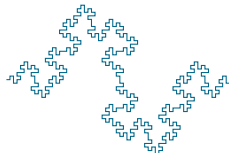


HLIN403 – Programmation Applicative

Christophe Dony – Annie Chateau

Université de Montpellier – Faculté des Sciences



INTRODUCTION

PROGRAMMATION APPLICATIVE

SCHEME

RAPPELS

CONTENU DU COURS

OUVRAGES LIÉS

- ▶ “Passeport pour l’algorithmique et Scheme” de R.Cori et P.Casteran.
- ▶ “Structure and Interpretation of Computer Programs” de Abelson, Fano, Sussman.
- ▶ “Programmation récursive (en Scheme)” de Christian Queinnec

PROGRAMMATION APPLICATIVE

- ▶ Programmation dans laquelle un programme est un ensemble de **définitions** de fonctions
- ▶ L'exécution d'un programme est une succession d'**applications** de fonctions à des arguments au sens algébrique du terme (calcul utilisant des opérations, des lettres et des nombres).
- ▶ Programmation où la mémoire utilisée par les calculs est automatiquement gérée par l'**interpréteur** du langage.

PROGRAMMATION RÉCURSIVE

- ▶ **Fonction récursive** : C'est une fonction capable de s'appeler elle-même.
- ▶ Les fonctions récursives sont idéales pour implanter un algorithme pour tout problème auquel peut s'appliquer une résolution par récurrence, ou pour traiter des structures de données récursives (listes, arbres).
- ▶ Omniprésentes en info et dans le web

APPLICATIONS

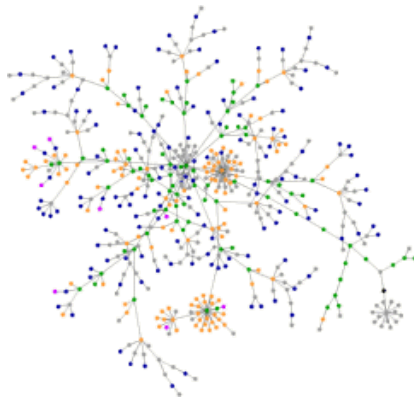


FIGURE – Web 3.0 - The Semantic, Implicit, Mobile or Distributed Web ? Jonas Bolinder 2008

APPLICATIONS



FIGURE – by Naughty Dog (*Crash Bandicoot*, *Uncharted*)

LISP, SCHEME

- ▶ Langages dédiés à la programmation applicative
- ▶ Langage de programmation universel, historiquement majeur, permettant notamment la programmation applicative.
- ▶ Syntaxe simple
- ▶ Fort pouvoir d'expression

LISP, SCHEME

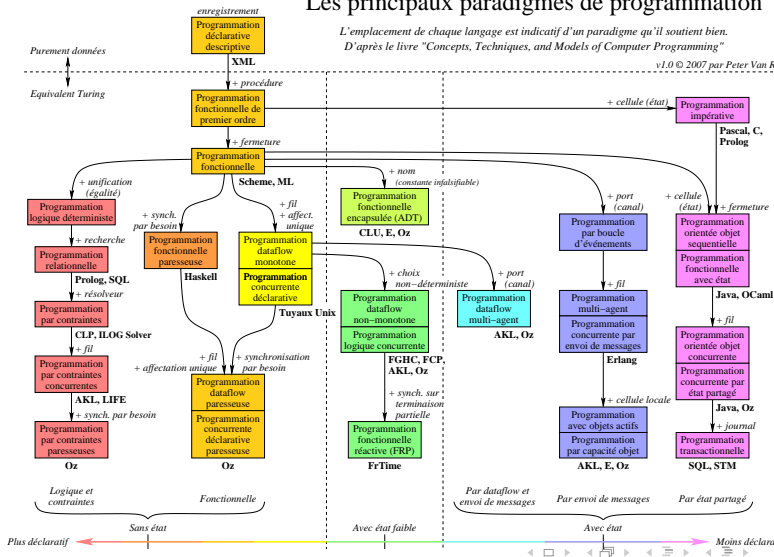
- ▶ **Lisp**, 1960, John Mc Carthy : pour du calcul numérique classique et du calcul symbolique (calcul dont les données ne sont pas des nombres mais des chaînes de caractères, des mots, des collections de choses).
- ▶ **Scheme**, 1970, Gerald J. Sussman et Guy L. Steele :
Héritier de LISP à liaison strictement lexicale :
enseignement de la programmation.
“Structure and Interpretation of Computer Programs -
Abelson, Fano, Sussman - MIT Press - 1984”.
- ▶ Excellent préalable à l'étude des langages à objets, plus complexes.

