TP Informatique 2 - Types de variables

1 Affectation d'une variable

Nous avons vu que pour donner une valeur à une variable `u`, l'utilisateur peut le faire lui-même à l'aide de la commande `READLN`. On peut aussi écrire dans le programme `u:=1 ;` pour affecter la valeur 1 à la variable `u`. Si `v` est une autre variable à laquelle on a déjà affecté une valeur, on peut aussi écrire `u :=2*v ;`.

Exercice 02.1

1. Déterminer les valeurs contenues par les variables `x` et `y` à la fin de l'éxécution de ce programme :

```
PROGRAM melange ;
    VAR x,y :REAL ;
BEGIN
    WRITE('Donner la valeur de x :') ;
    READLN(x) ;
    WRITE('Donner la valeur de y :') ;
    READLN(y) ;
    x := x+y ;
    y := x-y ;
    x := x-y ;
END.
```

2. Vérifier en exécutant le programme suivant (le même sauf qu'il affiche les valeurs successives prises par `x` et `y`.

```
PROGRAM melange ;
    VAR x,y :REAL ;
BEGIN
    WRITE('Donner la valeur de x :') ;
    READLN(x) ;
    WRITE('Donner la valeur de y :') ;
    READLN(y) ;
    x := x+y ;
    WRITELN('après la première modification, x vaut ',x,' et y vaut ',y) ;
    y := x-y ;
    WRITELN('après la seconde modification, x vaut ',x,' et y vaut ',y) ;
    x := x-y ;
    WRITELN('après la troisième et dernière modification, x vaut ',x,' et y vaut
    READLN ;
END.
```

3. Que se passe-t-il lorsqu'on échange les deux dernières affectations `x :=x-y` et `y :=x-y` ? Quelles sont les valeurs finales de `x` et de `y` ?

2 Quelques types à connaître

2.1 Le type REAL

On utilise des variables de type `REAL` lorsque les valeurs prises par ces variables seront des réels. Ils sont stockés avec une précision de 11 à 12 chiffres, sous forme "scientifique" : par exemple, 117 sera stocké et affiché sous la forme `1.17E+02` (car $117 = 1,17 \times 10^2$). Vous remarquez que la virgule se note par un point, ce sont les conventions des anglophones.

On dispose :
- des opérations `+`, `-`, `*`, `/`,
- des fonctions usuelles `ln`, `exp`, `cos`, `sin`, `trunc` pour la partie entière, `abs` pour la valeur absolue, `sqrt` pour la racine carrée (de l'anglais *square root*).

Remarquez qu'il n'y a pas la fonction puissance. Attention aussi à bien parenthéser les expressions. Par exemple, pour écrire $\dfrac{u^4 + \cos(w)}{2v}$, il faudra écrire `(u*u*u*u+cos(w))/(2*v)`.

Turbo Pascal ne peut manipuler que les réels compris entre $2,9 \times 10^{-39}$ et $1,7 \times 10^{38}$. Si un programme nécessite le calcul d'une valeur non comprise dans cet intervalle, un message d'erreur s'affichera.

3 Le type INTEGER

C'est pour les variables dont on est sûr qu'elles prendront toujours des valeurs entières (positives ou négatives).
On dispose :
- des opérations `+`, `-`, `*`, mais pas de `/` car le résultat n'est pas toujours un entier,
- des fonctions `DIV` et `MOD` qui sont liées à la division euclidienne des entiers.

Rappel : soient a et b des entiers avec $b \neq 0$. Alors, il existe un unique couple (q, r) d'entiers tels que

$$a = qb + r \quad \text{et } 0 \leq r < b$$

q s'appelle le quotient de la division de a pet b, il est donné par `a DIV b`.
r s'appelle le reste de la division de a par b, il est donné par `a MOD b`.

Exemple : $37 = 9 \times 4 + 1$, donc `37 DIV 4` donne 9 et `37 MOD 4` donne 1.
Cependant, si on fait `37/4`, l'ordinateur interprète cette opération comme une opération entre réels et rend donc `9.2500000000E+00`.

Exercice 02.2

Ecrire un programme qui demande un temps T (entier) exprimé en secondes, et qui affiche ce temps en heures, minutes et secondes.

4 Le type BOOLEAN

Une variable de type `BOOLEAN` (booléenne, du mathématicien Boole) est une variable qui ne peut prendre que deux valeurs, `TRUE` ou `FALSE`.
Par exemple, si `x` est une variable de type `REAL`, on peut former `(x<0)` qui est de type `BOOLEAN`. Si la valeur de `x` est 1 par exemple, alors `(x<0)` vaut `FALSE`.

