Travaux Pratiques d'Informatique

Classe préparatoire économique, option économique, première année

Lycée Gambetta, Arras

• Mélissa Bailloeuil •

année scolaire 2012-2013

Travaux Pratiques d'Informatique 1

Introduction

- LES LOGICIELS
  Dans la majorité des cas, on achète des programmes (logiciels) tout faits qui correspondent plus ou moins au besoin (Traitement de texte, Tableur, Base de données))

- ORGANISATION DE L'ORDINATEUR
  Un ordinateur est une machine bête, ne sachant qu'obéir, et à très peu de choses : 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.

- LANGAGE DE PROGRAMMATION
  Le PASCAL, créé par WIRTH au début des années 70, possède des instructions assez claires (si vous comprenez l'anglais), et favorise une approche méthodique et disciplinée (on dit "structurée").
  Le PASCAL est un langage compilé, c'est-à-dire qu'il faut :
  1. entrer un texte dans l'ordinateur (à l'aide d'un programme appelé EDITEUR),
  2. le traduire en langage machine (c'est-à-dire en codes binaires compréhensibles par l'ordinateur) : c'est la compilation,
  3. l'exécuter.
- LES ACTEURS
  Lors de la création d'un programme et de son utilisation, plusieurs acteurs entrent en jeu :
  1. Le programmeur, être humain qui construit le programme en turbo pascal et qui, lors de la conception, devra penser à toutes les interactions possibles entre deux autres personnes : l'ordinateur et l'utilisateur.
  2. L'ordinateur, machine qui exécutera le programme.
  3. L'utilisateur, être humain qui inter-agira avec l'ordinateur lors de l'exécution du programme.

Premier programme Un programme PASCAL est composé

1. d'une entête composée du mot PROGRAM, suivi du nom du programme.
2. des déclarations de constantes et variables (CONST, VAR)
3. des instructions (délimitées par BEGIN et END.).

EXEMPLE

```
PROGRAM cercle;
VAR perimetre,diametre : REAL;
CONST PI=3.141592;
BEGIN
   writeln('Donner le diamètre du cercle');
   readln(diametre);
   perimetre := PI * diametre;
   writeln('Le diamètre est');
   writeln(diametre);
   writeln('Le périmètre est');
   writeln(perimetre);
   readln;
END.
```

Remarquez la présentation du programme qui suit une certaine "indentation" afin de le rendre plus lisible au programmeur.

La partie déclarative de notre programme est limitée à la déclaration de deux variables (mot clef VAR) et d'une constante mot clef CONST). Une variable ou une constante est une "case" mémoire de l'ordinateur, à laquelle on donne ici un nom. Chaque case peut contenir une valeur. On a précisé ici que notre constante prendrait toujours la valeur 3.141592 au cours de l'exécution du programme et que nos deux variables PERIMETRE et DIAMETRE contiendraient des réels. Les types simples connus en PASCAL sont : REAL, INTEGER (entier naturel), CHAR (contient UN et un seul caractère), et BOOLEAN (booléen, c.a.d qui peut valoir soit TRUE (vrai) soit FALSE (faux). En TURBO PASCAL, les entiers admissibles sont compris entre -32768 et +32767. Les réels doivent être compris en TURBO entre + et -1.7E37 (c.a.d 1,7 fois 10 puissance 37), avec 11 chiffres significatifs. 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, certains PASCAL sont plus limités (31 caractères par ex). 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 :
- affichage à l'écran de "Donner le diamètre du cercle".
- 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 DIAMETRE et continue le programme lorsque l'on appuie sur la touche "ENTREE" ou "RETURN".
- calcul et affectation : on multiplie le contenu de la case DIAMETRE par le contenu de la case PI, et on met le résultat dans la case PERIMETRE. Le **:=** symbolise le verbe "recevoir".
- affichage à l'écran de "Le diamètre est".
- affichage à l'écran du contenu de la case DIAMETRE .
- affichage à l'écran de "Le périmètre est".
- affichage à l'écran du contenu de la case PERIMETRE .

Effectuons quelques remarques sur ce programme :
- Les textes doivent être entourés de cotes ('). Les majuscules/minuscules sont significatives. Pour afficher une apostrophe utiliser deux cotes ('l"exemple'). Pour sauter une ligne utiliser WRITELN seul.
- Les instructions doivent toujours être séparées par des " ;". Le programme doit toujours se terminer par un point.
- On peut insérer des remarques dans le programme (qui ne seront pas lues par le compilateur) en les entourant par (* et *) ou { et }.

Exercice 1
Qu'affichera à l'écran la séquence "writeln('Diamètre : ',diametre,', Périmètre : ',perimetre) ;" ajoutée avant "readln ;" dans notre programme ?

Exercice 2 (TVA)
Faire un programme demandant le prix unitaire HT d'un article et sa quantité, puis qui affiche : le total Hors Taxes, le montant de la TVA (pour un taux de 18,6 %) et le total TTC.

La commande **write** effectue un affichage à l'écran.
La commande **read** attend une saisie au clavier de la part de l'utilisateur.

Travaux Pratiques d'Informatique 2

CORRECTION DES EXERCICES

Exercice 1 (TVA)

Faire un programme demandant le prix unitaire HT d'un article et sa quantité, puis qui affiche : le total Hors Taxes, le montant de la TVA (pour un taux de 18,6 %) et le total TTC.

```
PROGRAM PRIX;
VAR Q : integer;
VAR PUHT, THT, MTVA, TTC : real;
CONST TVA=0.186;
BEGIN
     writeln('Donner le prix unitaire HT.');
     readln(PUHT);
     writeln('Donner la quantité.');
     readln(Q);
     THT:=PUHT*Q;
     MTVA:=THT*TVA;
     TTC:=THT+MTVA;
     writeln('Le total hors taxe est ');
     writeln(THT);
     writeln('Le montant de la TVA est ');
     writeln(MTVA);
     writeln('Le total toutes taxes est ');
     writeln(TTC);
     readln;
END.
```

INSTRUCTION D'AFFECTATION

On appelle AFFECTATION la mise d'une valeur dans une variable. Le signe := signifie "**mettre la VALEUR à droite du := dans la mémoire désignée à gauche**". *Une affectation du type B*C :=A est donc IMPOSSIBLE.*

Une affectation ne peut se faire qu'entre une variable et une expression de même type (caractère, bouléen, réel, entier...).

- La seule exception est de mettre un entier dans un réel (le .0 est rajouté automatiquement).
- L'inverse est impossible directement. Soient I entier et X réel, pour mettre X dans I il faut utiliser I :=ROUND(X) (arrondi) ou I :=TRUNC(X) (partie entière).
- **ENTIERS**
  - Déclaration : VAR varia1,varia2,...,variaN : INTEGER ;
  - Opérations sur entiers : + - * div (division) mod (reste de la division). Elles sont toutes à résultat entier, et nécessitent deux arguments entiers.
- **REELS**
  - Déclaration : VAR liste de variables : REAL ;
  - Opérations : + - * /
  - Quand une opération comprend un argument réel et un entier, le résultat est réel. / donne toujours un résultat réel, même si les deux arguments sont entiers.
  - * et / sont de priorité supérieure à + et -, mais entre * et / tout dépend du compilateur (en général de gauche à droite). En cas d'ambiguïté, utilisez des parenthèses (il n'y a aucun inconvénient à mettre plus de parenthèses que nécessaire).
- **BOOLEENS**
  - Déclaration : VAR liste de variables : BOOLEAN ;
  - Ces variables peuvent prendre soit la valeur TRUE (vrai), soit la valeur FALSE (faux).
  - Opérations booléennes : AND, OR, NOT, XOR (ou exclusif). Ces opérations nécessitent des arguments booléens.
  - Opérations à valeur booléenne : > (supérieur), < (inf), >= (sup ou égal), <=, = (égal),<>(différent). Ces opérations comparent tous éléments de type simple (les 2 arguments doivent être de même type, sauf entiers et réels qui peuvent être comparés entre eux), et renvoient un booléen.
  - AND (et), OR (ou), NOT (non), sont de priorité supérieure aux précédents et ne peuvent opérer que sur des booléens : $A > B$ et C doit être écrit : $(A > B) and (A > C)$. Les parenthèses sont obligatoires pour ne pas faire en premier B and A.

  Exemple :

```
VAR test:boolean;
VAR a,b:real;
...
on peut écrire : TEST:=(A<B)and(A>0)
```

Exercice 1

Ecrire un programme qui pour un taux de croissance annuel fourni par l'utilisateur renvoie le taux de croissance mensuel moyen correspondant.

Exercice 2

Ecrire un programme qui fait la moyenne de trois nombres demandés à l'utilisateur.

Travaux Pratiques d'Informatique 3

CORRECTION DES EXERCICES

Exercice 1

Ecrire un programme qui pour un taux de croissance annuel fourni par l'utilisateur renvoie le taux de croissance mensuel moyen correspondant.

```
PROGRAM TAUX;
VAR A,M : real;
BEGIN
     writeln('Donner le taux annuel.');
     readln(A);
     M:=(exp(ln(1+A/100)/12)-1)*100;
     writeln('Le taux mensuel est ');
     writeln(M);
     readln;
END.
```

Exercice 2

Ecrire un programme qui fait la moyenne de trois nombres demandés à l'utilisateur.

```
PROGRAM MOYENNE;
VAR A,B,C,m : real;
BEGIN
     writeln('Donner le premier nombre.');
     readln(A);
     writeln('Donner le deuxieme nombre.');
     readln(B);
     writeln('Donner le troisieme nombre.');
     readln(C);
     m:=(A+B+C)/3;
     writeln('La moyenne est :');
     writeln(m);
     readln;
END.
```

LES FONCTIONS STANDARDS

On peut utiliser des fonctions qui renvoient une valeur au programme. Par exemple, A :=sqrt(B*C) met dans A la racine carrée de B fois C. B*C est appelé ARGUMENT de la fonction.
Les principales fonctions standard connues par Turbo Pascal sont :

- ABS : renvoie la valeur absolue.
- SQR : renvoie le carré.
- SQRT : renvoie la racine carrée.
- EXP : exponentielle. Le résultat est forcément un réel.
- LN : log népérien. Le résultat est forcément un réel.
- TRUNC : partie entière (permet de mettre un réel dans un entier : trunc(4.5)=4). Le résultat est forcément un entier.
- DIV et MOD : renvoie le quotient et le reste de la division euclidienne. Le résultat est forcément un entier. Par exemple 37 DIV 5 donne 7 et 37 MOD 5 donne 2.

INSTRUCTIONS

On appelle instruction soit

- une affectation ($m := (A + B + C)/3$;).
- un appel à une procédure (nous l'étudierons plus tard).
- une structure de contrôle (nous le découvrirons au fur et à mesure des TP).

On regroupe plusieurs instructions sous la forme BEGIN instruction1 ;instruction2 ; ... ;instructionN ; END
La lisibilité du programme sera meilleure en mettant une instruction par ligne, et en décalant à droite les instructions comprises entre un begin et un end.

INSTRUCTION IF - THEN - ELSE

Structure :

- Premier cas : IF condition $THEN$ instruction1.
- Second cas : IF condition $THEN$ instruction1 $ELSE$ instruction 2.

Si la condition est vraie, alors on exécute l'instruction 1. Sinon, on passe à la suite (cas 1), ou on exécute l'instruction 2 (cas 2).
Si dans "*instruction*1" ou "*instruction*2", on désire mettre plusieurs instructions, la structure devient :

- Premier cas :

```
IF condition THEN BEGIN
     instruction;
     instruction;
END;
```

- Second cas :

```
IF condition THEN BEGIN
     instruction;
     instruction;
END;
ELSE BEGIN
     instruction;
     instruction;
END;
```

Exercice 1

Ecrire un programme qui pour a, b et c donnés par l'utilisateur renvoie les racines du trinôme $ax^2 + bx + c$.

Travaux Pratiques d'Informatique 4

CORRECTION DU TP 03

Exercice 1 (Trinôme)

Ecrire un programme qui pour a, b et c donnés par l'utilisateur renvoie les racines du trinôme $ax^2 + bx + c$.

```
PROGRAM TRINOME;

VAR A,B,C,DELTA,X,Y : real;

BEGIN
     {Saisie des coefficients a, b et c}
     writeln('Donner a.');
     readln(A);
     writeln('Donner b.');
     readln(B);
     writeln('Donner c.');
     readln(C);

     {Calcul de DELTA}
     DELTA:=sqr(B)-4*A*C;

     {Affichage du resultat}
     if DELTA>0 then begin
       X:=(-B-sqrt(DELTA))/(2*A);
       Y:=(-B+sqrt(DELTA))/(2*A);
       writeln('Deux racines :');
       writeln(X);
       writeln(Y);
     end;
     if DELTA=0 then begin
       X:=-B/(2*A);
       writeln('Une racine :');
       writeln(X);
     end;
     if DELTA<0 then begin
       writeln('Pas de racines.')
     end;
     readln;
END.
```

LA FONCTION RANDOM

Il est quelquefois nécessaire d'avoir recours à l'utilisation de valeurs de variables qui soient le fruit du hasard. Mais l'ordinateur n'est pas capable de créer du vrai hasard. Il peut cependant fournir des données dites aléatoires, c'est-à-dire issues de calculs qui utilisent un grand nombre de paramètres eux-mêmes issus de l'horloge interne. On appelle cela un pseudo-hasard car il est très difficile de déceler de l'ordre et des cycles dans la génération de valeurs aléatoires. Ainsi, on admettra que Turbo Pascal 7.0 offre la possibilité de générer des nombres aléatoires.

