(2) Programmation en Pascal : objets et instructions élémentaires

0.) généralités-exemples

Le programme le plus simple en Pascal — commentaires

```
program tressimple ;        nom du programme (facultatif sous TPas), avec point-virgule
BEGIN                       début du programme (N.B. : casse indifférente)
END.                        fin du programme, terminé par un point.
```

Que fait ce programme : rien!

Un programme simple en Pascal — l'algorithme associé

```
program echange ;                          Algorithme Echange

(* partie déclarative *)                   donnée : deux entiers saisis au claviers
var x, y, z : integer ;                    résultat : permutation des valeurs de x et de y

(* partie exécutive *)                     procédure :
BEGIN                                          saisir x
write(' premier entier ? ') ;
readln(x) ;
write('deuxième entier? ') ;                   saisir y
readln(y) ;
z:=x ; x:=y ; y:=z ;                           z<-x ; x<-y ; y<-z
writeln('x= ', x, 'y= ', y) ;                  afficher x puis y.
(* readln ; *)
END.
```

Commentaires
Les commentaires depuis (* jusqu'à *) (facultatifs!), ignorés par le compilateur.
Tout le reste est lu par le compilateur, avec caractères non-accentués et confusion MAJ-MIN.
Partie déclarative : on déclare les variables (cf mémoires d'un calculette) qu'on veut utiliser.
Partie exécutive : on peut afficher du texte et des valeurs, saisir une valeur dans une variable.
On peut changer la valeur d'une variable (opérateur d'affectation :=).
Ecriture d'un algorithme : voir ch(1).

Exercice : Ecrire un programme pascal qui demande l'année de naissance (voir gestion des apostrophes dans les chaînes) et la date (jour, mois, année) puis calcule l'âge correspondant. Votre programme résout-il exactement le problème de détermination de l'âge ?

Commentaire : l'algorithme (le programme) essaie de répondre au problème posé, mais il manque des données pour résoudre le problème (qui est donc mal posé). On a donc trois niveaux conceptuels : problème, algorithme, programme.

Un programme est donc organisé en trois parties :
- nom du programme en en-tête,
- partie déclarative,
- partie exécutive (commençant par begin et finissant par end.).

Un programme plus compliqué : avec blocs imbriqués (voir le TP n°1, voir le chapitre 3 pour l'analyse)
donné simplement pour information, sans commentaire

```
program PIQUET;

(* partie declarative *)

var jeu, compteur : integer ;
    nom1, nom2, nom : string[20] ;

(* partie executive *)

BEGIN (* 0 *)
(* règle du jeu *)
writeln('Jeu du piquet (2 joueurs)') ;
writeln('Vous tapez un nombre entre 1 et 10 qui s''ajoute à un compteur.') ;
writeln('Le premier qui atteint 100 gagne') ;
writeln ;

(* saisie des noms des joueurs *)
write('Nom du premier joueur ? ') ; readln(nom1) ;
write('Nom du deuxième joueur ? ') ; readln(nom2) ;
writeln ;

(* initialisation du compteur *)
compteur:=0 ;

(* deroulement du jeu *)
writeln('C''est ',nom1, ' qui commence') ;
while compteur<100 do
   begin (* 1 *)
   write('jeu de ',nom1) ;
   (* saisie avec contrôle de validité *)
   repeat
        begin (* 2 *)
        write(' (entre 1 et 10) ? ') ;
        readln(jeu) ;
        end (* 2 *)
   until (jeu>=1) and (jeu<=10) ;
   compteur:=compteur+jeu ;
   writeln('         compteur : ',compteur) ;
   (* échange des noms *)
   nom:=nom1 ; nom1:=nom2 ; nom2:=nom ;
   end ;(* 1 *)

(* ici compteur >=100 : affichage du gagnant *)
if compteur >=100 then writeln('C''est ',nom2,' qui a gagné')
                  else writeln('C''est ',nom1,' qui a gagné') ;
END. (* 0 *)
```

1.) Les trois parties d'un programme

1.1.) en-tête

Facultative en Turbo Pascal, demandée sous GPC.