Exercice 02.3

Taper, compiler et exécuter le programme suivant :

```
PROGRAM signe ;
    VAR x0 :REAL ;
        b :BOOLEAN ;
BEGIN
```

```
        WRITELN('Donner la valeur de x0 :') ;
        READLN(x0) ;
        b :=((x0*x0+x0-6)>0) ;
        WRITELN('It is ',b,' that x->x*x+x-6 is positive at the point x0.') ;
    END.
```

C'est un type utile pour certaines structures comme `IF...THEN...ELSE`. Par exemple, le programme ci-dessus peut être amélioré de la façon suivante :

```
PROGRAM signe ;
    VAR x0 :REAL ;
BEGIN
    WRITELN('Donner la valeur de x0 :') ;
    READLN(x0) ;
    IF ((x0*x0+x0-6)>0) THEN
        WRITELN('La fonction est strictement positive en x0')
        ELSE WRITELN('La fonction est négative en x0') ;
END.
```

Remarques importantes :

- Pour avoir la variable booléenne qui vaut `true` si x est positif ou nul et `false` si x est strictement négatif, on écrit `x>=0`,
- Pour avoir la variable booléenne qui vaut `true` si x est non nul et `false` si x vaut 0, on écrit `x<>0` (le symbole `<>` est donc la façon d'écrire $\neq$).
- Soient `b1` et `b2` des variables de type `BOOLEAN`, on peut former `b1 and b2` et `b1 or b2`. Exemple, on peut écrire `(x<>1) and (x<>2)`.

Exercice 02.4

Corriger les programmes suivants pour qu'ils fonctionnent et écrire une phrase expliquant ce que renvoie chaque programme.

1.
```
PROGRAM exercice_4_question_1 ;
    VAR a,b,c : INTEGER ;
        d : BOOLEAN ;
BEGIN
    READLN(a) ;
    READLN(b) ;
    READLN(c) ;
    d := a < (b and c) ;
    WRITELN(d) ;
END.
```

2.
```
PROGRAM exercice_4_question_2 ;
    VAR a,b,c : INTEGER ;
        d : BOOLEAN ;
BEGIN
    READLN(a) ;
    READLN(b) ;
    READLN(c) ;
    d := (a <= b) or b ;
    WRITELN(d) ;
END.
```

5 Compléments

Turbo Pascal ne peut manipuler que les entiers entre -2^{15} et $2^{15}-1$, c'est-à-dire entre -32768 et 32767. Cette fois, nous allons l'expliquer.
C'est dû au fait que l'ordinateur n'utilise que 0 et 1 pour coder tout objet. Un **bit** informatique est une mini-case mémoire destinée à contenir soit 0, soit 1, et pour coder un objet on utilise un certain nombre de bits. Les bits sont utilisés par paquets de 8, un tel paquet est appelé un **octet**.
En particulier, avec Turbo-Pascal, chaque entier a droit à 2 octets pour être codé, c'est-à-dire 16 bits. Ainsi, chaque suite de 0 et de 1 avec 16 éléments va correspondre à un nombre entier. Comme le nombre de telles suites possibles est 2^{16}, on ne peut coder que 2^{16} entiers. C'est bien le nombre d'entiers dans l'intervalle ci-dessus (les 2^{15} entiers de $-2^{15}-1$ et les 2^{15} entiers de 0 à $2^{15}-1$).
On peut se demander : comment code-t-on un entier de cette manière ? En fait, le code associé à un entier $n \in \mathbb{Z}$ est l'écriture en base 2 de $|n|$, plus un bit pour le signe (0 si n est positif, 1 si n est négatif par exemple).

Propriété. Tout entier $n \in \mathbb{N}$ s'écrit de manière unique sous la forme

$$n = a_0 + a_1 \times 2 + a_2 \times 2^2 + \cdots + a_p \times 2^p,$$

où $p \in \mathbb{N}$ et $a_0, a_1, \ldots, a_{p-1}$ valent 0 ou 1, $a_p = 1$.

Puisqu'on a existence et unicité de cette écriture, on va se servir de la suite $a_0 a_1 \ldots a_p$ pour coder n. Nous ne pouvons utiliser que 15 bits (le 16-ième est réservé pour le signe), donc p vaut 14 au maximum ; comme n est maximal, lorsque tous les a_i valent 1, le plus grand entier que l'on puisse coder est

$$1 + 2 + 2^2 + \cdots + 2^{14} = \sum_{k=0}^{14} 2^k = \frac{1-2^{15}}{1-2} = 2^{15} - 1,$$

comme annoncé plus haut.

Exemples : l'entier 1 est codé par (on omet l'octet réservé au signe) 100000000000000 ; l'entier 2 est codé par 010000000000000 ; l'entier 29 vaut $1 + 2.14 = 1 + 2.2.7 = 1 + 2.2.(1 + 2 + 2^2) = 1 + 2^2 + 2^3 + 2^4$, donc il est codé par 101110000000000.

Exercice 02.5

Trouver le code binaire des entiers suivants : $227, 1008$.