

# Chapitre 2

## Les types simples et leurs opérations

### Chapitre 2 Les types simples et leurs opérations

#### I Introduction

Les types simples utilisés sont :

- entier
- réel
- caractère
- chaîne (de caractères)

A chaque type est associé un ensemble *d'opérations spécifiques*.

Un nouveau type simple : le type **booléen**.

## Chapitre 2 Les types simples et leurs opérations

### II le type booléen

#### **II.1) Exemple :** Calcul de la paye

Si l'employé est un cadre, alors son salaire brut est donné sinon il est calculé à partir du nombre d'heures et du salaire horaire.

algo calcul_Paie	Ord	Lexique
<u>si</u> cadre = vrai <u>alors</u>   lire(salbrut) <u>sinon</u>   salbrut ← salhor*nbh fsi		cadre (booléen) : à vrai si l'employé est cadre, faux s'il ne l'est pas

**NB:** la variable *cadre* doit avoir une valeur (*vrai* ou *faux*) pour être comparée

## Chapitre 2 Les types simples et leurs opérations

### **II.2) caractéristiques du type booléen**

#### **a) définition**

Le type booléen est un type simple.

Une variable de type booléen prend ses valeurs dans l'ensemble {**vrai**, **faux**}.

 : **vrai**, **faux** ne sont pas des chaînes de caractères mais sont des symboles à utiliser tels quels.

## Chapitre 2 Les types simples et leurs opérations

### b) utilisation

✓ en affectation simple :

$a \leftarrow \text{vrai}$

$b \leftarrow \text{faux}$

✓ en résultat d'une évaluation de condition :

$\text{salinferieur} \leftarrow \text{salbrut} \leq \text{plafond} \Leftrightarrow$

si salbrut  $\leq$  plafond alors  
| salinferieur  $\leftarrow$  vrai  
sinon  
| salinferieur  $\leftarrow$  faux  
fsi

## Chapitre 2 Les types simples et leurs opérations

### c) Construction d'une condition

Une condition est construite à l'aide :

- ☐ de booléens
- ☐ d'opérations de comparaison entre objets de même type
- ☐ d'opérateurs logiques : et, ou, non

## Chapitre 2 Les types simples et leurs opérations

### Exemple :

Si un employé est cadre et qu'il possède + 3 ans d'expérience, alors on rajoute à son salaire une prime de 200 euros.

```
si cadre et (nbannees > 3) alors      | | cadre (booléen): à vrai si c'est un cadre
| salaire ← salnet + 200                | | nbannees (entier): nbre années ancienneté
|                                     | | salaire (entier) : salaire avec prime ....
fsi
```

### Autre formulation :

```
si cadre et ancien alors              | | cadre(booléen): à vrai si c'est un cadre
| salaire ← salnet + 200                | | ancien(booléen): à vrai si +3 ans ancienneté
fsi
```

## Chapitre 2 Les types simples et leurs opérations

La valeur de la variable **ancien** peut être évaluée de deux manières :

ancien ← nbannees > 3    ⇔

```
si nbannees > 3 alors
| ancien ← vrai
sinon
| ancien ← faux
fsi
```

**Remarque :** le résultat de la comparaison qui est **vrai** ou **faux** est affecté à **ancien**.

## Chapitre 2 Les types simples et leurs opérations

### III Opérations et types simples :

A chacun des types simples est associé un ensemble d'opérations.

