ESATIC ECOLE SUPERIEURE AFRICAINE DES TIC	Année scolaire 2014 - 2015 PROF : KONAN HYACINTHE LICENCE 1 SRIT

INTRODUCTION A L'ALGORITHMIQUE ET A LA PROGRAMMATION

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 **REEL**
'lettre' est de type **CARACTERE**

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**	+ - * /	→ **réel**

Les opérateurs **div** et **mod** sont ceux de la division euclidienne :
a = b * q + r avec r < b
q = a **div** b : quotient entier dans la division entière de a par b
r = a **mod** b : reste dans la division entière de a par b

exemple :
32 **div** 5 vaut 6
32 **mod** 5 vaut 2

Priorité :
les opérateurs {* / 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 ← **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 ← **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**
vrai	faux
faux	vrai

A	**B**	**A et B**	**A ou B**
vrai	vrai	vrai	vrai
vrai	faux	Faux	vrai
faux	vrai	faux	vrai
faux	faux	faux	faux

non (46 > 100) = **VRAI**
(20 > 3) et (20 < 15) = **FAUX**
VRAI et FAUX = **FAUX**

(20 > 3) ou (20 < 15) = **FAUX**
VRAI ou FAUX = **VRAI**

Priorité :
NON est prioritaire sur **ET** qui est prioritaire sur **OU**

Lois de De Morgan
non (A et B) ≡ (non A) ou (non B)
non (A ou B) ≡ (non A) 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 >= 10] et [non ((n1 < 5) ou (n2 < 5) ou (n3 < 5))]
est équivalente à :
[(n1 + n2 + n3) / 3 >= 10] et [((n1 >= 5) et (n2 >= 5) et (n3 >= 5))]

- Les expressions booléennes peuvent être construites avec l'opérateur ∈ et un ensemble :
  mois ∈ {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 : **REEL**
DEBUT
    LIRE (n1, n2, n3)
    SI (n1 + n2 + n3) / 3 ≥ 10 **ET NON** (n1 < 5 OU n2 < 5 OU n3 < 5) **ALORS**
        ECRIRE ('admis')
    SINON
        ECRIRE ('refusé')
    FINSI
FIN

ou encore

ALGORITHME : ExprBooleennesV2
    VARIABLES n1, n2, n3 : **REEL**
                estAdmis : **BOOLEEN**
DEBUT
    LIRE (n1, n2, n3)
    estAdmis ← (n1 + n2 + n3) / 3 ≥ 10 **ET NON** (n1 < 5 ou n2 < 5 ou n3 < 5)
    SI estAdmis **ALORS**
        ECRIRE ('admis')
    SINON
        ECRIRE ('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.

a = vrai ≡ a et a = faux ≡ ! a

Preuve :

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

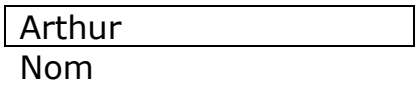
5. Expressions caractères et chaînes de caractères

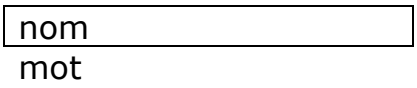
type **CARACTERE** : un seul caractère
type **CHAINE** : suite de caractères (éventuellement un seul)

nom : **CHAINE**
nom ← "Arthur"
ECRIRE ("Son nom est : ", nom)

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 mot.





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

"Bonjour" + "Arthur" = "BonjourArthur"
"Bonjour" + " Arthur" = "Bonjour Arthur"
"51"+"200" = "51200"
51 + 200 = 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" = **VRAI**
"100" < "40" = **VRAI** car "1" est placé avant "4"

Mais

100 < 40 = **FAUX**

FIN DU CHAPITRE 3