

## 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


**Correction.** Mêmes question qu'au partiel de mi-semestre.

**Barème.**

1. 1.5 pt (barème augmenté) dont 1 pt pour “déclaration de la variable `minimum`” + 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 constante symbolique, préprocesseur, `TAILLE` remplacé par 3. 0.5 pt si seulement notion de constantes et pas de mise en relation avec la substitution au moment de la compilation.
3. 2.5 pt pour le code juste (compilation correction quel que soit `TAILLE`). Sinon, maxi 2 pt :
  - pas de boucle -> zro pt
  - 0.5 pt une (et une seule) boucle (for, while accepté)
  - 0.5 pt 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 `minimum` à zéro.
  - 0.5 pt pour un if imbriqué dans la boucle.
  - 0.5 pt si le if est correct, c'est à dire s'il réalise le bon test (comparaison avec le `minimum` courant) et change la valeur du `minimum` courant en conséquent.

## 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

**Correction.** Table 1 page 4.

**Barème.** Maximum 4 pt. Si des erreurs, maximum 3,75 pt.

- (a) +1 pt deux premières lignes de la trace du main sont correctes (identification des variables et leurs initialisations).
- \* -0,5 pt par variable manquante ou en trop
  - \* -0,5 pt par initialisation manquante ou en trop
- (b) +1,25 pt Pour l'appel à foo :
- \* +0,25 pt pour `foo(3)` ligne 16
  - \* +0,25 pt une colonne pour le paramètre formel `n` bien initialisé à 3
  - \* +0,5 pt pour le déroulement correct de la boucle (deux tours, `i = -1`), 0 si une erreur.
  - \* +0,25 pt pour le retour de 42 et l'affectation ligne 16.
- (c) +1,25 pt pour l'appel récursif à bar :
- \* +0,25 pt pour `bar(3)` ligne 19
  - \* +0,5 pt pour le déclenchement du second appel ligne 42.
  - \* +0,25 pt une colonne pour le paramètre formel `n` bien initialisé à 3 et à 1 dans le second appel.
  - \* +0,25 pt pour une profondeur de exactement 2 (pas plus).
- (d) +0,25 pt pour au moins l'un des deux affichages (`3 1 \n`) de foo ou de bar correct.

```

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

main()																																																
ligne	x	res	Affichage																																													
initialisation	3	?																																														
16	<div>foo(3)</div> <table><tr><td>ligne</td><td>n</td><td>i</td><td>reponse</td><td>Affichage</td></tr><tr><td>initialisation</td><td>3</td><td>?</td><td>42</td><td></td></tr><tr><td>29</td><td></td><td>3</td><td></td><td></td></tr><tr><td>31</td><td></td><td></td><td></td><td>3</td></tr><tr><td>32</td><td></td><td>1</td><td></td><td></td></tr><tr><td>31</td><td></td><td></td><td></td><td>1</td></tr><tr><td>32</td><td></td><td>-1</td><td></td><td></td></tr><tr><td>33</td><td></td><td></td><td></td><td>\n</td></tr><tr><td>36</td><td colspan="4">renvoie 42</td></tr></table>			ligne	n	i	reponse	Affichage	initialisation	3	?	42		29		3			31				3	32		1			31				1	32		-1			33				\n	36	renvoie 42			
ligne				n	i	reponse	Affichage																																									
initialisation				3	?	42																																										
29					3																																											
31							3																																									
32					1																																											
31							1																																									
32					-1																																											
33							\n																																									
36				renvoie 42																																												
16		42																																														
17			foo(3) = 42\n																																													
19	<div>bar(3)</div> <table><tr><td>ligne</td><td>n</td><td>Aff.</td></tr><tr><td>init.</td><td>3</td><td></td></tr><tr><td>41</td><td></td><td>3</td></tr><tr><td>42</td><td colspan="2" rowspan="3"><div>bar(1)</div><table><tr><td>ligne</td><td>n</td><td>Aff.</td></tr><tr><td>init.</td><td>1</td><td></td></tr><tr><td>46</td><td></td><td>1\n</td></tr></table></td></tr></table>			ligne	n	Aff.	init.	3		41		3	42	<div>bar(1)</div> <table><tr><td>ligne</td><td>n</td><td>Aff.</td></tr><tr><td>init.</td><td>1</td><td></td></tr><tr><td>46</td><td></td><td>1\n</td></tr></table>		ligne	n	Aff.	init.	1		46		1\n																								
ligne				n	Aff.																																											
init.				3																																												
41					3																																											
42	<div>bar(1)</div> <table><tr><td>ligne</td><td>n</td><td>Aff.</td></tr><tr><td>init.</td><td>1</td><td></td></tr><tr><td>46</td><td></td><td>1\n</td></tr></table>		ligne	n	Aff.	init.	1		46		1\n																																					
ligne			n	Aff.																																												
init.			1																																													
46		1\n																																														
22	renvoie EXIT_SUCCESS																																															

TABLE 1 – Trace du programme de l'exercice 2.

## 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
```

**Correction.**

```
#include <stdlib.h> /* EXIT_SUCCESS */
#include <stdio.h> /* printf, scanf */

/* declarations constantes et types utilisateurs */

/* fonction principale */
int main()
{
    int n = 0;          /* saisie utilisateur */
    int produit = 1;    /* accumulateur pour le produit */
    int i;              /* var de boucle */

    printf("Saisissez un entier : ");
```

```

scanf("%d", &n);

for (i = 1; i <= n; i = i + 1)
{
    produit = produit * i;
}

printf("factorielle de %d = %d\n", n , produit);

return EXIT_SUCCESS;
}

