

Éléments d'informatique, rattrapage

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 Minimum des éléments d'un tableau

Nous voulons écrire un programme qui calcule le minimum des éléments d'un tableau d'entiers. Une partie du programme est déjà écrite, figure 1, et Pippo pense qu'il ne reste plus qu'à écrire la partie qui calcule effectivement le minimum.

```
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 */
11
12
13
14     /* calcul du minimum (À FAIRE) */
15
16
17
18
19
20     /* affichage du résultat */
21     printf("Le minimum est %d\n", minimum);
22
23     /* valeur fonction */
24     return EXIT_SUCCESS;
25 }
```

FIGURE 1 – Minimum à compléter

Question A. 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

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

1 pt
9 min

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


2.5 pt
22 min

Exemple d'exécution pour le tableau donné dans le code :

Le minimum est 10

1.2 Trace d'un programme avec fonctions

Question D. Simulez l'exécution du programme figure 2, en réalisant sa **trace**, comme cela a été vu en TD et en cours.

 4 pt
36 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 n);
8  void bar(int n);
9
10 /* fonction principale */
11 int main()
12 {
13     int x = 3;
14     int res;
15
16     res = foo(x);
17     printf("foo(%d) = %d\n", x, res);
18
19     bar(x);
20
21     return EXIT_SUCCESS;
22 }
23
24 /* definitions de fonctions utilisateurs */
25 int foo(int n)
26 {
27     int i;
28     int reponse = 42;
29     for (i = n; i > 0; i = i - 2)
30     {
31         printf("%d ", i);
32     }
33     printf("\n");
34     return reponse;
35 }
36
37 void bar(int n)
38 {
39     if (n > 1)
40     {
41         printf("%d ", n);
42         bar(n - 2);
43     }
44     else
45     {
46         printf("%d\n", n);
47     }
48 }
```


FIGURE 2 – Programme pour la trace

2 Sans fonctions, for ou while, if, structures

Il est demandé de résoudre les trois problèmes suivants sans définir de fonctions utilisateurs. L'ensemble du code sera à écrire dans la fonction principale `main`.

2.1 Factorielle

Question E. Écrire le programme qui : demande à l'utilisateur d'entrer un entier ; calcule la factorielle de l'entier saisi ; affiche le résultat du calcul, comme dans l'exemple suivant.

 2 pt
18 min

On supposera que l'entier saisi est toujours supérieur ou égal à zéro (on ne teste pas la saisie de l'utilisateur). Rappel : la factorielle d'un entier n , notée habituellement $n!$, vaut le produit de tous les entiers de 1 à n : $n! = 1 \times 2 \times \dots \times n$.


Exemple d'exécution :

```
Saisissez un entier : 5
factorielle de 5 = 120
```

2.2 Halte à la dette

Pippo emprunte 100 brouzoufs à 6% d'intérêts annuel. Il rembourse par versements de 25 brouzoufs chaque année, en commençant un an après la date d'emprunt. Combien d'années devra t'il rembourser son emprunt ? Chaque année, on calcule d'abord les intérêts à ajouter à la dette actuelle, puis on soustrait le montant du versement annuel.

Question F. Écrire un programme qui affiche le plan de remboursement de Pippo, c'est à dire, pour chaque année : les intérêts de l'année, le versement, la dette restant à rembourser.


 2.5 pt
22 min

Votre programme doit pouvoir être exécuté pour n'importe quelle autre initialisation des données du prêt.

Exemple d'exécution :


```
annee 1, interets : 6, versement : 25, dette : 81
annee 2, interets : 4.86, versement : 25, dette : 60.86
annee 3, interets : 3.6516, versement : 25, dette : 39.5116
annee 4, interets : 2.3707, versement : 25, dette : 16.8823
annee 5, interets : 1.01294, versement : 25, dette : -7.10477
```

Question G. Le dernier versement est trop important. Comment modifier votre programme de manière à ne rembourser que le reste à payer, dernière année ? La dernière ligne affichée sera alors :

 1 pt
9 min

```
annee 5, interets : 1.01294, versement : 17.8952, dette : 0
```


Question H. Si le versement est inférieur aux intérêts, que dire de l'exécution de ce programme ?


 0.5 pt
4 min


2.3 Deux jours avant

Question I. Compléter le programme de la figure 3 :

- définir le type utilisateur `struct date_s`, pour stocker une date sous la forme numéro du jour, numéro du mois et année, de manière à ce que la valeur initiale de la variable `demain` corresponde à la date du premier juin 2011 ;
- à partir de la date de demain, calculer la date d'hier (deux jours avant demain). Vous supposerez que la date de demain appartient à l'année 2011. Pour connaître le nombre de jours de chaque mois, vous utiliserez le tableau `nbjours`. Ce tableau contient à chaque indice i entre 1 et 12 le nombre de jours du i ème mois de l'année 2011 et à l'indice 0 le nombre de jours du mois de décembre 2010 ;
- afficher le résultat comme dans les exemples suivants.

 1 pt
9 min

 2 pt
18 min

 0.5 pt
4 min

Votre programme doit fonctionner et calculer la date correcte pour n'importe quelle autre initialisation de la variable `demain` par une date de l'année 2011.

```

1  #include <stdlib.h> /* EXIT_SUCCESS */
2  #include <stdio.h> /* printf, scanf */
3
4  /* déclaration constantes et types utilisateurs */
5
6  /* Fonction principale */
7  int main()
8  {
9      int nbjours[13] = {31, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
10     /* nombre de jours de chaque mois, de decembre 2010 a decembre 2011 */
11
12     struct date_s demain = {1, 6, 2011}; /* exemple */
13     struct date_s hier;
14
15     /* calcul de la date d'hier */
16
17     /* affichage du résultat */
18
19     /* valeur fonction */
20     return EXIT_SUCCESS;
21 }

```

FIGURE 3 – Deux jours avant (à compléter)

Exemple d'exécution :

Demain : 1/6/2011
Hier : 30/5/2011

Autres exemples (en modifiant l'initialisation de la variable `demain`) :

Demain : 2/1/2011
Hier : 31/12/2010

Demain : 12/5/2011
Hier : 10/5/2011

3 Écriture de fonctions


1. Déclarer et définir la procédure `menu` qui n'a pas d'entrée et affiche :


```

*****
* MENU *
*****

```

2. Déclarer et définir une fonction qui calcule la valeur absolue d'un réel.

 1 pt
9 min

 1 pt
9 min

