scanf("%lg", &x);
20
21     /* Calcul du carre */
22     res = x * x;
23
24     /* affichage du resultat */
25     printf("%g exposant 2 = %g\n", x, res);
26
27     /* Valeur fonction */
28     return EXIT_SUCCESS;
29 }
30
31 /* Definition des fonctions utilisateur */

```

- Un point pour une saisie correcte
- un point pour le calcul et l’affichage
- des pénalités de 0.25 pt si il y a des erreurs : manque stdio, return EXIT_SUCCESS (on admet return 0)

Question I. Réécrire ce programme (ou indiquer clairement les modifications à apporter) de manière à ce qu’il demande également à l’utilisateur l’exposant n (un entier positif) auquel doit être élevé x et affiche le résultat du calcul de x^n . Il faut programmer ce calcul (ne pas faire appel à une bibliothèque offrant la fonction ad-hoc). L’algorithme utilisé devra apparaître clairement.

2 pt
18 min

Exemple d’exécution :

```

Entrer un nombre reel : 12.35
Entrer son exposant (entier positif) : 10
12.35 exposant 10 = 8.2541e+10

```

Correction.

```

1  /* Declaration de fonctionnalites supplementaires */
2  #include <stdlib.h> /* pour EXIT_SUCCESS */
3  #include <stdio.h> /* pour printf() et scanf() */
4
5  /* Declaration des constantes et types utilisateur */
6

```

```

7  /* Declaration des fonctions utilisateur */
8
9  /* Fonction principale */
10 int main()
11 {
12     /* Declaration et initialisation des variables */
13     double x;
14     double res = 1.0;
15     int exposant;
16     int i; /* var de boucle */
17
18     /* Demander a l'utilisateur de saisir un nombre reel */
19     printf("Entrer un nombre reel : ");
20     scanf("%lg", &x);
21     printf("Entrer son exposant (entier positif) : ");
22     scanf("%d", &exposant);
23
24     /* Calcul de la puissance */
25     for (i = 0; i < exposant; i = i + 1) /* repeter exposant fois */
26     {
27         res = res * x; /* multiplication par x */
28     }
29
30     /* affichage du resultat */
31     printf("%g exposant %d = %g\n", x, exposant, res);
32
33     /* Valeur fonction */
34     return EXIT_SUCCESS;
35 }
36
37 /* Definition des fonctions utilisateur */

```

On ne regarde que la partie modifiée depuis le programme précédent et on compte 1 point pour un algorithme correct :


- mettre le résultat à 1
- Répéter n fois
 - multiplier le résultat par x
- afficher le résultat

Et 1 point pour son écriture en C.

Pénalités de 0.25 pt s'il manque l'initialisation à 1, ou la saisie, ou l'affichage, ou la déclaration de la variable de boucle.

4 Histogrammes (5 points)

Question J. 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;
}

```

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 K. 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. 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 L. 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. On ne compte pas de pénalité si le scanf est fait dans un format "%c" au lieu de " %c". On compte un demi point pour le scanf correct 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 */

```

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.