Travaux Pratiques d'Informatique

Classe préparatoire économique, option économique, seconde année

Lycée Gambetta, Arras

• Mélissa Bailloeuil •

année scolaire 2012-2013

Travaux pratiques d'informatiques : introduction

Notion de programme :

```
program   nom_du_prog ;    (on donne un nom au programme)
uses crt ;                (on pourra utiliser l'instruction "clrscr" pour effacer l'écran)
var  a,b,c : real ;      (on déclare les variables, elles peuvent être de type :)
                            -> integer : entier de -32768 à 32767.
                            -> longint : entier de -2147483648 à 2147483647
                            -> real : nombre réel positif ou négatif.
                            -> array ou boolean
begin                      (on commence le programme)
clrscr ;                   (pour effacer l'écran précédent)
instructions ;
end.
```

Notion de fonction :

Les fonctions ont une syntaxe identique au programme.
Lorsqu'on a besoin d'un petit programme dans un programme, on crée une fonction.

```
function nom_fonction( i : integer ) :integer;
     (la fonction s'appelle "nom" elle prend par exemple ici un entier et renvoie un entier.
     La variable i, déjà déclarée, correspond à la valeur entrée dans la fonction.)
var  a,b,c : interger ;       (on déclare les variables de la fonction.)
begin
instructions ;
nom_fonction :=a ; (quand on appelle cette fonction, la valeur en sortie est celle de a.)
end ;       (end avec un point virgule pour terminer une fonction.)
```

Communiquer avec l'ordinateur :

1. L'ordinateur peut afficher du texte :
   – "writeln('bonjour') ;" à l'écran on lit : "bonjour"
   – "writeln('a') ;" à l'écran on lit : "a"
2. L'ordinateur peut afficher la valeur d'une variable, on ne met pas d'apostrophe :
   – "writeln(a) ;" à l'écran on lit la valeur de la variable a.
3. L'ordinateur peut stocker une valeur dans une variable, on utilise readln.
   "readln(a) ;" stocke la valeur donnée par l'utilisateur dans la variable a.

A la fin du programme, on met un "readln ;" tout seul pour avoir le temps de lire les résultats à l'écran.
Pour mettre des commentaires dans un programme, on les met entre accolades.

Structures :

1.
```
if  (condition)
then   begin
    instructions;
end;
else   begin
    instructions ;
end;
```

2.
```
for k:=1 to n do  begin
    instructions ;
end ;
```

3.
```
repeat
    instructions;
until  (condition);
```

4.
```
while (condition) do begin
    instructions;
end;
```

Travaux Pratiques d'Informatique 1

REVISIONS : PETIT MEMO I

Quelques morceaux de programmes remarquables :

1. **Suite définie explicitement**

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

2. **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;
```

3. **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;
```

4. **Calcul d'une somme**

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

REVISIONS ECE1 I

Exercice 1

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

$$\sum_{k=2}^{N}\left(\frac{k^2}{k+1}\right)$$

Exercice 2

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

$$u_0 = 1 \quad u_{n+1} = 2u_n + 6$$

Exercice 3

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

$$u_0 = 1 \quad u_{n+1} = 2u_n + 6n$$

Exercice 4

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

$$u_0 = 5 \quad u_1 = 3 \quad u_{n+2} = u_n \times u_{n+1}$$

Exercice 5

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

$$u_0 = 5 \quad u_1 = 3 \quad u_{n+2} = u_n \times (n - u_{n+1})$$

Exercice 6

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

$$\sum_{k=4}^{2N}\left(k^2 \times |N-k|\right)$$

Travaux Pratiques d'Informatique 2

Suites et Sommes

Dans une même variable, on range successivement plusieurs valeurs.
C'est à vous de savoir ce qui s'y trouve à chaque instant.
Vérifier particulièrement les valeurs stockées au début et à la fin des boucles.

Exemple épineux :
modifier deux variables à chaque itération

Pour calculer

$$\sum_{k=1}^{10} \frac{1}{k!}$$

on a

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

avec

$$(n+1) = (n+1)\, n!$$

que l'on comprend ainsi :
- La factorielle suivante est le produit de la précédente et de l'indice suivant.
- la somme suivante est le total de la somme précédente et de la factorielle suivante.

On obtient par conséquent le code Pascal :

```
F:=1;
S:=0;
for n:=1 to 10 do begin
   F:=F*n;
   S:=S+1/F ;
end;
writeln(S);
```

Pour calculer

$$\sum_{k=3}^{10} \frac{1}{k!}$$

on obtient le code Pascal qui varie légèrement :

```
F:=...;
S:=0;
for n:=3 to 10 do begin
   F:=F*n;
   S:=S+1/F ;
end;
writeln(S);
```

Pour calculer

$$\sum_{k=0}^{10} \frac{1}{k!}$$

on obtient :

```
F:=...;
S:=...;
for n:=1 to 10 do begin
   F:=F*n;
   S:=S+1/F ;
end;
writeln(S);
```

L'initialisation s'adapte en fonction des valeurs à avoir dans les variables à la première itération.

