

Algorithmique et UML

CM1-2 : Algorithmique

Mickaël Martin Nevot

V1.8.1



Cette œuvre de [Mickaël Martin Nevot](#) est mise à disposition selon les termes de la [licence Creative Commons Attribution – Pas d'Utilisation Commerciale – Partage à l'Identique 3.0 non transposé](#).

Algorithmique et UML

- I. Prés. du cours
- II. Init. à la prog.
- III. Algo.
- IV. APP
- V. Java
- VI. Java avancé
- VII. Algo. avancée
- VIII. UML
- IX. Génie log.

Algorithmique

- Suite finie et non-ambiguë d'opérations (instructions) permettant de donner la réponse à un problème
- IXe siècle : Al Khuwarizmi (*Algoritmi* en latin)
- Pseudo-code



Pseudo-code

- Règle d'or : **le pseudo-code doit être suffisamment précis pour que quelqu'un d'autre l'ayant lu sans connaître l'algorithme soit capable de le coder (raffinement)**
- Pas de norme officielle
- Proposition de langage de description algorithmique
- **Un algorithme juste est un algorithme faux qui s'ignore !**



Pseudo-code

- Une seule page : généralement 20 – 25 lignes
- Laisser des lignes blanches régulièrement
- Avoir une **bonne indentation**
- Choisir une langue : l'**anglais** ?
- Choisir une **convention** d'écriture
- Factoriser le code (modularité et règle **DRY**)
- Éviter les zones troubles
- Éviter les cas particuliers
- Éviter les commentaires

Pseudo-code

- **Pas de trivialité :**

- Éviter les déclarations des variables locales (mais préserver les initialisations)
- Éviter les indices de début et de fin des boucles triviales
- Éviter de préciser les types de certaines variables ou paramètres

- **Pas de détail** sur les morceaux de code usuels :

- Entrées et sorties simples
- Structures ou méthodes courantes
- Fonctions de comparaison standards

Mise en forme / commentaires

- Mise en forme :
 - Indentation à chaque niveau de bloc
 - Convention de nommage :
 - myProg
 - myVar
 - MY_CONST

- Commentaires (de type C/C++) ← **À éviter**

```
// Commentaire (une seule ligne).
```

```
/* Autre commentaire (une ou plusieurs lignes). */
```

```
/*  
    Commentaire  
    (sur plusieurs lignes).  
*/
```


Structure générale



Types/opérateurs

- Types :
 - Entier (`int`) : 3
 - Réel (`float`) : 2.75
 - Chaîne de caractère (`string`) : "hello!"
 - Booléen (`boolean`) : true
 - Tableau (`array`) : {500, 200, 100, 50, 20, 10, 5, 2, 1}
- Opérateurs :
 - Numériques : +, /, *, -, div (division entière), % (modulo)
 - Affectation : ←
 - Comparaison : =, ≠, >, <, >, ≤, ≥
 - Logiques: or (ou inclusif), and, not, xor (ou exclusif)
 - De chaînes : + (concaténation), size (taille)
 - De tableau : size (nombre d'éléments)

Déclaration

```
const myVar:string
```

Mot clé const (optionnel)

Identificateur

Type (optionnel)

Absence de ;

```
var1, var2:float  
var3
```

Déclaration (tableaux)

```
const myVar:array [0..5] of int
```

Un seul type
par tableau
(optionnel avec
mot de clé of)

Absence de ;

Mot clé const (optionnel)

Identificateur

Type (optionnel)

Indices minimum et
maximum (optionnel)

