

Faculté des Sciences
كلية العلوم
جامعة بنغازي

Département d'Informatique
Filière MIP - Semestre 2
2023-2024

Informatique II

Algorithmique II / Python

Pr. Khadija Louzaoui

Grandes lignes du cours

- ✚ Chapitre 1 : Algorithmique I
- ✚ Chapitre 2 : Les fonctions et les procédures
- ✚ Chapitre 3 : La récursivité
- ✚ Chapitre 4 : Les enregistrements et fichiers
- ✚ Chapitre 5 : La complexité
- ✚ Chapitre 6 : La preuves des algorithmes

Faculté des Sciences
كلية العلوم
جامعة بنغازي

Département d'Informatique
Filière MIP - Semestre 2
2023-2024

Module : Informatique II (Algorithmique II / Python)

Chapitre 3

La récursivité

Pr. Khadija Louzaoui

Chapitre 3


La récursivité

Introduction

La récursivité est un concept général qui peut être illustré dans (quasiment) tous les langages de programmation, et qui peut être utile dans de nombreuses situations. La définition la plus simple d'une fonction récursive est la suivante :

C'est une fonction qui s'appelle elle-même.

Si dans le corps (le contenu) de la fonction, vous l'utilisez elle-même, alors elle est récursive.




4

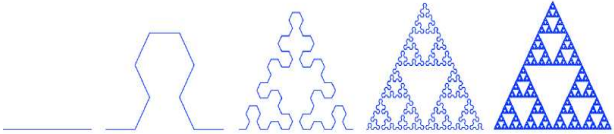
Chapitre 3 La récursivité

Introduction

Exemple : le dessin de la Vache qui rit



Autre exemple : le triangle de Sierpinski



5

Chapitre 3 La récursivité

Algorithmes Récursifs : Exemple

Exemple 1 : Factoriel d'un nombre

1- Ecrire une fonction récursive **fact(n)** permettant de calculer le factoriel d'un nombre entier naturel n.

2- Donner la trace d'exécution des appels récursifs pour n=4


Version itérative:

```

fonction fact ( n: entier): entier
Variables i,f: entier
Début
  f ← 1
  Pour i de 2 à n pas 1 faire
    f ← f* i
  FinPour
  Retourner (f)
Fin
                    
```

```

def fac_iterative(n):
    f=1
    for i in range(2,n+1):
        f=f*(i)
    return f
                    
```



```

>>> fac_iterative(10)
3628800
                    
```

6

Chapitre 3 La récursivité

Algorithmes Récursifs : Exemple

Exemple 1 : Factoriel d'un nombre

1- Ecrire une fonction récursive **fact(n)** permettant de calculer le factoriel d'un nombre entier naturel n.

fact(0)= fact(1)=1

Version itérative:	Version récursive:
<pre> fonction fact (n: entier): entier Variables i,f: entier Début f ← 1 Pour i de 2 à n pas 1 faire f ← f* i FinPour Retourner (f) Fin </pre>	<div style="border: 1px solid black; height: 100px;"></div>

7

Chapitre 3 La récursivité

Algorithmes Récursifs : Exemple

Exemple 1 : Factoriel d'un nombre

1- Ecrire une fonction récursive **fact(n)** permettant de calculer le factoriel d'un nombre entier naturel n.

fact(0)= fact(1)=1
fact(4)=4*3*2*1

Version itérative:	Version récursive:
<pre> fonction fact (n: entier): entier Variables i,f: entier Début f ← 1 Pour i de 2 à n pas 1 faire f ← f* i FinPour Retourner (f) Fin </pre>	<div style="border: 1px solid black; height: 100px;"></div>

8

Chapitre 3 La récursivité

Algorithmes Récursifs : Exemple

Exemple 1 : Factoriel d'un nombre

1- Ecrire une fonction récursive $fact(n)$ permettant de calculer le factoriel d'un nombre entier naturel n.

