TP N ° 2

I. LOGICIELSEt LANGAGES DE PROGRAMMATION

Dans la majorité des cas, on achète des programmes (logiciels) tous faits qui correspondent plus ou moins au besoin :

* Traitement de texte - P.A.O : avec mise en page, justification, numérotation chapitres-pages, table des matières, dictionnaire...
* Tableur : tableau de nombres à 2 dimensions et calculs
* Base de données : ensemble de fiches (nom, adresse...) et recherche par rubrique, publipostage...
* C.A.O, Dessin par ordinateur : propre, modification aisée, archivage...
* Gestion : paye, facturation, stock...
* Communication : transfert de programmes par modem et ligne téléphonique, serveur minitel...

Un Intégré regroupe plusieurs de ces possibilités.

Soit on achète un logiciel général : très bon niveau, parfaitement testé, documentation, formation... mais trop général (fonctions inutiles, fonctions utiles avec trop de paramètres ou difficilement accessibles). Soit on fait (ou fait faire) un logiciel particulier : plus pratique, mais plus hasardeux (erreurs, SAV, doc...). et l'obligation d'un cahier des charges qui doit être très précis.

Un ordinateur donc est une machine ne sachant qu'obéir à des opérations comme:

- addition, soustraction, multiplication en binaire, uniquement sur des entiers,
- sortir un résultat ou lire une valeur binaire (dans une mémoire par exemple),
- comparer des nombres.

Sa puissance vient du fait qu'il peut être PROGRAMME, c'est à dire que l'on peut lui donner, à l'avance, la séquence (la suite ordonnée) des ordres à effectuer l'un après l'autre. Le grand avantage de l'ordinateur est sa rapidité. Par contre, c'est le programmeur qui doit TOUT faire. L'ordinateur ne comprenant que des ordres codés en binaire (le langage machine), des langages dits "**évolués"** ont été mis au point pour faciliter la programmation.

II. PROGRAMMATION EN EN PASCAL

Le **PASCAL**, créé par WIRTH au début des années 70, possède des instructions assez claires, et favorise une approche méthodique et disciplinée (on dit "structurée"). Le langage Pascal offre une très bonne approche de la programmation. Très utilisé dans le milieu scolaire, il permet d'acquérir des notions solides que l'on retrouve dans tous les autres langages.

Le PASCAL est un **langage compilé**, c'est à dire qu'il faut :

* entrer un texte dans l'ordinateur (à l'aide d'un programme appelé **EDITEUR**),
* le traduire en langage machine (c'est à dire en **codes binaires** compréhensibles par l'ordinateur) : c'est la compilation et éventuellement l'édition de liens **(LINK**),
*le compiler : c'est à dire détecter les erreurs et les corriger.
* l'exécuter pour avoir les résultats attendus.

Un programme PASCAL est composé **d'une entête**, **des déclarations** et **des instructions** (délimitées par **BEGIN** et **END**).

L'entête est composée du mot PROGRAM, suivi du nom du programme (cercle), et d'indications sur les Entrées/Sorties (ici le clavier et l'écran).

La partie déclarative de notre programme est limitée à la déclaration de deux variables (mot clef VAR). Une variable est une "case" mémoire de l'ordinateur, à laquelle on donne ici un nom. Chaque case peut contenir une valeur.

Les types simples connus en PASCAL sont : REAL, INTEGER (entier naturel), CHAR (contient UN est un seul caractère), et BOOLEAN (booléen, c.à.d. qui peut valoir soit TRUE (vrai) soit FALSE (faux). En TURBO PASCAL,

La virgule décimale est toujours représentée par un point en informatique.

Un identificateur (tout nom que vous choisissez : variable, programme...) peut être formé de lettres (A à Z), de chiffres et (pas sur toutes les versions de PASCAL) du signe _ (souligné).

TURBO PASCAL accepte des noms de 127 caractères maximum.Le premier caractère doit être une lettre. Par exemple, VALEUR1 ou PREM_VALEUR sont possibles mais pas 1ERE_VALEUR.

