

Chapitre 3

Expressions

1. Définitions

Une expression simple est :

- une valeur (constante explicite)

13 21.73 't'

- l'identificateur d'une variable (c'est à dire l'accès à sa valeur) : nb, g ou lettre

158	9.81	x
nb	g	lettre

- la combinaison de valeurs et/ou identificateurs de variables et d'opérateurs (et d'appels de fonctions qui seront abordés ultérieurement :

nb + 10 où nb et 10 sont les opérandes et + l'opérateur

Dans les expressions complexes, les opérandes peuvent être eux mêmes des expressions :

(nb + 10) * G

2. Type d'une expression

C'est le type de son résultat :

nb + 10 est de type **entier**
 (nb + 10) * G est de type **réel**
 'lettre' est de type **caractère**

Les expressions numériques ne posent pas de problèmes car les étudiants y sont habitués, mais l'extension de la notion de calcul à d'autres types que numériques est moins évidente.

3. Expressions numériques

Soit l'expression " a **op** b " où a , b sont les opérandes et **op** l'opérateur.

	a	entier	réel
b			
entier	+ - * div mod	➔ entier	+ - * / ➔ réel
Réel	+ - * /	➔ réel	+ - * / ➔ réel

Les opérateurs *div* et *mod* sont ceux de la division euclidienne : $a = b * q + r$ avec $r < b$

$$q = a \text{ div } b$$

$$r = a \text{ mod } b$$

quotient entier dans la division entière de a par b reste dans la division entière de a par b

exemple : $32 \text{ div } 5$ vaut 6 et $32 \text{ mod } 5$ vaut 2

Priorité : les opérateurs $\{ * / \text{ div mod}, \}$ sont prioritaires sur les opérateurs $\{ + - \}$

4. Expressions booléennes

➤ deux valeurs portées par les symboles *vrai* et *faux*.

➤ on peut déclarer des variables de type booléen.

exemples :

possible : booléen

trouve : booléen

possible \leftarrow *vrai*

Attention : *vrai* est un symbole et non pas un identificateur de variable

➤ Une expression booléenne peut être réduite à une variable booléenne

exemple :

possible \leftarrow trouve

➤ Une expression booléenne peut être construite à partir d'entiers et de réels (plus tard à partir de caractères et de chaînes) avec les opérateurs :

$<$ $<=$ $>$ $>=$ $=$ $<>$

exemples :

$46 > 100$

est une expression booléenne dont la valeur est *faux*

Si x est une variable réelle contenant la valeur 5.45,

$x < 10$

est une expression booléenne dont la valeur est *vrai*.

➤ Une expression booléenne peut aussi être construite à partir d'autres expressions booléennes et des opérateurs :

non , et , ou

Tables d'opération de ces opérateurs (on dit aussi tables de vérités car ce sont des opérateurs booléens)

A	non A
<i>vrai</i>	<i>faux</i>
<i>faux</i>	<i>vrai</i>

A	B	A et B	A ou B
<i>vrai</i>	<i>vrai</i>	<i>vrai</i>	<i>vrai</i>
<i>vrai</i>	<i>faux</i>	<i>faux</i>	<i>vrai</i>
<i>faux</i>	<i>vrai</i>	<i>faux</i>	<i>vrai</i>
<i>faux</i>	<i>faux</i>	<i>faux</i>	<i>faux</i>

non ($46 > 100$) \rightarrow *vrai*

$20 > 3$ et $20 < 15$

vrai et *faux* \rightarrow *faux*

$20 > 3$ ou $20 < 15$

vrai ou *faux* \rightarrow *vrai*

Priorité : *non* est prioritaire sur *et* qui est prioritaire sur *ou*

Lois de De Morgan	$\text{non}(A \text{ et } B) \equiv \text{non } A \text{ ou non } B$	$\text{non}(A \text{ ou } B) \equiv \text{non } A \text{ et non } B$
--------------------------	--	--

On peut les démontrer en construisant des tables de vérité.

exemple : 3 notes sont rangées dans 3 variables $n1$, $n2$, $n3$. L'expression booléenne qui exprime l'admission à un module (moyenne arithmétique ≥ 10 et pas de note éliminatoire inférieure à 5)

