Introduction à la programmation en Pascal

Module L1 - Info

Octobre 2004

Gédéon LÉGAUT

1 Introduction au langage Pascal

On va définir les briques de bases du langage Pascal.

J'ai demandé aux élèves de se faire un petit Mémo pour avoir la syntaxe sous la main en salle de TP et pour leur faire relire le cours.

1.1 Les lettres

A à Z, a à z et _

pas de différences minuscules/majuscules

Pour les noms de variables : ils doivent commencer par une lettre, mais peuvent contenir des chiffres.

1.2 Les chiffres et les nombres

0 à 9

La partie décimale est notée .

1.3 Les symboles réservés

symboles	opération	exemple
+ - * /	les quatres opérations usuelles	
div mod	division euclidienne et son reste	3 div 2, 3 mod 2
= > >= < <= et <> (différent de)	tests	2=j teste si j vaut 2
[ ]	crochets pour les tableaux	$T[i]$ avec i entier
..	intervalle	[0..9] ou [a-e]
'	délimiteur de chaîne de caractères	'toto'
and, not, or et xor (ou exclusif)	opérateurs booléens	(j>2) and (j<10)
;	fin d'instruction ou de déclaration	
:=	affectation	i :=2 ; s :='toto' ; f :=2.3 ;
//...	commentaires sur une seule ligne	//s :=2 ;
{...}	commentaires sur plusieurs lignes	

Les " ne sont pas utilisés en pascal.

Exercices : Que signifie les expressions suivantes ?

j=2

pourquoi pas de ; après "j=2" ?

j :=2 ;

pourquoi un ; après "j :=2" ?

//j :=2 ;

7 div 3

7 mod 3

1.4 Mots réservés

Il y en a beaucoup. On en verra tout au long des TD et TP. Par exemple,
begin end ; Un *begin* se finit toujours par un *end ;*, sauf celui du programme qui se finit par *end.*.
if ... then ... else ... ; –> IF condition THEN instruction ELSE instruction ;
(pas de ; avant ELSE)
do

1.5 Types de variables

variable	type	exemples
entière	**integer**	1 2 235 6325874
réelle	**real**	2.3 1 0.3698 1e-5
chaîne de caractères	**string**	'toto' 't"as rien dit ?'
tableau	**array**	T[4]
booléen	**boolean**	false true

2 Structure d'un programme

```
PROGRAM nom_du_programme ;
USES unité ;
déclaration des constantes et des variables GLOBALES
BEGIN
      corps du programme = suite d'instructions
END.
```

USES : utilisation d'une bibliothèque. Pour avoir la bibliothèque mathématique, USES math ;
Déclaration des constantes et des variables :
CONST nom = valeur ;
VAR nom : type ;
Exemple :

```
PROGRAM Surface_du_cercle ;
CONST pi=3.14159;
VAR   r:integer;
      s:real;
BEGIN
      r:=10;
      s:=pi*r*r;
END.
```

3 Fonction

```
FUNCTION nom (variables : type ) : type renvoyé par la fonction ;
déclaration des constantes et des variables LOCALES
BEGIN
      corps de la fonction
      nom:= ... ; //pour que la fonction renvoie un résultat
END;
```

Exercice : Ecrire la fonction qui renvoie le maximum de deux entiers.

```
FUNCTION max (a,b:integer) : integer;
BEGIN
IF a>b THEN max:=a ELSE max:=b;
END;
```

4 Procédures

Les procédures sont une généralisation de la notion de fonction. Elles ne renvoient aucun résultat, mais peuvent écrire dans des variables qu'on leur spécifie. Il y a donc deux cas de figure, suivant qu'on leur spécifie des variables ou non.

$$\text{PROCEDURE nom} \left(\underbrace{\text{variables : type}}_{\text{variables d'entrée}} \; ; \underbrace{\text{VAR variables : type}}_{\text{variables de sortie}} \right)$$

Cas 1 :

```
PROCEDURE nom (variables : type) ;
déclaration des constantes et des variables LOCALES
BEGIN
      corps de la procédure
END;
```

Cas 2 :

```
PROCEDURE nom (variables : type ; VAR variables : type) ;
déclaration des constantes et des variables LOCALES
BEGIN
      corps de la procédure
END;
```

Pour faire les exercices suivants, on va introduire deux nouvelles fonctions :