Opérations de comparaison (Communes à tous les types)		
type de comparaison	opérateur	type résultat
égalité	=	booléen
supériorité	>	booléen
infériorité	<	booléen
supériorité ou égalité	≥	booléen
infériorité ou égalité	≤	booléen
différence	≠	booléen

## Chapitre 2 Les types simples et leurs opérations

### III.1) Les entiers

- correspondent aux entiers relatifs ( $\cong \mathbb{Z}$ )
- nombre limité : dépend de la taille du mot-mémoire de la machine utilisée

Opérations spécifiques :		
type opération	opérateurs	types résultat
addition	+	entier : a+b
soustraction	-	entier : a-b
multiplication	*	entier : a*b
division réelle	/	réel : a/b
division entière	<u>div</u>	entier : a <u>div</u> b ( <i>quotient</i> )
reste de la division entière	<u>mod</u>	entier : a <u>mod</u> b ( <i>reste</i> )

## Chapitre 2 Les types simples et leurs opérations

### III.2) Les réels

- la précision est limitée (c'est le nombre de chiffres derrière la virgule)
- utilise une codification exponentielle appelée virgule flottante qui correspond à la notation scientifique (convention : virgule placée juste après le 1<sup>er</sup> chiffre). Après normalisation de l'écriture, on obtient une mantisse de type entier :

signe.mantisse.base<sup>exposant</sup>

**Exemple :**  $3,142.10^{20} = 0,3142.10^{21}$  => la valeur de l'exposant indique la position de la virgule

## Chapitre 2 Les types simples et leurs opérations

Opérations spécifiques :

type opération	opérateurs	type résultat
addition	+	réel : a+b
soustraction	-	réel : a-b
multiplication	*	réel : a*b
division réelle	/	réel : a/b

## Chapitre 2 Les types simples et leurs opérations

### III.3) Les caractères

Les caractères utilisés sont définis dans une table spécifique : la table **ASCII** (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange : code américain normalisé pour l'échange d'information).

**Notation** : Une valeur caractère sera noté entre simple cotes ' ' (ou guillemets)

**Exemple** : 'z' , 'Z' , '2' , '!'

Opérations		
type opération	opérateur	type résultat
concaténation		chaîne

**Exemple** : 'a' | 'e' ⇒ "ae"

**NB** : chaque caractère dans la table des codes ASCII possède un rang (sa position dans la table). Comparer 2 caractères revient à comparer leurs rangs dans la table.

## Chapitre 2 Les types simples et leurs opérations

### III.4) Les chaînes de caractères

**Définition** : Une chaîne de caractères est une suite de caractères possédant une taille qui représente le nombre de caractères qui la composent.

**Exemple** : nom ← "Dupond"      | | nom (chaîne)

Pour récupérer la **taille** d'une chaîne, on supposera qu'il existe une **opération** (fonction mathématique) **prédéfinie** **longueur** qui calcule le nombre de caractères qui composent la chaîne.

**Exemple d'utilisation** :

nom ← "Titi"                      | | nom (chaîne)

taille ← **longueur** (nom)        | | taille (entier)

(Contenu de la variable taille = 4)

**NB** : Tout objet **prédéfini** n'a pas à être défini dans le lexique. (C'est le cas de la fonction **longueur** par exemple)

## Chapitre 2 Les types simples et leurs opérations

### Chaîne vide :

❑ C'est la chaîne de **longueur 0**

❑ Représentation : " "

#### Rappel :

- une valeur caractère sera mise entre ' '
- une valeur chaîne sera mise entre " "

D'où : "a" et 'a' sont 2 valeurs de types différents : une chaîne et un caractère.

Opérations spécifiques		
type opération	opérateurs	type résultat
concaténation		chaîne
calcul de la taille	longueur	entier

L.ZERTAL

15

## Chapitre 2 Les types simples et leurs opérations

### Exemple :

debut ← "il fait "

fin ← "beau!"

message ← debut | fin

ecrire (message)

debut, fin (chaîne) : début et fin du texte

message (chaîne) : résultat

Texte affiché: "il fait beau! " : Les deux variables ont été concaténées.

#### Représentation :

	i					f		a		i		t			b		e		a		u		!	
1	2	3	4	5	6	7	8	9	10	11	12	13												

L.ZERTAL

16



## Chapitre 2 Les types simples et leurs opérations

### Autre exemple :

message[2] ← 'L'  
message[4] ← 'F'  
message[13] ← '?'

