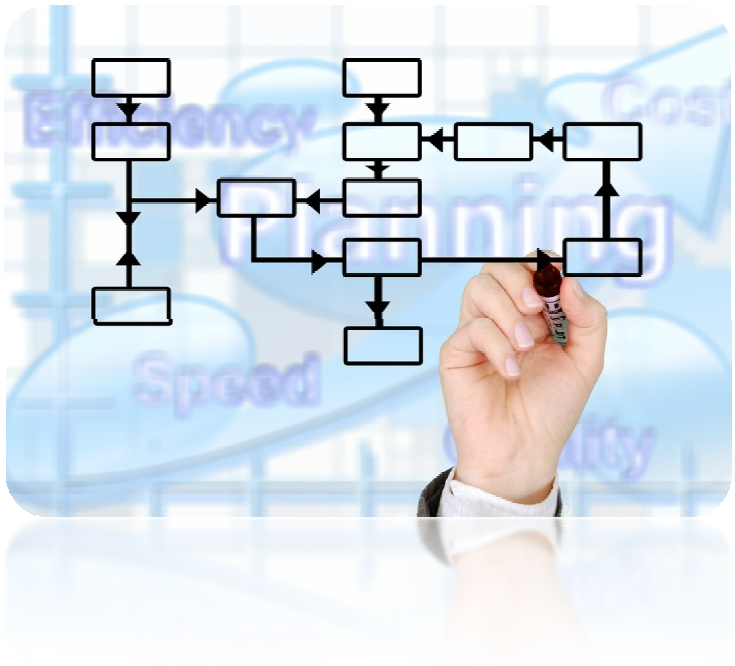
Instructions de base et types

Objectifs : comprendre les types et la structure d'un algorithme.



1- Lecture de données, affectation

2- L'écriture des résultats

3- Types, algèbre de Boole, opérateurs

4- Structure d'un algorithme

Lire les données depuis le clavier

1 Lecture des données : de quoi s'agit-il?

La lecture de données consiste à donner une valeur à une donnée tout en la nommant, c'est-à-dire en lui effectant un identificateur.

Exemples

LIRE N ← Ranger dans la case **N** la donnée lue

LIRE X, Y, Z ← Ranger dans les cases **X, Y, Z** les 3 données lues.

Lorsque l'exécutant (l'ordinateur ou le concepteur de l'algorithme) rencontre cette instruction, il s'arrête et attend que l'utilisateur donne une valeur à **N** dans le premier exemple, à **X, Y, Z** dans le deuxième.

Affectation = mettre dans la case

2 L'affectation

L'affectation consiste à affecter à une case mémoire (donc à un identifiant donné) une certaine valeur. Elle peut être représentée par le symbole ← ou par **:=** suivi de la valeur.

Exemples

TOTAL ← 350

PRIX ← TOTAL

SOMME ← TOTAL + 9

Après exécution, **PRIX** va contenir **350** et **SOMME 359**.

Nota

- L'affectation écrase la valeur précédente de l'identificateur par la nouvelle.
- La valeur de l'objet à droite de ← n'est pas changée par l'affectation.

Ecrire = à l'écran ou sur un disque

1 Opération d'écriture

Cette instruction est le symétrique de **LIRE**. Elle visualise les résultats. En l'absence d'ordre d'écriture dans un algorithme, les résultats ne sont pas fournis à l'utilisateur.

Exemples

ECRIRE TOTAL

ECRIRE 'Ceci est une chaine de caractères'

Nota

Dans le 2ème exemple, la chaîne de caractères est entourée de ' '. Ce sont des délimiteurs de chaîne qui indiquent que la suite de caractères situés entre eux est une constante et ne doit pas être traitée en tant qu'identificateur.

Toute donnée doit être typée.