Avant l'utilisation des fonctions qui vont suivre, il faut initialiser le générateur aléatoire avec la procédure Randomize. Cette initialisation est indispensable pour être sûr d'obtenir un relativement "bon hasard" bien que ce ne soit pas obligatoire.

Syntaxe :

```
Randomize ;
```

On peut générer un nombre réel aléatoire compris entre 0 et 1 grâce à la fonction Random. *Syntaxe :*

```
X := Random ;
```

On peut générer un nombre entier aléatoire compris entre 0 et Y-1 grâce à la fonction Random(Y). *Syntaxe :*

```
X := Random(Y) ;
```

Exercice 1

Ecrire un programme qui affiche cinq nombres au hasard :

1. le premier est un réel compris entre 0 et 1
2. le deuxième est un réel compris entre 1 et 2
3. le troisième est un réel compris entre 0 et 5
4. le quatrième est un entier compris entre 1 et 4
5. le cinquième est un entier pair compris entre 6 et 12

Exercice 2

Ecrire un programme qui affiche cinq nombres au hasard :

1. le premier est un réel compris ente 0 et 3
2. le deuxième est un réel compris ente 1 et 2
3. le troisième est un entier compris ente 0 et 5
4. le quatrième est un entier compris ente 3 et 7
5. le cinquième est un entier multiple de 7 compris ente 14 et 35

Travaux Pratiques d'Informatique 5

CORRECTION DU TP 04

Exercice 1 (Correction)

Ecrire un programme qui affiche cinq nombres au hasard :

1. le premier est un réel compris entre 0 et 1
2. le deuxième est un réel compris entre 1 et 2
3. le troisième est un réel compris entre 0 et 5
4. le quatrième est un entier compris entre 1 et 4
5. le cinquième est un entier pair compris entre 6 et 12.

```
PROGRAM HASARD;
VAR X : real;
VAR A : integer;
BEGIN
     randomize;
     {Reel compris entre 0 et 1}
     X:=random;
     writeln('Reel compris entre 0 et 1 :');
     writeln(X);
     {Reel compris entre 1 et 2}
     X:=random + 1;
     writeln('Reel compris entre 1 et 2 :');
     writeln(X);
     {Reel compris entre 0 et 5}
     X:=5 * random;
     writeln('Reel compris entre 0 et 5 :');
     writeln(X);
     {Entier compris entre 1 et 4}
     A:=random(4)+1;
     writeln('Entier compris entre 1 et 4 :');
     writeln(A);
     {Entier pair compris entre 6 et 12}
     A:=(random(4)+3)*2;
     writeln('Entier pair, entre 6 et 12 :');
     writeln(A);
     readln;
END.
```

Exercice 2

Ecrire un programme qui affiche cinq nombres au hasard :

1. le premier est un réel compris ente 0 et 3
2. le deuxième est un réel compris ente 1 et 2
3. le troisième est un entier compris ente 0 et 5
4. le quatrième est un entier compris ente 3 et 7
5. le cinquième est un entier multiple de 7 compris ente 14 et 35

```
PROGRAM HASARD;
VAR X : real;
VAR A : integer;
BEGIN
     randomize;
      X:=3*random;
     writeln('Reel compris entre 0 et 3 :');
     writeln(X);
     X:=random + 1;
     writeln('Reel compris entre 1 et 2 :');
     writeln(X);
     A:=random(6);
     writeln('Entier compris entre 0 et 5 :');
     writeln(X);
     A:=random(5)+3;
     writeln('Entier compris entre 3 et 7 :');
     writeln(A);
     A:=(random(4)+2)*7;
     writeln('Multiple de 7, entre 14 et 35 :');
     writeln(A);
     readln;
END.
```

Exercices

Exercice 1

Ecrire un programme qui affiche cinq nombres au hasard :

1. le premier est un réel compris entre 2 et 3
2. le deuxième est un réel compris entre 2 et 5
3. le troisième est un entier compris entre 3 et 6
4. le quatrième est un entier impair compris entre 3 et 17
5. le cinquième est un entier multiple de 4 compris entre 16 et 32

Exercice 2

Ecrire un programme proposant à l'utilisateur le jeu suivant : l'ordinateur tire au sort trois nombres entiers compris ente 1 et 11. L'utilisateur propose un nombre entre 1 et 11. Il a gagné si ce nombre fait parti des trois tirés au sort. Il a perdu sinon. L'ordinateur affiche l'issue du jeu.

Exercice 3

Ecrire un programme qui simule un lancer de trois dés et affiche le résultat.

Travaux Pratiques d'Informatique 6

CORRECTION DU TP 05

Exercice 1 (Correction)

Ecrire un programme qui affiche cinq nombres au hasard :

1. le premier est un réel compris entre 2 et 3
2. le deuxième est un réel compris entre 2 et 5
3. le troisième est un entier compris entre 3 et 6
4. le quatrième est un entier impair compris entre 3 et 17
5. le cinquième est un entier multiple de 4 compris entre 16 et 32

```
PROGRAM HASARD;
VAR X : real;
VAR A : integer;
BEGIN
     randomize;
      X:=random+2;
     writeln('Reel compris entre 2 et 3 :');
     writeln(X);
     X:=random*3 + 2;
     writeln('Reel compris entre 2 et 5 :');
     writeln(X);
     A:=random(4)+3;
     writeln('Entier compris entre 3 et 6 :');
     writeln(A);
     A:=((random(8)+1)*2)+1;
     writeln('Entier impair entre 3 et 17 :');
     writeln(A);
     A:=(random(5)+4)*4;
     writeln('Multiple de 4 entre 16 et 32 :');
     writeln(A);
     readln;
END.
```

Exercice 2 (Correction)

Ecrire un programme proposant à l'utilisateur le jeu suivant : l'ordinateur tire au sort trois nombres entiers compris ente 1 et 11. L'utilisateur propose un nombre entre 1 et 11. Il a gagné si ce nombre fait parti des trois tirés au sort. Il a perdu sinon. L'ordinateur affiche l'issue du jeu.

```
PROGRAM JEU;
VAR A,B,C,U : integer;
BEGIN
     randomize;
     {Tirage de trois entiers entre 1 et 11}
     A:=random(11)+1;
     B:=random(11)+1;
     C:=random(11)+1;
     {Nombre de l'utilisateur}
     writeln('Donner un entier entre 1 et 11');
     readln(U);
     if ((U=A) or (U=B) or (U=C)) then begin
       writeln('Vous avez gagne.');
     end
     else begin
          writeln('Vous avez perdu.');
     end;
     writeln('Les nombres etaient');
     writeln(A);
     writeln(B);
     writeln(C);
     readln;
END.
```

Exercice 3 (Correction)

Ecrire un programme qui simule un lancer de trois dés et affiche le résultat.

```
Program Lancers;
var A,B,C:integer;
begin
     randomize;
     A:=random(6)+1;
     writeln(A);
     B:=random(6)+1;
     writeln(B);
     C:=random(6)+1;
     writeln(C);
     readln;
end.
```

Exercices

Exercice 1

Ecrire un programme qui, pour la suite (u_n) suivante, affiche le terme dont le rang sera choisi au hasard par l'ordinateur entre 0 et 50.

$$\forall n \in \mathbb{N},\ u_n = n^2 + n + 1$$

Exercice 2

Ecrire un programme qui, pour la suite (u_n) suivante, affiche le terme dont le rang sera choisi par l'utilisateur.

$$\forall n \in \mathbb{N}^*,\ u_n = (4n+3) * |\ln\left(\frac{n}{10}\right)|$$

Exercice 3

Ecrire un programme qui, pour la suite (u_n) suivante, affiche le terme dont le rang sera choisi par l'utilisateur.

$$\forall n \in \mathbb{N}^*,\ u_n = \sqrt{|3-n|}$$

Travaux Pratiques d'Informatique 7

CORRECTION DU TP 06

Exercice 1 (Correction)

Ecrire un programme qui, pour la suite (u_n) suivante, affiche le terme dont le rang sera choisi au hasard par l'ordinateur entre 0 et 50.

$$\forall n \in \mathbb{N},\ u_n = n^2 + n + 1$$

```
Program Suite;
var n:integer;
var u:real;
Begin
     randomize;
     n:=random(51);
     u:=sqr(n)+n+1;
     writeln('Le terme de rang ',n,' est ',u);
     readln;
end.
```

Exercice 2 (Correction)

Ecrire un programme qui, pour la suite (u_n) suivante, affiche le terme dont le rang sera choisi par l'utilisateur.

$$\forall n \in \mathbb{N},\ u_n = (4n+3) * |\ln\left(\frac{n}{10}\right)|$$

```
Program Suite2;
var n:integer;
var u:real;
Begin
     writeln('Donner n.');
     readln(n);
     u:=(4*n+3)*abs(ln(n/10));
     writeln(u);
     readln;
end.
```

Exercice 3 (Correction)

Ecrire un programme qui, pour la suite (u_n) suivante, affiche le terme dont le rang sera choisi par l'utilisateur.

$$\forall n \in \mathbb{N}^*,\ u_n = \sqrt{|3-n|}$$

```
Program Suite3;
var n:integer;
var u:real;
Begin
     writeln('Donner n.');
     readln(n);
     u:=sqrt(abs(3-n));
     writeln(u);
     readln;
end.
```

LES AFFICHAGES

Pour afficher, on utilise les instructions **write** (écrit à l'écran) ou **writeln** (écrit à l'écran puis retour à la ligne), le message à afficher étant entre parenthèses.
Pour afficher :

- un message en français, on encadre le message par des apostrophes
- un message qui serait le contenu d'une variable, on cite le nom de la variable
- plusieurs messages sur une même ligne, on les sépare par des virgules.

Par exemple

writeln('Le prix est ' , RES , ' euros.') ;

affichera *Le prix est 30 euros.* si le contenu de la variable RES est 30.

Exercices

Exercice 1

Soit la suite (u_n) géométrique de raison q et de premier terme u_0. Ecrire un programme qui demande à l'utilisateur les valeurs de q et de u_0, demande un rang n à l'utilisateur puis affiche la valeur du terme de rang n de cette suite.

Exercice 2

Soit (u_n) la suite récurrente linéaire d'ordre 2 de premier terme 0 et de second terme 1 défini par :

$$\forall n \in \mathbb{N},\ u_{n+2} = a \times u_{n+1} + b \times u_n$$

Ecrire un programme qui demande les valeurs de a et de b à l'utilisateur puis renvoie le Δ de l'équation caractéristique, la ou les racines correspondantes puis u_n pour n choisi par l'utilisateur.
Cette exercice difficile nécessite de résoudre le système en λ et μ avec des paramètres pour trouver les formules finales.

Travaux Pratiques d'Informatique 8

CORRECTION DU TP 07

Exercice 1 (Correction)

Soit la suite (u_n) géométrique de raison q et de premier terme u_0. Ecrire un programme qui demande à l'utilisateur les valeurs de q et de u_0, demande un rang n à l'utilisateur puis affiche la valeur du terme de rang n de cette suite.

```
Program Suite1;
var q, u_0, u_n : real;
var n: integer;
Begin
  writeln('Donner q.');
  readln(q);
  writeln('Donner u_0.');
  readln(u_0);
  writeln('Donner n.');
  readln(n);
  u_n:=u_0*exp(n* ln(q));
  writeln('Le terme de rang ',n,' vaut ',u_n);
  readln;
end.
```

Exercice 2 (Correction)

Soit (u_n) la suite récurrente linéaire d'ordre 2 de premier terme 0 et de second terme 1 défini par :

$$\forall n \in \mathbb{N},\ u_{n+2} = a \times u_{n+1} + b \times u_n$$

Ecrire un programme qui demande les valeurs de a et de b à l'utilisateur puis renvoie la forme du terme général de u.

```
Program Suite2;
var a, b, delta, x_un, x_deux, mu : real;
Begin
  writeln('Donner a.');
  readln(a);
  writeln('Donner b.');
  readln(b);
  {Résolution de x^2-ax-b=0}
  delta:=a*a+4*b;
  if delta<0 then begin
   writeln('Pas de formule');
  end;
  if delta=0 then begin
    x_un:=a/2;
    writeln('u_n=n*',x_un,'^(n-1)');
  end;
  if delta>0 then begin
    x_un:=(a-sqrt(delta))/2;
    x_deux:=(a+sqrt(delta))/2;
    mu:=1/(x_deux-x_un);
    writeln('u_n=',mu,'(',x_deux,
                          '^n-',x_un,'^n)');
  end;
  readln;
end.
```

INSTRUCTION FOR - TO - DO

Cette structure de contrôle s'appelle une boucle définie. Elle permet de répéter une même séquence d'instructions (délimitée par begin et end ;) un nombre de fois déterminé par un **compteur** (variable de type integer) qui va énumérer les nombres entiers de la première valeur (**debut**) à la dernière valeur (**fin**).

