Informatique

ECE2

Année 2011-2012

1. L'essentiel du langage Pascal

1.1 Structure d'un programme

```
PROGRAM nom;    {Commentaires}
CONST constante=valeur ;
VAR variable :type ;
BEGIN
        …
        …
END.
```

1.2 Types de variable

integer : les entiers entre –32768 et 32767
longint : les entiers entre ≈ -2.10^9 et 2.10^9
real : les réels en écriture scientifique (attention, ils sont stockés en valeur approchée !)
boolean : booléen, c'est-à-dire true ou false
char : caractère, entre apostrophes : 'A', …
array : tableau

1.3 Entrées – Sorties

writeln('Texte'); Affiche le texte entre apostrophe.
writeln(variable); Affiche la valeur de la variable
Si plusieurs paramètres, séparer par des virgules

readln(variable); Stocke dans la variable la valeur tapée au clavier.
Jamais de texte après readln !

Exemples :
1) x est une variable. Afficher la valeur de x :

2) x est une variable. Demander la valeur de x à l'utilisateur.

1.4 Stockage et calculs

variable := valeur; stocke la valeur de droite dans la variable de gauche.

Opérateurs mathématiques : +, –, *, / (pour les réels)
+, –, *, DIV, MOD (pour les entiers)

Fonctions prédéfinies : ln, exp, trunc (partie entière), abs, sqr (carré), sqrt (racine carrée)
Il n'y a pas de puissance prédéfinie en Pascal !

Compteur :

Initialisation : i :=0 ;
Relation de récurrence : i := i + 1 ;

1.5 L'instruction IF

1ère syntaxe :

```
IF condition THEN instruction ;
```

2ème syntaxe :

```
IF condition THEN instruction1    (pas de point-virgule avant ELSE)
                 ELSE instruction2;
```

Si plusieurs instructions après THEN, ne pas oublier : BEGIN … END ;

Exemple :
n est un entier déjà calculé. Afficher à l'écran s'il est pair ou impair.

2. Boucles, sommes et suites

2.1 FOR, REPEAT, WHILE

Pour effectuer plusieurs fois la même instruction, on effectue une boucle.

_ soit on connaît par avance le nombre d'itérations : on utilise FOR

_ soit on ne connaît pas par avance le nombre d'itérations : on utilise REPEAT ou WHILE

Syntaxes :

```
FOR compteur :=debut TO fin DO instruction;

FOR compteur := debut TO fin DO BEGIN        {si plusieurs instructions dans la boucle}
                                    instruction1;
                                    instruction2;
                                     …
                                  END;
```

Attention, compteur doit être une variable **entière.**

```
REPEAT instruction1 ;
          instruction2 ;
          ….
UNTIL condition ;
```

La condition est testée après les instructions. La boucle s'arrête quand la condition est vraie.

```
WHILE condition DO instruction ;

WHILE condition DO BEGIN      {si plusieurs instructions dans la boucle}
                              instruction1 ;
                              instruction2;
                              …
                            END;
```

La condition est testée avant les instructions. La boucle s'arrête quand la condition est fausse.

Attention, dans REPEAT et WHILE, il n'y a pas de compteur. Si vous avez besoin d'un compteur (pour compter le nombre d'itérations effectuées), c'est à vous de le rajouter. (Voir paragraphe 1.4).

2.2 Sommes et produits

Si on veut calculer $S = \sum u_k$

Initialisation : S :=0 ; Formule de récurrence : S := S + uk ;

Si on veut calculer $P = \prod u_k$

Initialisation : P :=1 ; Formule de récurrence : P := P * uk ;

Exemples :

1) n étant connu, calculer $S = \sum_{i=1}^{n} \frac{1}{i}$

2) Calculer le premier entier n tel que $2^n > 10^6$

2.3 Suites récurrentes

2.3.1 Suites récurrentes d'ordre 1 : $u_{n+1} = f(u_n)$

On considère une suite $(u_n)_{n \in \mathbb{N}}$ définie par : $\begin{cases} u_0 \in \mathbb{R} \\ \forall\, n \in \mathbb{N},\ u_{n+1} = f(u_n) \end{cases}$