$$fact(0) = fact(1) = 1$$

$$fact(4) = 4 * 3 * 2 * 1$$

$$fact(5) = 5 * 4 * 3 * 2 * 1 = 5 * fact(4)$$

Version itérative:	Version récursive:
<pre> fonction fact (n: entier): entier Variables i, f: entier Début f ← 1 Pour i de 2 à n pas 1 faire f ← f * i FinPour Retourner (f) Fin </pre>	<pre> fonction fact (n: entier): entier Début Fin </pre>

9

Chapitre 3 La récursivité

Algorithmes Récursifs : Exemple

Exemple 1 : Factoriel d'un nombre

1- Ecrire une fonction récursive $fact(n)$ permettant de calculer le factoriel d'un nombre entier naturel n.

$$n! = \begin{cases} 1 & \text{si } n \leq 1 \\ n(n-1)! & \text{sinon} \end{cases}$$

Version itérative:	Version récursive:
<pre> fonction fact (n: entier): entier Variables i, f: entier Début f ← 1 Pour i de 2 à n pas 1 faire f ← f * i FinPour Retourner (f) Fin </pre>	<pre> fonction fact (n: entier): entier Début Fin </pre>

10

Chapitre 3 La récursivité

Algorithmes Récursifs : Exemple

Exemple 1 : Factoriel d'un nombre

1- Ecrire une fonction récursive $fact(n)$ permettant de calculer le factoriel d'un nombre entier naturel n.

$$n! = \begin{cases} 1 & \text{si } n \leq 1 \\ n(n-1)! & \text{sinon} \end{cases}$$

Version itérative:	Version récursive:
<pre> fonction fact (n: entier): entier Variables i, f: entier Début f ← 1 Pour i de 2 à n pas 1 faire f ← f * i FinPour Retourner (f) Fin </pre>	<pre> fonction fact (n: entier): entier Début Fin </pre>

11

Chapitre 3 La récursivité

Algorithmes Récursifs : Exemple

Exemple 1 : Factoriel d'un nombre

1- Ecrire une fonction récursive $fact(n)$ permettant de calculer le factoriel d'un nombre entier naturel n.

$$n! = \begin{cases} 1 & \text{si } n \leq 1 \\ n(n-1)! & \text{sinon} \end{cases}$$

Version itérative:	Version récursive:
<pre> fonction fact (n: entier): entier Variables i, f: entier Début f ← 1 Pour i de 2 à n pas 1 faire f ← f * i FinPour Retourner (f) Fin </pre>	<pre> fonction fact (n: entier): entier Début si (n<=1) alors Retourner 1 Sinon Fin </pre>

12

Chapitre 3 La récursivité

Algorithmes Récursifs : Exemple

Exemple 1 : Factoriel d'un nombre

1- Ecrire une fonction récursive $fact(n)$ permettant de calculer le factoriel d'un nombre entier naturel n.

$$n! = \begin{cases} 1 & \text{si } n \leq 1 \\ n(n-1)! & \text{sinon} \end{cases}$$

Condition d'arrêt

Version itérative:	Version récursive:
<pre> fonction fact (n: entier): entier Variables i, f: entier Début f ← 1 Pour i de 2 à n pas 1 faire f ← f* i FinPour Retourner (f) Fin </pre>	<pre> fonction fact (n: entier): entier Début si (n<=1) alors Retourner 1 Sinon Finsi Fin </pre>

13

Chapitre 3 La récursivité

Algorithmes Récursifs : Exemple

Exemple 1 : Factoriel d'un nombre

1- Ecrire une fonction récursive $fact(n)$ permettant de calculer le factoriel d'un nombre entier naturel n.

$$n! = \begin{cases} 1 & \text{si } n \leq 1 \\ n(n-1)! & \text{sinon} \end{cases}$$

Version itérative:	Version récursive:
<pre> fonction fact (n: entier): entier Variables i, f: entier Début f ← 1 Pour i de 2 à n pas 1 faire f ← f* i FinPour Retourner (f) Fin </pre>	<pre> fonction fact (n: entier): entier Début si (n<=1) alors Retourner 1 Sinon Retourner n*fact(n-1) Finsi Fin </pre>