```


**Barème.** Si tout juste à l'aide d'un for : 2 pt sinon maxi 1, 75 pt.

- 0.25 pt déclaration (avec ou sans initialisation) d'une variable entière pour la saisie
- 0.25 pt saisie à l'aide d'un scanf avec ou sans
- 0.5 pt boucle for pas de point en moins si oublie de déclaration de la variable de boucle
- 0.25 pt bloc de boucle exécuté  $n$  ou  $n - 1$  fois
- 0.25 pt déclaration d'un accumulateur pour stocker le produit
- 0.25 pt initialisation à 1 de l'accumulateur
- 0.25 pt affichage du résultat du calcul avec les deux conversions (faux ou juste)

## 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

```

**Correction.**

```

#include <stdlib.h> /* EXIT_SUCCESS */
#include <stdio.h> /* printf, scanf */

/* déclaration constantes et types utilisateurs */

/* Fonction principale */
int main()
{
    double dette = 100.0;
    double versement = 25.0;
    double taux = 0.06;
    int annee = 0;
    double interets;
    double total = 0.0;

    while (dette > 0)

```

```

{
    interets = dette * taux;
    dette = dette + interets - versement;
    annee = annee + 1;
    printf("annee %d, interets : %g, versement : %g, dette : %g\n",
        annee, interets, versement, dette);
    total = total + versement;
}

return EXIT_SUCCESS;
}

```

**Barème.** Si tout juste à l'aide d'un `while` : 2.5 pt sinon maxi 2, 25 pt.

- 0.25 pt déclaration et initialisation de la dette et du taux comme des double (on ne regarde pas versement).
- 0.5 pt une et une seule boucle `while`.
- 0.5 pt modification de la valeur de la dette dans la boucle (0.25 pt si mauvaise modification)
- 0.25 pt déclaration et initialisation du compteur d'années (une erreur de 1 est acceptée)
- 0.25 pt incrément du compteur d'année à chaque tour de boucle
- 0.25 pt affichage dans la boucle
- 0.25 pt affichage des bonnes valeurs avec les bonnes conversions

**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

**Correction.** juste avant affichage dans le `while` on insère :

```

if (dette < 0)
{
    versement = versement + dette; /* on supprime le trop vers'e */
    dette = 0;
}

```

**Barème.** Si tout juste : 1 pt sinon maxi 0.75 pt.

- 0.25 pt pour un `if` dans le `while` (ou après si condition du `while` modifiée)
- 0.25 pt pour un `pour` dette mise à zéro avant affichage
- 0.25 pt pour un versement égal au reste à percevoir avant affichage

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

0.5 pt  
4 min

**Correction.** Si le versement est inférieur aux intérêts la première année, la dette va augmenter et elle ne sera jamais nulle (ou négative). Donc le programme va boucler et la dette atteindre une valeur représentant l'infinie.

**Barème.** 0.5 pt si mention du fait que le programme ne s'arrête jamais.

### 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 ;

1 pt  
9 min

```

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)

- 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.

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.

 2 pt  
18 min

 0.5 pt  
4 min

### 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

### Correction.

```

#include <stdlib.h> /* EXIT_SUCCESS */
#include <stdio.h> /* printf, scanf */

/* déclaration constantes et types utilisateurs */
struct date_s
{
    int j; /* jour */
    int m; /* mois */
    int a; /* annee */
};

/* Fonction principale */
int main()
{
    int nbjours[13] = {31, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
    /* nombre de jours de chaque mois, de decembre 2010 a decembre 2011 */

```

```

struct date_s demain = {2, 1, 2011}; /* exemple */
struct date_s hier;

/* calcul de la date d'hier */
hier = demain;
hier.j = demain.j - 2;
if (hier.j < 1)
{
    hier.m = demain.m - 1;
    hier.j = nbjours[hier.m] + hier.j;
}
if (0 == hier.m)
{
    hier.m = 12;
    hier.a = 2010;
}

/* affichage du résultat */
printf("Demain : %d/%d/%d\n", demain.j, demain.m, demain.a);
printf("Hier : %d/%d/%d\n", hier.j, hier.m, hier.a);

/* valeur fonction */
return EXIT_SUCCESS;
}

```

**Barème.** Si tout juste : 3.5 pt sinon maxi 2, 25 pt.

- 0.5 pt déclaration correcte syntaxiquement d'une structure contenant au moins trois champs (entiers, double ou char).
- 0.5 pt la déclaration de la structure est en accord avec son usage dans la suite.
- 2 pt calcul exact sinon maxi 1.75 pt
  - 0.5 pt si utilisation d'un ou plusieurs if pour distinguer plusieurs cas pour les jours (cas du premier et deuxième jours du mois vs autres jours du mois).
  - 0.5 pt on compte 0.25 pt si utilisation de exactement une case du tableau (pour les cas de début de mois) et +0.25 pt s'il s'agit de la case du mois précédent le mois courant
  - 0.25 pt Si calcul du jour toujours correct
  - 0.25 pt Si calcul du mois toujours correct
  - 0.25 pt Si calcul de l'année toujours correct,
- 0.5 pt Affichage au bon format de demain et hier (décomposition en affichage de trois valeurs chacune). zéro pt sinon.


### 3 Écriture de fonctions

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


```

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

```

 1 pt  
9 min

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

 1 pt  
9 min

**Correction.** Difficultés diverses mais c'est voulu, on compte : 0.5 pt déclaration correcte et 0.5 pt code correct.