On veut calculer les valeurs successives de (u_n), stockées dans la variable u :

Initialisation : u := u0 ; Formule de récurrence : u := f(u) ;

Exemple : Soit $(u_n)_{n \in \mathbb{N}}$ la suite définie par : $\begin{cases} u_0 = 2 \\ \forall\, n \in \mathbb{N},\ u_{n+1} = \frac{1}{2}\left(u_n + \frac{2}{u_n}\right) \end{cases}$

Déterminer le premier entier n tel que $\left|u_n - \sqrt{2}\right| \leq 10^{-3}$

2.3.2 Suites récurrentes d'ordre 2 : $u_{n+2} = f(u_n, u_{n+1})$

On considère une suite $(u_n)_{n \in \mathbb{N}}$ définie par : $\begin{cases} u_0 \in \mathbb{R} \\ u_1 \in \mathbb{R} \\ \forall\, n \in \mathbb{N},\ u_{n+2} = f(u_n, u_{n+1}) \end{cases}$

Pour calculer les valeurs de (u_n), il faut conserver deux valeurs successives, stockées dans u et v.

Initialisations : u := u0 ; v := u1 ; Relations de récurrence : w:=f(u, v); (faire un schéma pour s'y retrouver) u :=v ; v :=w ;

Remarque : v commençant à u_1, il faut n-1 itérations pour que v contienne u_n.

Exemple : Soit $(u_n)_{n \in \mathbb{N}}$ la suite définie par : $\begin{cases} u_0 = 2 \\ u_1 = -1 \\ \forall n \in \mathbb{N},\ u_{n+2} = u_{n+1} + 3u_n \end{cases}$. Déterminer u_{10}

3. Fonctions, fonctions récursives

3.1 Syntaxe d'une fonction

Dans la partie déclarative du programme :

```
function nom(var1 : type1 ;var2 : type2…) : typerésultat ;
var {variables locales}
begin
   …
   …
   nom :=…. ;
end ;
```

_ ni writeln ni readln dans une fonction.
_ nom n'est affecté qu'une fois, à la fin. Utiliser une variable locale pour des calculs intermédiaires.

Exemples :
1) Ecrire une fonction f qui à un réel x associe $f(x) = 3x - \ln(x^2 + 1)$

2) Ecrire une fonction factorielle, qui à un entier n associe le nombre n !

Appel d'une fonction : Dans une fonction définie après, ou dans le programme principal :

```
nom(val1, val2, …)
```

Il s'agit d'un nombre de type typerésultat

3.2 Fonctions récursives

Une fonction récursive est une fonction qui s'appelle elle-même.
Elle est utile notamment pour des expressions définies par leur premier terme et une relation de récurrence.

Attention : Il faut toujours une initialisation (ou point d'appui) et il faut toujours s'assurer qu'avec un nombre d'appels fini de cette fonction, on arrive à ce point d'appui.

En général, il s'agit de fonction sur $\mathbb{N}$ où l'initialisation se fait pour n = 0, et où le calcul de f(n) nécessite la valeur de f(n-1). Dans ce cas, elle s'écrit sous la forme :

```
IF n=0 THEN f :=…
        ELSE f :=….  f(n-1)… ;
```

Exemples :

1) Soit (u_n) la suite définie par : $\begin{cases} u_0 = 1 \\ \forall\, n \in \mathbb{N},\ u_{n+1} = 3u_n^{\,2} - 2 \end{cases}$
Ecrire une fonction récursive u, qui à un entier n associe la valeur de u_n.

2) Ecrire une fonction récursive puissance, qui à un entier n associe 2^n

Remarque : Un processus qui utilise une boucle FOR est appelé un processus itératif.
Un processus récursif est souvent plus simple à écrire, mais n'est pas toujours le plus efficace : les appels successifs à la fonction encombrent la mémoire, on ne contrôle que difficilement le nombre d'opérations, et celui-ci peut augmenter très rapidement.

