

Schéma itératif [it17]

Sujets d'examens (axiomatique impérative)

Karine Zampieri, Stéphane Rivière

Unisciel  algoprog 

Version 17 mai 2018

Table des matières

1	Décomposition en facteurs	2
1.1	Décomposition en facteurs (1) / pgndecom1 (6 points)	2
1.2	Décomposition en facteurs (2) / pgndecom2 (7 points)	3
2	Numération romaine	5
2.1	Ancienne numération romaine / pgromain (6 points)	5
2.2	Codage romain / pgromain	5
3	Nombre de Hamming / pghamming (6 points)	6
4	Les onze voleurs / pgonzev (6 points)	7
5	Miroir d'un entier / pgnpalind (6 points)	9
6	Poids d'un entier / pgnpoids (6 points)	10
7	Personnes dans l'ascenseur (4 points)	11
8	Compte à rebours (5 points)	12
9	Kaprekar à trois chiffres (7 points)(Examen)	13
10	Nombre heureux (1) / pgheureux1 (6 points)	15
11	Nombre heureux (2) / pgheureux2 (6 points)	17
12	Multiplication égyptienne / pgegypt1 (6 points)	19

Java - Sujets d'examens (Examen)

1 Décomposition en facteurs

1.1 Décomposition en facteurs (1) / pgndecom1 (6 points)



Objectif

Affichez la décomposition en facteurs premiers d’un **entier positif** ainsi que le produit recomposé. Exemple du résultat attendu :

```
Entier a décomposer? -123
Entier a décomposer? 265880
2 * 2 * 2 * 5 * 17 * 17 * 23 * 1 = 265880
```

Méthode

Pour réaliser la décomposition d’un entier n , nous allons utiliser la méthode suivante. Soit p le produit recomposé et soit d un diviseur :

1. Diviser n par d (valant 2) tant que possible et multiplier p par d .
2. Ensuite traverser les **entiers impairs** (à partir de d valant 3), et le diviser par d tant que possible (et donc multiplier p par d) avant de passer à l’impair d suivant.



(2 points) Écrivez un programme qui demande et saisit un entier dans **n** **tant qu'il est négatif ou nul**. Affichez l’invite :

```
Entier a decomposer?
```



Testez votre saisie.



(0.5 point) Déclarez deux entiers **p** (le produit recomposé) et **d** (un diviseur) et initialisez-les correctement.



(1.5 point) Réalisez l’étape (1) de la méthode ci-dessus.

Dans le cas où **d** divise **n**, affichez (où **[x]** désigne le contenu de **x**) :

```
[d]*
```



(2 points) Réalisez l’étape (2) de la méthode ci-dessus.

Attention ! vous ne devez parcourir que les entiers impairs : en effet, après l’étape (1), l’entier qui reste à décomposer est forcément un entier impair.



Finalement affichez :

```
1 = [p]
```



Testez. Exemples d’exécution :

```
Entier a décomposer? -123
Entier a décomposer? 265880
2 * 2 * 2 * 5 * 17 * 17 * 23 * 1 = 265880
```

```
Entier a decomposer? 1001880
2 * 2 * 2 * 3 * 3 * 5 * 11 * 11 * 23 * 1 = 1001880
```

1.2 Décomposition en facteurs (2) / pgndecomp2 (7 points)



Objectif

Affichez la décomposition en facteurs premiers d’un **entier positif** ainsi que le produit recomposé. Exemple du résultat attendu :

```
Entier a décomposer? -123
Entier a décomposer? 265880
2 * 2 * 2 * 5 * 17 * 17 * 23 * 1 = 265880
```

Méthode rapide

Pour réaliser la décomposition d’un entier n , nous allons utiliser la méthode suivante. Soit p le produit recomposé et soit d un diviseur :

1. Diviser n par d (valant 2) tant que possible et multiplier p par d .
2. Diviser n par d (valant 3) tant que possible et multiplier p par d .
3. Ensuite, à partir de d valant 5, traverser tous les diviseurs de la forme $6k - 1$ et $6k + 1$, et le diviser par l’impair d tant que possible (et donc multiplier p par d) avant de passer à l’impair suivant, en augmentant d de 2 ou de 4.