1.2.) la partie déclarative

Un exemple :

```
program achat ;                              Programme en Turbo-Pascal
USES crt ;
CONST pi=3.14 ;
          euro = 6.55957 ;
          franc = 1/6.55957 ;                voir les expressions §4
VAR prixunitaire, prixtotal : real ;         voir les variables §3
       nombre : integer ;
       prixeuros : real ;
BEGIN
clrscr;
write('prix à l'unité des CD (en francs) ? ') ;
readln(prixunitaire) ;                       voir la saisie §3.2
write('nombre de CD ?') ;
readln(nombre) ;
prixtotal:=prixunitaire*nombre ;             voir l'affectaztion §3.1
writeln('prix total :',prixtotal:0:2,' francs') ;
writeln('soit : ',(prixtotal/euro):0:2) ;    voir l'affichage §5
readln ;
END.
```

Commentaires :

La partie déclarative contient trois zones successives et facultatives commençant respectivement par les mots réservés USES, CONST et VAR.

La clause USES ajoute des "unités" au programme et permet d'avoir des intructions supplémentaires (clrscr pour crt). On leur dira quand une telle clause est utile.

La clause CONST contient la déclaration des constantes : cases-mémoire dont la valeur sera fixe. On peut y placer le résultat d'un calcul représenté par une **expression**.

La clause VAR contient la déclaration des variables. Il faut donner aux variables et constantes des noms suffisemment longs et clairs pour etre explicites. Les lettres uniques seront réservées aux variables intermédiaires fréquemment utilisées et peuvent suivre la convention suivante :

i,j,k pour des BYTE ou WORD et plus généralement des indices,
m,n,p,q pour des INTEGER,
x,y,z pour des REAL,
c,d,e pour des CHAR,
s,t,u pour des STRING.

Pourquoi déclarer les variables et constantes :

Le compilateur, quand il crée le programme à partir du code source a besoin de savoir quelles cases-mémoire doivent être réservées pour les variables. Cette obligation de déclarer au préalable les variables est une contrainte de rigueur dans l'écriture des programmes, car il faut prévoir avant l'édition du programme les variables que l'on va utiliser, ou au moins, en faire l'inventaire après usage.

De plus, le Pascal est un langage typé, cela signifie qu'une variable (mais aussi les constantes) ne peut prendre de valeurs que dans un ensemble de valeurs défini par son type. On utilisera ce semestre 4 types simples : integer (entiers), real (nombres à virgule), char (caractères), string (texte=chaîne de caractères) et boolean (booléens) qui sont décrits dans le paragraphe 2, et auxquels on rajoutera le type élaboré array (tableaux indicés).

Le typage de Pascal est dit fort, on ne peut mettre un caractère dans une variable numérique, même si dans la machine le caractère est mémorisé par un code qui est un nombre. Certaines opérations, naturelles, sont cependant possibles (Voir paragraphe 3).

De plus, les constantes et les variables peuvent former, grâce aux opérateurs, aux fonctions et aux parenthésage, des expressions plus ou moins complexes que le compilateur devra évaluer, et selon les cas stocker en mémoire et/ou afficher au cours de l'exécution du programme. Voir le paragraphe 4.

1.2.) la partie exécutive

Cette partie contient toutes les **instructions** destinées à réaliser ce qu'on veut faire au programme. Il est important de savoir que ce modèle de programmation où on indique en détail au programme tout ce qu'il doit réaliser (modèle impératif) n'est pas le seul modèle de programmation qui existe. Le programme peut être une simple liste d'instructions (schémas séquentiel) ou bien avoir une structure plus élaborée (schémas itératifs, voir chapitre 3).

Toutes les instructions se terminent par un point-virgule, sauf BEGIN. Autre exceptions qu'on verra ensuite : l'instruction précédant un ELSE (en Turbo Pascal).
N.B. le point-virgule est facultatif avant un END (règle pouvant être ignorée).

Parmi ces instructions, certaines servent à donner une valeur à une variable, soit par saisie (readln) soit par affectation (opérateur :=). Voir paragraphe 3.

2.) les principaux types et leur représentation en mémoire

le type INTEGER

Entiers relatifs de l'intervalle [-32768,32767] (en Turbo Pascal). Soit 65636 valeurs correspondant à un codage sur 2 octets (2^16=65536). Avec gpc, le codage est sur 32 ou 64 bits, et donc permet un intervalle de valeurs beaucoup plus vaste.
N.B. : le type BYTE correspond aux entiers naturels de l'intervalle [0..255] et le type WORD à [0..65535].