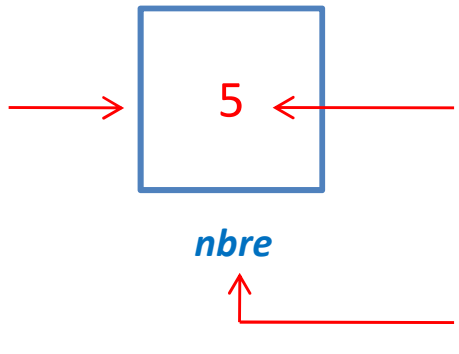
1 La notion de type

Chaque donnée devra être représentée par un identificateur et doit être d'un certain type. Le type d'une donnée désigne à la fois l'ensemble des valeurs qu'elle peut prendre et les opérations qu'il est possible de faire.

type :

1- ensemble de valeurs possibles.
Exemple : **entier**, **chaine**, **date**, etc.
5 est de type entier.

2- opérations possibles :
Exemple : +, - , & , / , etc.



5 est la ***valeur*** stockée dans la case.

Cette case s'appelle ***nbre***. C'est son ***identificateur***.

Donnée : *identificateur + valeur + type*

Toute donnée doit être typée.

2 Les types de base

Il existe plusieurs types de base ou types de données prédéfinis, mais l'utilisateur peut créer ses propres types. On peut citer les entiers, les réels, les chaînes et les booléens par exemple.

- **Type NUMERIQUE**

  Une donnée de ce type est destinée à des ***opérations mathématiques***. Elle peut contenir une valeur ***entière*** ou ***décimale***, signée ou non. Sa longueur indique le nombre maximal de chiffres (« digits ») pouvant figurer avant et après la virgule.

  Exemples

  'Entier contiendra 99999 ou -568 mais pas 458125
  Compteur NUMERIQUE 5

  'Avec 7 digits dont 2 décimales
  Montant NUMERIQUE 7 , 2

Toute donnée doit être typée.

2 Les types de base (suite)

- **Type ENTIER**

Une donnée de type **ENTIER** peut prendre une valeur entière comprise entre **–32768** et **+32767**. Pour un langage de programmation donné, les bornes de ce sous-ensemble de nombres entiers dépendent de l'implémentation du compilateur.

-32768	.....	+32768

Exemple

'Ici on utilise ENTIER au lieu de NUMERIQUE, parce qu'on sait qu'on a un entier.
Compteur ENTIER 5

Toute donnée doit être typée.

2 Les types de base (suite)

- **Type REEL**

Une donnée de type **REEL** peut prendre une valeur décimale comprise dans l'intervalle (l'intervalle dépend en fait du langage et de l'ordinateur cible) :

[5.3976E-79, 7.2370E75] U [0]

Exemple

'Ici on utilise REEL au lieu de NUMERIQUE, parce qu'on sait qu'on a un réél.
Montant REEL 7 , 2

Toute donnée doit être typée.

2 Les types de base (suite)

- **Type CHAINE**

Une donnée de type chaîne (encore appelé type « alphanumérique ») peut contenir des lettres, des chiffres et tout autre symbole que l'on peut obtenir au clavier.

Ce type est ordonné par une table où sont recensées toutes les valeurs :

• Table ASCII sur la plupart des systèmes micro.

• Table EBCDIC dans le monde IBM.

On a ainsi :

'a' < 'b'	'a' < > 'A'	'9' > '8'

Nota :
7 est différent de **'7'**

Toute donnée doit être typée.

2 Les types de base (suite)

- **Type CHAINE**

Les différents langages de programmation définissent plusieurs opérations (fonctions) sur les sur les chaînes de caractères.

Exemples :

- Longueur d'une chaîne
  length(chaine) : en Java
- Concaténer 2 chaînes
  String test = "Jean" + " " + "Dupont"; : en Java
- Extraire une sous chaîne d'une chaîne
  String substring(int d) : en Java

Toute donnée doit être typée.

2 Les types de base (suite)

- **Type BOOLEEN**

Une donnée de type BOOLEEN prend sa valeur dans l'ensemble { VRAI, FAUX }.

VRAI	FAUX

Nota

Les données de type **TABLEAU** seront vues plus tard dans cette formation.

Vrai ou Faux ?

