

LIF3

Plan Licence / Grippe
Présentation de l'UE
Modalité de Contrôle des Connaissances
Objectifs pédagogiques

1

Plan Licence

- Mise en place en septembre 2008
- **Enseignant Référent** qui suit votre évolution (2 rencontres)
- Toutes les UEs du L1 et L2 en Contrôle Continu Intégral
 - plusieurs notes pour vous évaluer
 - plus de notion d'examen et de seconde session d'examen
- Calendrier adapté (contrôle terminal la semaine du 18 janvier)
- Heures d'enseignements supplémentaire (2 séances de TD)
- Séances de soutien
 - sur avis des intervenants de TD
 - 3 séances sur le semestre
- Harmonisation des notes en fin de semestre

2

Licence Lyon1 - UE LIF3

N. Guin – F. Zara

Grippe A : H1N1

- Si vous souhaitez obtenir des informations sur le virus / les modes de transmission / les symptômes suivez le lien « consultez les consignes à suivre » sur SPIRAL.
- Si vous contractez le virus de la grippe A : H1N1
 - Connectez-vous sur SPIRAL
 - Déclarez-vous porteur du virus (suivez le lien « accédez au formulaire de déclaration de grippe »



3

Licence Lyon1 - UE LIF3 N. Guin – F. Zara

Grippe A H1N1 : conséquences

- Si vous avez la grippe :
 - Restez chez vous
 - Déclarez vous sur SPIRAL
 - Les notes des CC manqués seront neutralisées
 - Essayez de faire chez vous les exercices + mails à votre intervenant
- Si les enseignants sont touchés
 - Dans la mesure du possible des remplacements seront organisés
- Si l'Université ferme
 - Le programme sera abrégé
 - pas le temps matériellement de récupérer les séances annulées

4

Licence Lyon1 - UE LIF3

N. Guin – F. Zara

Présentation de l'UE

- Responsables de l'UE
 - Florence Zara – bâtiment Nautibus (semestre d'automne)
 - Nathalie Guin – bâtiment Nautibus (semestre de printemps)
- Site Web de l'UE pour le semestre d'automne 2009
 - <http://bat710.univ-lyon1.fr/~fzara/LIF3>
 - Planning (salles, horaires)
 - Répartition des groupes de TDs et TP
 - Supports des CMs, TDs et TP
 - Corrigés des TP
 - etc.

5

Licence Lyon1 - UE LIF3

N. Guin – F. Zara

Planning de l'UE

- Planning détaillé sur la page Web de LIF3
 - Début des TDs : semaine du 5 octobre
 - Début des TP : semaine du 12 octobre
- Attention
 - **Lundi 16/11** : TD de 11h45 à 13h15 à la place du cours
 - **Lundi 14/12** : TD de 11h45 à 13h15 à la place du cours
- Les créneaux de soutien mis sur le planning sont destinés aux étudiants convoqués

6

Licence Lyon1 - UE LIF3

N. Guin – F. Zara

Modalité de Contrôle des Connaissances

- Cette UE est en Contrôle Continu Intégral
 - **il n'y aura donc pas d'examen écrit pendant la « session d'examens », ni de seconde session**
- Le contrôle continu est constitué de plusieurs épreuves :
 - **interrogations écrites en début de TD** (coefficient 0,20)
 - TP noté en conditions d'examen (TP7, coefficient 0,25) : **mardi 1 décembre**
 - TP projet (TP 8 et 9, coefficient 0,15) : évaluation le **mardi 5 janvier en salle de TP**
 - épreuve écrite en amphi après le dernier cours (coefficient 0,4) : **semaine du 18 janvier**

7

Licence Lyon1 - UE LIF3

N. Guin – F. Zara

Objectifs pédagogiques de l'UE (1)

- Cours 1 : Algorithmes, début en Scheme
 - Algorithmes itératif vs. récursif
 - Syntaxe du langage de programmation Scheme
- Cours 2 : Complexité, listes
 - Complexité en temps et complexité en espace
 - Utilisation des listes en Scheme
- Cours 3 : Let
 - Mémorisation
- Cours 4 : Tris
 - Tris par insertion et par fusion

8

Licence Lyon1 - UE LIF3

N. Guin – F. Zara

Objectifs pédagogiques de l'UE (2)

- Cours 5 : Récursivité profonde
 - Plusieurs niveaux de récursivité
- Cours 6 : Arbres
 - Utilisation des arbres en Scheme
- Cours 7 : Programmation d'ordre supérieur
 - Fonctions en arguments, fonctions en résultats
 - Abstraction
- Cours 8 : Logique
 - Logique des propositions
 - Algèbre de Boole

9

Licence Lyon1 - UE LIF3

N. Guin – F. Zara

Algorithmique et programmation fonctionnelle et récursive

Algorithme itératif / récursif
Programmation impérative / fonctionnelle

10

Qu'est-ce qu'un algorithme ?