le type REAL

Nombres réels de l'intervalle [1.E-38 , 1.E+38] avec 11 chiffres significatifs (en Turbo Pascal). La mantisse est codée sur 5 octets dont un bit pour le signe (2^39#5,5x10^11) ; l'exposant est codée sur 1 octet (de -128 à 127) comme une puissance de 2 (2^-128#3,4x10^-38 et 2^127#1,7x10^38). Le codage diffère selon les Pascal (gpc). Entiers et réels forment l'ensemble de types numérique et peuvent être ordonnés.

le type BOOLEAN

Valeurs de vérité TRUE et FALSE (deux constantes). On ne s'occupe pas de la manière dont ce type est codé.

le type CHAR

Caractères codées sur un octet et correspondant à une convention appelée ASCII (et ASCII-étendus multiples). Les caractères écrits explicitement (constantes explicites) sont placés entre apostrophes. Caractères, boolean et entiers forment l'ensemble de types scalaire.

Exemples :	caractère	code décimal	code binaire
	'M'	77	01001101
	'I'	83	01001001
	'A'	65	01000001
	'S'	83	01010011
	' '	32	00100000
	'7'	55	00110111
	'<'	60	00110000
	'a'	97=65+32	01100001

Les caractères sont ordonnés par ordre croisant de code et peuvent être comparés :
'A'>'M' est vrai, 'a'>'A' est vrai, ' '>'I' est faux.

Attention, cet ordre ne correspond pas au classement alphabétique habituel où 'a' et 'A' sont considérés comme identiques, il faudra en tenir compte dans les programmes utilisant un classement alphabétique.

Comme tous les types scalaires, le type caractère possède les fonctions succ et pred (cf barreaux d'une échelle).

le type STRING[n] (avec n dans [0..255] en Turbo Pascal, snas limite sous gpc)

Chaînes de 0 à 255 caractères. Une chaîne de taille n occupe n+1 octets en mémoire : 1 octet mémorise la taille effective de la chaîne, les autres contenant la chaîne elle même. Ce type n'est pas un type élémentaire (voir les paramètres de sous-programmes) mais reste SIMPLE.

Exemple :
```
Var S string[6] ;
...
S:='ouf!' ;                          [4][o][u][f][!][ ][ ]
writeln(length(S)) ;
```

N.B. : il existe un deuxième mode de codage des chaînes de caractère (codage dit A Zéro Terminal).

Classification (non conventionnelle) des types :
- scalaires (s.s.) : boolean, entiers (integer, byte, word) et caractères (char),
- élémentaires : + réels (real),
- simples : + chaînes (string[n]).

3.) Les variables

La valeur d'une variable peut être modifiée par saisie (READLN) ou par affectation (:=). Avant l'initialisation (première saisie/affectation), la valeur de la variable est indéterminée (tout est possible !). Il est donc fortement recommandé de vérifier l'initialisation de chaque variable.

3.1.) l'affectation

C'est le procédé qui permet de changer la valeur d'une variable sans faire appel à l'utilisateur. L'écriture prend une des quatre formes ci-dessous. L'affectation place une valeur dans la variable qui est A GAUCHE du symbole d'affectation.

variable:=valeur;	la valeur est placée dans la case-mémoire de la variable en écrasant la valeur ancienne.
variable1:=variable2;	c'est la valeur de variable2 qui est placée dans variable1 (variable2 ne change pas).
variable:=constante;	comme précédemment.
variable:=expression;	l'expression est évaluée et son résultat placé dans la variable (voir paragraphe 4).

N.B. bien évidemment on ne peut réaliser l'affectation constante:=...

Attention : A:=B signifie que la variable A prend la valeur contenue dans la variable B. Dans les deux cas l'ancienne valeur (éventuelle) de A disparaît. Par conséquent, on ne peut donc pas échanger les valeurs de A et B en écrivant A:=B ; B:=A; (voir le programme ECHANGE).

3.2.) la saisie

L'instruction READLN(variable) a pour effet de :
- suspendre le programme
- saisir ce que l'utilisateur tape au clavier
- vérifier que cette saisie est compatible avec ce qu'il attend (vérifiaction de type et de bornes)
- si la saisie est conforme, placer la valeur correspondante dans la variable.
- poursuivre le programme à la suite de cette instruction.

De bonnes habitudes : ne pas utiliser READ, ne saisir qu'une variable à la fois (pas READLN(v1,v2) pexp).

3.3.) la compatibilité entre les types

L'affectation nécessite qu'il y ait compatibilité de types et de valeurs :

– Entre types numériques, l'affectation ascendante est toujours possible : byte<<ord<<integer<<real. L'affectation descendante n'est possible entre types entiers que s'il n'y a pas dépassement de valeurs (sinon plantage ou errement selon réglages du compilateur).

