(3) Structures itératives

Un algorithme plus élaboré : le jeu de Piquet (voir le programme dans le chapitre 2)

Donnée : deux joueurs
Résultat : le joueur gagnant
Algorithme :
Afficher la règle du jeu : <<les deux joueurs ajoutent alternativement un nombre entier compris entre 1 et 10 à un compteur dont la valeur initiale est 0. Le premier qui atteint 100 a gagné. Celui qui dépasse le premier 100 a perdu.>>
Saisir les noms des joueurs ;
Initialiser le compteur à 0 ;
Le premier joueur a la main ;
Tant que le compteur est inférieur à 100 faire
    | le joueur qui a la main saisit une valeur entre 1 et 10 ;
    | incrémenter le compteur de cette valeur ;
    | afficher la valeur du compteur ;
    | la main passe à l'autre joueur ;
Si le compteur a pour valeur 100
    | alors le joueur qui n'a pas la main a gagné
    | sinon le joueur qui a la main a gagné ;
Afficher le joueur gagnant.

Une exécution de ce programme : 5, 10, 3 (etc.). Notion de trace d'un programme.

Commentaires : alternative, répétitive indéterminée, notion de bloc, mise en page.

Commentaires : Notion de bloc "naturel" ou structuré (dans ce dernier cas, en Pascal : begin ... end comme délimiteurs). Importance ergonomique de la pagination (indentation). Du bon usage des commentaires (fréquence, taille, pertinence et cohérence).

1.) Structures séquentielles

Ce qu'on a vu au chapitre précédent : les instructions se suivent "de gauche à droite" et "de haut en bas". La pagination a un rôle très important pour mettre en évidence certains aspects : affichage d'information précédent une saisie, regroupement de blocs de programme autour d'un commentaire, etc. Cependant cette mise en page est transparente pour le compilateur qui saisit le programme <<au kilomètre>>.

2.) Structures alternatives

On a vu dès le chapitre précédent que le programmeur a besoin de réaliser des tests sur les valeurs qu'il rentre (voir l'exercice 3 du ch2), tests qui orienteront la suite du programme. D'où la présence de l'alternative :

2.1.) alternative "complète"

Ecriture algorithmique : **SI** *condition* **ALORS** *premièreconséquence* **SINON** *deuxièmeconséquence*
Ecriture en Pascal : **IF** *condition* **THEN** *premièreconséquence* **ELSE** *deuxièmconséquence* ;

Commentaires : En Pascal : pas de point-virgule avant ELSE, un point-virgule en fin d'alternative.

Exemple : ch(2)/exc3/c/ : IF ((c<'A') OR (c>'Z')) THEN WRITELN('Ca va pas la tête!') ELSE ...

Exercice : Construire un programme qui calcule le prix TTC d'un achat à partir du prix HT, avec deux possibilités : TVA à 5,5% pour la nourriture, à 18,6% pour le reste. (les renseignements sont fournis par l'utilisateur).

Exercice : Construire un programme qui saisit deux réels a et b puis qui les affiche par ordre croissant.

Réponse1 :

```
programme PremierTri ;
Var a, b : real ;
BEGIN
write('a? ') ; readln(a) ;
write('b? ') ; readln(b) ;
IF a>b
   THEN writeln('a=',a,' b=', b)
   ELSE writeln('b=', b,' a=', a) ;
END.
```

Les deux nombres sont affichés sur une même ligne

Réponse2 :

```
programme PremierTriBis ;
Var a, b : real ;
BEGIN
write('a? ') ; readln(a) ;
write('b? ') ; readln(b) ;
IF a>b
   THEN
         begin
         writeln('a=',a) ;
         writeln(' b=', b)    (* pas de point-virgule *)
         end ;
   ELSE begin
         writeln('b=', b) ;
         writeln(' a=', a) ;
         end ;                (* avec point-virgule *)
END.
```

Les deux nombres sont affichés sur deux lignes successives

Commentaire : Les conséquences du test contiennent ici plusieurs instructions qui doivent être exécutées en bloc (séquentiel ici) ; les mots BEGIN et END servent à indiquer le regroupement de ces instruction dans un bloc.