En PASCAL les minuscules sont traitées comme des majuscules (SURface et surFACE désignent la même case mémoire).

Les accents et autres ç ne sont pas autorisés (var diamètre:real est interdit à cause de l'accent). Un blanc dans un identificateur est également interdit (utilisez _ pour séparer des mots dans un même identificateur).

Toute variable utilisée dans un programme doit être déclarée. Ceci évite la plupart des erreurs de frappe, et rend le programme plus compréhensible.Les instructions de notre programme sont :

* lecture sur le clavier : le programme s'arrête, attend que l'on donne une valeur à l'aide du clavier, met cette valeur dans la case (variable)correspondante en RAM et continue le programme lorsque l'on appuie sur la touche "ENTREE" ou "RETURN". La commande `read` permet à l'utilisateur de rentrer une valeur qui sera utilisée par leprogramme. Cette commande ne provoque pas de retour Chariot, c'est-à-dire que le curseur nepasse pas à la ligne, La commande `ReadLn provoque` le curseur de passe à la ligne suivante.

Syntaxe :

`ReadLn (`*variable1*`,` *variable2*`) ;`

* calcul et affectation : après réalisation du calcul le résultat sera et dans la case correspondante.
Le **:=** symbolise une flèche à gauche. Ce n'est PAS une égalité au sens mathématique, mais la copie d'une valeur dans une mémoire.

*** écriture sur l'écran**: La commande `write` permet d'afficher du texte et de laisser le curseur à la fin du texte affiché, La commande `WriteLn` est semblable à la précédente à la différence près que le curseur est maintenant renvoyé à la ligne suivante. Les textes doivent être entourés de cotes ('). Les majuscules/minuscules sont significatives

Syntaxe :

`Write ('`*Texte à afficher*`',` *variable1*`,` *variable2*`, '`*texte2*`') ;`

`Write ('`*L''apostrophe se double.*`') ;`

Exemples

```
Program exemple1;
Var nom : String ;
BEGIN
Write('Entrez votre nom : ') ;
```

```
ReadLn(nom) ;
WriteLn('Votre nom est ', nom) ;
ReadLn ;
END.
```

Exemple2 :

```
PROGRAM cercle (input,output); (* entête *)
VAR perimetre,diametre : REAL; (* déclarations *)
BEGIN
        readln(diametre);       (* instruction *)
        perimetre := 3.141592 * diametre;      (* instruction *)
        writeln(diametre,perimetre)    (* instruction *)
END.
```

Remarque :

Par défaut, lorsque l'on affiche un nombre réel avec **Write** ou **Writeln**, il est affiché sous forme scientifique, correspond à une mantisse réelle m à 20 chiffres (1<=m<10) et un exposant e signé à 4 chiffres. Ainsi le nombre 15 serait 1.50000000000000E +0001. Pour l'écrire sous forme 15.0, Il faut utiliser les spécificateurs de précision. Pour ce faire on ajoute derrière le nombre 2 points « : », la taille du champ, puis encore 2 points et le nombre de chiffres souhaités après la virgule, comme ceci :**(15 :4 :2).**Cet exemple affiche le nombre 15, sur un total de 4 caractères et avec **2** chiffres après la virgule. Il est à noter que la syntaxe est la même pour transformer un nombre réel en chaine avec l'instruction **Str .**

Les instructionsdoivent toujours être séparées par des "**;**" (j'ai dit "séparées", pas "terminées").
Le fait de passer à la ligne n'est interprété par l'ordinateur que comme un blanc. On aurait donc pu écrire notre programme sur une seule ligne (peut-être un peu longue pour l'éditeur).
Le programme doit toujours se terminer par **un point**.

Remarques :On peut insérer des remarques dans le programme (qui ne seront pas lues par le compilateur) en les entourant par (* et *) ou { et }. On ne peut en standard pas imbriquer des commentaires. Les commentaires peuvent faire plus d'une ligne, ceci permet de supprimer momentanément une partie d'un programme.