On souhaite résoudre le problème :

Trouver le minimum d'une liste de nombres :
par exemple (3 6,5 12 -2 0 7)

- ➔ Expliquer à une machine comment trouver le minimum
de n'importe quelle liste de nombres
- Langage commun entre la machine et nous : Scheme

11

Licence Lyon1 - UE LIF3

N. Guin – F. Zara

Définition d'un algorithme

- Un algorithme est une méthode
 - Suffisamment générale pour pouvoir traiter toute une classe de problèmes
 - Combinant des opérations suffisamment simples pour être effectuées par une machine

12

Licence Lyon1 - UE LIF3

N. Guin – F. Zara

Rappel : la machine n'est pas intelligente

- L'exécution d'un algorithme ne doit normalement pas impliquer des **décisions subjectives**, ni faire appel à l'utilisation de l'**intuition** ou de la **créativité**
- Exemple : une recette de cuisine n'est pas un algorithme si elle contient des notions vagues comme « ajouter de la farine »

13

Licence Lyon1 - UE LIF3

N. Guin – F. Zara

Mon premier algorithme

- Déterminer le minimum d'une liste de nombres
 - par exemple

(3 6,5 12 -2 0 7)

14

Licence Lyon1 - UE LIF3

N. Guin – F. Zara

Méthode itérative

- Je parcours la liste de gauche à droite, en retenant à chaque pas le minimum « pour l'instant »

(3 6,5 12 -2 0 7)

- Entre 3 et 6,5, c'est 3
- Entre 3 et 12, c'est 3
- Entre 3 et -2, c'est -2
- Entre -2 et 0, c'est -2
- etc.

15

Licence Lyon1 - UE LIF3

N. Guin – F. Zara

Algorithme itératif

Soient - L la liste

- premier, longueur, ième et écrire des primitives (*i.e.* algorithmes déjà définis)

Définition de minimum(L)

Début

min ← premier(L)

i ← 2

TantQue i ≤ longueur(L) Faire

Si ième(L,i) < min Alors

min ← ième(L,i)

FinSi

i ← i+1

FinTantQue

Écrire(« Le minimum est »)

Écrire(min)

Fin

16

Licence Lyon1 - UE LIF3

N. Guin – F. Zara

Méthode récursive (1)

- Pour trouver le minimum de la liste (3 6,5 12 -2 0 7)
- On suppose le problème résolu pour la liste privée de son premier élément *i.e.* (6,5 12 -2 0 7)
- Soit **min** le minimum de cette sous-liste, ici -2
- Le minimum de la liste complète s'obtient par comparaison entre le premier élément de la liste (ici 3) et min (ici -2)

17

Licence Lyon1 - UE LIF3

N. Guin – F. Zara

Méthode récursive (2)

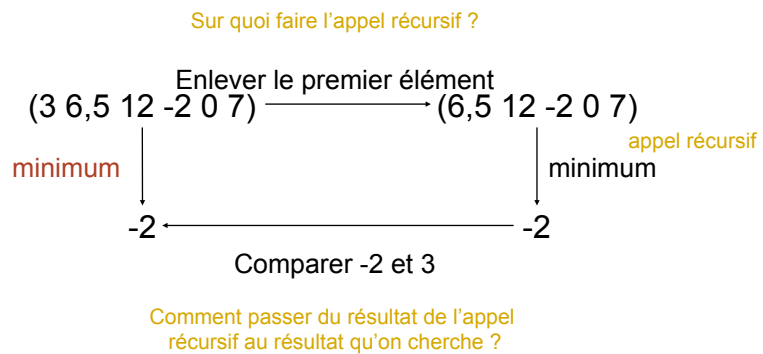
- Comment résout-on le problème pour la sous-liste ?
En faisant le même raisonnement.
C'est la **récurtivité**.
- Quand s'arrête-t-on ?
Quand on ne peut plus parler de sous-liste, *i.e.* quand la liste a un seul élément. C'est alors le minimum.