Exemple : Suite de Fibonacci : (u_n) définie par : $\begin{cases} u_0 = 0 \\ u_1 = 1 \\ \forall\, n \in \mathbb{N},\ u_{n+2} = u_{n+1} + u_n \end{cases}$ Calculer u_5

_ de manière itérative : calculs de u_2, u_3, u_4, u_5 : 4 opérations
_ de manière récursive : $u_5 =$ u_4 + u_3

4. Equations, intégrales

4.1 Résolution d'équation par dichotomie

Cadre :
Soit f une fonction continue et monotone sur un intervalle [a ; b].
Par le théorème de la bijection, on a montré que l'équation f(x) = 0 admet une unique solution α sur [a ;b].
Soit $\varepsilon > 0$. On cherche à déterminer une valeur approchée de α à ε près.

Principe :
On construit deux suites a_n et b_n telles que $\forall\, n \in \mathbb{N}$, $\alpha \in [a_n ; b_n]$.
_ Initialisations : $a_0 = a$ et $b_0 = b$
_ Récurrence : Quand a_n et b_n sont calculés :
on calcule le milieu $c_n = \frac{a_n + b_n}{2}$ de l'intervalle $[a_n ; b_n]$.
(Si f est croissante) Si $f(c_n) > 0$, alors on pose $\begin{cases} a_{n+1} = \\ b_{n+1} = \end{cases}$
sinon on pose $\begin{cases} a_{n+1} = \\ b_{n+1} = \end{cases}$
(inverser si f est décroissante)
Ainsi $\alpha \in [a_{n+1}, b_{n+1}]$.
_ On s'arrête quand $b_n - a_n \leq \varepsilon$.
a_n et b_n sont alors des valeurs approchées de α à ε près.

Remarques :
_ $b_n - a_n$ est une suite géométrique de raison $\frac{1}{2}$, donc $\forall\, n \in \mathbb{N}$, $b_n - a_n = \frac{b_0 - a_0}{2^n}$
_ $a_{n+1} = a_n$ ou $b_{n+1} = b_n$ n'a pas besoin de s'écrire en informatique, puisque la valeur reste inchangée.

Algorithme à retenir :

```
program dichotomie;
var a, b, c : real;

begin
  a:=...  ; b:=….;  {mettre les bornes de l'intervalle}
  repeat
      c := (a + b)/2;
      {si f es croissante}      if f(c) > 0 then b:=c      {remplacer f par son expression si
                                else a:=c;                  vous n'avez pas défini de fonction}
      {si f est décroissante}   if f(c) > 0 then a:=c
                                else b:=c;
  until b – a <  epsilon;
  writeln('Une valeur approchée de alpha est ', a);  {ou b à la place de a}
  readln;
end.
```

Remarque :
On peut regrouper les cas où f est croissante et où f est décroissante de la manière suivante :
$\alpha \in [a_n; c_n]$ si et seulement si f change de signe entre a_n et c_n, c'est-à-dire si et seulement si $f(a_n) \times f(c_n) \leq 0$.
On peut donc écrire également :

```
if f(a) × f(c) < 0  then b := c          {Valable pour f croissante ou décroissante}
                    else  a := c;
```

Exemple
Soit f(x) = xln(x) – 1 sur]0 ; + ∞[. On peut montrer que f est croissante sur [1 ;2], et que l'équation f(x) = 0 admet une unique solution α sur [1 ;2].
Ecrire un programme qui détermine une valeur approchée de α à 10^{-4} près.

