Algorithmique fonctionnelle - Principes de base

2 octobre 2009

Présentation

Objectifs du cours

- Ecrire la solution à un problème donné : **algorithmes**
- Résoudre un problème en utilisant des **fonctions** : **algorithmique fonctionnelle**
- Implémentation des algorithmes en langage **Scheme**
- Développement d'applications

Concepts de base

Définitions

- **Algorithme** : Enoncé d'une suite d'instructions permettant de donner la réponse à un problème
- **Algorithmique** : L'ensemble des techniques qui sont impliquées dans la définition et la conception des algorithmes

Quelques illustres prédecesseurs

Euclide, Archimède, Al Khuwarizmi, Fibonacci, Hilbert, Turing, Church, Goedel, Von Neumann, Knuth, Karp, ...

Concepts de base

Programme - Définitions

- Une suite d'instructions indiquant, dans un langage compréhensible par la machine, à l'ordinateur ce qu'il doit faire
- Une relation fonctionnelle entre deux ensembles (données en entrée et en sortie)

Concepts de base

Algorithme

- indépendant du langage de programmation
- écrit en pseudo-code

Concepts de base

Critères d'évaluation

- Correction
- Temps de calcul
- Taille mémoire
- Consommation d'énergie
- Machine : séquentielle, parallèle, distribué
- Exact ou approché ?, Prévisible ou imprévisible ? ...

Variables

Présentation

- Définies par le développeur
- Stocker temporairement des valeurs
- Données issues du disque dur, saisies par l'utilisateur, calculées par une autre partie du programme, ...
- Image d'une boite contenant une donnée accessible via une étiquette, son **nom**
- Contient une donnée d'un certain **type**

Variables

Types de variables

- Entier (ex : 12)
- Réel (ex : 19,6)
- Chaine (ex : "toto")
- Booléen (ex : faux)

Syntaxe

nom : type

Déclaration

```
a: Entier
b: Réel
b: Chaîne
d: Booléen
```

Expressions

Définition

Une expression est un ensemble de valeurs reliées par des opérateurs et équivalente à une seule valeur.

Opérateurs

Ils sont liés aux types des valeurs qu'ils manipulent :

- Entier : $+, -, *, /, div, mod$
- Réel : $+, -, *, /$
- Booléen : *non*, *et*, *ou*
- Chaîne : &
- Divers :
  - Parenthèses (gestion des priorités) : $(,)$
  - Comparaison (résultats booléens) : $=, \neq, >, <, \geq, \leq$

Notez : Des règles de priorités implicites sont liées aux différents opérateurs.

Expressions

Exemples

Considérons deux variables a et b ayant respectivement les valeurs 2 et 3

- $a * 5/2 + b$ aura pour valeur : 8
- $a * 5/(2 + b)$ aura pour valeur : 2
- $"toto" \& "titi"$ aura pour valeur : $"tototiti"$

Affectations

Pourquoi

L'affectation permet d'associer une valeur à une variable

Comment : instruction d'affectation

Syntaxe : *variable* ← *expression*

Remarques

- Seule la partie gauche d'une affectation est modifiée
- L'expression qui est affectée à la variable doit être d'un type compatible avec cette variable
- La dernière affectation d'une variable écrase sa valeur précédente

Affectations

Exemples

$a \leftarrow 5$

$b \leftarrow 6$

$a \leftarrow b$

$b \leftarrow 2 * a + b$

Exercice

Comment permuter les valeurs de 2 variables ?

Affectations

Exemples

$a \leftarrow 5$

$b \leftarrow 6$

$a \leftarrow b$

$b \leftarrow 2 * a + b$

Exercice

Comment permuter les valeurs de 2 variables ?

$a \leftarrow 2$

$b \leftarrow 5$

$a \leftarrow b$

$b \leftarrow a$

Affectations

Exemples

$a \leftarrow 5$

$b \leftarrow 6$

$a \leftarrow b$

$b \leftarrow 2 * a + b$

Exercice

Comment permuter les valeurs de 2 variables ?

$a \leftarrow 2$

$b \leftarrow 5$

$a \leftarrow b$

$b \leftarrow a$

... n'est pas correct ! Comment résoudre le problème ?