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

Exercices

Exercice 1

Quel sera l'affichage à l'écran pour les séquences d'instructions suivantes ?

1.
```
FOR k:=1 TO 6 DO BEGIN
  writeln(k);
END;
```
2.
```
FOR i:=0 TO 5 DO BEGIN
  writeln('coucou');
END;
```
3.
```
FOR j:=4 TO 7 DO BEGIN
  a:=j*j;
  write('le carré de ', j ,' : ')
  writeln(a);
END;
```

Exercice 2

Ecrire un programme qui fournit tous les termes de la suite u jusqu'au rang 30.

$$\forall n \in \mathbb{N},\ u_n = e^{-n}$$

Exercice 3

Ecrire un programme qui fournit tous les termes de la suite u du rang 5 au rang 25.

$$\forall n \in \mathbb{N},\ u_n = n^2 + n + 1$$

Travaux Pratiques d'Informatique 9

CORRECTION DU TP 08

Exercice 1 (Correction)

Ecrire un programme qui fournit tous les termes de la suite u jusqu'au rang 30.

$$\forall n \in \mathbb{N},\ u_n = e^{-n}$$

```
Program Suites;
var n : integer;
var u : real;
begin
  for n:=0 to 30 do begin
     u:=exp(-n);
     writeln('Le terme de rang ',n,' est ',u);
  end;
  readln;
end.
```

Exercice 2 (Correction)

Ecrire un programme qui fournit tous les termes de la suite u du rang 5 au rang 25.

$$\forall n \in \mathbb{N},\ u_n = n^2 + n + 1$$

```
Program Suites;
var n : integer;
var u : real;
begin
  for n:=5 to 25 do begin
     u:=sqr(n)+n+1;
     writeln('Le terme de rang ',n,' est ',u);
  end;
  readln;
end.
```

Compléter le tableau suivant :

$n =$	$u =$
5	
6	
7	
8	
13	
14	
15	

$n =$	$u =$
16	273
17	307
18	343
19	381
20	421
21	463
22	507
23	553
24	601
25	651

INSTRUCTION FOR - TO - DO

Exercice 1

Compléter ce programme qui fournit tous les termes de la suite u jusqu'au rang choisi par l'utilisateur.

$$\forall n \in \mathbb{N}^*,\ u_n = \frac{n}{2}$$

```
Program Suites;
VAR N,k:...;
VAR U:...;
Begin
     {Saisie de N}
     writeln('Donner N.');
     readln(N);
     {Calcul et affichage des termes}
     for k:=... to ... do begin
         U:=.../...;
         write('Terme de rang ',k);
         writeln(U);
     end;
     readln;
End.
```

Compléter le tableau relatif à la boucle pour $N = 8$:

Etape	N	k	u
Initialisation			
Boucle			

Exercice 2

$$\forall n \in \mathbb{N}^*,\ u_n = \frac{1}{n}$$

1. Ecrire un programme qui fournit tous les termes de la suite u jusqu'au rang N choisi par l'utilisateur.
2. Ecrire un programme qui fournit tous les termes de rang pair de la suite u du rang 2 au rang 20.

Travaux Pratiques d'Informatique 10

CORRECTION DU TP 09

Exercice 2 (Correction)

Ecrire un programme qui fournit tous les termes de la suite u jusqu'au rang N fourni par l'utilisateur avec $\forall n \in \mathbb{N}^*,\ u_n = \frac{1}{n}$

```
Program Suites;
VAR N,k:integer;
VAR U:real;
Begin
     {Saisie de N}
     writeln('Donner N.');
     readln(N);
     {Calcul et affichage des termes}
     for k:=1 to N do begin
         U:=1/k;
         write('Terme de rang ',k);
         writeln(U);
     end;
     readln;
End.
```

INSTRUCTION FOR - TO - DO

Exercice 1

Voici un programme qui fournit tous les termes de la suite u jusqu'au rang N choisi par l'utilisateur.

$$u_0 = 1 \text{ et } \forall n \in \mathbb{N},\ u_{n+1} = -2u_n - 3$$

```
Program Suites;
VAR N,k:integer;
VAR U:real;
Begin
     {Saisie de N}
     writeln('Donner N.');
     readln(N);
     {Initialisation du terme de rang 0}
     U:=1;
     write('Terme de rang 0');
     writeln(U);
     {Calcul et affichage des termes}
     for k:=1 to N do begin
         U:=-2*U-3;
         write('Terme de rang ',k);
         writeln(U);
     end;
     readln;
End.
```

Compléter le tableau suivant en prenant $N = 5$ sans utiliser l'ordinateur :

Etape	N	k	U
Initialisation			
Boucle			

Exercice 2

Ecrire un programme qui fournit tous les termes de la suite u jusqu'au rang N choisi par l'utilisateur avec u_0 choisi par l'utilisateur et :

$$\forall n \in \mathbb{N},\ u_{n+1} = u_n + |u_n|$$

Exercice 3

Ecrire un programme qui fournit tous les termes de la suite u jusqu'au rang 7.

$$u_0 = 1 \text{ et } \forall n \in \mathbb{N},\ u_{n+1} = \ln(u_n) + 2$$

On pourra démontrer par récurrence que cette suite est correctement définie.

Travaux Pratiques d'Informatique 11

CORRECTION DU TP 10

Exercice 2 (Correction)

Ecrire un programme qui fournit tous les termes de la suite u jusqu'au rang N fourni par l'utilisateur avec u_0 donné par l'utilisateur et $\forall n \in \mathbb{N},\ u_{n+1} = u_n + |u_n|$

```
Program Suites;
VAR N,k:integer;
VAR U :real;
Begin
     {Saisie de N}
     writeln('Donner N.');
     readln(N);
     writeln('Donner U_0');
     readln(U);
     {Calcul et affichage des termes}
     for k:=1 to N do begin
         U:=U+abs(U);
         write('Terme de rang ',k);
         writeln(U);
     end;
     readln;
End.
```

Exercice 3 (Correction)

Ecrire un programme qui fournit tous les termes de la suite u jusqu'au rang 7.

$$u_0 = 1 \text{ et } \forall n \in \mathbb{N},\ u_{n+1} = \ln(u_n) + 2$$

```
Program Suites;
VAR k:integer;
VAR U:real;
Begin
     {Initialisation du terme de rang 0}
     U:=1;
     write('Terme de rang 0');
     writeln(U);
     {Calcul et affichage des termes}
     for k:=1 to 7 do begin
         U:=ln(U)+2;
         write('Terme de rang ',k);
         writeln(U);
     end;
     readln;
End.
```

Au lieu de travailler avec une variable U représentant le terme courant de la suite, on peut travailler avec deux variables A et B qui représentent des termes consécutifs de la suite.

```
Program Suites;
VAR k:integer;
VAR A,B :real;
Begin
     {Initialisation du terme de rang 0}
     A:=1;
     write('Terme de rang 0');
     writeln(A);
     {Calcul et affichage des termes}
     for k:=1 to 7 do begin
         B:=ln(A)+2;
         write('Terme de rang ',k);
         writeln(B);
         A:=B;
     end;
     readln;
End.
```

Compléter le tableau suivant :

Etape	k	A	B
Initialisation			
Boucle			

INSTRUCTION FOR - TO - DO

Exercice 1

Voici un programme qui fournit tous les termes de la suite u jusqu'au rang 8 avec $u_0 = 2$, $u_1 = 1$ et :

$$\forall n \in \mathbb{N},\ u_{n+2} = u_n \times u_{n+1} + 1$$

```
Program Suites;
VAR A,B,C:real;
VAR k:integer;
Begin
     {Initialisation de u_0 et u_1}
     A:=2;
     B:=1;
     {Calcul des termes consécutifs}
     for k:=2 to 8 do begin
         C:= A*B+1;
         write('Terme de rang ',k);
         writeln(C);
         A:=B;
         B:=C;
     end;
     readln;
End.
```

Compléter le tableau suivant :

Etape	k	A	B	C
Initialisation				
Boucle				

Exercice 2

Ecrire un programme qui fournit tous les termes de la suite u jusqu'au rang N choisi par l'utilisateur avec $u_0 = 1$, $u_1 = 0$ et $\forall n \in \mathbb{N},\ u_{n+2} = 2u_{n+1} + 3u_n$.

Exercice 3

Ecrire un programme qui fournit tous les termes de la suite u jusqu'au rang N choisi par l'utilisateur avec $u_0 = 1$, $u_1 = 0$ et $\forall n \in \mathbb{N},\ u_{n+2} = 2u_{n+1} - 3u_n + 2n$.

Travaux Pratiques d'Informatique 12

CORRECTION DU TP 11

Exercice 2 (Correction)

Ecrire un programme qui fournit tous les termes de la suite u jusqu'au rang N choisi par l'utilisateur avec $u_0 = 1$, $u_1 = 0$ et $\forall n \in \mathbb{N},\ u_{n+2} = 2u_{n+1} + 3u_n$.

```
Program Suites;
VAR A,B,C:real;
VAR k,N:integer;
Begin
     {Saisie de N}
     writeln('Donner N.');
     readln(N);
     {Initialisation de u_0 et u_1}
     A:=1;
     B:=0;
     {Calcul des termes consécutifs}
     for k:=2 to N do begin
         C:= 3*A+2*B;
         write('Terme de rang ',k);
         writeln(C);
         A:=B;
         B:=C;
     end;
     readln;
End.
```

Exercice 3 (Correction)

Ecrire un programme qui fournit tous les termes de la suite u jusqu'au rang N choisi par l'utilisateur avec $u_0 = 1$, $u_1 = 0$ et $\forall n \in \mathbb{N},\ u_{n+2} = 2u_{n+1} - 3u_n + 2n$.

```
Program Suites;
VAR A,B,C:real;
VAR k,N:integer;
Begin
     {Saisie de N}
     writeln('Donner N.');
     readln(N);
     {Initialisation de u_0 et u_1}
     A:=1;
     B:=0;
     {Calcul des termes consécutifs}
     for k:=2 to N do begin
         C:= 2*B-3*A+2*(k-2);
         write('Terme de rang ',k);
         writeln(C);
         A:=B;
         B:=C;
     end;
     readln;
End.
```

INSTRUCTION FOR - TO - DO

Nous avons vu pour le moment trois modèles de programmes. Révisons-les.

Exercice 1 (Suite avec terme général fourni)

Ecrire un programme qui fournit tous les termes de la suite u jusqu'au rang N choisi par l'utilisateur.

$$\forall n \in \mathbb{N}^*,\ u_n = \frac{1}{n^2}$$

Exercice 2 (Suite avec formule de récurrence) liant deux termes consécutifs

Ecrire un programme qui fournit tous les termes de la suite u jusqu'au rang N choisi par l'utilisateur.

$$u_0 = 3, \quad \forall n \in \mathbb{N},\ u_{n+1} = (u_n - 1)^2$$

Exercice 3 (Suite avec formule de récurrence) liant trois termes consécutifs

Ecrire un programme qui fournit tous les termes de la suite u jusqu'au rang N choisi par l'utilisateur.

$$u_1 = 4,\ u_2 = 7, \quad \forall n \in \mathbb{N}^*,\ u_{n+2} = \frac{u_n}{(u_{n+1})^2 + 1}$$

Il reste deux modèles de programmes à voir au sujet de l'utilisation de l'instruction FOR - TO - DO.

