NFA031 Algorithmique, Pseudo-code

Algorithmique

Caractéristiques

- **Analyser** la nature du **problème**
- **Préciser** les **données** fournies
- **Définir** les **résultats** que l'on souhaite obtenir
- **Déterminer** le **processus de transformation** des données pour aboutir au **résultat**
- **lisible** : compréhensible pour un non informaticien
- **haut niveau** : traduisible dans n'importe quel langage, pas de notions techniques relatives à un programme ou un OS
- **précis** : pas de confusion, concis, sans ambiguïté

Pseudo-code

Type de l'algorithme : **procédure** ou **fonction**

- **Données** (en **entrée**)
- **Résultats** (en **sortie**)
- **Initialisation** des **variables locales**
- commence par **Début** et se termine par **Fin** Entre ces deux **marqueurs** se situent les **instructions**
- Commentaires sous la forme : **//texte de commentaire**

Exemple

```
Procédure MonAlgorithmeDeTest
Début
... //instruction 1
... //instruction 2
... //instruction 3 etc.
Fin
```

Etapes

1. **1 définition du problème** à résoudre (par un non informaticien)
2. **2 Ecriture** de la solution **en pseudo code** (par un non informaticien et un informaticien)
3. **3 transcription** en un **langage de programmation** par le développeur (par un informaticien)
4. **4 transcription** en « **langage Assembleur** » (par la machine)
5. **5 transcription** en **code machine** compréhensible par le **processeur** (par la machine)

Variables

Simples

```
a      //La premier cas n'est pas représentatif car non explicite
Age    //La second est lisable de fait de l'accent circonflexe
Age    //Parfait car représentatif et court
AgeDeLaPrisonnierSOuLaDetraqueur  //La dernier est bien trop long
```

Exemple :
Je veux mémoriser l'âge d'une personne dans une variable :

```
NbMn
```

Je veux mémoriser dans une variable un nombre de minutes ? Cm a ici utilisé les abrégés de nombre et Minute, l'utilisation des Majuscule permet de dissocier à la lecture les deux termes sans avoir à utiliser d'espace

Contenu / valeur

```
oui/non ; ouvert/fermé ; vrai/faux
```

Booléen > Faux ou Vrai

```
0 à 255
```

Byte (octet)

```
-32 768 à 32 767
```

Entier Simple

```
-2 147 483 648 à 2 147 483 647
```

Entier Long

```
nombre quelconque
```

Numérique > 45.890

```
-3,40282E38 à -1,40129E46 pour les valeurs négatives
3,40282E-45 à 3,40282E38 pour les valeurs positives
```

Réel Simple

```
1,79769E308 à -4,94065E-324 pour les valeurs négatives
4,94065E-324 à 1,79769E308 pour les valeurs positives
```

Réel Double

```
caractère seul
```

Caractère > "c"

Attention à ne pas confondre numériques et chaîne de caractères !! **123** est un **numérique** (cent vingt trois)
" 123 " est une chaîne (suite de chiffre 1, 2, 3)

```
un texte de zéro, un, ou plusieurs caractères
```

Chaîne de caractères > "hello #world"

```
67.90
```

Monetaire

```
13/03/2002
```

Date

Déclaration de variable

```
Var Prix en Numérique
```

Règle : toujours effectuer la déclaration en début d'algorithme

- « **Var** » : indique que l'on **déclare une variable**, quelle qu'elle soit
- « **Prix** » : **Nom que l'on donne** à la variable crée
- « **en Numérique** » : **type de la valeur** de la variable nommée juste avant

Exemple :
Imaginons que nous ayons à déclarer **une série de variables** utilisées dans le calcul du prix d'un objet. Nous avons comme variables à déclarer : le **prix ht**, le **prix ttc**, le **taux de tva**, le **nom de l'objet**.

```
Vars PrixHT, PrixTTC, TauxTVA en Numérique
Var NomObjet en Chaîne de caractères
```

ou

```
Var PrixHT en Numérique
Var PrixTTC en Numérique
Var TauxTVA en Numérique
Var NomObjet en Chaîne de caractères
```

Affectation de variable

Affecter une variable signifie lui **attribuer une valeur**. Cette valeur peut être **connue à l'avance, ou non**. Si cette valeur est **inconnue** nous lui attribuons une **valeur par défaut**.

```
PrixHT <- 10
```

Attribue la valeur numérique 10 à la variable PrixHT

```
NomObjet <- "Objet numéro 1"
```