- *writeln(s)* affiche la chaîne de caractères "s" à l'écran quand on exécute en ligne de commande le programme pascal. Avec Delphi, on ne fait rien en ligne de commande ; mais on va écrire sur des objets.
  Ex : *writeln('toto') ;* affiche la chaîne de caractères *toto*.
- *format('%X',[var])* renvoie une chaîne de caractères où %X est remplacé le contenu de la variable *var*.
  Ex : si *r :=2 ; p :=2*pi*r ; format('le périmètre d"un cercle de rayon %f vaut %f',[r,p])* renvoie
  le périmètre d'un cercle de rayon 2 vaut 12.56

type de la variable var	X
string	%s
integer	%d
real	%f, %1.5f

Exercice : écrire un programme avec une procédure qui calcule le cosinus et le sinus d'un angle. Variable d'entrée : un angle en degré. Deux variables de sorties pour le sinus et le cosinus.

```
PROGRAM angle;
VAR x,y,z: real;
BEGIN
   PROCEDURE cosin(x:angle; VAR c,s:real);
   BEGIN
      c:=cos(x*pi/180);
      s:=sin(x*pi/180);
   END;

   x:=0.12; // radians
   cosin(x,y,z);
   // y contient le cosinus de l'angle x
   // z contient le sinus de l'angle x
END.
```

Discuter des variables LOCALES / GLOBALES.

5 Quelques boucles

On peut tout faire facilement avec deux boucles alors qu'il en existe beaucoup. On va regarder les boucles *for* et *while*.

5.1 Boucle for

```
FOR i:=1 TO 10 DO instruction ;
```

i est incrémenté de 1 automatiquement

la boucle porte PAR DÉFAUT sur la première instruction après le *do*. Pour avoir une boucle qui exécute plusieurs instructions pour chaque valeurs de i, mettre un *begin ... end ;*

```
FOR i:=1 TO 10 DO BEGIN
    instruction1;
    instruction2;
    ...
    instructionN;
END;
```

Un *begin* doit toujours être fermé par un *end ;*. On ne peut pas modifier la valeur de i dans une boucle *for*.

5.2 Boucle while

```
WHILE condition DO instruction;
```

Pour que la boucle *while* ne soit pas une boucle infinie, il faut que la condition devienne fausse à un moment donné. Il faut donc que la boucle *while* porte sur deux instructions (ce qu'on veut + modifier la condition) : on mettra donc un *begin* fermé par un *end ;*. Exemple

```
i:=1; s:=0;
WHILE i<=10 DO BEGIN
    i:=i+1;
    s:=s+1;
END;
```

Que fait ce programme ?

i	s
1	0
2	0+1
3	1+1
...	...
n	n-1

Pour quitter une boucle sans être arrivé à la condition finale, on utilise la fonction *break*.

5.3 Exercices

1. Calculer la somme des N premiers entiers avec une boucle *for* et une boucle *while* :

$$s = \sum_{i=1}^{N} i = \frac{N(N+1)}{2}$$

```
PROGRAM somme_entier1;
VAR s,i,n:integer;
BEGIN
    n:=10;
    s:=0; // initialisation
    FOR i:=1 TO n DO s:=s+i;
END.
```

```
PROGRAM somme_entier2;
VAR s,i,n:integer;
BEGIN
    n:=10;
    s:=0; // initialisation
    i:=1;
    WHILE i<=n DO BEGIN
       s:=s+i;
       i:=i+1;
    END;
END.
```

2. Soit les fonctions $g(x) = (1-x)^2$ et $h(x) = x^2 - 1$. Calculer

$$s = \sum_{i=1}^{N} g(x_i) * h(x_i) \qquad x_i = i * dx \qquad dx = 3$$

```
PROGRAM somme_fonctions;
VAR i,n:integer;
    dx,s:real;
BEGIN
    FUNCTION g (x:real):real;
    BEGIN
       g:=(1-x)*(1-x);
    END;

    FUNCTION h (x:real):real;
    BEGIN
       h:=x*x-1;
    END;

    n:=10;
    dx=3;
    s:=0; // initialisation
    FOR i:=1 TO n DO s:=s+g(i*dx)*h(i*dx);
END.
```

6 Les tableaux

Déclaration d'un tableau :

```
CONST N=100;
VAR T : array [0..100] of real;
    T1 : array [0..N,0..N+10] of integer ;
```