Exercice 4 (Puissance)

Compléter ce programme qui calcule x^n pour x un réel et n un entier naturel fournis par l'utilisateur, autrement dit $x^n = x \times x \times x \times \cdots \times x$.

```
Program Puissance;
VAR k,n : integer;
VAR x,P : real;
Begin
     writeln('Donner x'); readln(x);
     writeln('Donner n'); readln(n);
     {Initialisation de la puissance}
     P:=1;
     {Calcul de x^n}
     for k:= ... to ... do begin
         P:=P* ...;
     end;
     writeln(P); readln;
End.
```

Exercice 5 (Factoriel)

Compléter ce programme qui calcule $n!$ pour n un entier naturel fourni par l'utilisateur, autrement dit $n! = 1 \times 2 \times 3 \times \cdots \times n$.

```
Program Factoriel;
VAR k,n,F :integer;
Begin
     writeln('Donner n'); readln(n);
     {Initialisation du factoriel}
     F:=1;
     {Calcul de n!}
     for k:= ... to ... do begin
         F:=F* ...;
     end;
     writeln(F);      readln;
End.
```

Travaux Pratiques d'Informatique 13

CORRECTION DU TP 12

Exercice 1 (Suite avec terme général fourni)

Ecrire un programme qui fournit tous les termes de la suite u jusqu'au rang N choisi par l'utilisateur.
$\forall n \in \mathbb{N}^*,\ u_n = \frac{1}{n^2} + 2^n$.

```
Program Suites;
VAR N,k:integer;
VAR U:real;
Begin
     {Saisie de N}
     writeln('Donner N.'); readln(N);
     {Calcul et affichage des termes}
     for k:=1 to N do begin
         U:=1/(sqr(k)+exp(k * ln(2));
         write('Terme de rang ',k); writeln(U);
     end;
     readln;
End.
```

Exercice 2 (Suite avec formule de récurrence)

Ecrire un programme qui fournit tous les termes de la suite u jusqu'au rang N choisi par l'utilisateur avec $u_0 = 3, \quad \forall n \in \mathbb{N},\ u_{n+1} = (u_n - 1)^2$.

```
Program Suites;
VAR k,N:integer;
VAR U:real;
Begin
     {Saisie de N}
     writeln('Donner N.'); readln(N);
     {Initialisation du terme de rang 0}
     U:=3;
     write('Terme de rang 0'); writeln(U);
     {Calcul et affichage des termes}
     for k:=1 to N do begin
         U:=sqr(U-1);
         write('Terme de rang ',k); writeln(U);
     end;
     readln;
End.
```

Exercice 3 (Suite avec formule de récurrence)

Ecrire un programme qui fournit tous les termes de la suite u jusqu'au rang N choisi par l'utilisateur avec $u_1 = 4,\ u_2 = 7, \quad \forall n \in \mathbb{N}^*,\ u_{n+2} = \frac{u_n}{(u_{n+1})^2+1}$.

```
Program Suites;
VAR A,B,C:real;
VAR k,N:integer;
Begin
     {Saisie de N}
     writeln('Donner N.'); readln(N);
     {Initialisation de u_1 et u_2}
     A:=4; B:=7;
     {Calcul des termes consécutifs}
     for k:=3 to N do begin
         C:= A/(sqr(B)+1);
         write('Terme de rang ',k);
         writeln(C);
         A:=B;
         B:=C;
     end;
     readln;
End.
```

FOR - TO - DO

Exercice 1

1. Ecrire un programme qui calcule $n!$ pour n fourni par l'utilisateur.
2. Compléter ce programme afin que le programme calcule la combinaison de k parmi n, noté $\begin{pmatrix} n \\ k \end{pmatrix}$ où n et k sont fournis par l'utilisateur avec :

$$\begin{pmatrix} n \\ k \end{pmatrix} = \frac{n!}{(n-k)!\ k!}$$

Exercice 2

Ecrire un programme qui affiche à l'écran une simulation de 20 lancers de dés.

Travaux Pratiques d'Informatique 14

CORRECTION DU TP 13

Exercice 1 (Correction)

1. Ecrire un programme qui calcule $n!$ pour n fourni par l'utilisateur.
2. Compléter ce programme afin que le programme calcule la combinaison de k parmi n, noté $\begin{pmatrix} n \\ k \end{pmatrix}$ où n et k sont fournis par l'utilisateur avec :

$$\begin{pmatrix} n \\ k \end{pmatrix} = \frac{n!}{(n-k)!\,k!}$$

```
Program Combinaison;
VAR k, n, Fn, Fk, Fnk, i : integer ;
VAR C : real;
Begin
     writeln('Donner n');
     readln(n);
     writeln('Donner k');
     readln(k);
     {Calcul du factoriel de k}
     {Initialisation du factoriel}
     Fk:=1;
     {Calcul de k!}
     for i:=1 to k do begin
         Fk:=Fk*i;
     end;
     {Calcul du factoriel de n}
     {Initialisation du factoriel}
     Fn:=1;
     {Calcul de n!}
     for i:=1 to n do begin
         Fn:=Fn*i;
     end;
     {Calcul du factoriel de n-k}
     {Initialisation du factoriel}
     Fnk:=1;
     {Calcul de (n-k)!}
     for i:=1 to (n-k) do begin
         Fnk:=Fnk*i;
     end;
     {Calcul de C}
     C:= Fn / (Fnk*Fk);
     writeln(C);
     readln;
End.
```

Exercice 2 (Correction)

Ecrire un programme qui affiche à l'écran une simulation de 20 lancers de dés.

```
Program DES;
VAR R:integer;VAR k:integer;
Begin
     randomize;
     for k:=1 to 20 do begin
         R:= random(6)+1;
         write('Jet ',k,' : ');
         writeln(R);
     end;
     readln;
End.
```

INSTRUCTION FOR - TO - DO

Exercice 1

Compléter le programme qui fournit tous les termes de la suite u jusqu'au rang N choisi par l'utilisateur avec : $u_0 = \frac{1}{2}$,

$$\forall n \in \mathbb{N},\ u_{n+1} = u_n + 3^n$$

```
Program Suites;
VAR N,k:integer;
VAR A,B,P:real;
Begin
     {Saisie de N}
     writeln('Donner N.');
     readln(N);
     {Initialisation}
     A:=...................
     P:=1;
     writeln('Le terme de rang 0 est ',A);
     {Calcul et affichage des termes}
     for k:=1 to N do begin
         B:=.....................
         write('Le terme de rang ',k);
         writeln(B);
         A:=.....................
         P:=.....................
     end;
     readln;
End.
```

Exercice 2

Ecrire un programme qui fournit tous les termes de la suite u jusqu'au rang N fourni par l'utilisateur avec : $u_0 = \frac{1}{2}, \qquad u_1 = 1$

$$\forall n \in \mathbb{N},\ u_{n+2} = u_{n+1} - u_n + \left(\frac{1}{2}\right)^n$$

Exercice 3

Ecrire un programme qui fournit tous les termes de la suite u jusqu'au rang N fourni par l'utilisateur avec : $u_0 = 1, \qquad u_1 = 3$

$$\forall n \in \mathbb{N},\ u_{n+2} = u_{n+1} + u_n - 2^{n+4}$$

Travaux Pratiques d'Informatique 15

CORRECTION DU TP 14

Exercice 1

Copmpléter le programme qui fournit tous les termes de la suite u jusqu'au rang N choisi par l'utilisateur avec : $u_0 = \frac{1}{2}$ et $\forall n \in \mathbb{N},\ u_{n+1} = u_n + 3^n$.

```
Program Suites;
VAR N,k:integer;
VAR A,B,C,P:real;
Begin
     {Saisie de N}
     writeln('Donner N.');
     readln(N);
     {Initialisation}
     A:=1/2;
     P:=1;
     writeln('Le terme de rang 0 est ',A);
     {Calcul et affichage des termes}
     for k:=1 to N do begin
         B:=A+P;
         write('Le terme de rang ',k);
         writeln(B);
         A:=B;
         P:=3*P;
     end;
     readln;
End.
```

Exercice 2 (Correction)

Ecrire un programme qui fournit tous les termes de la suite u jusqu'au rang N fourni par l'utilisateur avec : $u_0 = \frac{1}{2}$, $u_1 = 1$ et $\forall n \in \mathbb{N},\ u_{n+2} = u_{n+1} - u_n + \left(\frac{1}{2}\right)^n$.

```
Program Suites;
VAR N,k:integer;
VAR A,B,C,P:real;
Begin
     {Saisie de N}
     writeln('Donner N.');
     readln(N);
     {Initialisation}
     A:=1/2;
     B:=1;
     P:=1;
     writeln('Le terme de rang 0 est ',A);
     writeln('Le terme de rang 1 est ',B);
     {Calcul et affichage des termes}
     for k:=2 to N do begin
         C:=B-A+P;
         write('Le terme de rang ',k);
         writeln(C);
         A:=B;
         B:=C;
         P:=P/2;
     end;
     readln;
End.
```

FONCTIONS

Les fonctions sont des morceaux de code qui peuvent être appelés d'à peu près n'importe quel endroit du programme. Une fonction renvoit une valeur.

Exemple de programme avec deux fonctions

```
program TOTO;
var A,B,C : Real ;
var N : integer;

function machin(MA, MB : Real) : Real;
begin
   machin := MA+2*MB;
end;

function truc(TA : real ; TN : integer) : Real;
var k : integer;
var res : real;
begin
   res := 1;
   for k:= 1 to TN do begin
      res := res*TA;
   end;
   truc := res;
end;

BEGIN
   writeln('Donner N');
   readln(N);
   A:= 3;
   writeln('Donner B');
   readln(B);
   C:= machin(A,B)+truc(A,N);
   writeln(C);
   readln;
END.
```

Variables globales et variables locales

Une variable globale est une variable déclarée au début du programme, accessible par toutes les fonctions du programme.
Une variable locale est une variable déclarée au début d'une fonction, elle n'est accessible que par cette fonction.
Attention ! Dans les fonctions, les variables sont toujours réinitialisées à chaque appel de la dite fonction.

Architecture du programme

```
program nom;
const const1 : Type = valeur;
{Variables globales}
var gVar1 : Type;
var gVar2 : Type;

function function1(Arg1,Arg2) : Type;
var pVar1 : Type;
    pVar2 : Type;
begin
  [Instructions]
end;

BEGIN       {Procédure principale}
  [Instructions]
END.
```

Exercice 1

Utiliser une fonction qui calculera un factoriel pour obtenir un programme qui calcule une combinaison.

Exercice 2

Utiliser une fonction qui calculera une puissance pour obtenir un programme qui calcule $P(x) = 3x^5 - 6x^4 + 2x^3 - 8x^2$ pour x choisi par l'utilisateur.

Travaux Pratiques d'Informatique 16

CORRECTION DU TP 15

Exercice 1 (Correction)

```
PROGRAM Combinaison;

VAR N,K : integer;
VAR A,B,C : integer;
VAR Cb : integer;

Function factoriel(p : integer):integer;
VAR j,res : integer;
Begin
   res:=1;
   for j:=1 to p do begin
      res:=res*j;
   end;
   factoriel:=res;
End;

BEGIN
  writeln('Donner N.');
  readln(N);
  writeln('Donner K.');
  readln(K);
  A:= factoriel(N);
  B:= factoriel(K);
  C:= factoriel(N-K);
  Cb:= A div (B*C);
  writeln('Combi ',K,' parmi ',N,' : ',Cb);
  readln;
END.
```

FONCTIONS

Exercice 1

Ecrire un programme, contenant une fonction factoriel, qui calcule un arrangement.
Indication : l'en-tête de la fonction serait :

```
function factoriel(fn:integer) : integer;
```

Exercice 2

Ecrire un programme, contenant une fonction puissance et une fonction factoriel, qui calcule tous les termes de la suite u jusqu'au rang choisi par l'utilisateur.

$$\forall n \in \mathbb{N}^*,\ u_n = \frac{3^n}{n!}$$

Indication : l'architecture du programme serait :

```
PROGRAM Suite;

VAR N,K : integer;
VAR U : real;

Function factoriel(p : integer):integer;
...

Function puissance...


BEGIN
  writeln('Donner N.');
  readln(N);
  for k:=1 to N do begin
     U:=...
     writen('Terme de rang ',k,' : ',U);
  end;
  readln;
END.
```

REVISIONS

Exercice 3

Ecrire un programme qui affiche des nombres au hasard :

1. Un réel compris ente 0 et 1
2. Un réel compris ente -5 et 0
3. Un réel compris ente 3 et 7
4. Un entier compris ente 0 et 4
5. Un entier divisible par 3 et compris ente 6 et 27

Exercice 4

Ecrire un programme qui fournit tous les termes de la suite u jusqu'au rang choisi par l'utilisateur.

$$\forall n \in \mathbb{N}^*,\ u_n = n!$$

Travaux Pratiques d'Informatique 17

CORRECTION DU TP 16

Exercice 1

```
PROGRAM Arrangement;

VAR N,K : integer;
VAR A,B : integer;
VAR Ar : integer;

Function factoriel(p : integer):integer;
VAR j,res : integer;
Begin
   res:=1;
   for j:=1 to p do begin
      res:=res*j;
   end;
   factoriel:=res;
End;

BEGIN
  writeln('Donner N.');
  readln(N);
  writeln('Donner K.');
  readln(K);
  A:= factoriel(N);
  B:= factoriel(N-K);
  Ar:= A div B;
  writeln('Arrangement ',K,' parmi ',N,' : ',Ar);
  readln;
END.
```

Exercice 2

```
PROGRAM Suite;

VAR N,K : integer;
VAR U : real;

Function factoriel(p : integer):integer;
VAR j,res : integer;
Begin
   res:=1;
   for j:=1 to p do begin
      res:=res*j;
   end;
   factoriel:=res;
End;

Function puissance(x,p):real;
VAR j : integer;
VAR res : real;
Begin
   res := 1;
   for j:=1 to p do begin
       res := res*x;
   end;
   puissance := res;
End;

BEGIN
  writeln('Donner N.');
  readln(N);
  for k:=1 to N do begin
     U:=puissance(3,k)/factoriel(k);
     writen('Terme de rang ',k,' : ',U);
  end;
  readln;
END.
```

SIMULATION DE JEUX

Voici un programme qui simule un lancer de dé.

```
Program Jeu;
Var de : integer;
Begin

   de := random (6)+1;
   writeln('Le lancer de dé donne : ', de);
   readln;
End.
```

Exercice 1

Pile ou Face

Ecrire un programme qui simule un jeu de pile ou face (0 ou 1).