Attribue la valeur en chaîne de caractères "Objet numéro 1" à la variable NomObjet

```
NomVariable1 <- 2
```

Attribue la valeur numérique 2 à la variable NomVariable1

```
NomVariable1 <- 2 NomVariable2 <- NomVariable1
```

Attribue la valeur de NomVariable1 à NomVariable2

Une affectation ne modifie la valeur que de ce qui est à gauche de la flèche, nomVariable1 n'a ici pas changé de valeur

```
NomVariable1 <- 2
NomVariable2 <- NomVariable1 + 10
```

Avec NomVariable1 qui vaut 2, NomVariable2 vaut maintenant 12 (2 + 10)

Faire attention aux guillemets

```
Debut
    Toto <- "titi"
    Tutu <- "Toto"
Fin
```

On affecte à la variable Toto la valeur «titi» sous la forme d'une chaîne de caractères, et à la variable Tutu la valeur « Toto » sous la forme d'une chaîne de caractère.
Les valeurs des deux variables sont ainsi différentes.

```
Debut
    Toto <- "titi"
    Tutu <- Toto
Fin
```

On affecte à la variable tutu **la valeur de la variable Toto**, au final les variables Toto et Tutu ont donc la même valeur, à savoir la chaîne de caractère « titi » (titi)

Ordre des instructions

```
Debut
    PrixHT <- 15
    PrixHT <- 20
Fin
```

la valeur finale de la variable PrixHT est 20

Par exemple il peut exister un PrixHT avec une valeur affectée par défaut, puis la variable PrixHT peut changer en cours d'algorithme (application d'une réduction sur le prix de départ par exemple).

```
Debut
    PrixHT <- 20
    PrixHT <- 15
Fin
```

la valeur finale de la variable PrixHT est 15

Exercices

```
Vars A, B en Entier
Debut
    A <- 1
    B <- A + 3 A <- 3
Fin
```

A vaut 3, B vaut 4

```
Vars A, B, C en Entier
Debut
    A <- 2
    B <- 10
    C <- A + B
    B <- A + B
    A <- C
Fin
```

A vaut 12, B vaut 12, C vaut 12

```
Vars A, B en Entier
Debut
    A <- 5
    B <- 2
    A <- B
    B <- A
Fin
```

A vaut 2, B vaut 2

```
Vars A, B, Temp en Entier
Debut
        Temp <- A
        A <- B
        B <- Temp
Fin
```

Nous utilisons ici une variable temporaire « Temp » qui va nous permettre de stocker temporairement la valeur de A afin de l'affecter eu final à la variable B.

```
Vars A, B, en Entier
Debut
        A <- B
        B <- A
Fin
```

Si nous avions écrit ceci. A aurait pris la valeur de B mais ensuite B aurait pris la valeur de A **modifiée** juste avant, au final les valeurs n'aurait pas été échangées, Les deux variables auraient pris la valeur de B.

Expressions et opérateurs

```
NomVariable2 <- NomVariable1 + 10
```

Définition : **Une expression est un ensemble de valeurs, reliées par des opérateurs, et équivalent à une seule valeur.**

Types

- "+" : addition
- "-" : soustraction
- "*" : multiplication
- "/" : division
- "^" : puissance

Priorités opératoires

```
NombreA <- 10+20
```

La variable Nombre1 aura pour valeur 20

```
NombreA <- (10+20)*2
```

La variable Nombre1 aura pour valeur 60

Concaténation

```
Debut
    Nom <- "Martin"
    Prenom <- " pierre"
    NomPrenom <- Nom & Prenom
Fin
```

concaténer deux chaînes de caractère, « Concaténer » signifier rassembler, joindre. **NomPrenom = Martin pierre**

Opérateurs logiques (booléens)

A	OUI a
0	0
1	1

A est VRAI seulement si a est OUI

A	NON a
0	1
1	0

A est VRAI si et seulement si A est NON

a	b	a ET b
0	0	0
0	1	0
1	0	0
1	1	1

a ET b est VRAI si et seulement si a est OUI et b est OUI

a	b	a OU b
0	0	0
0	1	1
1	0	1
1	1	1

a OU b est VRAI si et seulement si a est OUI **et/ou b** est OUI

a	b	a XOR b
0	0	0
0	1	1
1	0	1
1	1	0

a OU b est VRAI si et seulement si a est OUI ou b est OUI