On notera que l’alternance entre 2 et 4 s’effectue simplement comme suit :

Soit s valant initialement 2. Pour passer à l’impair suivant, on calcule :

```
s <-- 6 - s // l'opération <-- désigne l'affectation
```

En effet, pour s valant 2, le suivant est 4 et son suivant est 2, et ainsi de suite.



(2 points) Écrivez un programme qui demande et saisit un entier dans **n tant qu'il est négatif ou nul**. Affichez l’invite :

```
Entier a decomposer?
```



Testez votre saisie.



(0.5 point) Déclarez deux entiers **p** (le produit recomposé) et **d** (un diviseur) et initialisez-les correctement.



(1.5 point) Réalisez l’étape (1) de la méthode ci-dessus.

Dans le cas où **d** divise **n**, affichez (où **[x]** désigne le contenu de **x**) :

[d]*



(0.5 point) De même, réalisez l’étape (2) de la méthode ci-dessus avec **d** valant 3.



(2.5 points) Enfin réalisez l’étape (3) de la méthode ci-dessus.



Finalement affichez :

1 = [p]



Testez. Exemples d’exécution :

Entier a décomposer? -123

Entier a décomposer? 265880

$2 * 2 * 2 * 5 * 17 * 17 * 23 * 1 = 265880$

Entier a decomposer? 1001880

$2 * 2 * 2 * 3 * 3 * 5 * 11 * 11 * 23 * 1 = 1001880$

2 Numération romaine

2.1 Ancienne numération romaine / pgromain (6 points)



Objectif

Cet exercice affiche un entier positif en chiffres romains en utilisant l’ancienne notation. Ce système suppose que les chiffres sont lus de la gauche vers la droite avec la correspondance :

```
I: 1, V:5, X: 10, L: 50, C: 100, D: 500, M: 1000
```

Exemple d’exécution :

```
Votre entier positif? -20
Votre entier positif? 81
LXXXI
```



(2 points) Écrivez un programme qui saisit un entier **tant qu'il est négatif ou nul**. Affichez l’invite :

```
Votre entier positif?
```



(4 points) Écrivez la décomposition.

Testez. Exemples d’exécution :

```
Votre entier positif? 163
CLXIII
```

```
Votre entier positif? 639
DCXXXVIIII
```

```
Votre entier positif? 1981
MDCCCCLXXXI
```

2.2 Codage romain / pgromain



Inventez un système de comptage inspiré de celui des romains et faites un programme qui convertit des entiers naturels entrés par l’utilisateur.

3 Nombre de Hamming / pghamming (6 points)



Objectif

Les nombres de HAMMING sont les **entiers naturels** de la forme :

$$H = 2^a \cdot 3^b \cdot 5^c \quad (\text{avec } a, b, c \in \mathbb{N})$$

Exemples :

- $100 = 2^2 3^0 5^2$ et $15 = 3^1 5^1$ sont des entiers de HAMMING.
- Mais ni 21, ni 22, ni 23 ne le sont.



(2 points) Écrivez un programme qui saisit un entier **tant qu'il est négatif ou nul**. Affichez l'invite :

Entier à tester?



(1 point) Calculez l'entier **a** qui représente la puissance de 2.



(0.5 point) De même, calculez l'entier **b** qui représente la puissance de 3.



(0.5 point) Enfin calculez l'entier **c** qui représente la puissance de 5.



(1 point) Calculez un booléen **h** qui sera **Vrai** si l'entier est de HAMMING, **Faux** sinon.



(1 point) Affichez (où **[x]** désigne le contenu de **x**) :

`2^[a] * 3^[b] * 5^[c] * [entier restant]`

`Hamming : [h]`

(Le C++ indiquera 1 pour **true** et 0 pour **false**.)



Testez.

4 Les onze voleurs / pgonzev (6 points)



Objectif

Onze voleurs veulent se partager équitablement n pièces d’or.
En restera-t-il, et si oui, combien ?