Exercice 2

Tirages dans une urne

On effectue un tirage au hasard dans une urne contenant des boules : 5 boules rouges et 3 boules vertes.

1. Ecrire un programme qui simule le tirage d'une boule.
2. Soit X la VAR valant 1 si au cours du tirage la boule est verte et 0 sinon. Compléter le programme précédent pour qu'il renvoie la valeur de X.
3. Ecrire un programme qui simule le tirage de 10 boules avec remise.
4. Soit Y la VAR égale au nombre de boules vertes obtenues au cours de 10 tirages **avec remise**. Compléter le programme précédent pour qu'il renvoie la valeur de Y.
5. Voici un programme :

```
Program jeu;
Var X,V,R : integer;
Begin
   randomize;
   V:=3;
   R:=5;
   X:=0;
   for i:=1 to 4 do begin
     a := random;
     if a<=(V/(R+V)) then begin
         X:=X+1;
         V:=V-1;
     end;
     else
         R:=R-1;
   end;
   writeln(X);
   readln;
```

Que représente la VAR X ?

Travaux Pratiques d'Informatique 18

Lois usuelles

Le but n'est pas de donner la loi de probabilité d'une VAR mais de donner une simulation d'une expérience dont la VAR étudiée suit une loi donnée. Autrement dit Bernoulli(p) fournira 1 avec la probabilité p et 0 avec la probabilité $1-p$.

```
program simulations;
uses crt;
var p : real;

function BERNOULLI(proba:real):integer;
var a: real;
begin
     a := random;
     if a<=proba then BERNOULLI:=1;
     if a>proba then BERNOULLI:=0;
end;

BEGIN
    clrscr;
    randomize;
    writeln('Donner p');
    readln(p);
    writeln(BERNOULLI(p));
    readln;
END.
```

1. Ecrire une fonction qui simule une loi uniforme sur $[\![1;n]\!]$ avec n comme paramètre d'entrée.
2. Voici un programme :

```
program simulations;
uses crt;
var p : real;
var n:integer;

function TRUC(proba:real, nb : integer):real;
var a: real;
var k,res : integer;

begin
     res:=0;
     for k:=1 to nb do begin
         a := random;
         if a<=proba then res:= res+1;
     end;
     TRUC:=res;
end;

BEGIN
    clrscr;
    randomize;
    writeln('Donner p');
    readln(p);
    writeln('Donner n');
    readln(n);
    writeln(TRUC(p,n));
    readln;
END.
```

Quelle est la loi simulée par la fonction TRUC ?

Lancements d'une pièce

On lance 100 fois une pièce truquée dont la probabilité de donner Pile est 0,3.

1. Quelle est la loi du nombre de "Pile" obtenus ? Quelle est le nombre de "Pile" obtenus en moyenne.
2. Ecrire un programme qui simule une telle série de lancers et qui affiche le nombre de Pile obtenus.

Tirages sans remise

On effectue un tirage au hasard dans une urne contenant des boules : 5 boules rouges et 8 boules vertes.

1. Ecrire une fonction qui simule le jeu suivant : on tire 6 boules **sans remise**, et on compte le nombre de boules vertes obtenues, on notera X cette VAR.
2. Quelle est la loi suivie par X ?
3. Quelle est l'espérance de X ?

Travaux Pratiques d'Informatique 19

SIMULATION DES LOIS USUELLES

1. Simulation de Bernoulli(proba)

```
function BERNOULLI(proba:real):real;
var a: real;
begin
     a := random;
     if a<=proba then BERNOULLI:=1;
     if a>proba then BERNOULLI:=0;
end;
```

2. Simulation de Uniforme(n)

```
function UNIFORME(n : integer):integer;
begin
     UNIFORME:=random(n)+1;
end;
```

3. Simulation de Binomiale(n,p)

```
function BINOMIALE(p:real;n:integer):real;
var a: real;
var k,res : integer;
begin
     res:=0;
     for k:=1 to n do begin
         a := random;
         if a<=p then res:= res+1;
     end;
     BINOMIALE:=res;
end;
```

4. Hypergéométrique(Nb,nt,p)

On prend $M = Nb * p$ le nombre d'éléments de type 1, Nb le nombre total d'éléments et nt le nombre de tirages.

```
function HYPER(Nb,nt,M:integer):real;
var a: real;
var k,res : integer;
begin
     res:=0;
     for k:=1 to nt do begin
         Nb:=Nb-1;
         a := random;
         if (a<=(M/Nb)) then begin
            res:= res+1;
            M:=M-1;
         end;
     end;
     HYPER:=res;
end;
```

STRUCTURE DE CONTROLE : BOUCLES

Soit (u_n) la suite définie par $u_0 = 1$ et $u_{n+1} = u_n - u_n^2$.
On peut démontrer que cette suite converge vers 0.
Nous allons utiliser cette suite dans différents programmes.

1. **BOUCLE FOR TO - DO**
   Elle permet de répéter des instructions un nombre de fois déterminé.
2. **BOUCLE FOR DOWNTO - DO**

```
Program suite;
VAR A,B:real;
VAR n,k : integer;
Begin
     A:=1;
     n:=100;
     for k:=n downto 1 do begin
          B:=A-sqr(A);
          A:=B;
     end;
     writeln(A);
     readln;
END.
```

Ce programme calcule les termes de la suite u du rang 1 au rang 100 mais en faisant décroître le compteur.

3. **BOUCLE WHILE - DO**
   Elle permet de répéter des instructions tant qu'une condition reste vraie.

```
Program suite;
VAR A,B:real;
VAR n : integer;
Begin
     A:=1;
     N:=0;
     while (A>0.01) do begin
          B:=A-sqr(A);
          N:=N+1;
          A:=B;
     end;
     writeln(N);
     readln;
END.
```

Ce programme calcule les termes de la suite u tant que ceux-ci sont supérieurs à 0,01.
Il faut noter que si la condition est fausse dès le début, l'instruction n'est jamais exécutée.

4. **BOUCLE REPEAT - UNTIL**
   Elle permet de répéter des instructions jusqu'à ce qu'une condition soit vraie.

```
Program suite;
VAR A,B:real;
VAR N : integer;
Begin
     A:=1;
     N:=0;
     repeat
          B:=A-sqr(A);
          N:=N+1;
          A:=B;
     until (A=<0.01);
     writeln(N);
     readln;
END.
```

Ce programme calcule les termes de la suite u jusqu'à ce que ceux-ci soient inférieurs ou égaux à 0,01.
Il faut noter que si la condition est vraie dès le début, elles sont au moins exécutées une fois.

Exercice 1

Ecrire un programme qui fournit tous les termes de la suite u définie par $\forall n \in \mathbb{N}^*,\ u_n = \frac{1}{n}$, jusqu'à ce que les termes deviennent inférieurs à $0,01$.

Exercice 2

Ecrire un programme qui simule des lancers successifs d'un dé jusqu'à obtenir le premier 6. On affichera le rang d'apparition de ce 6.

Travaux Pratiques d'Informatique 20

Correction de l'exercice du TP 19

Exercice 1

Ecrire un programme qui fournit tous les termes de la suite u jusqu'à ce que les termes deviennent inférieurs à $0,01$.

$$\forall n \in \mathbb{N}^*,\ u_n = \frac{1}{n}$$

Remarquez que cette suite est positive, décroissante et converge vers 0. On utilisera les boucles "for to do", "while do" et "repeat until".

1. **BOUCLE WHILE - DO**

```
Program suite;
VAR U :real;
VAR k : integer;
Begin
     U:=1;
     k:=1;
     while (U>=0.01) do begin
          U:=1/k;
          k:=k+1;
          writeln(U);
     end;
     readln;
END.
```

2. **BOUCLE REPEAT - UNTIL**

```
Program suite;
VAR U :real;
VAR k : integer;
Begin
     k:=0;
     repeat
          k:=k+1;
          U:=1/k;
          writeln(U);
     until (U<0.01);
     readln;
END.
```

Exercices du TP 20 : Boucles

Exercice 1

```
Program suite;
VAR A,B,C,epsilon:real;
Begin
     A:=1/2;
     writeln(A);
     repeat
          B:=A-sqr(A);
          C:=abs(B-A);
          write('A= ',A);
          write('B= ',B);
          write('C= ',C);
          A:=B;
          writeln(A);
     until (C<=0.04);
     readln;
END.
```

Que fait ce programme ? On pourra compléter le tableau suivant à l'aide de l'ordinateur :

Etape	A	B	C
Initialisation			
Boucle			

Exercice 2

Ecrire un programme qui fournit tous les termes de la suite u jusqu'à ce que l'écart entre deux termes consécutifs soit inférieur à une quantité *epsilon* choisie par l'utilisateur.

$$u_0 = \frac{3}{2},\ \forall n \in \mathbb{N},\ u_{n+1} = (u_n + 1)^2 e^{-u_n}$$

Exercice 3

Ecrire un programme qui simule une loi binomiale de paramètre $\frac{2}{3}$ et de taille n choisie par l'utilisateur.

Exercice 4

Ecrire un programme qui simule le tirage de 3 boules successives sans remise dans une urne contenant 6 noires et 2 blanches et comptant le nombre de boules blanches obtenues.

Travaux Pratiques d'Informatique 21

Correction de l'exercice du TP 20

Exercice 2

Ecrire un programme qui fournit tous les termes de la suite u jusqu'à ce que l'écart entre deux termes consécutifs soit inférieur à une quantité *epsilon* choisie par l'utilisateur.

$$u_0 = \frac{3}{2},\ \forall n \in \mathbb{N},\ u_{n+1} = (u_n + 1)^2 e^{-u_n}$$

BOUCLE WHILE - DO

```
Program suite;
VAR A,B,C,epsilon :real;
Begin
     writeln('Donner epsilon');
     readln(epsilon);
     A:=1.5;
     writeln(A);
     B:=sqr(A+1)*exp(-A);
     C:=abs(B-A);
     A:=B;
     while (C>epsilon) do begin
          B:=sqr(A+1)*exp(-A);
          C:=abs(B-A);
          A:=B;
          writeln(A);
     end;
     readln;
END.
```

BOUCLE REPEAT - UNTIL

```
Program suite;
VAR A,B,C,epsilon:real;
Begin
     writeln('Donner epsilon');
     readln(epsilon);
     A:=1.5;
     writeln(A);
     repeat
          B:=sqr(A+1)*exp(-A);
          C:=abs(B-A);
          A:=B;
          writeln(A);
     until (C<=epsilon);
     readln;
END.
```

STRUCTURE DE CONTROLE : BOUCLES

Exercice 1

Soit $a \in R^{*+}$, et

$$f_a : x \mapsto \frac{1}{2}\left(x + \frac{a}{x}\right)$$

Soit (u_n) définie par $u_0 > 0$ et $\forall n \in \mathbb{N}$,

$$u_{n+1} = f_a(u_n)$$

On peut démontrer que (u_n) converge vers $\sqrt{a}$. (Ecricome 2007)

On peut aussi démontrer que :

$$\forall n \in \mathbb{N},\ |u_n - \sqrt{a}| \leq \left(\frac{1}{2}\right)^n |u_0 - \sqrt{a}|$$

1. Ecrire un programme qui affiche les 20 premiers termes de la suite u, de premier terme 1 et convergeant vers $\sqrt{2}$.
2. Décrire ce que fait le programme suivant :

```
Program TOTO;
VAR u,p :real;
Begin
     u:=1;
     p:=1;
     while (p>0,01) do begin
          p:=p/2;
          u:=(u+3/u)/2;
     end;
     writeln(u);
     readln;
END.
```

Compléter le tableau suivant :

Etape	u	p
Initialisation		
Boucle		
	$1,73$	
	$1,732$	
	$1,73205$	
	$1,7320508$	

Le programme affiche une valeur approchée à $0,01$ près de $\sqrt{\cdots}$.

