Algorithmique

Variables en algorithmique

Anicet E. T. Ebou, ediman.ebou@inphb.ci

Ce travail est soumis à une licence internationale Creative Commons Attribution 4.0.

01

Introduction

Introduction

Un ordinateur ne se souvient pas naturellement d'un travail qu'il a effectué. Il faut explicitement lui donner l'ordre de le faire. Suivant la quantité de données, la structure ou la durée du « souvenir » que l'on veut créer, on utilise différents mécanismes :

- Écrire dans un fichier;
- Écrire dans une base de donnée;
- Écrire dans la mémoire vive (RAM).

Introduction

C’est cette dernière méthode que nous allons étudier ici. Les variables mutables (qui peuvent être modifiées) sont stockées le plus souvent dans la mémoire vive.

02

Déclaration

Rôle de la déclaration

Tous les langages ne nécessitent pas de déclaration (Python n'en a pas besoin par exemple), mais on le fait quand même en algorithmique. Cela rend les algorithmes plus lisibles et prépare le terrain pour les langages qui en ont besoin.

Les déclarations réservent un espace dans la mémoire vive pour que le programmeur puisse y stocker une valeur. Il faut bien sûr que la taille de cet espace soit adaptée à la taille de l'information à stocker.

Notation de la déclaration

En algorithmique :

- `nom_variable` : `son_type` pour une variable;
- `nom1`, `nom2` : `son_type` pour plusieurs variables d'un même type.

En Python, il n'y a pas de déclaration, mais on peut mettre un commentaire pour expliquer ce qu'est censée contenir certaines variables (c'est inutile pour la machine, mais lisible par les humains).

02

Affectation

Rôle de l’affectation

L’affectation consiste à mettre la valeur d’une expression dans une case mémoire repérée par le nom de la variable (penser à une boîte).

Notation de l’affectation

En algorithmique : `nom_variable <- expression;`

`<-` se lit « prend la valeur » ou « prend pour valeur ».

Remarques: Erreurs fréquentes

- Il faut que la partie gauche de l'affectation soit bien un nom de variable : attention en à ne pas mettre de tiret '-' dans les noms de variable. Le signe (moins) - est compris comme une soustraction.
- Seul le contenu de la variable dont le nom est à gauche est changé, et ceci seulement au moment de l'affectation. L'affectation n'attache pas à la variable le comportement à venir de l'expression (même si c'est parfois le cas dans certains langages quand la valeur est « passée par référence »).

Remarques: Incohérences avec les mathématiques

- L’expression est souvent calculée entièrement avant que le résultat ne prenne place dans la case correspondant au nom de la variable, ce qui amène à des incohérences avec le monde des mathématiques.
- Le signe ‘=’ de certains langages a un sens totalement différent de celui qu’il a en mathématiques.

Remarques: Incohérences avec les mathématiques

Texte à comparer	Maths	Info
x = 1	égalité, proposition comparant x et 1	en général, c'est une affectation
x == 1	ne veut rien dire	expression qui compare x et 1, retourne souvent un booléen
x = x + 1	proposition toujours fausse	la valeur de la variable x est augmentée de 1 (incrémentation)

Remarques: Inconvénients des variables mutables

Il n'y a pas « de transparence référentielle » : un x à un endroit d'un programme n'a peut-être pas la même valeur deux lignes plus loin, alors qu'en maths, x représente toujours le même nombre. Cela devient un vrai jeu de piste pour savoir quelle est la valeur d'une variable à un moment donné.

L'ordre des affectations est important. Exécuter x <- 0 puis x <- 1 n'aura pas le même effet que x <- 1 puis x <- 0.

04

Types

Introduction

Les opérateurs travaillent souvent sur des valeurs du même type. Par exemple, `4 / 2` signifie quelque chose, alors que `Faux / Vrai` ne signifie rien. Nous aborderons en détails une liste non exhaustive des types que l'on peut rencontrer en informatique.

Booléens

On déclare les booléens en utilisant le nom ‘Booléen’ dans la partie déclaration des variables.

Exemple:

```
Var
    my_bool: Booléen
```

Les booléens peuvent prendre deux valeurs: Faux et Vrai.

Booléens: opérations

Il existe trois grands types d’opérations qu’on peut effectuer sur les booléens.

Opérations	**Algo**
négation	non
disjonction	et
conjonction	ou

Entiers

On déclare les entiers en utilisant le nom ‘Entier’ dans la partie déclaration des variables.

Exemple:

```
Var
    my_integer: Entier
```

Entiers: opérations sur les entiers

Opérations	**Algo**
Addition	+
Soustraction	-
Multiplication	*
Division (euclidienne ou entière)	div
Reste	mod
Puissance	**

Entiers: opérations sur les entiers

Incrémenter signifie « augmenter d'une valeur entière fixée », en général de 1. On peut par exemple incrémenter `n` grâce à l'instruction `n <- n + 1` en algorithmique.

Flottants

Dans beaucoup de langages, on se sert d'un point pour la virgule (c'est ce que font les anglophones). Pour qu'un nombre entier soit vu comme flottant par la machine, on se sert aussi du point, éventuellement suivi d'un zéro. On déclare les flottants en utilisant le mot-clé 'Réel'.

Exemple:

```
Var
    my_float: Réel
```

Flottants: les opérations

Nous retrouvons l’addition, la soustraction, la multiplication et la division.

Caractères

On déclare les caractères en utilisant le nom ‘Caractère’ dans la partie déclaration des variables. Un caractère représente une lettre.

Exemple:

```
Var
    my_char: Caractère
```

Chaînes de caractères

C’est le mot technique pour le type qui contient du texte. Les chaînes de caractères sont souvent délimitées par des guillemets simples ' ou doubles ". On les déclarent en utilisant le nom ‘Chaîne de caractères’ dans la partie déclaration des variables. Un caractère représente une lettre.

Exemple:

```
Var
    my_char: Chaîne de caractères
```