2.2.) alternative écourtée

Dans ce cas, ELSE est absent :

Ecriture algorithmique : **SI** *condition* **ALORS** *conséquence* ;
Ecriture en Pascal : **IF** *condition* **THEN** *conséquence* ;

Commentaire : on écrit un point-virgule après la conséquence, car l'instruction est terminée.

N.B. : cela revient à écrire : SI condition ALORS conséquence SINON ne rien faire.

Exemple :

1) Programme calculant l'âge à partir de l'année en cours et de l'année de naissance et précisant si la personne est majeure. (voir exercice AGE1 dans le chapitre 2)
2) Même question mais en ayant les dates JJ,MM,AA. (voir TP)

```
Programm AGE2 ;
Var N, C : integer ;
BEGIN
...
IF (C-C)>=18 THEN writeln('et vous êtes majeur') ;
END.
```

2.3.) Evaluation de la condition

La condition peut être complexe. On a souvent la possibilité de décomposer le test s'il est compliqué, mais ce n'est pas forcément avantageux :

Exemple : Programme mettant au pluriel le nom saisi s'il appartient à une liste donnée.

Solution 1 : Avec une expression élaborée, mais...

```
Programm Pluriel1 ;
Var nom : string ;
BEGIN
readln(nom) ;
IF ((nom='joujou') OR  (nom='pou') OR (nom='chou')
        THEN writelen('Le pluriel de ce nom est : ',nom+'x')
        ELSE writeln('Je ne connais pas ce nom') ;
END.
```

Solution 2 : Avec des expressions simples, mais...

```
Programm Pluriel2 ;
Var nom : string ;
BEGIN
readln(nom) ;
IF (nom='joujou') THEN writeln('Le pluriel de ce nom est : joujoux')
ELSE IF (mot='pou') THEN writeln('Le pluriel de ce nom est : poux')
ELSE if (nom='chou') THEN writeln('Le pluriel de ce nom est : choux')
ELSE writeln('Je ne connais pas ce nom') ;
END.
```

Exercice : Construire un programme qui détermine si un entier naturel saisi est divisible à la fois par 2, 3 et 5.

Exercice : une entreprise donne à ses ouvriers un salaire mensuel de base de 8000frs auquel s'ajoute une prime de 100frs par année d'ancienneté à partir de la troisième. A Noël, l'entreprise verse à chaque employé 300frs par enfant à charge, plus 500frs pour les familles de plus de 10 enfants dont un des conjoints reste au foyers. Donner le programme qui saisit les caractéristiques d'un employé et calcule son revenu annuel.

EXERCICES

1) On donne le programme Pascal suivant (voir TP3) :

```
program (*bissextile*) ;
var     E : integer ;
        q,c,m : boolean ;
        S : boolean ;
BEGIN
write('donnez-moi une année ') ; readln(E) ;
if ((E mod 4)=0) then q:=true else q:=false ;
if ((E mod 100)=0) then c:=true else c:=false ;
if ((E mod 400)=0) then m:=true else m:= false ;
S:=(q and (not c or m)) ;
writeln(..., S) ;
END.
```

a) Que donne le programme pour E=2000, E=1999, E=1996, E=1900. En conclusion, que fait le programme ? Compléter la dernière instruction en remplaçant les pointillés par un texte adéquat.
b) Pour E=1996, construire la trace du programme : indiquer les affichages et les saisies et les valeurs prises par les différentes variables du programme à mesure de son exécution.
c) Simplifier ce programme en regroupant les trois tests en un seul. On peut simplifier encore en supprimant toutes les alternatives, comment ?
d) Ecrire l'algorithme correspondant à cette simplification ultime (titre, donnée, résultat, procédure)puis le programme Pascal.

2) Ecrire un programme qui résout une équation $ax^2+bx+c=0$ avec a,b,c paramètre ENTIERS donnés.

4)
a) Ecrire un programme qui transforme un temps HMS en temps décimal (et inversement).
b) Un train part [à HMS1] de Ville1 pour arriver [à HMS2] à Ville2 [distante de D km], [en circulant à V km/h]. Construire un programme qui calcule une des quatre valeurs en fonction des trois autres.
N.B. : pour la saisie, une valeur négative indiquera la valeur manquante (une seule !).