3. Ecrire un programme qui donne une valeur approchée à 10^{-3} près de $\sqrt{5}$.

Travaux Pratiques d'Informatique 22

CORRECTION DU TP 21

Exercice 1

1. Ecrire un programme qui affiche les 20 premiers termes de la suite u, de premier terme 1 et convergeant vers $\sqrt{2}$.

```
Program suite;
VAR A, B : real;
VAR k : integer;
Begin
     A := 1;
     writeln(A);
     for k := 1 to 19 do begin
          B := (A+2/A)/2;
          A := B;
          writeln(A);
     end;
     readln;
END.
```

2. Le programme TOTO génère les résultats suivants :

Etape	u	p
Initialisation	1	1
Boucle	2	$\frac{1}{2}$
	$\frac{7}{4}$	$\frac{1}{4}$
	$\frac{97}{56}$	$\frac{1}{8}$
	~ 1.732	$\frac{1}{16}$
	~ 1.732	$\frac{1}{32}$
	~ 1.732	$\frac{1}{64}$
	~ 1.732	$\frac{1}{126}$

Le programme affiche une valeur approchée à $0,01$ *près de* $\sqrt{3}$.

3. Ecrire un programme qui donne une valeur approchée à 10^{-3} près de $\sqrt{5}$.
D'après l'inégalité, on a $\forall n \in \mathbb{N},\ |u_n - \sqrt{5}| \leq \left(\frac{1}{2}\right)^n |1 - \sqrt{5}|$
or $|1 - \sqrt{5}| < 2$, dès lors $\forall n \in \mathbb{N},\ |u_n - \sqrt{5}| \leq \left(\frac{1}{2}\right)^n \times 2$.
Au rang 0, le majorant vaut 2. Au rang 1, il vaut $\left(\frac{1}{2}\right) \times 2$. Au rang 2, il vaut $\left(\frac{1}{2}\right)^2 \times 2$. On passe d'un rang au suivant en multipliant le majorant par $\left(\frac{1}{2}\right)$.
Le calcul des termes de la suite se fera tant que le majorant sera supérieur à 0,001.
D'autre part, $\forall n \in \mathbb{N}$, $u_{n+1} = f(u_n)$ avec $f : x \mapsto \frac{1}{2}\left(x + \frac{5}{x}\right)$.

```
Program TOTO;
VAR u,p :real;
Begin
     u:=1;
     p:=2;
     while (p>0,001) do begin
          p:=p/2;
          u:=(u+5/u)/2;
     end;
     writeln(u);
     readln;
END.
```

STRUCTURE DE CONTROLE : BOUCLES

BOOLEENS

Dans une boucle "repeat... until", on effectue un test d'arrêt qui provoque l'arrêt de la boucle si le test rend la réponse VRAIE et poursuit la boucle si le test rend la réponse FAUX. Le résultat d'un test (VRAI ou FAUX) est un résultat dit "Booléen".
On peut mettre en place dans un programme une variable booléenne :

- Déclaration : VAR liste de variables : BOOLEAN ;
- Ces variables peuvent prendre soit la valeur TRUE (vrai), soit la valeur FALSE (faux).
- Opérations booléennes : AND, OR, NOT, XOR (ou exclusif). Ces opérations nécessitent des arguments booléens.
- Opérations à valeur booléenne : > (supérieur), < (inf), >= (sup ou égal), <=, = (égal),<>(différent). Ces opérations comparent tous éléments de type simple (les 2 arguments doivent être de même type, sauf entiers et réels qui peuvent être comparés entre eux), et renvoient un booléen.
- AND (et), OR (ou), NOT (non), sont de priorité supérieure aux précédents et ne peuvent opérer que sur des booléens : $A > B$ et C doit être écrit : $(A > B) and (A > C)$. Les parenthèses sont obligatoires pour ne pas faire en premier B and A.

Exercice 1 (HEC II 2004)

Deux joueurs J et J' s'affrontent dans un jeu utilisant le lancer d'une pièce équilibrée avec les règles suivantes :

- le joueur J est gagnant si la configuration " pile, pile, face " apparaît dans la suite des résultats des lancers, avant que la configuration " face , pile, pile " n'apparaisse ;
- le joueur J' est gagnant si la configuration " face, pile, pile " apparaît dans la suite des résultats des lancers, avant que la configuration " pile, pile, face " n'apparaisse ;

On considère le programme PASCAL suivant :

```
PROGRAMME Quigagne;

VAR x,y,r,k :INTEGER;

BEGIN
   x := 0;
   y := 0;
   k := 0;
   WHILE ((x<3) AND (y<3)) DO BEGIN
      k := k+1;
      r := RANDOM(2);
      IF r=1 THEN BEGIN
         IF x>=1 THEN x := 2 ELSE x :=1;
         IF y>=1 THEN y :=y+1;
      END;
      ELSE BEGIN
         IF x = 2 THEN x := 3 ELSE x := 0;
         y :=1;
      END;
   END;
   IF x=3 THEN WRITE (' ......... ') ELSE WRITE('.......');
END.
```

1. Donner sous forme d'un tableau les valeurs successives prises par les variables x, y et k lors de l'exécution de ce programme, si les valeurs données à la variable r par la fonction " RANDOM(2)" sont successivement :
   (a) 1, 1, 1, 1, 0
   (b) 1, 0, 1, 0, 0, 0, 1, 1
   (c) 0, 1, 0, 1, 0, 1, 1
2. Que représente la dernière valeur prise dans le programme par la variable k et quels textes pourrait-on substituer aux pointillés de la dernière instruction ?
   Qu'afficherait alors l'ordinateur dans les trois exemples de la question précédente ?

Travaux Pratiques d'Informatique 23

Correction de l'exercice du TP 22

Exercice 1 (HEC II 2004)

1. Les tableaux :

   (a) 1, 1, 1, 1, 0

Etape	r	x	y	k
Initialisation		0	0	0
Boucle	1	1	0	1
	1	2	0	2
	1	2	0	3
	1	2	0	4
	0	3	1	5

   (b) 1, 0, 1, 0, 0, 0, 1, 1

Etape	r	x	y	k
Initialisation		0	0	0
Boucle	1	1	0	1
	0	0	1	2
	1	1	2	3
	0	0	1	4
	0	0	1	5
	0	0	1	6
	1	1	2	7
	1	2	3	8

   (c) 0, 1, 0, 1, 0, 1, 1

Etape	r	x	y	k
Initialisation		0	0	0
Boucle	0	0	1	1
	1	1	2	2
	0	0	1	3
	1	1	2	4
	0	0	1	5
	1	1	2	6
	1	2	3	7

2. La dernière valeur prise dans le programme par la variable k est le nombre de lancers effectués lors de la partie. On pourrait substituer aux pointillés de la dernière instruction :

```
IF x=3
 THEN WRITE ('J a gagné.')
 ELSE WRITE('J' a gagné.');
```

   Ici, J a gagné la première partie et J' les deux suivantes.

SOMMES

Exercice 1

Que fait le programme suivant ? On complétera le tableau suivant sans utiliser l'ordinateur.

```
Program BOUCLE;
var k : integer;
var u, S : real;
begin
     S:=0;
     for k:=4 to 13 do begin
        u:=2*k;
        S:=S+u;
        write('k = ', k);
        write('u= ', u);
        writeln('S= ', S);
     end;
     readln;
end.
```

Etape	k	u	S
Initialisation			
Boucle			

Que représente la dernière valeur de S ?

Exercice 2

Ecrire un programme qui calcule, pour N choisi par l'utilisateur ($N \geq 3$), la somme suivante :

$$\sum_{k=3}^{N} k^2$$

Il nous faudra 4 variables :

- N entier choisi par l'utilisateur.
- Le compteur k qui est un entier variant de 3 à N.
- La variable réelle u qui recevra le terme courant de rang k dans le calcul de la somme.
- La variable réelle S qui recevra la valeur de la somme.

On pourra compléter le tableau suivant en prenant $N = 8$:

Etape	N	k	u	S
Initialisation				
Boucle				

Exercice 3

Ecrire un programme qui calcule, pour N choisi par l'utilisateur, la somme suivante :

$$\sum_{k=0}^{N} \frac{k+1}{k^2+1}$$

Travaux Pratiques d'Informatique 24

REVISIONS : PETIT MEMO

Quelques morceaux de programmes remarquables :

1. **Fonction factoriel**

```
Function factoriel(p : integer):integer;
VAR j,res : integer;
Begin
   res:=1;
   for j:=1 to p do begin
      res:=res*j;
   end;
   factoriel:=res;
End;
```

2. **Fonction puissance**

```
Function puissance(x:real,p:integer):real;
VAR j : integer;
VAR res : real;
Begin
   res := 1;
   for j:=1 to p do begin
       res := res*x;
   end;
   puissance := res;
End;
```

3. **Suite définie explicitement**

```
for k:= 1 to N do begin
   U:= "expression en k"
end;
```

4. **Suite définie par une formule de récurrence d'ordre 1**

```
"Initialisation de A";
for k:= 1 to N do begin
   B:="expression en A";
   A:=B;
end;
```

5. **Suite définie par une formule de récurrence d'ordre 2**

```
"Initialisation de A";
"Initialisation de B";
for k:= 2 to N do begin
   C:="expression en A et B";
   A:=B;
   B:=C;
end;
```

6. **Calcul d'une somme**

```
S:=0;
for k:= 0 to N do begin
   S:= S + "terme de rang k";
end;
```

7. **repeat...until** Soit (u_n) définie par la donnée de $u_0 > 0$ et $\forall n \in \mathbb{N},\ u_{n+1} = f(u_n)$. On obtient, par exemple, au cours de l'exercice :

$$\forall n \in \mathbb{N},\ |u_n - \alpha| \leq \left(\frac{1}{2}\right)^{n+1}$$

On peut alors écrire un programme qui affiche une valeur approchée à 10^{-3} près de α.

```
Program TOTO;
VAR A,B,P :real;
Begin
     A:=1.5;
     P:=0.5;
     repeat
          p:=p/2;
          B:="expression en A";
          A:=B;
     until(P<0.001);
     writeln(A);
     readln;
END.
```

8. **Et lois U, B, H**

Exercice 1

Ecrire un programme qui fournit tous les termes de la suite u jusqu'au rang choisi par l'utilisateur.

$$\forall n \in \mathbb{N},\ u_n = 3n^2 + 4n + 1$$

Exercice 2

Ecrire un programme qui fournit tous les termes de la suite u jusqu'au rang choisi par l'utilisateur.

$$\forall n \in \mathbb{N},\ u_{n+1} = 3u_n^2 + 4u_n + 1 \text{ et } u_0 = 1$$

Exercice 3

Ecrire un programme qui fournit tous les termes de la suite u jusqu'au rang choisi par l'utilisateur.

$$\forall n \in \mathbb{N},\ u_{n+2} = 3u_{n+1}^2 + 4u_n + 1 \text{ et } u_0 = 1,\ u_1 = 0$$

Exercice 4

Ecrire un programme qui calcule, pour n choisi par l'utilisateur, la somme suivante :

$$\sum_{k=0}^{n} \frac{k}{k+1} \qquad \sum_{k=0}^{n} \frac{1}{k!}$$

Exercice 5

Dans les fiches précédentes, vous retrouverez les informations qui répondent aux questions suivantes ?

1. Quel est le résultat de

   `trunc(4.3)`

2. Quel est le résultat de

   `23 div 5`

3. Quel est le résultat de

   `23 mod 5`

4. Quel est le résultat de

   `4*random(5)+3`

5. Quel est le résultat de

   `4*random+3`

Travaux Pratiques d'Informatique 25

Correction de l'exercice du TP 24

Exercice 5 (Correction)

1. "trunc(4.3)" donne la partie entière de 4.3 soit 4.
2. "24 div 5" donne le résultat de la division euclidienne de 24 par 5, autrement dit 4.
3. "24 mod 5" donne le reste de la division eculidienne de 24 par 5, autrement dit 4.
4. "4*random(5)+3" donne un nombre au hasard parmi 3, 7, 11, 15 et 19.
5. "4*random+3" donne un nombre réel au hasard entre 3 et 7.