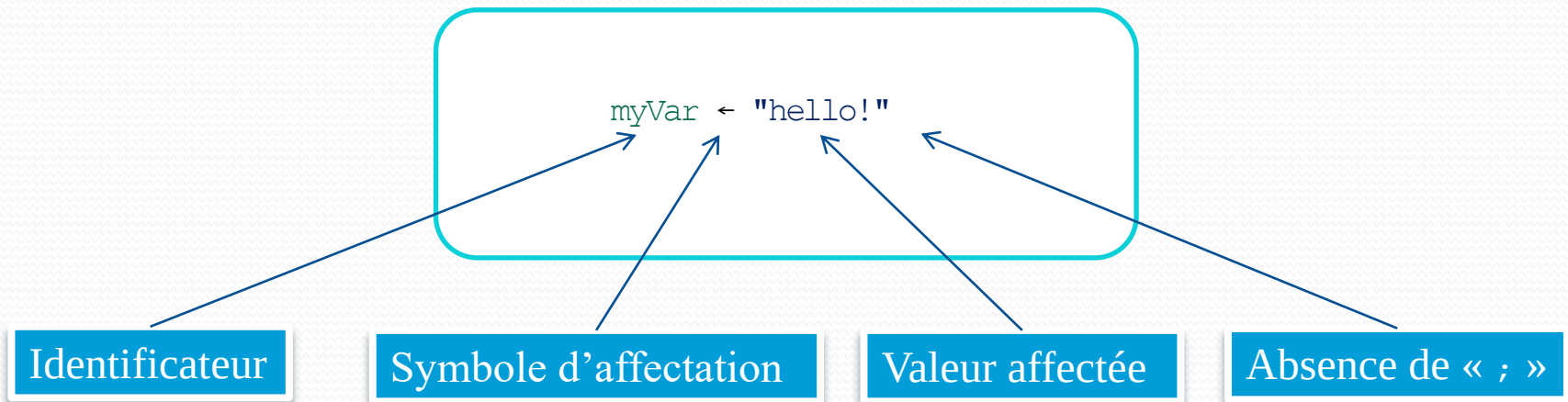
```
const myVar:array [0..5] [0..8] of int
```

Tableau multidimensionnel

```
var1:array [0..8]
```

```
var3:array [1..5][1..10]
```

Affectation

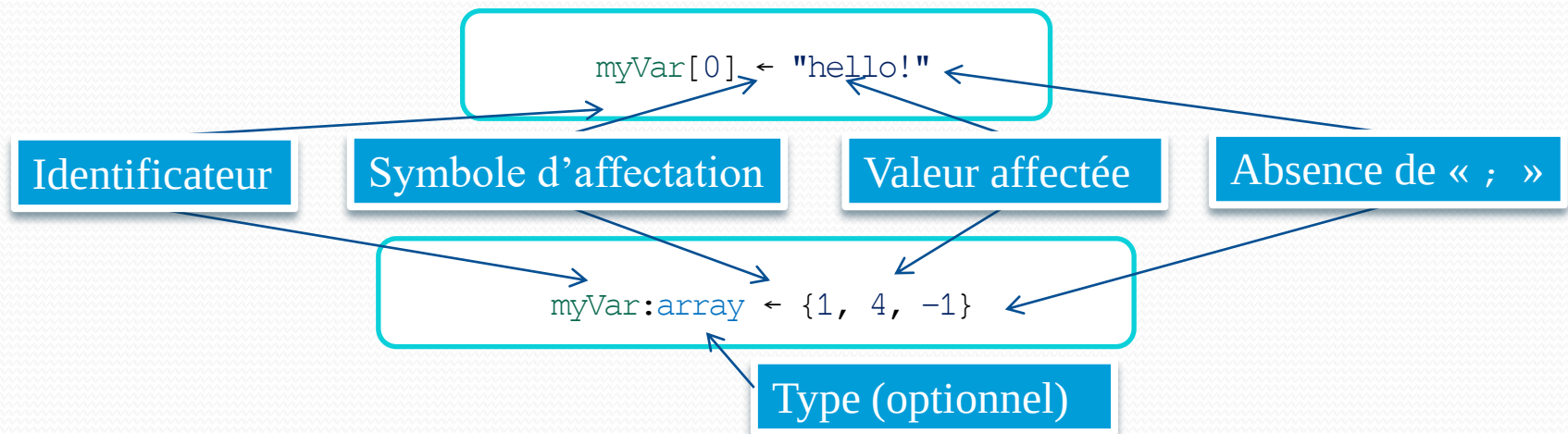


```
const CONST_1 <- 10
```

```
var1:float <- 3.14
```

```
var1 <- 10.5
```

Affectation (tableaux)



```
const CONST_1:array [0..10] of int ← {1, 4, -1}
var1:array ← {500, 200, 100, 50, 20, 10, 5, 2, 1}
var2:array [0..5][0..18]
```

```
var1[4] ← 15
var2[0][1] ← "hello!"
```