MOTS RESERVES DU LANGUAGE PASCAL

```
AND, ARRAY, ASM, BEGIN, CASE, CONST, CONSTRUCTOR, DESTRUCTOR,
DIV, DO, DOWNTO, ELSE, END, EXPORTS, FILE, FOR, FUNCTION,
GOTO, IF, IMPLEMENTATION, IN, INHERITED, INLINE, INTERFACE,
LABEL, LIBRARY, MOD, NIL, NOT, OBJECT, OF, OR, PACKED,
PROCEDURE, PROGRAM, RECORD, REPEAT, SET, SHL, SHR, STRING,
THEN, TO, TYPE, UNIT, UNTIL, USES, VAR, WHILE, WITH, XOR
```

OPERATEURS MATHEMATIQUES

Addition (et union1) +
Soustraction (et complément1) −
Division /
Multiplication (et intersection1) *
Egalité =
MOD : renvoie le reste de la division x MOD y
DIV : renvoie le quotient de la division x DIV y
*Opérateurs prioritaires : *, /, DIV et MOD. Opérateurs secondaires : + et −. Vous pouvez utiliser des parentèses.*

Opérateurs relationnels

Inférieur strict <
Inférieur ou égale (et inclu1) <=
Supérieur strict >
Supérieur ou égale (et contenant1) >=
Différent <>
Opérateur ultra-prioritaire : `NOT`. Opérateur semi-prioritaire : `AND`. Opérateur non prioritaires : `OR` et `XOR`.

Opérateurs logiques :

`AND` : le "et" logique des maths (voir chapitre 15 sur les booléens et tables de vérité)
`OR` : le "ou"
`XOR` : le "ou" exclusif
`NOT` : le "non"

Priorité des opérateurs

Niveau 1 : `NOT`.
Niveau 2 : `*`, `/`, `MOD`, `DIV`, `AND`.
Niveau 3 : `+`, `-`, `OR`, `XOR`.
Niveau 4 : =, <, >, <=, >=, <>.

LES PRINCIPALES FONCTIONS STANDARD CONNUES PAR TOUS LES COMPILATEURS SONT :

```
ABS : renvoie la valeur absolue SIN : sinus
SQR : renvoie le carré ARCTAN : arc tangente
SQRT : racine carrée EX : exponentielle
COS : cosinus LN : log népérien
SUCC : variable énumérée suivante PRED : précédent
ROUND : arrondi à l'entier le plus proche
TRUNC : partie entière (permet de mettre un réel dans un entier:
trunc(4.5)=4)
```

Comme toute variable, une fonction possède un type (entier, réel,...) défini, et ne peut donc être utilisée que comme une variable de ce type.

DEROULEMENT D'UN PROGRAMME EN PASCAL

Pour ouvrir un fichier, aller dans le menu **File/Open...** ou taper la touche fonction **F3**. Pour exécuter un programme, aller dans le menu **Run/Run** ou taper la combinaison de touches **Ctrl+F9**.
Pour compiler "correctement" un exécutable, aller dans le menu **Compile/Make** (ou **/Compile**) ou taper **F9** on obtient ainsi des exécutables de meilleurs qualité qui pourront être utilisés sur d'autres ordinateurs.

Si vous avez ommis de mettre une pause à la fin d'un programme, ou si vous désirez tout simplement avoir sous les yeux, la dernière page d'écran, il vous suffit d'alllez dans le menu :
Debug/User Screen ou tapez **ALT+F5**.

Pour une aide, aller dans le menu **Help/Index** ou taper **Shift+F1**. Pour obtenir de l'aide surune instruction qui apparait dans un script, placez le curseur de la souris dessus et allez dansle menu **Help/Topic Search**, une fenêtre apparaîtra alors.

Si un problème a lieu lors de l'exécution d'un programme, utilisez le débugger : **Debug/Watch**. Une fenêtre apparait en bas de page. Cliquez sur **Add** et tapez le nom de la variable dont vous désirez connaître la dernière valeur.