PROCEDURES

Les procédures comme les fonctions sont des morceaux de code qui peuvent être appelés d'à peu près n'importe quel endroit du programme. Une procédure se distingue d'une fonction par le fait qu'elle ne renvoit pas de valeur.
Créons une procédure qui calculera la résistance d'un composant électrique en étant en connaissance de la tension et de l'intensité. Les arguments à transmettre seront donc : une variable I pour l'intensité, et une variable U pour la tension. Nous appellerons la procédure LoiOhm.

```
program elec;
var Res : Real;

procedure LoiOhm(I, U, R : Real);
begin
  R := U / I;
end;

BEGIN
  writeln('Avant -> Res = ', Res);     { Sortie -> 0 }
  LoiOhm(15, 250, Res);
  writeln('Après -> Res = ', Res);    { Sortie -> 0 }
END.
```

Vous remarquez que la sortie est toujours égale à 0. Pourquoi ? Il faut spécifier que R est une variable dans la procédure, à l'aide du mot-clef var. Notre exemple ainsi modifié devient :

```
program elec;
var Res : Real;

procedure LoiOhm(I, U : Real; var R : Real);
begin
  R := U / I;
end;

BEGIN
  writeln('Avant -> Res = ', Res);     { Sortie -> 0 }
  LoiOhm(15, 250, Res);
  writeln('Après -> Res = ', Res);    { Sortie -> 16.66..}
END.
```

Architecture du programme

Il est temps de dresser un squelette d'un programme, avec nos connaissances actuelles :

```
program nom;
const const1 : Type = valeur;
{Variables globales}
var gVar1 : Type;
var gVar2 : Type;

procedure procedure1(Arg1,Arg2);
{Variables locales}
var pVar1 : Type;
var pVar2 : Type;
begin
  [Instructions]
end;

function function1(Arg1,Arg2) : Type;
var pVar1 : Type;
    pVar2 : Type;
begin
  [Instructions]
end;

BEGIN       {Procédure principale}
  [Instructions]
END.
```

REVISIONS : EXERCICES

Exercice 1

Ecrire un programme qui fournit tous les termes de la suite u jusqu'au rang N fourni par l'utilisateur avec $u_0 = 3$, $u_1 = 3$ et :

$$\forall n \in \mathbb{N},\ u_{n+2} = \frac{1}{n^2+1} u_{n+1} \times u_n$$

Exercice 2

Ecrire un programme qui calcule, pour n choisi par l'utilisateur, la somme suivante :

$$\sum_{k=1}^{n} (k + \ln(k))$$

Exercice 3

Ecrire un programme qui simule une loi binomiale de paramètre $\frac{1}{2}$ et de taille n choisie par l'utilisateur.

Exercice 4

Ecrire un programme qui simule une loi uniforme sur $[\![3; 10]\!]$.

Exercice 5

Ecrire un programme qui calcule le terme de rang n de la suite u définie pour tout n entier par $u_n = 3n!$.

Exercice 6

Ecrire un programme qui calcule tous les termes jusqu'au rang 20 de la suite u définie pour tout n entier par $u_{n+1} = 3u_n + (n-1)^2$ avec $u_0 = 1$.

Exercice 7

Ecrire un programme qui simule le tirage de 5 boules successives sans remise dans une urne contenant 8 noires et 6 blanches et comptant le nombre de boules noires obtenues.

Travaux Pratiques d'Informatique 26

REVISIONS : EXERCICES

Exercice 1

Ecrire un programme qui fournit tous les termes de la suite u jusqu'au rang N fourni par l'utilisateur avec $u_0 = 1$, $u_1 = 2$ et :

$$\forall n \in \mathbb{N},\ u_{n+2} = 3u_{n+1} + nu_n$$

Exercice 2

Ecrire un programme qui fournit tous les termes de la suite u jusqu'au rang N fourni par l'utilisateur avec $u_0 = 1$ et :

$$\forall n \in \mathbb{N},\ u_{n+1} = \frac{u_n}{3^n}$$

Exercice 3

Ecrire un programme qui calcule, pour N choisi par l'utilisateur, la somme suivante :

$$\sum_{k=1}^{N} \frac{k^2+1}{k+1}$$

Exercice 4

Ecrire un programme qui calcule, pour N choisi par l'utilisateur, la somme suivante :

$$\sum_{k=1}^{N} k!$$

Exercice 5

Ecrire un programme qui calcule, pour N choisi par l'utilisateur, la somme suivante :

$$\sum_{k=1}^{N} \frac{k}{2^k}$$

Exercice 6

Ecrire un programme qui simule une loi binomiale de paramètre $\frac{1}{3}$ et de taille N choisie par l'utilisateur.

Travaux Pratiques d'Informatique 27

METHODE DE DICHOTOMIE

Soit f continue et strictement croissante sur $[a;b]$ et telles que $f(a) < 0$ et $f(b) > 0$.
Notons α l'unique solution dans $[a;b]$ de $f(x) = 0$.

Objectif : Déterminer un processus itératif qui permet de donner une valeur approchée de α à une précision donnée.

Méthode : On construit deux suites $(a_n)_{n\in\mathbb{N}}$ et $(b_n)_{n\in\mathbb{N}}$ telles que

$$\forall n \in \mathbb{N},\ a_n \leq \alpha \leq b_n$$

et $(b_n - a_n)_{n\in\mathbb{N}}$ soit une suite décroissante de limite nulle.

Initialisation On a $a \leq \alpha \leq b$, on prend donc $a_0 = a$ et $b_0 = b$.

Itération On part de $a_n \leq \alpha \leq b_n$.

Si $f\left(\dfrac{a_n + b_n}{2}\right) > 0$

Alors $a_{n+1} = a_n$ et $b_{n+1} = \dfrac{a_n + b_n}{2}$ (dessin :)

Sinon $a_{n+1} = \dfrac{a_n + b_n}{2}$ et $b_{n+1} = b_n$ (dessin :)

Arrêt lorsque $|b_n - a_n| <$ précision souhaitée.

Résultat les derniers termes a_n et b_n sont des valeurs approchées de α à la précision souhaitée, a_n par défaut et b_n par excès.

Remarque : Que se passe-t-il si f est décroissante ?

Exercice 1

Ecrire un programme qui utilise la méthode de dichotomie à la précision ϵ afin de déterminer l'unique solution positive de

$$x^3 + 4x^2 - 1 = 0$$

Pour justifier mathématiquement l'emploi de la dichotomie, il faut effectuer les étapes suivantes :

1. Montrer que f réalise une bijection de $\mathbb{R}^+$ sur $f(\mathbb{R}^+)$ avec 0 appartenant à $f(\mathbb{R}^+)$.
2. Noter α l'unique solution de $x^3 + 4x^2 - 1 = 0$.
3. Montrer que cette unique solution positive est comprise entre 0 et 1.
4. Expliquer le processus de construction des suites (a_n) et (b_n) en n'oubliant pas de s'appuyer sur le sens de variation de f.
5. Rappeler que $\forall n \in \mathbb{N},\ b_n - a_n = (b_0 - a_0) \times \left(\frac{1}{2}\right)^n$.
6. La suite $(b_n - a_n)$ converge vers 0.

Programmation :
On note *epsilon* la précision voulue.
A et B représentent le terme courant des suites (a_n) et (b_n).
C représente $\dfrac{a_n + b_n}{2}$.
F représente $f\left(\dfrac{a_n + b_n}{2}\right)$.

```
Program dichotomie;
VAR A,B,C,F,epsilon :real;
Begin
     writeln('Donner epsilon');
     readln(epsilon);
     A:=0;
     B:=1;
     repeat
          C:=(A+B)/2;
          F:=C*C*C+4*C*C-1;
          if F>0
          then B:=C
          else A:=C;
     until(B-A<epsilon);
     writeln(A);
     readln;
END.
```

Exercice 2

Ecrire un programme qui utilise la méthode de dichotomie à la précision $0,01$ afin de déterminer l'unique solution positive de $x^2 = 5$.

Exercice 3 (Révision)

Ecrire un programme qui fournit tous les termes de la suite u jusqu'à ce que l'écart entre deux termes consécutifs soit inférieur à 0.001.

$$u_1 = \frac{3}{2},\ \forall n \in \mathbb{N}^*,\ u_{n+1} = \sqrt{u_n} + \frac{1}{n}$$

Exercice 4 (Révision)

Ecrire un programme qui simule le tirage de 6 boules successives avec remise dans une urne contenant 4 rouges et 6 vertes et comptant le nombre de rouges obtenues.

Exercice 5 (Révision)

Même programme mais avec des tirages sans remise .

Travaux Pratiques d'Informatique 28

Correction de l'exercice du TP 27

Exercice 2

Ecrire un programme qui utilise la méthode de dichotomie à la précision ϵ afin de déterminer l'unique solution positive de

$$x^2 = 5$$

Programmation :

On note *epsilon* la précision voulue.
A et B représentent le terme courant des suites (a_n) et (b_n).
C représente $\dfrac{a_n + b_n}{2}$.
$x \mapsto x^2 - 5$ est croissante sur R^+.

On peut utiliser une fonction :

```
Program dichotomie;
VAR A,B,C,epsilon :real;

function Fonc(x:real):real;
begin
    Fonc :=x*x-5;
end;

Begin
     writeln('Donner epsilon');
     readln(epsilon);
     A:=0;
     B:=4;
     repeat
          C:=(A+B)/2;
          if Fonc(C)>0
          then B:=C
          else A:=C;
     until(B-A<epsilon);
     writeln(A);
     readln;
END.
```

Exercices du TP 28

Exercice 1

ECRICOME 2008

Soit

$$f : x \mapsto 1 + ln(x + 1)$$

et (u_n) la suite définie par $u_0 = 1$ et

$$\forall n \in \mathbb{N},\ u_{n+1} = f(u_n)$$

1. Montrer que l'équation $f(x) = x$ admet une unique solution dans $\mathbb{R}_+^*$, que l'on notera α. On admettra que $\alpha \in [1; 3]$
2. Montrer que $\forall n \in \mathbb{N}$,
$$u_n \geq 1$$
3. Appliquer à f l'inégalité des accroissements finis entre u_n et α et en déduire que $\forall n \in \mathbb{N}$:
$$|u_n - \alpha| \leq \left(\frac{1}{2}\right)^{n-1}$$
4. Ecrire un programme qui détermine le premier rang n_0 et la valeur du terme de ce rang n_0 tel que, $\forall n \geq n_0$, l'écart entre u_n et α soit inférieur à 10^{-4}.
5. Ecrire un programme qui utilise la méthode de dichotomie à la précision voulue afin de déterminer l'unique solution positive de $f(x) = x$.

Remarque : l'équation $f(x) = x$ est équivalente à $f(x) - x = 0$.

Exercice 2 (Révision)

Ecrire un programme qui fournit le terme de la suite u de rang N fourni par l'utilisateur avec $u_0 = 0$, $u_1 = 3$ et :

$$\forall n \in \mathbb{N},\ u_{n+2} = u_{n+1} + \frac{u_n}{n+1}$$

Exercice 3 (Révision)

Ecrire un programme qui fournit tous les termes de la suite u jusqu'au rang N fourni par l'utilisateur avec

$$\forall n \in \mathbb{N}^*,\ u_n = \frac{2^n}{n!}$$

Exercice 4 (Révision)

Ecrire un programme qui calcule, pour N choisi par l'utilisateur, la somme suivante :

$$\sum_{k=2}^{N} \frac{2^n}{n!}$$

Travaux Pratiques d'Informatique 29

Correction de l'exercice du TP 28

Exercice 1

Ecrire un programme qui utilise la méthode de dichotomie à la précision voulue afin de déterminer l'unique solution positive de $f(x) = x$.
Attention : la fonction $x \mapsto f(x) - x$ est décroissante sur $[1; 3]$.

```
Program dichotomie;
VAR A,B,C,F,epsilon :real;
VAR k : integer;
Begin
     writeln('Donner epsilon');
     readln(epsilon);
     A:=1;
     B:=3;
     repeat
          C:=(A+B)/2;
          F:=1+ln(C+1)-C;
          if F>0
          then A:=C;
          else B:=C;
     until(B-A<epsilon);
     writeln(A);
     readln;
END.
```

Exercices du TP 29

SCHEMA DE HÖRNER

Calcul de la valeur d'un polynôme en x_0.
Soit, par exemple, un polynôme

$$P : x \mapsto a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$$

Il faut, dans le pire des cas 4 additions et 10 multiplications pour calculer $P(x_0)$ correspondant à une valeur donnée x_0. On dit alors que le coût de cet algorithme de calcul en fonction du nombre d'opérations est de 14.

On peut diminuer considérablement ce nombre de multiplications en utilisant une écriture différente dite schéma de HÖRNER (mathématicien et physicien anglais 1786-1837) :

$$P(x) = [([a_4x + a_3)x + a_2]x + a_1)x + a_0]$$

Il n'y a plus que 4 additions et 4 multiplications!!!