3 Algèbre de BOOLE

L'algèbre de Boole, ou calcul booléen, s'intéresse à une approche algébrique de la logique vue en termes de variables, d'opérateurs et de fonctions sur les variables logiques, ce qui permet d'utiliser des techniques algébriques pour traiter les expressions à deux valeurs du calcul des propositions.

Elle fut lancée en 1854 par le mathématicien britannique George Boole. Elle est est appliquée en informatique et dans la conception des circuits électroniques (***Wikipedia***) .

L'algèbre de Boole permet :

- de représenter par une notation simple et précise toutes les propositions impliquées dans un raisonnement,
- de remplacer un raisonnement logique par un calcul et de bénéficier des automatismes que celui-ci permet.

Vrai ou Faux ?

3 Algèbre de BOOLE (suite)

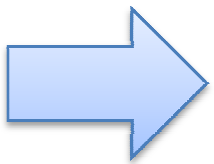
Soit **E** un ensemble d'éléments e_i et une propriété **p**.
*Chaque élément e_i a ou n'a pas la propriété **p***. A chaque e_i, on associe une variable, dite variable de Boole, qui vaut :

- **1** si **p** est vraie,
- **0** si **p** est fausse.

Exemple

E = ensemble des développeurs Java
ei appartiennent à **E**
p : « connaît Spring».

a : variable booléenne.



Valeurs de **a** :

- **1** connaît Spring
- **0** ne connait pas Spring

Vrai ou Faux ?

4 Opérations booléennes

- **Conjonction booléenne : ET**

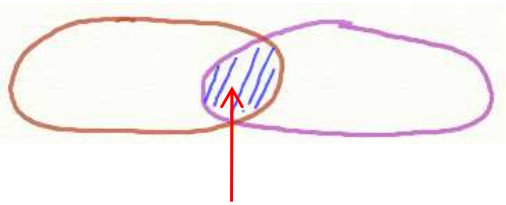
E = ensemble des développeurs Java
e_i appartiennent à **E**
p : « connait Spring».
q : « ayant travaillé sous Linux».

Les e_i « connaissant Spring **ET** ayant travaillé sous Linux » forment la conjonction **ET**.

Elle est notée « **x** » ou « **.** ».

a	b	a ET b
True	True	True
True	False	False
False	True	False
False	False	False

ET : table de vérité



p vraie, **q** vraie dans la zone hachurée.

Vrai ou Faux ?

4 Opérations booléennes (suite)

- **Conjonction booléenne : OU**

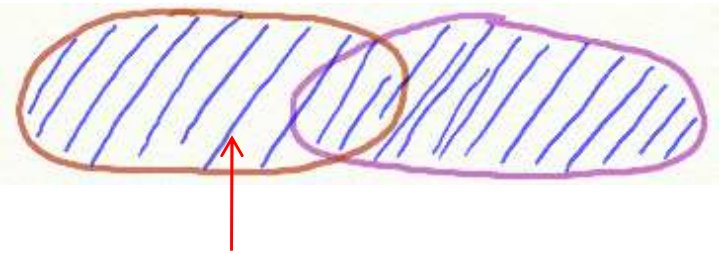
E = ensemble des développeurs Java
e_i appartiennent à **E**
p : « connait Spring».
q : « ayant travaillé sous Linux».

Les e_i « connaissant Spring **OU** ayant travaillé sous Linux » forment la conjonction **OU**.

Elle est notée « **+** ».

a	b	a OU b
True	True	True
True	False	True
False	True	True
False	False	False

OU : table de vérité



p vraie partout, **q** vraie partout dans la zone hachurée.

Vrai ou Faux ?

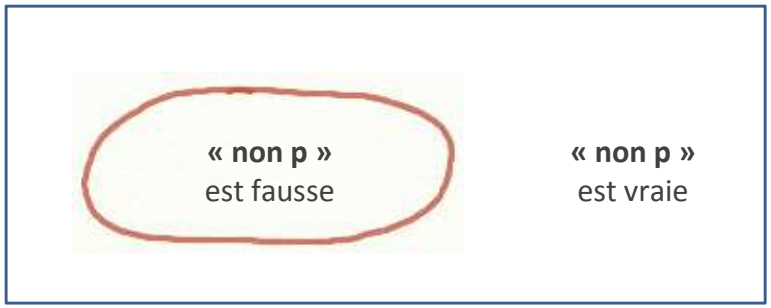
4 Opérations booléennes (suite)

- **Conjonction booléenne : NON**

E = ensemble des développeurs Java
e_i appartiennent à **E**
p : « connait Spring».
Non p : vraie si **p** fausse et inversement.