4.2 Valeur approchée d'une intégrale (méthode des rectangles)

Cadre : Soit f une fonction continue sur un intervalle [a;b]. On cherche à approcher numériquement l'intégrale $I = \int_a^b f(x)dx$

Principe : On découpe l'intervalle [a;b] en n intervalles $[x_k, x_{k+1}]$ de longueur $\frac{b-a}{n}$,

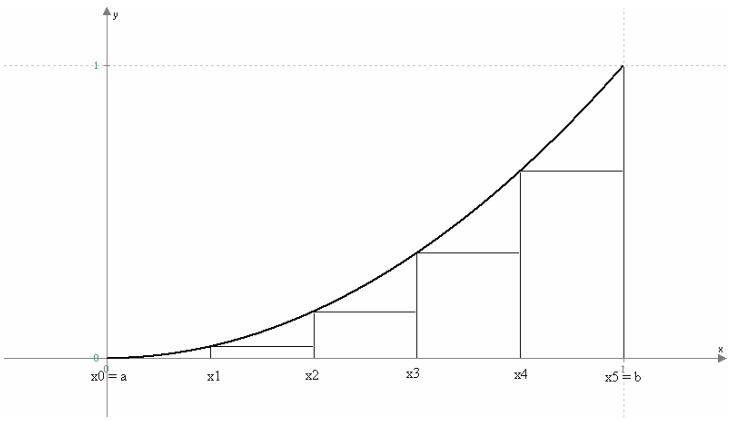
avec $x_k = a + k\times\frac{b-a}{n}$ pour $k \in \{0, \ldots, n\}$.

Considérons une fonction f à valeurs positives. I correspond à l'aire sous la courbe de f. Graphiquement, on approche cette aire par la somme des aires des rectangles délimités par les (x_k (à gauche ou à droite). Soit S_n cette somme.

En approchant à gauche, on a donc $S_n = \sum_{k=0}^{n-1} \frac{b-a}{n} \times f(x_k) = \frac{b-a}{n} \sum_{k=0}^{n-1} f(x_k)$

En approchant à droite, on a donc $S_n = \sum_{k=1}^{n} \frac{b-a}{n} \times f(x_k) = \frac{b-a}{n} \sum_{k=1}^{n} f(x_k)$

Exemple avec n = 5 (à gauche)



Propriété : Si f est une fonction continue sur [a;b], alors S_n converge vers $I = \int_a^b f(x)dx$.

Exemple : Ecrire un programme qui détermine une valeur approchée de $I = \int_1^2 \frac{e^x}{x}dx$ par la méthode des rectangles avec un pas de 1/1000.

5. Tableaux et gestion de listes

5.1 Définition et opérations sur les tableaux

Pour définir des tableaux, il faut d'abord définir un nouveau type qui détermine les tailles des tableaux.

Tableau à une dimension :

```
TYPE tableau=ARRAY[debut..fin] of typeélément;
```

Tableau à deux dimensions :

```
TYPE tableau=ARRAY[debut1..fin1,debut2..fin2] of typeélément;
```

Déclaration d'une variable de type tableau : `VAR T : tableau;`

Le coefficient n°i de T est `T[i]` (si une dimension),
Le coefficient d'indices i et j est `T[i,j]` (si deux dimensions).

Opérations sur les tableaux :
A l'exception du transfert simple d'un tableau (`U:=T;`), toutes les opérations sont à effectuées élément par élément, donc à l'aide d'une boucle `FOR` (ou de deux boucles `FOR` imbriquées si le tableau est à deux dimensions).

Exemples :
On définit `TYPE vecteur=ARRAY[1..10] of real;` représentant les vecteurs de $M_{10,1}(\mathbb{R})$.
`VAR T, U, V: vecteur;`
Demandez les valeurs de T et U à l'utilisateur, puis calculer V = T – 3U

Remarque :
Un polynôme $P(X) = a_0 + a_1X + \ldots + a_nX^n$ peut être représenté par un tableau (`TYPE polynome=ARRAY[0..n] of real;`) qui contient les coefficients $a_0, a_1, \ldots, a_n$.

5.2 Procédures

Une fonction ne peut pas avoir comme résultat un tableau.
Dans ce cas, il faut utiliser une procédure.