14

Chapitre 3 La récursivité

Algorithmes Récursifs : Exemple

Exemple 1 : Factoriel d'un nombre

2- Donner la trace d'exécution des appels récursifs pour n=4

<pre> fonction fact (n: entier): entier Début si (n<=1) alors Retourner 1 Sinon Retourner n*fact(n-1) Finsi Fin </pre>	<pre> def fac_recursive(n): if (n<=1): return 1 else: return n*fac_recursive(n-1) </pre>
<pre> def fac_recursive(n:int)->int : if (n<=1): return 1 else: return n*fac_recursive(n-1) </pre>	

15

Chapitre 3 La récursivité

Algorithmes Récursifs : Exemple

Exemple 1 : Factoriel d'un nombre

2- Donner la trace d'exécution des appels récursifs pour n=4

Fact(4)

<pre> fonction fact (n: entier): entier Début si (n<=1) alors Retourner 1 Sinon Retourner n*fact(n-1) Finsi Fin </pre>	
--	--

16

Chapitre 3 La récursivité

Algorithmes Récursifs : Exemple

Exemple 1 : Factoriel d'un nombre

2- Donner la trace d'exécution des appels récursifs pour n=4

```

fonction fact ( n: entier): entier
Début
  si (n<=1) alors
    Retourner 1
  Sinon
    Retourner n*fact(n-1)
Finsi
Fin
        
```

```

graph TD
    A[Fact(4)] --> B[4*Fact(3)]
        
```

17

Chapitre 3 La récursivité

Algorithmes Récursifs : Exemple

Exemple 1 : Factoriel d'un nombre

2- Donner la trace d'exécution des appels récursifs pour n=4

```

fonction fact ( n: entier): entier
Début
  si (n<=1) alors
    Retourner 1
  Sinon
    Retourner n*fact(n-1)
Finsi
Fin
        
```

```

graph TD
    A[Fact(4)] --> B[4*Fact(3)]
    B --> C[3*Fact(2)]
        
```

18

Chapitre 3 La récursivité

Algorithmes Récursifs : Exemple

Exemple 1 : Factoriel d'un nombre

2- Donner la trace d'exécution des appels récursifs pour n=4

```

fonction fact ( n: entier): entier
Début
  si (n<=1) alors
    Retourner 1
  Sinon
    Retourner n*fact(n-1)
Finsi
Fin
        
```

```

graph TD
    A[Fact(4)] --> B[4*Fact(3)]
    B --> C[3*Fact(2)]
    C --> D[2*Fact(1)]
    D --> E[1]
        
```

19

Chapitre 3 La récursivité

Algorithmes Récursifs : Exemple

Exemple 1 : Factoriel d'un nombre

2- Donner la trace d'exécution des appels récursifs pour n=4

```

fonction fact ( n: entier): entier
Début
  si (n<=1) alors
    Retourner 1
  Sinon
    Retourner n*fact(n-1)
Finsi
Fin
        
```

```

graph TD
    A[Fact(4)] --> B[4*Fact(3)]
    B --> C[3*Fact(2)]
    C --> D["2= 2*Fact(1)"]
    D --> E[1]
    E --> D
        
```

20

Chapitre 3 La récursivité

Algorithmes Récursifs : Exemple

Exemple 1 : Factoriel d'un nombre

2- Donner la trace d'exécution des appels récursifs pour n=4

```

fonction fact ( n: entier): entier
Début
  si (n<=1) alors
    Retourner 1
  Sinon
    Retourner n*fact(n-1)
Finsi
Fin
        
```

21

Chapitre 3 La récursivité

Algorithmes Récursifs : Exemple

Exemple 1 : Factoriel d'un nombre

2- Donner la trace d'exécution des appels récursifs pour n=4

```

fonction fact ( n: entier): entier
Début
  si (n<=1) alors
    Retourner 1
  Sinon
    Retourner n*fact(n-1)
Finsi
Fin
        
```