$(n1 + n2 + n3) / 3 \geq 10$ et **non** ($n1 < 5$ **ou** $n2 < 5$ **ou** $n3 < 5$)
est équivalente à :

$(n1 + n2 + n3) / 3 \geq 10$ et ($n1 \geq 5$ **et** $n2 \geq 5$ **et** $n3 \geq 5$)

➤ Les expressions booléennes peuvent être construites avec l'opérateur \in et un ensemble :

$\text{mois} \in \{1, 3, 5, 7, 8, 10, 12\}$ exprime que l'entier rangé dans la variable *mois* représente un mois à 31 jours.

Remarques :

Les conditions qui contrôlent les instructions structurées sont des expressions booléennes. C'est leur valeur *vrai* ou *faux* qui détermine le fonctionnement de l'instruction.

Exemple d'utilisation dans un algorithme :

Algorithme : ExprBooleennesV1

Variables

$n1, n2, n3$: réel

Début

lire($n1, n2, n3$)

si $(n1 + n2 + n3) / 3 \geq 10$ et non ($n1 < 5$ ou $n2 < 5$ ou $n3 < 5$) alors

 écrire('admis')

sinon

 écrire('refusé')

finsi

Fin

ou encore

Algorithme : ExprBooleennesV2

Variables

$n1, n2, n3$: réel

estAdmis : booléen

Début

lire($n1, n2, n3$)

estAdmis $\leftarrow (n1 + n2 + n3) / 3 \geq 10$ et non ($n1 < 5$ ou $n2 < 5$ ou $n3 < 5$)

si estAdmis alors

 écrire('admis')

sinon

 écrire('refusé')

finsi

Fin

Dans le second exemple, l'expression booléenne qui suit le "si" est réduite à une variable booléenne. Au lieu de noter `si estAdmis = vrai`, on a utilisé `si estAdmis`, car les 2 expressions booléennes sont équivalentes.

<code>a = vrai</code> \equiv <code>a</code>	et	<code>a = faux</code> \equiv <code>! a</code>
---	----	---

Preuve :

a	a = vrai	! a	a = faux
vrai	vrai = vrai vrai	faux	vrai = faux faux
faux	faux = vrai faux	vrai	faux = faux vrai

5. Expressions caractères et chaînes de caractères

type **caractère** : un seul caractère

type **chaîne** : suite de caractères (éventuellement un seul)

Arthur

nom

nom : chaîne de caractère

nom \leftarrow 'Arthur'

écrire('Son nom est : ', nom)

\rightarrow Son nom est Arthur

Attention : Comme les identificateurs sont constitués de caractères, il y a ambiguïté entre un identificateur et une chaîne de caractères. Pour lever cette ambiguïté, toutes les valeurs de type caractère ou chaîne sont encadrées par des **séparateurs**.

nom	'nom'
est l'identificateur d'une variable de type chaîne de caractères, représente une zone mémoire qui contient la valeur 'Arthur'	est une valeur de type chaîne de caractères, valeur qui peut être rangée dans la variable <i>mot</i> .

Arthur

nom

nom

mot

Il existe un opérateur de construction : l'opérateur de **concaténation** qui met bout à bout deux caractères/chaînes

'Bonjour' + 'Arthur' \rightarrow 'BonjourArthur'

'Bonjour' + ' Arthur' \rightarrow 'Bonjour Arthur'

'51' + '200' \rightarrow '51200'

51 + 200 \rightarrow 251

Toute expression résultat d'une concaténation entre caractères et/ou chaînes est de type chaîne.

Expressions booléennes construites à partir de caractères ou de chaînes

L'ordre est **alphabétique**. Il provient du fait que les caractères sont codés en respectant l'ordre alphabétique. Par contre, en machine, ce code ne respecte ni les accents, ni les majuscules.

'ELEPHANT' < 'SOURIS' \rightarrow *vrai*

'100' < '40' \rightarrow *vrai* car '1' est placé avant '4'

mais 100 < 40 \rightarrow *faux*