18

Licence Lyon1 - UE LIF3

N. Guin – F. Zara

Illustration de la méthode



19

Licence Lyon1 - UE LIF3

N. Guin – F. Zara

Algorithme récursif

Soient vide?, reste et premier des fonctions primitives

Définition de la fonction **minimum** (L)

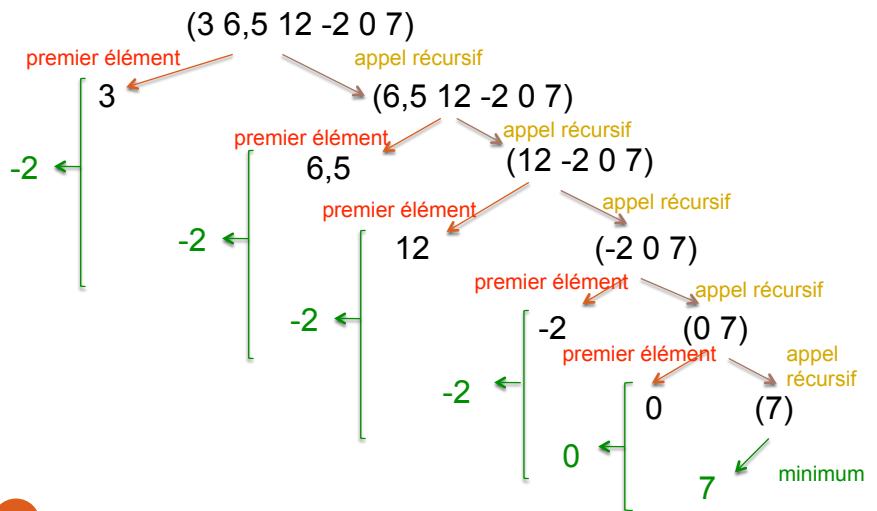
```
Si vide?(reste(L)) Alors
    retourne premier(L)
Sinon
    Si premier(L) < minimum(reste(L)) Alors
        retourne premier(L)
    Sinon
        retourne minimum(reste(L))
FinSi
FinSi
```

20

Licence Lyon1 - UE LIF3

N. Guin – F. Zara

Illustration de l'algorithme



21

Licence Lyon1 - UE LIF3

N. Guin - F. Zara

Remarques

- Minimum est ici une fonction, le mot **retourne** permet de dire quel est son résultat
- **Minimum** est l'**appel récursif**
- En programmation fonctionnelle, on n'a plus besoin de la séquence
- En programmation récursive, on n'a plus besoin de la boucle

22

Licence Lyon1 - UE LIF3

N. Guin - F. Zara

Pour écrire un algorithme récursif

- Il faut choisir
 - Sur quoi on fait l'appel récursif
 - Comment on passe du résultat de l'appel récursif au résultat que l'on cherche
 - Le cas d'arrêt

23

Licence Lyon1 - UE LIF3

N. Guin – F. Zara

Structure type d'une fonction récursive

Si je suis dans le cas d'arrêt
Alors voilà le résultat
Sinon le résultat est le résultat de
l'application d'une fonction sur
le résultat de l'appel récursif

24

Licence Lyon1 - UE LIF3

N. Guin – F. Zara

Les bugs

- Mon algorithme est faux car son résultat n'est pas défini si la liste est vide ou si elle contient autre chose que des nombres
- Pour éviter les bugs il faut :
 - Définir le problème aussi bien que possible
C'est la **spécification**
 - Tenter de prouver que son programme répond à la spécification
 - Passer des **jeux d'essai**, aussi divers et tordus que possible

25

Licence Lyon1 - UE LIF3

N. Guin – F. Zara

Pour résumer

LIF1 : Programmation

Langage C



LIF3 : Programmation

Langage Scheme

Impérative

Séquence
(faire les choses
l'une après l'autre)



Fonctionnelle

Appliquer une fonction à des arguments pour obtenir un résultat
Composer les fonctions pour enchaîner les traitements

Itérative

Boucle
(répéter)