5)
Ecrire un programme qui vérifie si un triplet d'entiers donnés est pythagoricien, c'est à dire qui vérifie si les trois nombres mesurent les côtés d'un triangle rectangle.

3)
a) Ecrire un programme qui "rend la monnaie" : le client achète avec un billet de 100 francs, un objet, le marchand rend la monnaie en pièces de 10 et 1 francs (on suppose qu'il en a autant qu'il veut).
b) Quels sont les changements dans le programme si le marchand rend la monnaie avec des pièces de 20, 10, 5, 2 et 1fr.
c) Votre programme rend-il bien un minimum de pièces au client ?
d) On suppose maintenant que le marchant a des quantités limitées de chaque type de pièces. Quelle contrainte supplémentaire ce fait introduit-il ?

3.) L'alternative étendue (=choix multiple =conditionnelle multiple) SELON (CASE)

(* remplacer des ifs imbriqués par des CASE *)

Forme algorithmique : SELON *variable* VALEURS [*valeur*[,*valeur*]* ALORS *bloc*]$^+$ [SINON *alternative*].
En Pascal : CASE *variable* OF *valeur*[, *valeur*]* **:** *bloc* [...]* [ELSE *alternative*] END;

La variable sélectrice doit (en Turbo Pascal) être un scalaire à bornes comprises dans [-32768..32767]. Ne sont donc pas acceptés les types string, longint, word. Concrètement, on choisit un integer ou un char.

Exemple : variable N indiquant le signe

Avec CASE :

```
...
CASE N OF
     -1 : writeln('négatif') ;
     +1 : writeln('positif') ;
     0 : writeln('nul') ;
     else writeln('problème') ;
     END;
...
```

signification :

```
...
if (N=-1) then writeln('négatif')
else if (N=+1) then writeln('positif')
else if (N=0) then writeln('nul')
else writeln('problème') ;
```

Exemple multivaleur : reconnaître si un caractère C est une voyelle, majuscule ou minuscule.

Avec CASE

```
...
CASE C OF
    'a','e','i','o','u' : writeln('voyelle minuscule') ;

    'A','E','I','O','U' : writeln('voyelle majuscule') ;
    (* point-virgule avant else *)
    ELSE writeln('pas voyelle');
    END ;
...
```

signification :

```
...

if (C='a')or(C='e')or(C='i')or(C='o')or(C='u')
   then writeln('voyelle minuscule')
else if (C='A')or(C='E')or(C='I')or(C='O')or(C='U')
   then writeln('voyelle majuscule')
else writeln('pas voyelle');
```

Contre -exemple : la mise au pluriel des noms est impossible sur ce format CASE.

Exercice :
a) Ecrire un programme qui saisit un entier entre 1 et 7 correspondant à un jour de la semaine et qui affiche le nom correspondant.
b) Prévoir un message d'erreur au cas où on saisirait une valeur hors de 1..7.

Cette structure de choix multiples permet de créer des menus :

Exemple : menu du restaurant (voir TP).

Exemple : choix de la langue pour les E/S :

```
...
writeln('Choisissez votre langue : ') ;
writeln('-1- Allemand') ;
writeln('-2- Anglais') ;
writeln('-3- Espagnol') ;
writeln('-4- Français') ;
writeln('-5- Italien) ;
writeln('---  autre') ;
readln(Langue) ;
CASE Langue OF
  1 : writeln('Auf wiedersehen!') ;
  2 : writeln('Bye !') ;
  3 : writeln('Asta la vista!') ;
  4 : begin writeln('bonjour') ; writeln('on se met au travail ?') ; end ;
  5 : writeln('Ciao!') ;
  ELSE ...
  END;
```

Exercice : Les schémas conditionnels suivants sont-ils équivalents ?