Ces 3 actions modifient le texte qui devient :

	i		L				F		a		i		t				b		e		a		u		?	
1	2	3	4	5	6	7	8	9	10	11	12	13														

## Chapitre 2 Les types simples et leurs opérations

### Utilisation :

Soit l'instruction :

taille ← **longueur**(message)

**alors :**

message[13] ← '?'

**ou**

message[**taille**] ← '?'

**ou**

message[**longueur**(message)] ← '?'

donnent le même résultat

### Comparaison :

On utilise l'ordre alphabétique ( c.f dictionnaire) :

"AABC" > "AABBZ "

"AABCA" < "AABCZ"

## Chapitre 2 Les types simples et leurs opérations

### III.5) Les booléens :

Domaine de valeurs : {vrai,faux}

Opérateurs	type résultat
<u>et</u>	booléen
<u>ou</u>	booléen
<u>non</u>	booléen

#### Tables de vérité :

a	<u>non</u> a
vrai	faux
faux	vrai

a	b	a <u>et</u> b	a <u>ou</u> b
vrai	vrai	vrai	vrai
vrai	faux	faux	vrai
faux	vrai	faux	vrai
faux	faux	faux	faux

L.ZERTAL

19

## Chapitre 2 Les types simples et leurs opérations

#### Propriétés :

Soient a et b deux expressions/variables de type booléen :

✓ ou (disjonction) et et (conjonction) sont commutatifs et associatifs :

$$a \text{ et } b \Leftrightarrow b \text{ et } a \quad a \text{ ou } b \Leftrightarrow b \text{ ou } a$$

$$(a \text{ et } b) \text{ et } c \Leftrightarrow a \text{ et } (b \text{ et } c) \quad (a \text{ ou } b) \text{ ou } c \Leftrightarrow a \text{ ou } (b \text{ ou } c)$$

✓ ou et et sont distributifs l'un par rapport à l'autre :

$$a \text{ et } (b \text{ ou } c) \Leftrightarrow (a \text{ et } b) \text{ ou } (a \text{ et } c)$$

$$a \text{ ou } (b \text{ et } c) \Leftrightarrow (a \text{ ou } b) \text{ et } (a \text{ ou } c)$$

✓ Idempotence :  $a \text{ ou } (a \text{ et } b) \Leftrightarrow a$

#### Propriétés :

✓ Théorème de Morgan :

- $\text{non} (a \text{ et } b) \Leftrightarrow \text{non } a \text{ ou } \text{non } b$
- $\text{non} (a \text{ ou } b) \Leftrightarrow \text{non } a \text{ et } \text{non } b$

✓ Comparaison :

On peut comparer des variables de type booléen sachant que :

**FAUX < VRAI**

$$a \neq b, \quad a < b, \quad a > b, \quad a \leq b, \quad a \geq b$$

✓ Autres :

$$\text{si } a = \text{vrai} \Leftrightarrow \text{si } a$$

$$\text{si } a = \text{faux} \Leftrightarrow \text{si } \text{non } a$$

L.ZERTAL

20

## Chapitre 2 Les types simples et leurs opérations

### IV Le type scalaire

Définition : c'est un type dont l'ensemble de valeurs est

- ordonné
- discret (entre 2 valeurs consécutives, il n'existe pas une infinité de valeurs)

- Types scalaires simples : *entier, caractère, booléen*
- Réel et chaîne ne sont pas des types discrets (entre 2 valeurs successives, il existe une infinité de valeurs) .

On peut utiliser 2 fonctions prédéfinies sur le type scalaire : **Pred** et **Succ** (prédécesseur et successeur)

Exemple :

Pred ('Z') = 'Y'	Pred (2) = 1	Succ ('B') = 'C'
Pred (vrai) = faux	Succ (faux) = vrai	Succ(5) = 6