Réursive

La répétition est assurée par l'enchaînement des appels récursifs
Le test de la boucle est remplacé par le cas d'arrêt

26

N. Guin – F. Zara

Débuts en Scheme

Évaluer une expression
Définir une fonction

27

Notion de fonction

- Une fonction prend des **arguments** et retourne **un résultat**
- Arguments et résultats peuvent être de n'importe quel **type** :
 - Nombre
 - Booléen
 - Caractère
 - Chaîne de caractères
 - Liste
 - Fonction

28

Licence Lyon1 - UE LIF3

N. Guin – F. Zara

Écriture de l'appel à une fonction (1)

- Syntaxe :
 - Parenthèse ouvrante
 - Nom de la fonction
 - Espace (NomFct Arg1 Arg2 ... Argn)
 - Premier argument
 - Espace
 - Deuxième argument
 - Etc
 - Parenthèse fermante

29

Licence Lyon1 - UE LIF3

N. Guin – F. Zara

Écriture de l'appel à une fonction (2)

- Sémantique : il faut donner à la fonction le bon nombre d'arguments, et du bon type
- Exemples :
 - (+ 5 13) retourne 18
 - (- 10 b) retourne la différence si b a une valeur numérique, une erreur sinon
 - (+ (* 2 5) (- 3 1)) retourne 12
 - (* 5) n'est pas correct
 - (/ 5 "a") non plus

30

Licence Lyon1 - UE LIF3

N. Guin – F. Zara

Évaluation de l'appel à une fonction

- Lorsqu'on lui fournit un appel de fonction, Scheme
 - Évalue chacun des arguments
 - Regarde s'il connaît la fonction, sinon affiche un message d'erreur
 - Applique la fonction aux résultats de l'évaluation des arguments
 - Affiche le résultat
- C'est un processus récursif

31

Licence Lyon1 - UE LIF3

N. Guin – F. Zara

Exemples d'erreurs

- $(1 + 2)$ → erreur : 1 n'est pas une fonction
- $(\text{toto } (1\ 2\ 3))$ → erreur : 1 n'est pas une fonction
- Dans certains cas particuliers, les arguments ne sont pas évalués avant l'application de la fonction. On parle alors de **forme spéciale** plutôt que de fonction

32

Licence Lyon1 - UE LIF3

N. Guin – F. Zara

Les variables

- Dans le langage Scheme, une variable se nomme **symbole**
- L'affectation revient à attribuer une **valeur** à un symbole. On utilise pour cela la forme spéciale **define**
- Exemples :
 - (define a 5)
 - (define b 2)
 - (+ a b) → 7

33

Licence Lyon1 - UE LIF3

N. Guin – F. Zara

La forme spéciale quote

- La forme spéciale **quote** permet d'empêcher l'évaluation
- Exemples :
 - (define a 5)
 - a → 5

 - (quote a) → a
 - (quote (+ 1 2)) → (+ 1 2)
 - '(+ 1 2) → (+ 1 2)

34

Licence Lyon1 - UE LIF3

N. Guin – F. Zara

La forme spéciale eval

- À l'inverse de `quote`, `eval` force l'évaluation

- Exemples :

```
(eval '(+ 3 2)) → 5
```

```
(define toto 5)
```

```
(define tata toto)
```

```
tata → 5
```

```
(define titi 'toto)
```

```
titi → toto
```

```
(eval titi) → 5
```

35

Licence Lyon1 - UE LIF3

N. Guin – F. Zara

Définition d'une fonction

- Syntaxe :

```
(define fonction  
  (lambda (liste-des-arguments  
          corps-de-la-fonction))
```

- Exemple :

```
(define plus-1  
  (lambda (x)  
    (+ x 1)))
```

- Test : `(plus-1 3) → 4`

36

Licence Lyon1 - UE LIF3

N. Guin – F. Zara

Spécification d'une fonction

; description de ce que fait la fonction
(define fonction ; -> type du résultat
 (lambda liste-des-arguments ; type des arguments
 corps-de-la-fonction))

- Exemple :
 ; ajoute 1 à un nombre
 (define plus-1 ; -> un nombre
 (lambda (x) ; x un nombre
 (+ x 1)))