(1)
```
(*bloc1*)
case c of
  v1 : (*bloc2*) ;
  v2 : (*bloc3*) ;
  end;
(*bloc4*)
```

(2)
```
(*bloc1*)
if c=v2 then (bloc3*)
else if c=v1 then (*bloc2*)
(*bloc4*)
```

(3)
```
(*bloc1*)
if c=v1 then (*bloc2*)
if c=v2 then (*bloc3*)
(*bloc4*)
```

(4)
```
if c<>v1 then if c=v2
    then (*bloc2*)(bloc4*)
    else (*bloc4*)
else if c<>v2 then if c<>v1
    then (*rien*)
    else (*bloc1*)(*bloc4*)
else (*bloc4*)
```

4.) Structures répétitives

Dans ce qui précédait, on pouvait par exemple vérifier qu'une valeur saisie entrait bien dans l'intervalle de valeur souhaité, mais un défaut de format impliquait un arrêt du programme. On pouvait redonner une chance à l'utilisateur en programmant une deuxième entrée en cas d'échec, mais l'interface restait limitée. On peut vouloir redonner sa chance plusieurs fois à l'utilisateur, ou, plus généralement et dans le même esprit, pouvoir répéter l'exécution d'un bloc de programme sans avoir à l'écrire plusieurs fois dans le code source ni même s'occuper du nombre de fois où il sera réellement exécuté, nombre qui pourra varier d'une exécution à l'autre.

4.1.) Boucles postcontrôlées : REPETER JUSQU'A (REPEAT ...UNTIL)

Dans le programme Piquet la saisie du jeu d'un joueur est contrôlée, on répète la saisie jusqu'à ce que la valeur choisie soit bien comprise entre 1 et 10. La structure de l'instruction est :

Forme algorithmique : REPETER *bloc* JUSQU'A *condition*
Forme Pascal : REPEAT *bloc* UNTIL *condition* ;

Contrairement à l'instruction Tantque, cette instruction est toujours exécutée au moins une fois, ce n'est qu'après l'exécution du bloc qu'intervient le contrôle de la répétitive.

Exemple :
Dans le programme Piquet, cette répétitive est utilisée pour contrôler la saisie des valeurs par les joueurs :

```
Algorithme                          Pascal

REPETER                             REPEAT
    saisir la valeur du joueur          begin (* 2 *)
                                        write(' (entre 1 et 10) ? ') ;   N.B. fait suite à l'écriture ...
                                        readln(jeu)                      N.B. ; oublié avant end : OK.
                                        end ; (* 2 *)
    JUSQU'A valeur correcte         UNTIL (jeu>=1) AND (jeu<=10) ;
```

Trace d'une exécution : jeu=-5 puis jeu=13 puis jeu=6.

Ce rôle dans le contrôle de la saisie des valeurs est fréquent (surtout en Pascal ou l'univers est très cadré) en programmation. On peut l'utiliser dans le contexte d'un menu.

Remarque : le contrôle a posteriori permet de s'assurer que la variable a bien reçu une valeur (voir au §4.2. ce qui se passerait avec un WHILE).

Exercice : constuire un menu avec contrôle de saisie qui choisit les inscriptions à un club de sport (judo, badminton, danse, tennis). Premièr programme : un seul sport ; deuxième programme : plusieurs sports mais pas ensemble badminton-tennis ou danse-judo, pour des questions d'horaires)

4.2.) Boucles précontrôlées : TANTQUE (WHILE ...DO)

Forme algorithmique : TANTQUE *condition* FAIRE *bloc*
Forme Pascal : WHILE *condition* DO *bloc* ;

Exemple : Revenons sur l'algorithme PIQUET : poursuite du jeu TANT QUE la valeur 100 n'a pas été atteinte.
La partie correspondant au TANTQUE de l'algorithme Piquet donnerait en Pascal :

```
WHILE compteur<100 DO
        BEGIN (* 1 *)
        write('Jeu de ',nom1) ;                    c'est à nom1 de jouer
        (* saisir avec contrôle de validité *)
        ...                                        saisir la valeur jeu de nom1
        compteur:=compteur+jeu ;
        writeln('       compteur: ', compteur) ;
        (* échange des noms *)
        nom:=nom1 ; nom1:=nom2 ; nom2:=nom ;   bloc
        END ; (* 2 *)
```

N.B. : il existe une technique pour gagner automatiquement à ce jeu avec :
`jeu:=((101-compteur) mod 11).`

Exercice : Ecrire l'algorithme puis le programme Pascal qui permet de trouver, pour x in]0;1[et E in R donnés, Min({n in N | x^n<=E}).

Exercice : Soit la suite (Un) définie par U_o=1 et $U_{n+1}=U_n/2$.
Trouver n tel que $U_n<10^{-3}$, puis n tel que 2–Somme(U_i,0<=i<=n)<10^{-4}.