Exercice 1

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

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

Exercice 2

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 3

Quel est le calcul effectué par les algorithmes suivants :

```
u := -2;
FOR k :=4 TO 9 DO BEGIN
u :=u-3*u*u ;
END;
**********************************
u :=2 ; s :=2 ; FOR k :=1 TO 7 DO BEGIN
u :=u+ln(u);
s :=s+u ;
END;
**********************************
u :=0 ; p :=1 ;
FOR k :=1 TO 5 DO BEGIN
u :=u+2 ;
p :=p*u ;
END;
```

Exercice 4

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

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

Travaux Pratiques d'Informatique 3

REVISIONS : PETIT MEMO II

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. **Calcul d'un produit**

```
P:=1;
for k:= 0 to N do begin
   P:= P * "terme de rang k";
end;
```

ANNALES CONCOURS

Exercice 1 (ECRICOME 2003)

On considère les fonctions ch et sh définies sur $\mathbb{R}$ par :

$$sh(x) = \frac{e^x - e^{-x}}{2} \text{ et } f(x) = \frac{x}{sh(x)}$$

On s'intéresse dans cet exercice à la convergence de la suite $(u_n)_{n\in\mathbb{N}}$ définie par la relation de récurrence :

$$\begin{cases} u_0 = 1 \\ \forall n \in \mathbb{N} \;\; u_{n+1} = f(u_n) \end{cases}$$

1. Ecrire un programme en Turbo-Pascal permettant de calculer et d'afficher u_{10}

Exercice 2 (ESCP 2000)

On considère la suite $V = (v_n)_{n\geq 0}$ vérifiant $v_0 = 0$, $v_1 = \beta$ et, pour tout n positif, la relation

$$v_{n+2} = \sqrt{v_{n+1}} + \sqrt{v_n}$$

1. Ecrire un programme en Turbo-Pascal qui lise un entier N et un réel β et qui affiche, en sortie, les N premiers termes de la suite V.

Exercice 3 (EDHEC 1999)

$a_1 = 1$ et $b_1 = 1$
(...) On a obtenu que : $\forall n \in \mathbb{N}^*$,

$$a_{n+1} = \frac{a_n}{n+1} - \frac{b_n}{(n+1)^2}$$

et

$$b_{n+1} = \frac{b_n}{n+1}$$

Ecrire un programme en Turbo Pascal qui calcule et affiche les n premiers termes de chacune des suites (a_n) et (b_n) pour une valeur de n entrée par l'utilisateur.

Travaux Pratiques d'Informatique 4

Conditions d'arrêt, valeur approchée

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.
```

- $|u_n - \alpha| \leq \left(\frac{1}{2}\right)^{n+1}$ se traduit par u_n est une valeur approchée de α à $\left(\frac{1}{2}\right)^{n+1}$ près.
- De la même manière si on a $u_n \leq \alpha \leq v_n$ donc $0 \leq \alpha - u_n \leq v_n - u_n$ se traduit par u_n est une valeur approchée de α à $(v_n - u_n)$ près.

Exercice 1

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 2

Ecrire un programme qui fournit une valeur approchée de α à une quantité *epsilon* près choisie par l'utilisateur.

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

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

Compteur

Une variable "compteur" est un entier qui s'initialise puis augmente de 1 chaque fois que nécessaire.
Cela s'utilise pour

- mémoriser l'indice d'une suite, auquel cals l'initialisation se fait au rang de la première valeur de la suite et augmente de 1 à chaque nouveau terme calculé (souvent à chaque itération) :

```
u:=u0;
n:=0;
repeat
   n:=n+1;
   u:=f(u);
until (u>100);
writeln(n);
```

- compter le nombre de "succès" d"une expérience :

```
randomize;
C:=0;
writeln('probabilit\'{e} de succ\'{e}s ?')
readln(p);
for i:=1 to 100 do begin
    if (random<p) then c:=c+1;
end;
writeln('nombre de succ\'{e}s: ',c);}
```

Exercice 3 (ESC 2006)

Dans cette question on note $(u_k)_{k\in\mathbb{N}}$ la suite ainsi définie : $u_0 = -1$ et pour tout entier naturel k, $u_{k+1} = e^{u_k} - 2$
On a obtenu par récurrence sur k que pour tout entier naturel k :

$$0 \leqslant u_k - \alpha_2 \leqslant \left(\frac{1}{e}\right)^k$$

On considère le programme Turbo-Pascal suivant : (où `trunc` désigne la fonction partie entière)