Propriété

Si la différence entre la somme des chiffres de rang pair d’un entier et la somme de ses chiffres de rang impair est divisible par 11, alors l’entier est lui-même divisible par 11.

Réciproquement, si un entier est divisible par 11, alors la différence entre la somme de ses chiffres de rang pair et celle de ses chiffres de rang impair est aussi divisible par 11.

Exemple

```
.      n = 765935
Positions 543210
s0 = 5+9+6 = 20
s1 = 3+5+7 = 15
```

D’où $|15-20|=5$. Le reste 5 n’étant pas un multiple de 11, on peut affirmer que `n` n’est lui-même pas divisible par 11. Il restera donc des pièces d’or après le partage et le nombre de pièces restantes est égal à 5.



(2 points) Écrivez un programme qui saisit le nombre de pièces dans un entier **tant qu'il est négatif ou nul**. Affichez l’invite :

Nombre de pieces?



Testez la saisie.



Déclarez deux entiers `s0` (pour la somme des chiffres de rang pair) et `s1` (pour celle des chiffres impairs) et initialisez-les correctement.



(3 points) Calculez `s0` et `s1` pour l’entier saisi.

Aide méthodologique

Soit `n` l’entier saisi.

Il faudra le traiter de la droite vers la gauche :

- Soit par chiffre : dans ce cas, il vous faudra aussi utiliser un booléen pour savoir s’il faut sommer dans `s0` ou dans `s1`.
- Soit par deux chiffres à la fois : le rang pair qui sera sommé dans `s0` et celui de rang impair qui sera sommé dans `s1`. Cette stratégie fonctionne car on effectue des sommes.

Rappel de cours

L’opération modulo par 10 permet de récupérer le dernier chiffre d’un entier, la division par 10 fait perdre son dernier chiffre et la division par 100 fait perdre ses deux derniers chiffres.

 **(0.5 point)** Calculez un booléen `b` qui sera `Vrai` si le partage est réalisable, `Faux` sinon.

 **(0.5 point)** Affichez (où `[x]` désigne le contenu de `x`) :

```
s0=[s0] s1=[s1]
```

```
Partage : [b]
```

(Le C++ indiquera 1 pour `true` et 0 pour `false`.)

 Testez. Exemples d’exécution :

```
Nombre de pieces? -10
```

```
Nombre de pieces? 23561
```

```
s0=8 s1=9
```

```
Partage : 0
```

```
Nombre de pieces? 125191
```

```
s0=4 s1=15
```

```
Partage : 1
```

5 Miroir d’un entier / pgnpalind (6 points)



Objectif

On appelle le **renversé** (ou **miroir**) d’un entier positif, la lecture de la gauche vers la droite de ses chiffres. Exemples :

- Le renversé de 2035 est 5302.
- Celui de 8954070 est 704598.
- Et celui de 14500 est 541.



(2 points) Écrivez un programme qui saisit un entier dans **n** tant qu'il est négatif ou nul. Affichez l’invite :

Entier a tester?



Déclarez un entier **r** (pour le renversé) et initialisez-le.



(3 points) Calculez le renversé de **n** dans **r**.

Aide méthodologique

Des « modulo et division » par 10 permettent d’obtenir un à un tous les chiffres d’un entier. Le tableau ci-dessous montre la méthode pour $n = 12306$.

Valeurs successives de l’entier n	Restes successifs	Quotients successifs
12306	6	1230
1230	0	123
123	3	12
12	2	1
1	1	0



Affichez (où **[x]** désigne le contenu de **x**) :

Entier miroir = [r]



(0.5 point) Un entier positif est dit **palindromique** s’il est égal à son renversé. Exemple : Les entiers 12021, 134431 et 1047401 sont palindromiques.

Dans un booléen **b**, calculez la palindromie de **n**.



(0.5 point) Affichez finalement :

Entier palindromique = [b]

(Le C++ indiquera 1 pour **true** et 0 pour **false**.)

6 Poids d’un entier / pgnpoids (6 points)



Objectif

On appelle le **poids d’un entier naturel**, la somme des chiffres de cet entier.