Syntaxe :
Dans la partie déclarative du programme :

```
PROCEDURE nom(paramètres);
VAR
        variables locales
BEGIN
        ….
END;
```

Les paramètres sont de deux types :
_ passage par valeur : le paramètre n'est pas modifié dans la procédure : on pourra appeler la procédure avec une variable, mais aussi un nombre ou une expression arithmétique (du bon type évidemment)
_ passage par variable : le paramètre est modifié dans la procédure : on ne pourra appeler la procédure qu'avec une variable. Dans ce cas, le paramètre est précédé de VAR.

Dans la partie exécutive du programme : nom(paramètres); est une instruction.

Exemple :
On considère des vecteurs de $M_3(\mathbb{R})$. Dans un programme, écrire une procédure Addition qui, à deux vecteurs X et Y, associe le vecteur Z = X + Y.
Tester cette procédure avec $X = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$ et $Y = \begin{pmatrix} -1 \\ 5 \\ 4 \end{pmatrix}$

```
program test;
type vecteur = array[1..3] of real;

procedure Addition(                                    );
var
begin


end;

var

begin
  x[1]:=1;x[2]:=2;x[3]:=3;
  y[1]:=-1;y[2]:=5;y[3]:=4;


  writeln('La somme vaut ', z[1], z[2], z[3]);
  readln;
end.
```

6. Simulation de variables aléatoires

6.1 Généralités

Rappel : Un programme simulant un phénomène aléatoire doit toujours commencer par l'instruction : Randomize

random s'utilise de deux façons différentes :

- **random(n)** (où n est un entier naturel) est un **entier** compris entre 0 et n – 1
  on peut considérer qu'il suit la **loi uniforme sur {0, … , n – 1}**

- **random** est un **réel** compris entre 0 et 1
  on peut considérer qu'il suit la **loi uniforme sur [0;1[**

On peut considérer que des appels successifs à la fonction random donnent des résultats **indépendants.** Si vous voulez utiliser un résultat, stockez-le immédiatement dans une autre variable (un nouvel appel donnerait une valeur différente).

Remarque : Si $X \longrightarrow U([0;1[)$, alors pour tout $p \in [0;1]$, $P(X < p) = p$ et $P(X \geq p) = 1 - p$.

A retenir : **Si $p \in [0;1]$, (random<p) est un événement de probabilité p**

Simuler la loi d'une V.A.R. X, c'est écrire un programme contenant une variable x dont les valeurs sont aléatoires, et qui suit la loi de X.

Exemples :
1) On lance un dé équilibré. Soit X le numéro de la face obtenue.
Simuler l'expérience aléatoire.

2) On lance une pièce truquée, dont la probabilité de faire face est 1/3.
On définit une variable lancer de type char, qui contient 'F' si on fait face et 'P' si on fait pile.
Simuler l'expérience aléatoire.

6.2 Simulation des lois discrètes usuelles

6.2.1 Loi uniforme sur {1, …, n}, sur {n_1, …, n_2}

Soit $n \in \mathbb{N}$. random(n) $\longrightarrow$ U({0, …., n – 1} donc random(n) + 1 $\longrightarrow$ U({1, …, n}).

Si X $\longrightarrow$ U({1, …, n})
x : = random(n) + 1;

Attention : Ne pas confondre :

- random(n) + 1 : loi uniforme sur {1, ..., n}, probabilité 1/n
- random(n + 1) : loi uniforme sur {0, …, n}, probabilité 1/(n + 1)

Exemple :
Soit X une V.A.R. qui suit la loi uniforme sur [[1 ; 10]].
Simulation de X :

Soient n_1 et n_2 deux entiers naturels avec $n_1 < n_2$.
Soit X $\longrightarrow$ U({n_1, …, n_2}). Alors Y = X – n_1 suit la loi uniforme sur U({0, …., $n_2 – n_1$})
Donc X = n_1 + Y, avec Y $\longrightarrow$ U({0, …., $n_2 – n_1$})