Exemple d'application : Pour trouver une approximation de SQRT(X), on peut utiliser la suite (An) : A_o=1 et $A_{n+1}=(X+A_n)/(1+A_n)$ qui converge vers SQRT(X). Ecrire un algorithme puis un programme qui permet de trouver cette racine avec une précision E donnée.

Exercice : La suite (Pn) définie par P_1=2 et $P_{n+1}=(4\ n^2\ P_n)/(4\ n^2 -1)$ converge vers Pi. Valeur approchée de Pi avec une précision E donnée ?

Exercice : reprendre les exercices avec WHILE en les transformant en REPEAT. Quels sont les avantages et les inconvénients des deux méthodes ?

En conclusion : comment choisir entre Tantque et Répéter ?

Un choix possible : utiliser préférentiellement le Tantque, sauf :
– quand on est sûr que le bloc intérieur sera exécuté au moins une fois,
– quand il s'agit d'un contrôle de type d'une valeur saisie.

EXERCICES

1) (voir TP et chapitre sur les tableaux)
a) Ecrire un programme qui retourne un mot.
b) Modifier ce programme pour qu'il vérifie si le mot saisi est un palindrome. ("esope reste ici et se repose"...)
c) Ecrire un programme qui reconnaît si un mot est un palindrome sans le renverser.
N.B. : LENGTH(S)
T:=COPY(S,debut,longueur)

2) Ecrire un programme qui saisit des valeurs entières, la fin de la saisie étant indiquée par la saisie de 0, puis qui affiche le nombre d'entiers positifs et le nombres de négatifs saisis.
2bis) Ecrire un programme qui saisit [un nombre n inconnu de] des valeurs réelles positives et qui calcule leur moyenne, leur maximum et leur minimum. (voir TP)
N.B. : on peut choisir la saisie d'un nombre négatif comme indicateur de fin de liste.

4.3) Les boucles bornées (=énumération) POUR (FOR)

Forme algorithmique : POUR I DE *début* A *fin* FAIRE *bloc*
Forme Pascal : FOR I:=*début* TO *fin* DO *bloc*
variante : DOWNTO

4.3.1.) boucles simples

Exemple1 : Afficher les nombres de 1 à 10.

Algorithme :	Programme :
pour I de 1 à 10 faire :	var I : integer ;
Afficher I ;	BEGIN
	for I:=1 to 10 do write(I:4) ;
	END.

Les valeurs de début et de fin de boucles peuvent être choisies plus largement :

Exemple2 :
```
program toutafficher ;
var debut, fin, i : integer ;
BEGIN
write('début ? ') ; readln(debut) ;
write('fin ? ') ; readln(fin) ;
for i:=debut to fin do writeln(i) ;
END.
```

N.B. : ici, si a=b, on n'affiche qu'un nombre et si a>b, on n'affiche aucun nombre. Que se passe-t-il si a et/ou b sont nuls, voir négatifs ?

Exercice1 : Trouver au moins trois programmes différents qui affichent 32 fois une astérisque.

Exercice2 : Afficher les dix premiers nombres pairs par ordre croissant puis par ordre décroissant.

Exercice3 : Calculer n! pour n donné. Comment gérez-vous les bornes de votre algorithme ?

Exemple3 : Les lapins de Fibonacci (se reproduisent, après un an de latence, tous les ans). Suite $F_0=1$, $F_1=1$ et $F_{n+2}=F_n+F_{n+1}$. Calculer le 20ème terme.

Exercice4 : Soit la suite (Un) définie par $U_0=2$ et $U_{n+1}=SQRT(U_n^2-1)$. Calculer les cinq premiers terme de cette suite. Calculer la somme des dix premiers termes de la suite.

4.3.2.) boucles imbriquées (voir les tableaux)

Exemple1 : Deux boucles imbriquées qui indiquent ce qu'elles font :
```
var i, j : integer ;
BEGIN
writeln('début') ;
for i:=1 to 3 do
        begin
        writeln('début boucle en j') ;
        for j:=0 to 2 do
                writeln('i=', i, ' j=', j) ;
        writeln('fin de boucle en j') ;
        end ;
writeln('fin') ;
END.
```

Exemple2 : calculer n! pour toutes les valeurs n de 0 à 10

```
programme :     var i, j, factorielle : integer ;
                BEGIN
                for i:=0 to 10 do
                        begin
                        factorielle:=1
                        for j:=2 to i do factorielle:=factorielle*j ;
                        writeln(i,'!=',factorielle) ;
                        end ;
                END.
```

Exercice1 : Afficher les tables de multiplications de 1 à 10.
N.B. : on peut "simplifier" en ne calculant que les produits ab pour a<=b.