Utiliser ou écrire dans un tableau :

```
T[i]:=i;
k:=T1[i][j]; // colonne i, ligne j
```

Exercice : Soit la fonction $f(x) = \sin(x)\mathrm{e}^{-(x-6)^2}$. On recherche le maximum de cette fonction dans l'intervalle [0..100]. Soit $n = 200$, $x_i = i * 100/n$, $T[i] = f(x_i)$.

```
PROGRAM maximum;
CONST n=200;
VAR i:integer;
    x,max,xmax:real;
    T:array [0..n] of real;
BEGIN
    FUNCTION f (x:real):real;
    BEGIN
        f:=sin(x)*exp(-(x-6)*(x-6));
    END;

    // ecriture des valeurs dans le tableau T
    FOR i:=0 TO n DO T[i]:=f(i*100/n);

    // recherche du maximum global
    max:=T[0]; //pourquoi pas max=0 ?
    FOR i:=0 TO n DO
    if T[i]>max THEN BEGIN
        max:=T[i];
        xmax:=i*dx/n;
    END;
END.
```

7 Traitement des chaînes de caractères

Une chaîne de catactères est un tableau de caractères. s :='bonjour';

s[1] → b
s[2] → o
s[5] → o

s :='toto'; s[3] :='x'; s contient alors *toxo*.
Concaténation : s1 :='toto'; et s2 :='tata';

```
s1+s2  renvoie  'tototata'
s2+s1  renvoie  'tatatoto'
```

7.1 Manipulation de chaîne de caractères

Comme une chaîne de catactères est un tableau de caractères, tester si deux chaînes sont identiques ne se fait pas par *s1=s2*. On peut tester si deux caractères sont identiques (*i :=1 ; if s1[i]=s2[i] then ...*). Pour manipuler de façon générale les chaînes de caractères, il faut utiliser des fonctions telles *StrComp* ... rechercher dans l'aide

1. *pos('motif',s)* :

   pos('.','10.2') → 3
   pos('o','tota') → 2
   pos(',','10.2') → 0

2. *length(s)* renvoie la longueur de la chaîne de caractères s.
3. *copy(s,n,m)* renvoie une chaîne de caractères

   copy('toto',1,2) → 'to'
   copy('bonjour',4,length('bonjour')) → 'jour'
   copy('tram',1,length('tram')) → tram
   copy('tram',1,12) → tram
   s :='tram'; copy(s,pos(',',s)+1,pos('m',s)) → 'tram'
   s :='tram'; copy(s,1,2) → 'tr'
   s :='tram'; copy('s',1,2) → 's'

7.2 Conversion de type

Tout ce qu'on rentre dans un programme avec un clavier ou un fichier est compris comme étant une chaîne de caractères par l'ordinateur. Il existe donc des fonctions de conversion de type.

1. *StrToInt*

```
VAR s:string;
    n:integer;

s:='10';
n:=StrToInt(s);
```

2. *Format('%X',[var])*

```
VAR s,s1:string;
    n:integer;
    r:real;

s1:='Toto';
n:=10;
r:=120.35;
s:=Format('\%s a recu pour ses \%d ans \%f euros',[s1,n,r]);
```

Exercice : écrire une fonction de conversion de type *StrToReal*. Vos données sont de la forme '12.13' ou '12,563'.

```
// fonction qui transforme une chaine de caractères en réel
FUNCTION StrToReal (s:string):real;
VAR s1,s2:string;
BEGIN
   // si virgule, la changer en point
   IF pos(',',s)<>0 THEN s[pos(',',s)]:='.';
   // si nombre entier -> cas particluier
   IF pos('.',s)=0 THEN strtoreal:=strtoint(s)
   ELSE BEGIN
      s1:=copy(s,1,pos('.',s)-1);
      s2:=copy(s,pos('.',s)+1,length(s));
      // rajouter math tout en haut du prog dans la rubrique ``Uses''
      strtoreal:=strtoint(s1)+strtoint(s2)/power(10,length(s2));
   END;
END;
```

8 Exercices

1. Comment tester si un nombre est pair ? impair ?
2. Comment enlever tous les espaces dans une chaîne de caractères ?
3. Comment inverser une chaîne de caractères ?
4. Comment tester si un entier est un nombre premier ?
5. Comment faire une calculette horaire qui calcule 10h53min + 128h38min ?