Structure conditionnelle

```
if condition
    instruction1
    instruction2
    ...
else
    instruction3
    instruction4
    ...
```

Structure conditionnelle multiple

```
switch expression
  valeur1 : instruction1
  valeur2 : instruction2
           instruction3
  valeur3,
  valeur4 : instruction4
           ...
else
  instruction5
  ...
```


Répétition (boucles)

De 0 à n fois

```
while condition do  
  instruction1  
  instruction2  
  ...
```

De 1 à n fois

```
do  
  instruction1  
  instruction2  
  ...  
while condition
```

Répétition (boucles)

```
for  $x \leftarrow i$  to  $j$  do  
  instruction1  
  instruction2  
  ...
```

Avec un pas de 1

```
for  $x \leftarrow i$  to  $j$  with step  $k$  do  
  instruction1  
  instruction2  
  ...
```

Avec un pas de k
 k entier

Fonction

```
function myFunction(paramètres):type  
...  
instruction1  
instruction2  
...  
return expression
```

Signature

Type optionnel

```
myVar1:string, myVar2 ...
```

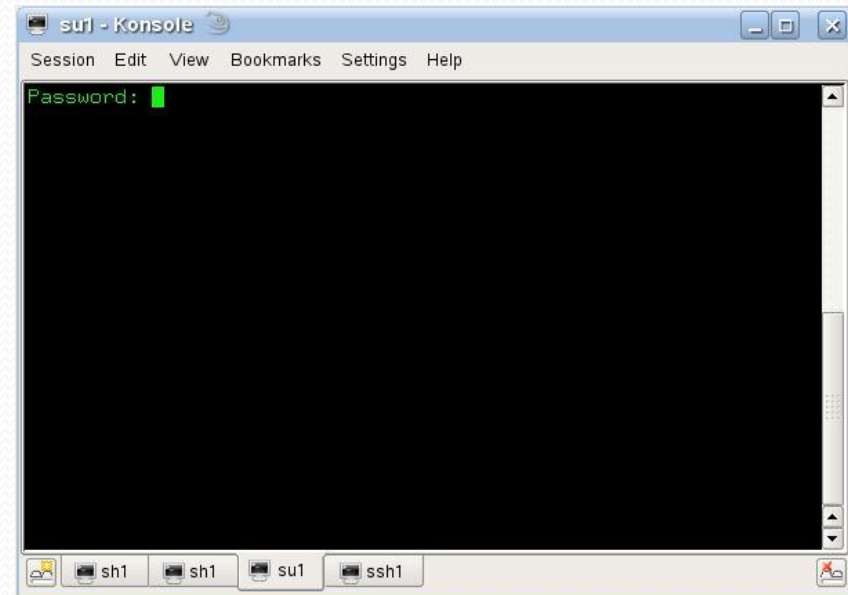
Paramètres

Identificateur

Type (optionnel)

Entrées/sorties

- Lire :
 - read `var1`
 - read `var1`, `var2` ...
- Écrire :
 - write `var1`
- Afficher :
 - print `var1`
 - print `"result : ", var1`
 - print `"3 x 2 = ", (3 * 2)`
 - print `"tab : ", {1, 4, -1}`



Quelques algorithmes

- Diviser pour régner :

- Décomposition
- Appels récurifs
- Recomposition

Recherche dichotomique

Tri fusion

Tri rapide

- Glouton :

- Étape par étape
- Optimum local = optimum global ?
- Heuristique gloutonne

Rendu de monnaie

Problème du sac à dos

- Recherche exhaustive (combinatoire) :

- Recherche de tous les cas

Attaque par force brute

- Probabiliste :