Exercice2 : Ecrire un algorithme puis un programme qui vérifie si un nombre n donné est premier (méthode brutale).

EXERCICES

1) Calculer la moyenne de n nombres, n étant connu.

2) Une boulangère désire établir une table des prix du pain (de un à 10 pains, 2frs/flûte et 32,40frs/baguette) sous forme d'un tableau : quantité prix-flûtes prix-baguettes.

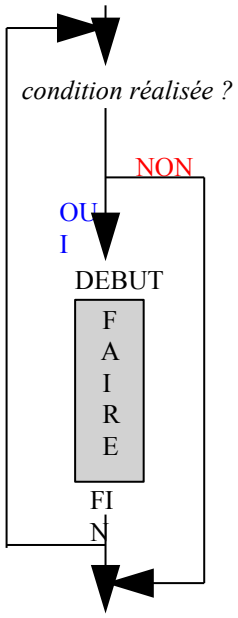
3) Calculer x^n pour x réel et n entier naturel donnés, en n'utilisant que des multiplications. Idem pour n entier relatif.

4)
a) Afficher tous les diviseurs d'un entier positif donné.
b) Afficher la somme des diviseurs d'un entier positif donné (y compris 1 mais pas lui-même).

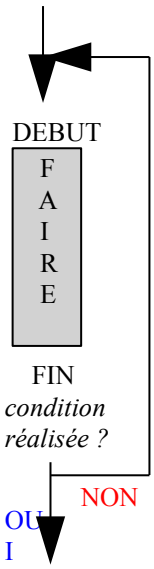
5) Soient une valeur filtre F, une valeur borne B, et une série de nombres saisis au clavier et se terminant par B. Ecrire un algorithme qui calcule la somme des cinq premières valeurs de la série qui sont inférieures ou égales à F. exp : F=12, B=-1 a) 1, 2, 12, 4, 18, 2, -1 donne 21 (1+2+12+4+2)

6) Un nombre est parfait s'il est égal à la somme de ses diviseurs (1 compris).
Ecrire un algorithme qui affiche tous les nombres parfaits compris entre 0 et un entier n donné.
N.B. : nombres amis égaux à la somme des diviseurs de l'autre : voir sous-programmes.

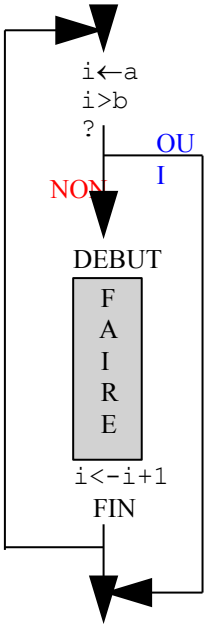
4.4.) Comparaison des trois schémas itératifs



TANTQUE *condition* FAIRE ...



REPETER ... JUSQU'A *condition*



POUR *i* DE *a* A *b* FAIRE ...

Les trois schémas itératifs sous forme IF-GOTO

```
...
:DebutBoucle
IF NOT condition THEN
      GOTO FinBoucle  ;
Begin
...
End ;
GOTO DebutBoucle
:FinBoucle
...
```

TANTQUE condition FAIRE

```
...
:DebutBoucle
Begin
 ...
End ;
IF NOT condition THEN
      GOTO DébutBoucle ;
...
```

REPETER ... JUSQU'A *condition*

```
...
i:=a ;
:DebutBoucle
IF i>b THEN
      GOTO FinBoucle
Begin
...
End ;
i:=i+1 ;
GOTO DebutBoucle
:FinBoucle
...
```

POUR *i* DE *a* A *b* FAIRE ...

TP3 (& TP3bis) <2001

Instruction de contrôle, alternative et répétitives.

TP4 <2001

Choix multiples, itératives, menus.