22

Chapitre 3 La récursivité

Algorithmes Récursifs : Exemple

Exemple 1 : Factoriel d'un nombre

2- Donner la trace d'exécution des appels récursifs pour n=4

```

fonction fact ( n: entier): entier
Début
  si (n<=1) alors
    Retourner 1
  Sinon
    Retourner n*fact(n-1)
Finsi
Fin
        
```

23

Chapitre 3 La récursivité

Algorithmes Récursifs : Exemple

Exemple 2 : Suite de Fibonacci

1- Ecrire une fonction récursive *fibonacci(n)* permettant de calculer le nième nombre de la suite Fibonacci F_n définie comme suit:

$$\begin{cases} F_n = 1 \text{ si } n = 0 \text{ ou } n = 1 \\ F_n = F_{n-2} + F_{n-1} \text{ si } n > 1 \end{cases}$$

2- Donner la trace d'exécution des appels récursifs pour n=5

24

Chapitre 3 La récursivité

Algorithmes Récursifs : Exemple

Exemple 2 : Suite de Fibonacci

1- Ecrire une fonction récursive *fibonacci* permettant de calculer le nième nombre de la suite Fibonacci F_n définie comme suit:

$$\begin{cases} F_n = 1 & \text{si } n = 0 \text{ ou } n = 1 \\ F_n = F_{n-2} + F_{n-1} & \text{si } n > 1 \end{cases}$$

Expression récursive Cas de base

fibonacci(0)= fibonacci(1)=1
fibonacci(n)= fibonacci(n-2)+fibonacci(n-1)

fonction fibonacci (n: entier): entier

Début

si ((n=1) ou (n=0)) alors

retourne 1

sinon

retourne fibonacci(n-2) + fibonacci(n-1)

Finsi

Fin

def Fibonacci(n):

if n<=1:

return 1

else:

return Fibonacci(n-2) + Fibonacci(n-1)

25

Chapitre 3 La récursivité

Algorithmes Récursifs : Exemple

Exemple 2 : Suite de Fibonacci

1- Ecrire une fonction récursive *fibonacci* permettant de calculer le nième nombre de la suite Fibonacci F_n définie comme suit:

$$\begin{cases} F_n = 1 & \text{si } n = 0 \text{ ou } n = 1 \\ F_n = F_{n-2} + F_{n-1} & \text{si } n > 1 \end{cases}$$

Expression récursive Cas de base

fibonacci(0)= fibonacci(1)=1
fibonacci(n)= fibonacci(n-2)+fibonacci(n-1)

fonction fibonacci (n: entier): entier

Début

si ((n=1) ou (n=0)) alors

retourne 1

sinon

retourne fibonacci(n-2) + fibonacci(n-1)

Finsi

Fin

def Fibonacci(n:int)-> int :