Si X $\longrightarrow$ U([[n_1, n_2]])
x:=n1 + random(n2 – n1 + 1);

Exemple : Soit X un nombre choisi au hasard entre 10 et 25 (inclus).
Simulation de X :

Remarque : Entre 10 et 25 (inclus), il y a 16 nombres entiers.
random(16) $\longrightarrow$ U({0, …,15}) donc random(16) + 10 $\longrightarrow$ U({10, ..., 25})

6.2.2 Loi de Bernoulli

Soit $p \in$]0;1[et X $\longrightarrow$ B(1,p). (c'est-à-dire : $\begin{cases} X(\Omega) = \{0,1\} \\ P(X = 0) = 1 - p \\ P(X = 1) = p \end{cases}$

Si X $\longrightarrow$ B(1,p) if random< p then x := 1
else x := 0;

Exemple : Une urne contient deux boules bleues et une boule verte. On tire une boule de l'urne. Soit X la V.A.R. qui est égale à 1 si la boule est bleue et à 0 sinon.

Donc X $\longrightarrow$ Simulation de la loi de X :

6.2.3 Loi binomiale

Rappel : On considère une épreuve de Bernoulli de succès de probabilité p. On répète n fois cette épreuve de manière indépendante. Soit X le nombre de succès. Alors X ⟶ B(n,p).

```
Si X ⟶ B(n,p)
                x := 0;
                for i :=1 to n do if random<p then x := x + 1;
```

Exemple : On considère l'urne précédente. On tire 10 boules en les remettant à chaque fois. Soit X le nombre de boules bleues tirées.

X ⟶

Simulation de la loi de X :

6.2.4 Loi géométrique

Rappel : On considère une épreuve de succès de probabilité p.
On répète cette épreuve de manière indépendante jusqu'à obtenir un succès.
On note X le nombre d'essais nécessaires. Alors X ⟶ G(p).

```
Si X ⟶ G(p) :    x := 0;
                 repeat x := x + 1; until random<p;
```

Exemple : On considère l'urne précédente. On tire une boule en la remettant, jusqu'à obtenir une boule bleue. Soit X le nombre de tirages nécessaires.

Donc X ⟶

Simulation de la loi de X :

6.3 Simulation des lois continues usuelles

6.3.1. Loi uniforme sur [0;1[

Rappel : random suit la loi uniforme sur [0;1[.

si X $\longrightarrow$ U([0;1[) **x := random;**

6.3.2 Loi uniforme sur [a;b[(a<b)

Soit X une V.A.R telle que X $\longrightarrow$ U([a;b[). Posons $Y = \frac{X - a}{b - a}$.

Alors Y $\longrightarrow$ U([0;1[). (valeurs prises entre 0 et 1, de manière uniforme)

Or $Y = \frac{X - a}{b - a} \Leftrightarrow Y(b - a) = X - a \Leftrightarrow X = a + (b - a)Y$

Simulation :

Si X $\longrightarrow$ U([a;b]) **x := a + (b – a)*random;**

Exemple : Soit X qui suit la loi continue uniforme sur [2;5].
Simulation de X :

Remarque : [2;5] est d'amplitude 3.
random $\longrightarrow$ U([0;1]) donc $3 \times$ random $\longrightarrow$ U([0;3]) et $3 \times$ random + 2 $\longrightarrow$ U([2;5]).

6.3.3 Simulation de la loi exponentielle

Propriété **(A redémontrer à chaque fois)**

Soit $a > 0$.

Si U est V.A.R. qui suit la loi $\mathcal{U}(]0;1[)$, alors $X = -\frac{1}{a}\ln(1 - U)$ suit la loi exponentielle $\mathcal{E}(a)$.

Démonstration :

Simulation informatique (A redémontrer aussi) :

Si $X \longrightarrow \mathcal{E}(a)$, on peut simuler X par :

```
x := - ln(1 – random)/a;
```

Exemple :
Soit X une V.A.R. qui suit la loi exponentielle $\mathcal{E}(2)$.
Simulation de X :