– Le type booléen est incompatible avec les autres types.

– L'affectation ascendante est possible des caractères vers les chaînes.

Exemple :

```
CONST t = 'chaîne longue' ;
VAR x : real ;
      n : integer ;
      i : byte ;
      c : char ;
      s : string[5]
BEGIN
(* possibles *)
x:=-5.2 ;
n:=-2 ;
i:=9 ;
n:=i ;
i:=i+1 ;  (*incrémentation*)
i:=n;      (* car 0<=n<=255 *)
x:=2*i ;
n:=30000;
s:=t ;     (* mais la chaîne sera tronquée : s='chaîn' *)
(* impossibles *)
n:=9.3 ;
x:=s;
n:=x ;
n:=3/2;
i:=365 ;
i:=n ;     (* car n>255 *)
t:=s ;
END.
```

4.) les expressions

4.1.) les expressions numériques

de type INTEGER

Les expressions utilisent :
Opérateurs : + - * div mod
fonctions : pred, succ (pour tous les scalaires)

p DIV q est le quotient entier (dividende de la division euclidienne) de p par q

```
exp :   -12 DIV 5      donne -2
        14 DIV 3       donne 4
```

p MOD q est les reste de la division euclidienne de p par q

```
exp :   14 mod 3       donne 2
        -14 mod 3      donne -2
```

Attention à certaines opérations qui peuvent faire sortir de l'étendue des entiers : SQR(182)=33124 >32767. On risque de faire planter (ici pour Turbo Pascal) le programme ou bien d'obtenir -32.412 (i.e. 33124-32767=357 et -32768-1+357=-32412).

de type REAL

Opérateurs : + - * /
fonctions : SQR(x), SQRT(x), trunc (partie entière), frac (partie fractionnaire), round (arrondi à l'entier le plus proche).

L'écriture mathématique habituelle avec traits de fraction étant impossible, il faut parfois ajouter des parenthèses selon les priorités des opérateurs.

Exp : 3.5+7/8.2/9.15 correspond à 3.5+ 7 / (8.2*9.15)
3.5+7/8.2*9.15 correspond à 3.5+ (7/8.2) * 9.15
3.5–8.2–9.15 correspond à 3.5 – (8.2+9.15)
(–b+SQRT(d))/(2*a) est différent de (-b+SQRT(d))/2*a et de -b+SQRT(d)/2*a.

EXERCICE : transformer les formules suivantes en expression numériques Pascal (a,b,x,y sont des variables).
$3x^2+5x-3,5$
...
Attention ici encore au dépassement de capacité ou aux opérations impossibles (1/0).

4.2.) les expressions alphanumériques (chaînes et caractères)

La concaténation de caractères et/ou de chaînes s'écrit comme une addition donnant une chaîne:
s:='MIAS'+'1'+' '+'S1 groupe 7' donnera à s la valeur 'MIAS1 S1 groupe 7'.

N.B. 'gérer l''apostrophe dans une chaîne'

T:=COPY(S,début,longueur) permet de recopier une partie de la chaîne S dans T.

LENGTH(S) retourne la longueur de la chaîne S.

Il est possible d'avoir accès à chaque caractère d'une chaîne par son indice (voir les tableaux).

4.3.) les expressions logiques (booléennes)

Ces expressions servent notamment à gérer les conditionnelles dans les alternatives et les répétitives

Opérateurs booléens (=connecteurs logiques) NOT AND OR par ordre de priorité décroissante (mais aussi XOR).

Une bonne habitude : mettre entre parenthèses les expressions booléennes (voire parenthéser complètement).

Les tests d'égalité (=,<>) sur tous les types, et de comparaison (<,<=,>,>=) sur les types ordonnés donnent des expressions logiques. Il est donc possible d'affecter le résultat d'une telle expression à une variable booléenne.

Exp : test:=((3<6) and ((30 mod 7)=2))

EXERCICE : pour quelles valeurs de A et B les expressions suivantes sont elles vraies.
A AND B OR A
(A OR (NOR B) OR B)
(A OR B) AND (NOT A AND B)

EXERCICE : quelle valeur reçoit la variable booléenne E dans chacun des cas suivants.
E:=('a'='A')=(5<2)
...