Exemple : Le poids de 12306 est $6 + 0 + 3 + 2 + 1 = 12$.



(2 points) Écrivez un programme qui saisit un entier dans **n** tant qu'il est négatif ou nul. Affichez l'invite :

Entier à tester?



Testez la saisie.



(2 points) Calculez la somme des chiffres de **n** dans un entier **s** (qu'il faudra déclarer et initialiser correctement).

Aide méthodologique

Des « modulo et division » par 10 permettent d’obtenir un à un tous les chiffres d’un entier. Le tableau ci-dessous montre la méthode pour $n = 12306$.

Valeurs successives de l’entier n	Restes successifs	Quotients successifs
12306	6	1230
1230	0	123
123	3	12
12	2	1
1	1	0



(0.25 point) Affichez (où **[x]** désigne le contenu de **x**) :

Poids de l’entier = **[s]**



(2 points) Complétez votre boucle de sorte à calculer également le produit des chiffres dans un entier **p**.



(0.25 point) Affichez aussi :

Produit des chiffres = **[p]**



Testez.

7 Personnes dans l’ascenseur (4 points)



Plusieurs personnes veulent prendre un ascenseur.

Définissez la constante `CPMAX=300` de la **capacité maximale** de l’ascenseur.



(4 points) Écrivez un programme qui saisit les poids des personnes (en kilogrammes) puis calcule le nombre de ces personnes dans un entier `n` pouvant prendre l’ascenseur. Utilisez la notion de sentinelle (poids négatif ou nul) pour terminer la saisie. A l’issu du calcul, si la capacité maximale est atteinte alors que la saisie n’est pas terminée, votre programme doit s’arrêter et afficher (où `[x]` désigne le contenu de `x`) :

```
STOP -- les [n] premières personnes entrent
```

Sinon il doit afficher :

```
Tout le monde entre dans l’ascenseur ([n] personnes)
```

Exemple : Exemples

- Pour les poids : 40 35 100 80 20 -1 (une famille avec trois enfants) :

```
Tout le monde entre dans l’ascenseur (5 personnes)
```

- Pour les poids : 100 80 110 40... :

```
STOP -- les 3 premières personnes entrent
```

8 Compte à rebours (5 points)

Il y a 60 minutes dans 1 heure et 60 secondes dans 1 minute.



(5 points) Écrivez un programme capable de simuler un compte à rebours, c.-à-d. de décompter, tout en l’affichant, une durée donnée sous la forme d’un nombre d’heures dans **hr**, d’un nombre de minutes dans **mn** et d’un nombre de secondes dans **ss** et d’afficher **Go** en fin de décompte.



Testez.

9 Kaprekar à trois chiffres (7 points)(Examen)



Objectif

En mathématiques, l’algorithme de KAPREKAR est un algorithme qui transforme un nombre entier en un autre, de façon répétitive jusqu’à arriver à un cycle. Cet exercice réalise l’algorithme de KAPREKAR pour un nombre à trois chiffres.

Algorithme de Kaprekar

Il consiste à associer à un entier n un autre entier $K(n)$ généré de la façon suivante :

1. On forme l’entier n_1 en arrangeant les chiffres de n dans l’ordre croissant et l’entier n_2 en les arrangeant dans l’ordre décroissant.
2. On pose $K(n) = n_2 - n_1$.
3. On itère ensuite le processus avec $K(n)$.

Exemple

Pour tout entier à trois chiffres, on obtient la constante 495 (cycle de longueur 1).

Par exemple :

```
644
==> 644-446=198
==> 981-189=792
==> 972-279=693
==> 963-369=594
==> 954-459=495
==> 954-459=495...
```

Rappel : Réordre de trois entiers

Il s’effectue en trois comparaisons :

```
ordonner a et b
ordonner b et c
ordonner a et b
```

où « ordonner x et y » signifie « avoir dans x la plus petite et dans y la plus grande des deux valeurs de x et y ».



(2 points) Écrivez un programme qui saisit un entier à trois chiffres dans `n`, c.-à-d. qui le demande et le saisit **tant qu'il n'est pas dans l'intervalle [100..999]**.