LISP, SCHEME

Les principaux paradigmes de programmation

L'emplacement de chaque langage est indicatif d'un paradigme qu'il soutient bien.
D'après le livre "Concepts, Techniques, and Models of Computer Programming"

v1.0 © 2007 par Peter Van Roy



CARACTÉRISTIQUES DES LANGAGES APPLICATIFS

- ▶ Toute phrase syntaxiquement correcte du langage (y-compris une instruction) est une expression algébrique ayant une valeur. Ceci n'est pas vrai dans les langages dits "impératifs".
- ▶ La mémoire est gérée automatiquement (allocation et récupération) si on le souhaite (programmation sans **Effets de bords**, lien avec la programmation dite fonctionnelle - Voir langage Haskell).

CARACTÉRISTIQUES DES LANGAGES APPLICATIFS

- Un très grand nombre d'abstractions simplifient la vie du programmeur. Exemple : les grands nombres, les itérateurs, etc.

```
> (fact 30)
```

```
265252859812191058636308480000000
```

```
> (map fact '(2 3 4 30))
```

```
(2 6 24 265252859812191058636308480000000)
```

RAPPELS GÉNÉRAUX SUR LA PROGRAMMATION

Informatique : De « INFORmation AutoMATIQUE ». Mot inventé en 1962 par P. Dreyfus.

L'informatique est la science du traitement automatisé de l'information. Le calcul au sens arithmétique du terme est un cas particulier de traitement de l'information.

« science du traitement rationnel, notamment par machines automatiques, de l'information considérée comme le support des connaissances humaines et des communication dans les domaines techniques, économiques et sociaux » (académie française).

RAPPELS GÉNÉRAUX SUR LA PROGRAMMATION

Traitement automatisé de l'information implique :

- ▶ codage de l'information pour sa représentation dans une machine.
- ▶ traitement de l'information selon une suite de calculs ou de transformations définis par un texte appelé programme

Données : entité manipulées dans les programmes.

programme : texte décrivant sous une forme interprétable par un ordinateur, une suite d'actions à réaliser sur des données.

Ordinateur : machine où un programme (un texte) est transformé en exécuté (transformé en actions).

RAPPELS GÉNÉRAUX SUR LA PROGRAMMATION

Algorithme¹ : Séquence d'opérations visant à la résolution d'un problème en un temps fini (mentionner la condition d'arrêt). Fondé sur la thèse de Church.

exemple : algorithme de l'école primaire pour multiplier deux nombres.

Processus, ou processus de calcul, ou Calcul : Exécution d'un programme par une machine.

Texte d'un programme : texte, écrit dans un langage de programmation, destiné à faire exécuter des actions à un ordinateur.

1. Terme venu du XIII^{ème} siècle, de la traduction en latin d'un mémoire de Mohammed Ibn Musa Abu Djefar Al-Khwarizmi commençant par : « Algoritmi dixit... ».

RAPPELS GÉNÉRAUX SUR LA PROGRAMMATION

Syntaxe : ensemble de règles de combinaison correcte des mots d'un langage pour former des phrases. (Syntaxe du français ou syntaxe de Lisp).

Grammaire : une représentation par intention, de la syntaxe d'un langage.

Analyse syntaxique, peut utiliser une grammaire, et dit si une phrase d'un langage est ou non syntaxiquement bien formée.

Sémantique : ensemble de règle définissant le sens d'une phrase.

Il possible d'écrire un texte syntaxiquement correct qui n'a pas de sens selon une sémantique donnée. Exemple en français :

“Quelle la différence entre un pigeon ?”

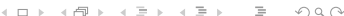
RAPPELS GÉNÉRAUX SUR LA PROGRAMMATION

Interpréteur : programme transformant un texte de programme syntaxiquement correct en actions. L'interpréteur fait faire à l'ordinateur ce que dit le texte.

Compilateur : Traducteur de texte d'un langage L1 vers un langage L2. Par exemple, compilateur de Java en instructions de la JVM, compilateur de C en assembleur.

Modèle de calcul : modèle permettant de représenter, afin de comprendre, mesurer prouver, la façon dont un programme est exécuté.

Exemples de modèles de calcul : machine de Turing, lambda-calcul²

2. Church-Alonso, (14/06/1903 - 11/08/1995). 

CONTENU DU COURS

- ▶ Syntaxe de *Scheme*
- ▶ Types prédéfinis. Base de l'interprétation des expressions.
- ▶ Fonctions, Identificateurs, Premières Structures de contrôle.
- ▶ Fonctions Récursives Simples.
- ▶ Listes - Symboles - Calcul Symbolique
- ▶ Fonctions récursives sur les listes. Récursions arborescentes
- ▶ Optimisation des fonctions récursives
- ▶ Abstraction de données - Arbres binaires
- ▶ Introduction à l'interprétation des langages