Non p contient les e_i « ne connaissant pas Spring».

Elle est notée **NON**(p) ou $\overline{p}$



Vrai ou Faux ?

5 Relations algébriques booléennes

$a \cdot a = a$ $\quad a + a = a$

$1 \cdot a = a$ $\quad 1 + a = 1$

$0 \cdot a = 0$ $\quad 0 + a = a$

$\overline{a} \cdot a = 0$ $\quad \overline{a} + a = 1$

$\overline{\overline{a}} = a$

$(a + b) + c = a + (b + c)$

$(a \cdot b) \cdot c = a \cdot (b \cdot c)$

associativité

$a + b = b + a$

$a \cdot b = b \cdot a$

commutativité

$a \cdot (b + c) = (a \cdot b) + (a \cdot c)$

$a + (b \cdot c) = (a + b) \cdot (a + c)$

distributivité

$a + (a \cdot b) = a$

$a \cdot (a + b) = a$

$a + (\overline{a} \cdot b) = a + b$

absorption

$\overline{a \text{ ou } b} = \overline{a} \text{ et } \overline{b}$

$\overline{a \text{ et } b} = \overline{a} \text{ ou } \overline{b}$

Lois de MORGAN

Obtenir des expressions

6 Quelques opérateurs

- **Opérateurs arithmétiques**

  Ils permettent d'obtenir des expressions en réalisant des opérations arithmétiques classiques.

Opérateur	Signification
+	addition
-	soustraction
*	multiplication
/	division
**	Élévation à la puissance

Par ordre de priorité croissante

Obtenir des expressions

6 Quelques opérateurs (suite)

- **Opérateurs relationnels**

Ils permettent de faire des comparaisons.
Le résultat est est un booléen.

Opérateur	Signification
=	égal
<>	différent de
<	inférieur
<=	Inférieur ou égal
>	supérieur
>=	Supérieur ou égal

Par ordre de priorité croissante

Obtenir des expressions

6 Quelques opérateurs (suite)

- **Opérateurs logiques**

  Ces opérateurs relient des expressions logiques ou des identificateurs de type booléen. On en utilise essentiellement 3 :

Opérateur	Signification
OU	Ou logique
ET	Et logique
NON	Non logique

Par ordre de priorité croissante

Exemple

1 Structure typique

On admet que tout algorithme sera composé de 3 parties distinctes.

- ***Entête***

  Permet de donner un nom à l'algorithme et de donner son utilité.

- ***Partie déclarative (= dictionnaire de données)***

  Recense toutes les données utilisées.
  Donnée : identificateur + type + [longueur] + [valeur]

- ***Le corps (= partie exécutable)***

  Indique les traitements à exécuter.

```
ALGO MONNOM

VAR Nom : Chaine

DEBUT
        ECRIRE  'Veuillez saisir votre nom'
        LIRE  Nom
        ECRIRE  'Vous êtes', Nom
FIN
```

Un petit exemple

Exemple

2 Structure typique : exemple

Voici le même algorthme avec des commentaires.

Pour les commentaires :

- Utilisez '
- Utilisez (* commentaire ici *)

```
ALGORITHME MONNOM
(* l'utilisateur doit saisir son nom *)

(* le dictionnaire de données*)
VAR Nom : Chaine

(* le corps commence ici*)
DEBUT
        ' afficher à l'écran
        ECRIRE  'Veuillez saisir votre nom'

        (* lire la donnée et la mettre dans Nom *)
        LIRE  Nom

        (* afficher le nom récupéré *)
        ECRIRE  'Vous êtes', Nom
FIN
```

Exemple commenté