Affichez l’invite :

```
Votre entier (trois chiffres)?
```



Testez la saisie.



(1 point) Écrivez la boucle qui itère tant que l’entier `n` ne vaut pas 495. Dans celle-ci, calculez les entiers `a`, `b` et `c` représentant l’unité, la dizaine et la centaine de `n`. Rappel : Utilisez les opérations de division et de modulo par 10.



(2 points) Complétez votre boucle de sorte à réordonner `a`, `b` et `c` en ordre croissant. A l’issue du réordre, on aura donc :

```
a <= b <= c
```



(1 point) Déduisez alors les entiers n_2 (dans `n2`) et n_1 (dans `n1`) qui représentent le réordre décroissant, respectivement croissant, des chiffres de `n`, ainsi que le terme $K(n)$ (qui vaut `n` ; principe de récurrence).



(1 point) Enfin complétez votre programme de sorte à afficher la suite de KAPREKAR issue d’un entier. Vérifiez que vous affichez **tous** les termes.



Testez. Pour l’exemple ci-dessus :

```
Votre entier (trois chiffres)? 20
Retapez? 644
644 198 792 693 594 495
```

10 Nombre heureux (1) / pgheureux1 (6 points)



Objectif

Cet exercice détermine la liste des nombres heureux d’au plus quatre chiffres (c.-à-d. qui sont inférieurs ou égaux à 9999).



Définition

En mathématiques, un entier naturel est un **nombre heureux** si, lorsque l’on calcule la somme des carrés de ses chiffres dans son écriture en base dix puis la somme des carrés des chiffres du nombre obtenu et ainsi de suite, on aboutit à l’entier 1.



Propriété

On démontre qu’en appliquant un tel processus, à partir d’un entier quelconque non nul, on finit par boucler sur un des cycles suivants : {1} ou {4, 16, 37, 58, 89, 145, 42, 20}. Un **nombre** est **malheureux** quand il boucle sur le cycle long.

Exemples

```
49 ==> 4*4+9*9=16+81=97 ==> 9*9+7*7=130 ==> 1*1+3*3+0*0=10 ==> 1*1+0*0=1
6 ==> 36 ==> 45 ==> 41 ==> 17 ==> 50 ==> 25 ==> 29 ==> 85 ==> 89 ==> 145
      ==> 42 ==> 20 ==> 4
```



(2 points) Écrivez un programme qui demande et saisit un entier dans `nmax` tant qu’il n’est pas dans l’intervalle [1..9999]. Affichez l’invite :

Jusqu’ à `nmax`?



Testez la saisie.



(3 points) Déterminez et affichez tous les nombres heureux compris entre 1 et `nmax`.

Aide simple

Soit l’entier n à tester. Comme il est au plus composé de quatre chiffres, la somme des carrés de ses chiffres est un entier d’au plus quatre chiffres. Par conséquent,

1. Déterminez les chiffres `a` (unité), `b` (dizaine), `c` (centaine) et `d` (milliers) de `n`.
2. Calculez la somme des carrés des chiffres dans `n` (principe de récurrence).
3. Testez s’il est heureux.
4. Bouclez les étapes (1) à (3) tant que **non** (`n` vaut 1 **ou** `n` vaut 4).



(1 point) Complétez votre code de sorte à les compter.



Testez. Exemple d’exécution :

```
Jusqu'a nmax? 10000
```

```
Retapez? 100
```

```
1 7 10 13 19 23 28 31 32 44 49 68 70 79 82 86 91 94 97 100 nt= 20
```

11 Nombre heureux (2) / pgheureux2 (6 points)



Objectif

Cet exercice teste si un entier positif est ou non heureux et affiche les entiers de son cycle.



Définition

En mathématiques, un entier naturel est un **nombre heureux** si, lorsque l’on calcule la somme des carrés de ses chiffres dans son écriture en base dix puis la somme des carrés des chiffres du nombre obtenu et ainsi de suite, on aboutit à l’entier 1.