37

Licence Lyon1 - UE LIF3

N. Guin – F. Zara

L'alternative

- L'alternative est définie en Scheme par la forme spéciale **if**
- Syntaxe :
(if condition valeur-si-vrai valeur-si-faux)
- Exemples :
(if (> 3 2) 'yes 'no) → yes
(if (> 2 3) 'yes 'no) → no
(if (> 3 2) (- 3 2) (+ 3 2)) → 1

38

Licence Lyon1 - UE LIF3

N. Guin – F. Zara

Quelques fonctions prédéfinies (1)

- Opérateurs arithmétiques :
+, -, *, /, `sqrt`, `min`, `max`, `abs`, etc.
(`sqrt 9`) → 3
(`min 5 2 1 3`) → 1
- Opérateurs booléens : `not`, `or`, `and`
(`not #t`) → `#f`
(`and (> 3 2) (> 2 5)`) → `#f`
(`or (> 3 2) (> 2 5)`) → `#t`

39

Licence Lyon1 - UE LIF3

N. Guin – F. Zara

Quelques fonctions prédéfinies (2)

- Opérateurs de comparaison :
 - =, <, >, <=, >= pour les nombres
 - `eq?` pour tout sauf les listes et les chaînes de caractères
 - `equal?` pour tout y compris les listes
- Pour tester le type d'un objet :
`boolean?`, `number?`, `symbol?`, `string?`

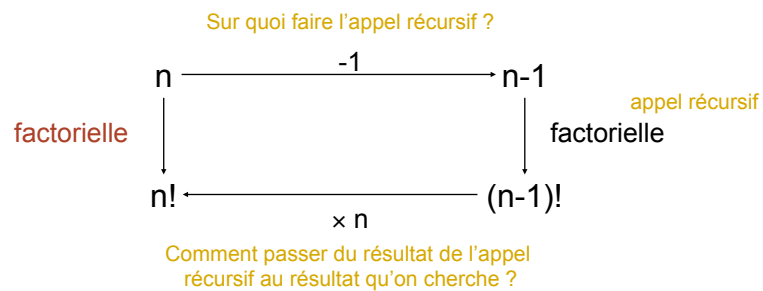
40

Licence Lyon1 - UE LIF3

N. Guin – F. Zara

Ma première fonction récursive : choix de la méthode

- On veut écrire une fonction qui étant donné un entier n calcule $n!$



41

Licence Lyon1 - UE LIF3

N. Guin – F. Zara

Ma première fonction récursive : écriture

- Cas d'arrêt : $0! = 1$
- Récursivité : $n! = 1 \times 2 \times 3 \times \dots \times n = n \times (n-1)!$

```
(define factorielle ; → entier positif
  (lambda (n) ; n entier positif
    (if (= n 0)
        1
        (* n (factorielle (- n 1))))))
```

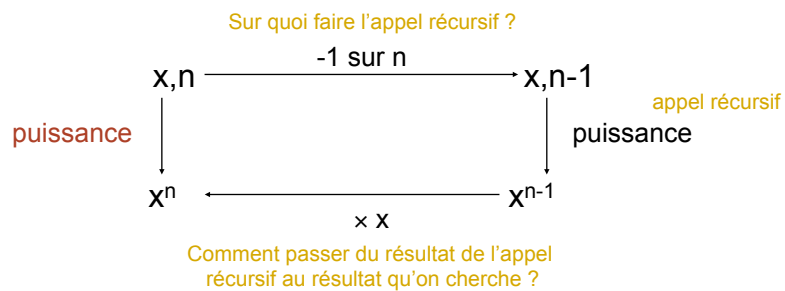
42

Licence Lyon1 - UE LIF3

N. Guin – F. Zara

Une autre fonction récursive : choix de la méthode

- On veut écrire une fonction qui étant donné un nombre x et un entier positif n calcule x^n



43

Licence Lyon1 - UE LIF3

N. Guin – F. Zara

Une autre fonction récursive : écriture

- Cas d'arrêt : $x^0 = 1$
- Récurtivité : $x^n = x \times x \times \dots \times x \times x = x \times x^{(n-1)}$

```
(define puissance ; → nombre
  (lambda (x n) ; x nombre, n entier positif
    (if (= n 0)
        1
        (* x (puissance x (- n 1))))))
```

44

Licence Lyon1 - UE LIF3

N. Guin – F. Zara