Ce schéma se généralise à un polynôme de degré quelconque.

Pour un polynôme de degré n, le calcul habituel de $P(x)$ nécessite :

- n additions
- $\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$ multiplications

autrement dit le nombre d'opérations croît comme n^2.
Pour un polynôme de degré n, le schéma de Hörner nécessite seulement n additions et n multiplications, le nombre d'opérations croît comme n.
Le complexité de l'algorithme passe ainsi de n^2 à n.

Exercice 1

Compléter :

```
PROGRAM HORNER;

VAR n, k : integer;
VAR res, x, coeff : real;

BEGIN
     {Saisie du degre de P et de la valeur de x}
     writeln('Donner le degre de P :');
     readln(n);
     writeln('Donner x :');
     readln(x);

     {Calcul de res}
     res:=0;
     for k:=... downto ... do begin
         writeln('Donner le coeff de degre ',k, ':');
         readln(coeff);
         res:=...
     end;
     writeln('Donner le coeff de degré 0 :');
     readln(coeff);
     res:=res+coeff;

     {Affichage du resultat}
     writeln;
     writeln('L''image de ',x,'par P est ',res);
     readln;
END.
```

Exercice 2 (Révision)

Ecrire un programme qui simule un tirage d'un jeton contenant 8 jetons numérotés 5, 10, 15, 20, 25, 30, 35, 40.

Exercice 3 (Révision)

Ecrire un programme qui simule le tirage de 6 jetons successifs avec remise dans une urne contenant 8 jetons numérotés 5, 10, 15, 20, 25, 30, 35, 40 et comptant le nombre de jetons, dont le numéro fini par 5, obtenus.

Travaux Pratiques d'Informatique 30

LOI GEOMETRIQUE

```
function GEOMETRIQUE(p:real):real;
var a: real;
var res : integer;
begin
     res:=0;
     repeat
         a := random;
         res:= res+1;
     until(a<=p);
     GEOMETRIQUE:=res;
end;
```

REVISIONS I

Exercice 1

Ecrire un programme qui calcule les N premiers termes, avec N fourni par l'utilisateur, de la suite u :

$$u_0 = 2 \quad u_{n+1} = 3u_n + 5$$

Exercice 2

Ecrire un programme qui calcule les N premiers termes, avec N fourni par l'utilisateur, de la suite u :

$$u_0 = 2 \quad u_{n+1} = 3u_n + 5n$$

Exercice 3

Ecrire un programme qui calcule les N premiers termes, avec N fourni par l'utilisateur, de la suite u :

$$u_0 = 2 \quad u_1 = 3 \quad u_{n+2} = u_n + n - u_{n+1}$$

Exercice 4

Ecrire un programme qui calcule les N premiers termes, avec N fourni par l'utilisateur, de la suite u :

$$u_0 = 2 \quad u_1 = 3 \quad u_{n+2} = u_n + n! - u_{n+1}$$

Exercice 5

Ecrire un programme qui calcule les N premiers termes, avec N fourni par l'utilisateur, de la suite u :

$$u_n = \frac{3^n}{n!}$$

Exercice 6

Ecrire un programme qui calcule avec N fourni par l'utilisateur :

$$\sum_{k=1}^{N} \left(\frac{1}{3}\right)^k \times k^3$$

D'après ECRICOME 2006

On considère la fonction f définie pour tout réel x par : $f(x) = x + 1 + 2e^x$.
On s'intéresse à la suite $(u_n)_{n\in\mathbb{N}}$ définie par son premier terme $u_0 = -1$ et par la relation

$$\forall n \in \mathbb{N},\ u_{n+1} = u_n - \frac{f(u_n)}{f'(u_n)}$$

On peut montrer que $0 \leqslant u_n - \alpha \leqslant \dfrac{1}{e^{2^n-1}}$.
Ecrire un programme permettant, lorsque l'entier naturel p est donné par l'utilisateur, de calculer une valeur approchée de α, de telle sorte que l'on ait :

$$0 \leqslant u_n - \alpha \leqslant 10^{-p}$$

D'après EML 2004

Une urne contient des boules blanches, des boules rouges et des boules vertes.

- La proportion de boules blanches est b.
- La proportion de boules rouges est r.
- La proportion de boules vertes est v.

Ainsi, on a $0 < b < 1$, $0 < r < 1$, $0 < v < 1$ avec $b + r + v = 1$. On effectue des tirages successifs avec remise et on s'arrête au premier changement de couleur.

1. Ecrire une fonction dont l'entête est

   ```
   function aleat(b,r:real):integer;
   ```

   qui donne 1 avec une probabilité de b, 2 avec une probabilité de r et 3 avec une probablité de $1-b-r$.

2. Complèter le programme suivant pour qu'il affiche le rang du premier changement de couleur :

```
begin
   writeln('proportion de rouges et
       de blanches ?');
   readln(r,b);
   randomize;
   x:=aleat(r,b);
   k:=...;
   repeat
       k:=k+1 ;
       y:=...
   until ....;
   writeln( 'il a fallu ',k,'tirages');
end.
```

Travaux Pratiques d'Informatique 31

REVISIONS II

Exercice 1
Ecrire un programme qui calcule avec N fourni par l'utilisateur :

$$\sum_{k=3}^{N}\left(\frac{k^2}{k!}\right)$$

Exercice 2
Ecrire un programme qui fournit tous les termes de la suite u jusqu'à ce que l'écart entre le terme u_n et α soit inférieur à une quantité *epsilon* choisie par l'utilisateur.

$$u_0 = 0 \quad u_{n+1} = \frac{1}{6}\left(1 - u_n^3\right)$$

On admettra que $|u_n - \alpha| \leq \left(\frac{1}{2}\right)^n$.

Exercice 3
Ecrire un programme qui fournit tous les termes de la suite u jusqu'à ce que l'écart entre le terme u_n et α soit inférieur à une quantité *epsilon* choisie par l'utilisateur.

$$u_0 = 2 \quad u_{n+1} = 1 - \frac{u_n}{3\left(u_n + 1\right)}$$

On admettra que $|u_n - \alpha| \leq 2\left(\frac{1}{3}\right)^n$.

Exercice 4
Ecrire un programme qui simule une loi binomiale de paramètre $\frac{1}{3}$ et de taille 5.

Exercice 5
Ecrire un programme qui simule une loi hypergéométrique de paramètres 10, 3 et $\frac{3}{10}$.

Exercice 6
Ecrire un programme qui simule une loi géométrique de paramètre $\frac{1}{3}$.

D'après EDHEC 2004

On désigne par n un entier naturel supérieur ou égal à 2.
On lance n fois une pièce équilibrée (cest-à-dire donnant pile avec la probabilité 1/2 et face également avec la probabilité 1/2), les lancers étant supposés indépendants.

On note Z la variable aléatoire qui vaut 0 si l'on n'obtient aucun pile pendant ces n lancers et qui, dans le cas contraire, prend pour valeur le rang du premier pile.

On rappelle que l'instruction `random(2)` renvoie un nombre au hasard parmi les nombres 0 et 1. Recopier et compléter le programme suivant pour qu'il simule l'expérience décrite ci-dessus, l'entier n é tant entré au clavier par l'utilisateur (pile sera codé par le nombre 1 et face par 0)

```
Program EDHEC2004 ;
var k, n, z, lancer : integer ;
Begin
   Randomize ;
   Readln(n) ;
   k := 0 ;
   z := 0 ;
   Repeat
        k := k + 1 ;
        lancer := random(2) ;
        If (lancer = 1) then .......... ;
   until ((lancer = 1) or (..........)) ;
   Writeln (z) ;
end.
```

D'après ESSEC 2003

On effectue des lancers successifs (indépendants) d'un dé cubique équilibré, dont les faces sont numérotées de 1 à 6, et on note $X_1, X_2, \ldots X_n, \ldots$, les variables aléatoires donnant le numéro amené par le dé aux premier lancer, deuxième lancer, ...

Pour tout entier naturel n non nul, on note Y_n, la somme des points obtenus aux n premiers lancers.

Enfin, pour tout entier naturel k non nul, la variable aléatoire T_k compte le nombre de celles des variables aléatoires $Y_1, Y_2, \ldots, Y_n, \ldots$ qui prennent une valeur inférieure ou égale à k .

Par exemple, si les cinq premiers numéros amenés par le dé sont, dans l'ordre : 3, 1, 2, 3, 6, alors les événements suivants sont réalisés : $(Y_1 = 3)$, $(Y_2 = 4)$, $(Y_3 = 6)$, $(Y_4 = 9)$, $(Y_5 = 15)$, et les variables aléatoires T_2 , T_3 , T_9 et T_{12} prennent respectivement pour valeurs 0, 1, 4 et 4.

Compléter les lignes marquées par les symboles... du programme Pascal ci-dessous, de façon qu'il simule l'expérience aléatoire étudiée et affiche la valeur de T_{12}.

```
Program ESSEC2003A;
var x,y,t,k:integer;
begin
   randomize;
   y:=0;
   t:=0;
   ...
   repeat
        x:=random(6)+1;
        k:=...
        y:=...
        if...       then t:=...
   until ...
   writeln(T=',t);
end.
```

Travaux Pratiques d'Informatique 32

REVISIONS : EXERCICES

Exercice 1

Quel est l'effet de cette séquence sur les contenus de A et B ?

```
C :=A ; A :=B ; B :=C ;
```

Exercice 2

On définit la suite u par $u_0 = 1$ et pour tout entier n,

$$u_{n+1} = u_n + \frac{1}{u_n}$$

1. Ecrire la séquence qui demande et saisit une valeur n et qui affiche u_n
2. Ecrire une séquence qui affiche la première valeur de n pour laquelle $u_n > 100$

Exercice 3

1. Ecrire une séquence qui simule 1000 lancers de dés et qui compte et affiche le nombre de 6 obtenus.
2. Ecrire une séquence qui simule des lancers de dés jusqu'à obtenir 6 et qui affiche le nombre de lancers effectués.

Exercice 4

Que font les séquences suivantes :

1.
```
Randomize ;
S :=0 ;
for i :=1 to 10 do
    begin D :=random(6)+1 ; S :=S+D end ;
moy :=S/10 ;
writeln(moy)
```
On précisera en particulier :
   - le rôle du `S :=0`
   - celui du couple `begin...end` ;

2.
```
max :=1 ; min :=6 ; Randomize ;
for j :=1 to 1000 do
begin
    S :=0 ;
    for i :=1 to 10 do
        begin
            D :=random(6)+1 ;
            S :=S+D
        end ;
    moy :=S/10 ;
    if moy > max then max :=moy ;
    if moy < min then min :=moy ;
end ;
writeln(min,' ',max) ;
```
On précisera en particulier :
   - pourquoi `max` est initialisé à 1 et `min` à 6
   - pourquoi l'initialisation `S :=0` se fait à l'interieur du `begin...end` ;

Exercice 5

(D'après CCIP 2000)

On dispose de deux jetons A et B que l'on peut placer dans deux cases C_0 et C_1, et d'un dispositif permettant de tirer au hasard et de manière équiprobable, l'une des lettre a, b ou c.

Au début de l'expérience, les deux jetons sont placés dans C_0.
On procède alors à une série de tirages indépendants de l'une des trois lettres a, b ou c.

A la suite de chaque tirage, on effectue l'opération suivante :

- si la lettre a est tirée, on change le jeton A de case,
- si la lettre b est tirée, on change le jeton B de case,
- si la lettre c est tirée, on ne change pas le placement des jetons.

On suppose qu'il existe un espace probabilisé dont la probabilité est notée p, qui modélise cette expérience et que l'on définit deux suites de variables aléatoires sur cet espace, $(X_n)_{n\geq 0}$ et $(Y_n)_{n\geq 0}$, décrivant les positions respactives de A et B, en posant :

- $X_0 = Y_0 = 0$
- Pour tout entier naturel n non nul, $X_n = 0$ si à l'issue de la $n^{ième}$ opération, le jeton A se trouve dans C_0 et $X_n = 1$ s'il se trouve dans C_1 ;
- De même, $Y_n = 0$ si à l'issue de la $n^{ième}$ opération, le jeton B se trouve dans C_0 et $Y_n = 1$ s'il se trouve dans C_1.

Simulation

Ecrire un programme en Turbo-Pascal permettant de simuler l'expérience, qui lira un entier N entré au clavier, représentant le nombre de tirages à effectuer, et qui affichera à l'écran la liste des couples observés (X_n, Y_n) pour $1 \leq n \leq N$.

Ce programme utilisera la fonction `RANDOM` qui renvoie, pour un argument m de type `INTEGER`, un nombre entier de l'intervalle $[0, m-1]$, tiré au hasard et de manière équiprobable.
(Cette fonction doit être initialisé par la commande `RANDOMIZE`)