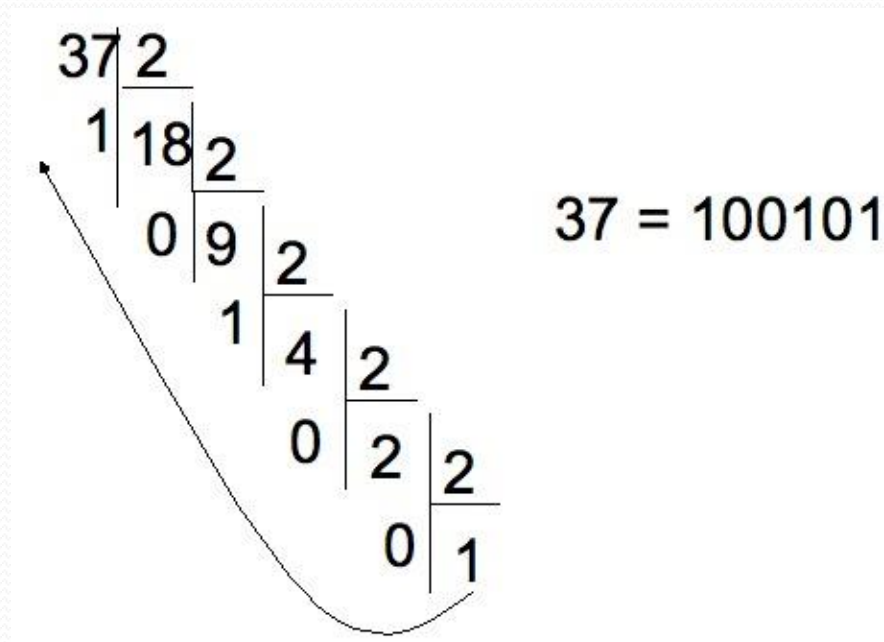
- Données tirées aléatoirement

Las Vegas

Monte-Carlo

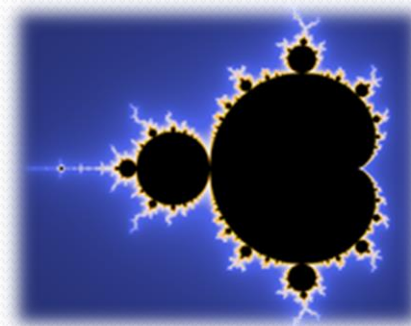
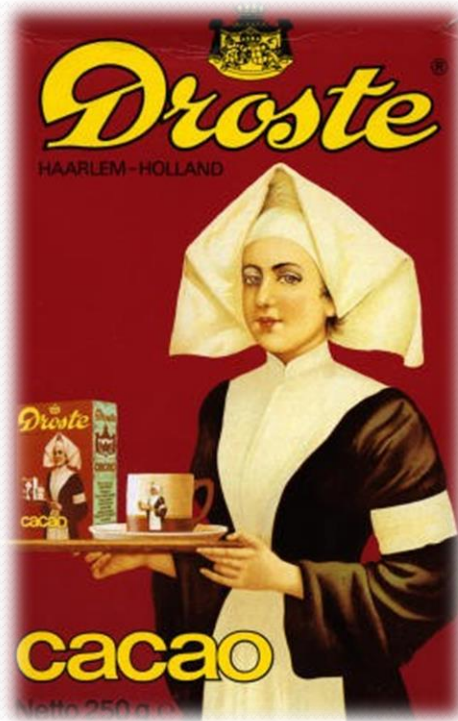
Trace d'un algorithme

- Exécute les instructions comme un ordinateur
- Permet de contrôler que l'algorithme correspond a priori aux attentes
- Ne constitue pas une preuve



Récurtivité

- Fonction qui s'appelle elle-même
- Condition d'arrêt



Récurtivité : un exemple

- Factorielle

$$n! = \prod_{i=1}^n i = 1 \times 2 \times 3 \times \cdots \times (n-1) \times n$$

$$n! = \begin{cases} 1 & \text{si } n = 0 \\ n \times (n-1)! & \text{sinon} \end{cases}$$

```
function factorial(n):int
  if n = 0
    return 1
  else
    n * factorial(n - 1)
```


Liens

- Documents électroniques :

- <http://www.france-ioi.org/train/algo/cours/>
- <ftp://ftp-developpez.com/lapoire/algorithmique/initiation-algorithmique.pdf>
- <http://www.apmep.asso.fr/spip.php?article3405>

Crédits

Auteur

Mickaël Martin Nevot

mmartin.nevot@gmail.com



Carte de visite électronique

Relecteurs

Cours en ligne sur : www.mickael-martin-nevot.com