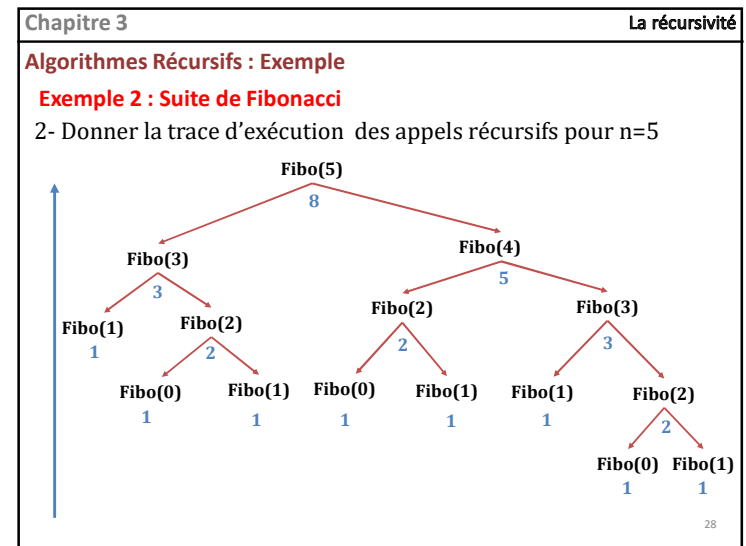
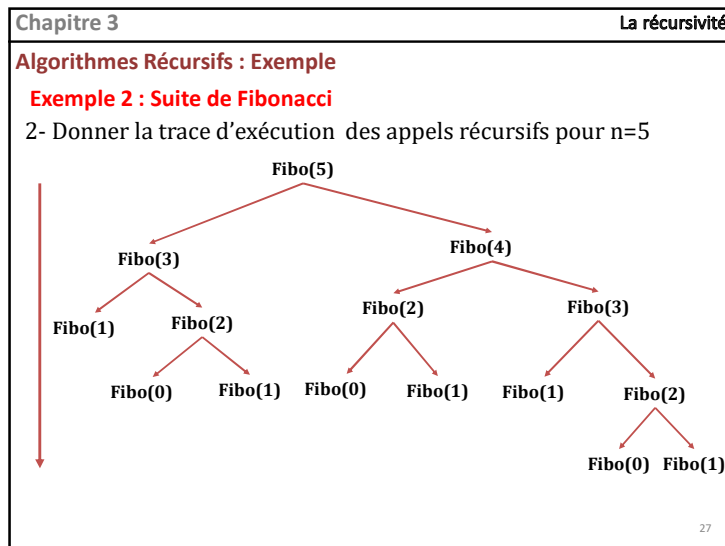
if n<=1:

return 1

else:

return Fibonacci(n-2) + Fibonacci(n-1)

26



Chapitre 3 La récursivité

Définition

- ❑ La **récursivité** est une méthode de description d’algorithmes qui permet à une fonction (ou procédure) de s’appeler elle-même directement ou indirectement.
- ❑ La formulation d'une solution récursive décrit plusieurs éléments.
 - Les **cas de base** : ces cas sont les **conditions d'arrêt** de la **chaîne des appels récursifs**.
 - Les **appels récursifs** eux-mêmes.
- ❑ La façon de formuler ces appels a un impact sur la **convergence** de la solution récursive.
 - Il faut que tout appel initial nous amène éventuellement à un des **cas de base**.
 - Pour ce faire, il faut que chaque appel nous rapproche des cas de base sans pour autant les dépasser.

29

Chapitre 3 La récursivité

Définition

- ❑ Généralement, la **condition d'arrêt** se présente sous la forme d’une instruction « **Si... Alors...Sinon** » qui permet de stopper la récurrence si la condition d’arrêt est satisfaite.
- ❑ Dans le cas contraire, la fonction ou la procédure continue à exécuter les appels récursifs.
- ❑ le paramètre de l’appel récursif doit converger toujours vers la condition d’arrêt.
- ❑ Un processus récursif remplace en quelque sorte une boucle, ainsi tout processus récursif peut être également formulé en tant qu’un processus itératif.

30

Chapitre 3 La récursivité

Critères de récursivité

- ❑ Expression récursive du problème :
C'est l' « équation » de la récursivité.
- ❑ Condition d’arrêt :
Quand est-ce qu’on arrête les appels récursifs ?
- ❑ Convergence (vers la condition d’arrêt):
Une petite « preuve » qui nous assure qu’un jour on va atteindre la condition d’arrêt.