Propriété

On démontre qu’en appliquant un tel processus, à partir d’un entier quelconque non nul, on finit par boucler sur un des cycles suivants : {1} ou {4, 16, 37, 58, 89, 145, 42, 20}. Un **nombre** est **malheureux** quand il boucle sur le cycle long.

Exemples

```
49 ==> 4*4+9*9=16+81=97 ==> 9*9+7*7=130 ==> 1*1+3*3+0*0=10 ==> 1*1+0*0=1
6 ==> 36 ==> 45 ==> 41 ==> 17 ==> 50 ==> 25 ==> 29 ==> 85 ==> 89 ==> 145
      ==> 42 ==> 20 ==> 4
```



(2 points) Écrivez un programme qui demande et saisit un entier dans **n** tant qu’il est **négatif ou nul**. Affichez l’invite :

Entier a tester?



Testez la saisie.



(3 points) Affichez la suite des entiers issu de **n**.

Aide méthodologique

Tant que **non** (**n** vaut 1 ou **n** vaut 4) :

1. Soit **s** (somme des carrés des chiffres de **n**) valant initialement 0.
2. Décomposez **n** (voir ci-après) et cumulez la somme des carrés des chiffres dans **s**.
3. Assignez **s** à **n** (principe de récurrence).

Aide : Décomposition chiffre par chiffre d’un entier

Des « modulo et division » par 10 permettent d’obtenir un à un tous les chiffres d’un entier. Le tableau ci-dessous montre la méthode pour $n = 12306$.

Valeurs successives de l’entier n	Restes successifs	Quotients successifs
12306	6	1230
1230	0	123
123	3	12
12	2	1
1	1	0



(0.5 point) Complétez votre code de sorte à compter la longueur du cycle et affichez-le.



(0.5 point) Enfin, dans un booléen `b`, évaluez si l’entier est heureux (valeur `Vrai`) ou malheureux (valeur `Faux`) et affichez-le.



Testez. Exemples d’exécution :

```
Entier a tester? -12
Retapez? 1211
1211 7 49 97 130 10 1
nt= 7
Heureux= 1
```

```
Entier a tester? 123
123 14 17 50 25 29 85 89 145 42 20 4
nt= 12
Heureux= 0
```

(Le C++ affiche 1 pour `true` et 0 pour `false`.)

12 Multiplication égyptienne / pgegypt1 (6 points)



Objectif

La multiplication égyptienne consiste à multiplier deux entiers positifs en n’utilisant que l’addition, la multiplication par deux et la division par deux.

Méthode

Pour rechercher un algorithme, on recommande de procéder en s’inspirant de l’exemple suivant :

$$\begin{aligned}
 25 * 19 &= 25*18 + 25 \\
 &= 50*9 + 25 \\
 &= (50*8 + 50) + 25 = 50*8 + 75 \\
 &\quad = 100*4 + 75 \\
 &\quad = 200*2 + 75 \\
 &\quad = 400*1 + 75 \\
 &\quad = 400*0 + 475 = 475
 \end{aligned}$$

Soient donc A et B , les deux variables qui vont recevoir les deux entiers. Elles vont évoluer ensuite comme suit :

$$P = A \times B = \begin{cases} A \times (B - 1) + A & \text{si } B \text{ est impair} \\ (2A) \times (B/2) & \text{si } B \text{ est pair et non nul} \end{cases}$$



(2 points) Écrivez un programme qui demande et saisit deux entiers dans `a` et `b` **tant qu’ils ne sont pas tous deux positifs**. Affichez l’invite :

Vos deux entiers?



Testez la saisie.



(4 points) Effectuez et affichez la multiplication égyptienne comme dans l’exemple ci-dessous :

Vos deux entiers? 25 -19

Retapez? 25 19

25*19

$$\begin{aligned}
 &= 25*18 + 25 \\
 &= 50*9 + 25 \\
 &= 50*8 + 75 \\
 &= 100*4 + 75 \\
 &= 200*2 + 75 \\
 &= 400*1 + 75 \\
 &= 400*0 + 475 \\
 &= 475
 \end{aligned}$$



Testez.