5) L'affichage

(* Entrée/sortie = gestion de la saisie et de l'affichage *)

WRITELN et WRITE permettent l'affichage à l'écran. WRITELN, où le suffixe LN signifie line-new, fait passer le curseur d'écran à la ligne après affichage.

Exp1 :
```
x:=35 ; y:=26 ;
WRITELN('x a pour valeur : ', x, ' et y a pour valeur : ', y) ;
WRITELN('x+y=',x+y) ;
```
affichera :
```
x a pour valeur : 35 et y a pour valeur : 26
        x+y=61
```

Exp2 :
```
S:='pas' ; T:='sable' ;
WRITE('S+T= ', S+T) ;
WRITE(' avec S=', S, ' et T=', T) ;
```
affichera : S+T= passable avec S=pas et T=sable

Les choix de présentation à l'écran devront éviter deux extrêmes :
– Concentrer l'affichage et le calcul en un seul affichage intégrant tous les calculs :
```
WRITELN('x=',x, ', y=',y, ', x+y=', x+y, '(x–y)(x–y^2)=', (x–y)*(x-y*y)) ;
```
– Atomiser la programmation en affichages multiples utilisant un grand nombre de variables :
```
WRITELN('x=', x) ;
WRITELN('y=', y) ;
z:=x–y ;
WRITELN('x–y=', z) ;
t:=(x–y)*(x–y*y) ;
WRITELN( '(x–y)(x–y^2)=', t) ;
```

L'affichage peut être formaté :

EXERCICE : Ecrire un programme qui affiche :
C'est la vie de château.
Pourvu que ça dure, pourvu que ça dure! pourvu que ça dure!!

Une solution :
```
program DEUG ;
USES CRT ;
const   A='C''est la vie de château' ;
        B='Pourvu que ça dure' ;
BEGIN
WRITELN(A,',')
WRITELN(B,', ',B,'! ',B,'!!') ;
END.
```

EXERCICES du chapitre :

1) affectations-opérations (M.H.)
"Fortunes" :
A, B et C possèdent respectivement 5, 4 et 3 millions.
Chaque jour :
A, qui a des loisirs, va porter à B la moitié de sa fortune
B, qui ne veut pas être en reste, va porter à C la moitié de sa (nouvelle) fortune
C, enfin, galvanisé par l'exemple, se précipite chez A pour lui donner la moitié de sa (nouvelle) fortune.
a) Ecrire un algorithme qui calcule les fortune de A, B et C le soir venu.
b) Faire une trace de cet algorithme.

2) Ecrire un programme qui, après avoir effacé l'écran, dessine une tête avec des caractères '|' et '–' :

```
 |||||
| – – |
||  |  ||
 |  –  |
  –––––
```

N.B. : on utilisera au besoin les capacités de formatage de l'affichage :
WRITEln(v : i) pour un affichage sur i case de la variable de tout type v, justifié à gauche.
WRITEln(x : i : j) pour un affichage sur i case d'un réel, avec k chiffres après la virgule.

3)
a) Ecrire un programme qui saisit le diamètre d'un cercle puis calcule son aire. On créera une constante Pi=3,1416.
b) Ecrire un programme qui saisit la longueur et la largeur d'un rectangle puis calcule son aire et son périmètre.

4)
a) Ecrire un algorithme qui tire quatre fois de suite au hasard un caractère de l'alphabet (de A à Z, on suppose que les saisies sont correctes) puis concatène ces quatre caractères en un mot de quatre lettres.
b) Transformer cet algorithme en programme Pascal.
Indications : RANDOMIZE initialise le générateur de nombres aléatoires,
RANDOM(n) est une fonction qui retourne un nombre tiré aléatoirement entre 0 et n–1,
CHR(n) retourne le caractère dont le code est n (issu de RANDOM).
c) Comment ferait-on pour vérifier que chaque caractère saisi est bien dans l'alphabet 'A'..'Z' ?
(réponse : test c>='A' et c<='Z')

TP1

Edition (recopie de blocs de texte), compilation d'un programme TurboPascal pour exécution immédiate.
Déclarations.
Gestion des entrées-sorties.
Affectations.
Commentaires.
Instructions :
RANDOMIZE pour initialiser le générateur de nombres aléatoires.
RANDOM(n) pour obtenir aléatoirement un entier entre 0 et n-1.
Familiarisation avec : test d'égalité, d'inégalité et alternative ; blocs de programme emboîtés ; répétitive.
Echange brutal de valeurs de deux variables.

TP2

...