31

Chapitre 3 Exercices

Types de récursivité

Récursivité non terminale

Une fonction récursive est dite **non terminale** si le **résultat de l'appel récursif est utilisé pour réaliser un traitement** (en plus du retour d'une valeur).

```
def facNT(n):
    if n==0:
        return 1
    return(n*facNT(n-1))
```

32

Chapitre 3 Exercices

Types de récursivité

Récursivité terminale

Une fonction récursive est dite **terminale** si aucun traitement n'est effectué à la remontée d'un appel récursif (sauf le retour d'une valeur). Concrètement il n'y a pas de calcul entre l'appel récursif et l'instruction return.

```
def fact(n,acc=1):
    if n==0:
        return acc
    return fact(n-1,n*acc)
```

Cette fois-ci, les calculs se font à la descente, comme c'est illustré ci-dessous, pour fact(3):

```
graph TD
    A[fact(3,1)] --> B[fact(2,3)]
    B --> C[fact(1,6)]
    C --> D[fact(0,6)]
```

33

Chapitre 3 Exercices

Types de récursivité

Récursivité simple

La **récursivité simple** où l'algorithme fait un seul appel récursif dans son corps.

Exemple : la fonction factorielle

```
fonction fact ( n: entier): entier
Début
    si (n<=1) alors
        Retourner 1
    Sinon
        Retourner n*fact(n-1)
    Finsi
Fin
```

34

Chapitre 3 Exercices

Types de récursivité

Récursivité multiple

La **récursivité multiple** où l'algorithme fait plusieurs appels récursifs dans son corps.

Exemple : la suite Fibonacci

```
fonction fibo ( n: entier): entier
Début
    si ((n=1) ou (n=0)) alors
        retourne 1
    sinon
        retourne fibo(n-2) + fibo(n-1)
    Finsi
Fin
```

35

Chapitre 3 Exercices

Types de récursivité

Récursivité imbriquée

La **récursivité imbriquée** consiste à faire un appel récursif à l'intérieur d'un autre appel récursif.

$$A(m,n) = \begin{cases} A(0,n) = n + 1 ; si m = 0 \\ A(m,0) = A(m-1,1) ; si n = 0 \\ A(m,n) = A(m-1, A(m,n-1)) ; si m \neq 0 \text{ et } n \neq 0 \end{cases}$$

Exemple : la suite d'Ackerman

```
fonction ackerman(n,m: entier ): entier
début
    si (m = 0) alors
        retourne n+1
    sinon
        si ((m>0) et (n=0)) alors
            retourne ackerman(m-1,1)
        sinon
            retourne ackerman(m-1,ackerman(m,n-1))
    fin
fin
```

36

Chapitre 3 Exercices

Types de récursivité

Récursivité imbriquée

La **récursivité imbriquée** consiste à faire un appel récursif à l'intérieur d'un autre appel récursif.

Exemple : la suite d'Ackerman

$$A(m,n) = \begin{cases} A(0,n) = n + 1 ; \text{si } m = 0 \\ A(m,0) = A(m-1,1) ; \text{si } n = 0 \\ A(m,n) = A(m-1, A(m,n-1)) ; \text{si } m \neq 0 \text{ et } n \neq 0 \end{cases}$$

```
def Ackermann(m,n:int)->int :
    if m==0:
        return n+1
    elif n==0:
        return Ackermann(m-1,1)
    else:
        return Ackermann(m-1, Ackermann(m,n-1))
```

Question : Que vaut Ackermann(2,2) ?

Réponse : 7

37

Chapitre 3 Exercices

Types de récursivité

Récursivité croisée ou mutuelle

La **récursivité croisée** consiste à écrire des fonctions qui s'appellent l'une l'autre (c.-à-d. un module P appelle un autre module Q qui fait à son tour un autre appel au module P).

Exemple : la définition de la parité d'un entier peut être écrite de la manière suivante :

$$Pair(n) = \begin{cases} vrai & \text{si } n = 0 \\ Impair(n-1) & \text{sinon} \end{cases}$$

et

$$Impair(n) = \begin{cases} faux & \text{si } n = 0 \\ Pair(n-1) & \text{sinon} \end{cases}$$

38

Chapitre 3 Exercices

Types de récursivité

Récursivité croisée ou mutuelle

Exemple :

```
fonction estPair(entier n):booléen
débüt
    si (m = 0) alors
        retourne VRAI
    sinon
        retourne estImpair(n-1)
    finsi
fin

fonction estImpair(entier n): booléen
débüt
    si (m = 0) alors
        retourne FAUX
    sinon
        retourne estPair(n-1)
    finsi
fin
```

39

Chapitre 3 La récursivité

Récursif versus itératif

- ❑ L'exécution d'une version récursive d'un algorithme est généralement un peu moins rapide que celle de la version itérative, même si le nombre d'instructions est le même (à cause de la gestion des appels de fonction).
- ❑ Il est souvent possible d'écrire un même algorithme en itératif et en récursif.