Variables booléennes

Pourquoi

- Stocker une information pouvant prendre 2 valeurs (en générale opposées) : vrai ou faux
- Permettre une alternative conditionnelle lors d'une suite d'instructions (structures de contrôles)

Variables booléennes

Rappels sur la logique booléenne

- valeurs : vrai/faux
- opérateurs : non, et, ou
- associativité : a et (b et c) = (a et b) et c
- commutativité : a ou b = b ou a

Variables booléennes

Rappels sur la logique booléenne

Distributivité

- a ou (b et c) = (a ou b) et (a ou c)
- a et (b ou c) = (a et b) ou (a et c)

Involution

- non non a = a

Loi de Morgan

- non (a ou b) = non a et non b
- non (a et b) = non a ou non b

Variables booléennes

Opérateurs

ET	*V*	*F*
V	*V*	*F*
F	*F*	*F*

OU	*V*	*F*
V	*V*	*V*
F	*V*	*F*

XOR	*V*	*F*
V	*F*	*V*
F	*V*	*F*

Variables booléennes

Exemple

$a \leftarrow vrai$
$b \leftarrow faux$
$c \leftarrow a \ ou \ b$
$d \leftarrow 4$
$e \leftarrow (d = 3) \ ou \ c$

Entrées sorties

Lecture

Entrer des valeurs externes (utilisateur) pour qu'elles soient utilisées dans la suite de l'algorithme. Implémenté en général via une saisie au clavier.

Ecriture

Communiquer des valeurs vers l'extérieur (utilisateur). Implémenté en général via un affichage à l'écran.

Syntaxe

Ecrire("Entrer une valeur")
Lire(val)

Entrées sorties

Exemple

Programme Exemple
Variables
  nb : Entier
Début
  Ecrire(”‘Entrer une valeur”’)
  Lire(nb)
  // nb contient la valeur saisie
  nb ← nb*2
  // on remplace la valeur de nb
  Ecrire(nb)
  // on affiche à l'écran le résultat
Fin

Instructions conditionnelles

Pourquoi

Les instructions conditionnelles permettent d'offrir lors du déroulement d'une suite d'instructions, une alternative selon la valeur d'un test booléen, **la condition**.

Comment : Syntaxe

```
Si Condition Alors
  Instructions
FinSi

Si Condition Alors
  Instructions si condition vraie
Sinon
  Instructions si condition fausse
FinSi
```

Instructions conditionnelles

Exemple

```
Programme Température
Variables
   temp: Entier
Début
   Ecrire("Entrez la température de l'eau")
   Lire(temp)
   Si temp <= 0 Alors
      Ecrire("C'est de la glace")
   Sinon
      Ecrire("C'est du liquide")
   FinSi
Fin
```

Assertions

Pourquoi

Améliorer la lisibilité d'un algorithme en ajoutant des expressions logiques (booléennes) toujours vraies décrivant formellement les instructions.

Comment

Les assertions seront entourées d'accolades { et }.
Elles sont toujours construites à partir de variables booléennes et de conditions élémentaires, à l'aide des opérateurs logiques *non*, *et*, *ou*

Assertions

Exemple

```
Programme Température
Variables
   temp: Entier
Début
   Ecrire("Entrez la température de l'eau")
   Lire(temp)
   Si temp <= 0 Alors
      Ecrire("C'est de la glace")
   Sinon {temp>0}
      Ecrire("C'est du liquide")
   FinSi
Fin
```

Instructions conditionnelles imbriquées

Pourquoi

Il est possible d'imbriquer des instruction conditionnelles lorsque le choix dépend de plusieurs critères dépendants les un des autres.

Comment : syntaxe

```
Si Condition 1 Alors
  Instructions 1
SinonSi Condition 2
  Instructions2
....
SinonSi Condition n
  Instructions n
Sinon
  Instructions n+1
FinSi
```

Instructions conditionnelles imbriquées

Exemple

Programme Température
Variables
  temp : Entier
Début
  Écrire("Entrez la température de l'eau")
  Lire(temp)
  Si temp ≤ 0 **Alors**
    Ecrire("C'est de la glace")
  SinonSi temp < 100 { $0 <$ temp < 100 }
    Ecrire("C'est du liquide")
  Sinon { temp ≥ 100 }
    Ecrire("C'est de la vapeur")
  FinSi
Fin