40

Chapitre 3		La récursivité
Récurif versus itératif		
PROPRIÉTÉ	RÉCURSION	ITÉRATION
Définition	La fonction s'appelle elle-même.	Un ensemble d'instructions exécutées de manière répétitive.
Application	Pour les fonctions.	Pour les boucles.
Terminaison	Par le cas de base, où il n'y aura pas d'appel de fonction.	Lorsque la condition de sortie de l'itérateur cesse d'être remplie.
Utilisation	Utilisé lorsque la taille du code doit être petite et que la complexité du temps ne pose pas de problème.	Utilisé lorsque la complexité temporelle doit être compensée par une taille de code plus importante.
Taille du code	Taille du code plus petite	Taille du code plus grande.
Complexité temporelle	Très grande (généralement exponentielle) complexité temporelle.	Complexité temporelle relativement plus faible (généralement polynomiale et logarithmique).

41

Chapitre 3	Exercices
Exercices	
1- Ecrire une fonction récurif et une autre itérative pour la calcul de la somme:	
$S_{(n,m)} = 1 + 2^m + 3^m + \dots + n^m$	


42

Chapitre 3	Exercices
Exercices	
$S_{(n,m)} = 1 + 2^m + 3^m + \dots + n^m$	$S_n = \sum_{i=0}^n i^m$
$S_{(n,m)} = f(S_{(n-1,m)}) ???$	$S_{(n,m)} = S_{(n-1,m)} + n^m$
<p>fonction somiter (n,m: entier): entier Variable i,som : entier Début som ← 0 Pour i de 1 à n alors som ← som+i^m FinPour Retourner (som) Fin</p>	<p>fonction somrec (n,m: entier): entier Début Si (n<=0) alors Retourne 0 Sinon Retourner (n^m+somrec(n-1)) FinSi Fin</p>

43

Chapitre 3	La récursivité
Exercices	
2- Ecrire une fonction récurif et une autre itérative qui calcul n ^{ième} terme de la suite (U _n) définie comme suit :	
$\begin{cases} U_0 = 2 \\ U_1 = 3 \\ U_n = \frac{2}{3}U_{n-1} - \frac{1}{4}U_{n-2} \end{cases}$	

44

Chapitre 3		La récursivité
<p>Exercices</p> <div style="display: flex; align-items: center; justify-content: center;"> <div style="margin-right: 20px;"> <p style="color: red;">Cas de base</p>  </div> <div style="margin-right: 20px;"> $U_n = \begin{cases} U_0 = 2 \\ U_1 = 3 \\ U_n = \frac{2}{3}U_{n-1} - \frac{1}{4}U_{n-2} \end{cases}$ </div> </div> <div style="text-align: center; margin-top: 10px;"> <div style="background-color: #f4a460; padding: 5px; display: inline-block; border: 1px solid black;"> suiterec(0)= 2 et suiterec(1)=3 </div> </div>		
<p style="text-align: center;">Version itérative:</p> <pre> fonction suiteiter (n: entier): réel Variables u,up,upp: réel Début upp ← 2 up ← 3 Pour i de 2 à n pas 1 faire u ← (2/3)*up-(1/4)*upp upp ← up up ← u finpour retourner u Fin </pre>	<p style="text-align: center;">Version récursive:</p> <pre> fonction suiterec (n: entier): réel Début Si (n==0) alors retourner 2 Sinon Si (n==1) alors retourner 3 Sinon retourner ((2/3)*suiterec(n-1)-(1/4)*suiterec(n-2)) Finsi Finsi Fin </pre>	