```
program ex2 ;
var N , k : integer ; epsilon , u : real ;
begin
       writeln ( ' Donnez un reel strictement positif') ;
       readln (epsilon ) ;
       N := trunc ( - Ln (epsilon ) ) + 1 ;
       u := -1 ;
       for k := 1 to N do ..... ;
       writeln(u) ;
end.
```

Montrer que l'entier naturel N calculé dans ce programme vérifie : $\left(\frac{1}{e}\right)^N \leq$ epsilon.

Compléter la partie pointillée de ce programme afin que la variable `u` contienne après son exécution une valeur approchée de α_2 à epsilon près.

Travaux Pratiques d'Informatique 5

ANNALES CONCOURS

Exercice 1 (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 montre que $0 \leqslant u_n - \alpha \leqslant \dfrac{1}{e^{2^n-1}}$

Écrire un programme en langage Pascal 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 :

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

Exercice 2 (ECRICOME 2007)

Soit a un réel strictement positif. On considère la fonction f_a définie pour tout réel t strictement positif par :

$$f_a(t) = \frac{1}{2}(t + \frac{a^2}{t})$$

ainsi que la suite $(u_n)_{n\epsilon\mathbb{N}}$ de nombre réels déterminée par son premier terme $u_0 > 0$ et par la relation de récurrence :

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

On a :

$$|u_n - a| \leq \left(\frac{1}{2}\right)^{n-1} |u_1 - a|$$

Ecrire un programme en langage Pascal permettant d'afficher les 100 premiers termes d'une suite (u_n), de premier terme 1, convergeant vers $\sqrt{2}$.

Exercice 3 (ECRICOME 2008)

On considère les fonctions suivantes :

$$g(x, y) = 1 + \ln(x + y)$$

et pour $p \in \mathbb{N}^*$,

$$\begin{cases} f_p(x) = g(x, p) \\ h_p(x) = x - f_p(x) \end{cases}$$

On admet que le réel α_1 appartient à l'intervalle $[1; 3]$.

On définit la suite $(u_n)_{n\in\mathbb{N}}$ par : $\begin{cases} u_0 = 1 \\ \forall n \in \mathbb{N}, u_{n+1} = f_1(u_n) \end{cases}$

$$|u_n - \alpha_1| \leq (\frac{1}{2})^{n-1}$$

Ecrire un programme en langage Pascal permettant d'obtenir les valeurs de n_0 et de u_{n_0} de telle sorte que si n est supérieur ou égal à n_0 alors $|u_n - \alpha_1|$ est inférieur ou égal à 10^{-4}.

REVISIONS

Exercice 4

Ecrire un programme qui calcule la somme

$$\sum_{k=2}^{n} \frac{2 \times k}{\sqrt{k}}$$

pour n choisi par l'utilisateur.

Exercice 5

Ecrire un programme qui calcule le produit

$$\prod_{k=1}^{n} (3 + k)$$

pour n choisi par l'utilisateur.

Exercice 6

Ecrire un programme qui calcule la somme

$$\sum_{k=1}^{n} \frac{2^k}{k!}$$

pour n choisi par l'utilisateur.

Exercice 7

Soit u la suite définie par $u_0 = 1$, $u_1 = 2$ et $\forall n \in \mathbb{N}$, $u_{n+2} = u_{n+1} + u_n$.

1. Ecrire un programme qui calcule les termes de la suite u jusqu'au rang $n = 100$.
2. Ecrire un programme qui calcule les termes de la suite u jusqu'à ce que $u_n > 100$.

Travaux Pratiques d'Informatique 6

REVISIONS : PETIT MEMO III

Quelques morceaux de programmes remarquables :

1. **Simulation de X↪Bernoulli(proba)**

```
a := random;
if a<=proba then X:=1;
if a>proba then X:=0;
```

2. **Simulation de X↪Uniforme(n)**

```
X:=random(n)+1;
```

3. **Simulation de X↪Binomiale(n,p)**

```
X:=0;
for k:=1 to n do begin
    a := random;
    if a<=p then X:= X+1;
end;
```

4. **Simulation de X↪Hypergéo(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.

```
X:=0;
for k:=1 to nt do begin
    Nb:=Nb-1;
    a := random;
    if (a<p)) then begin
       X:= X+1;
       M:=M-1;
    end;
    p := M/Nb
end;
```

5. **Simulation de X↪Géométrique(p)**

```
X:=0;
repeat
    a := random;
    X:= X+1;
until(a<=p);
```

PROBABILITES : LES OUTILS

1. Tirage au hasard
   - random(n) donne équiprobablement les entiers de $[[0, n-1]]$.
   - random(n)+1 pour avoir un entier de $[[1, n]]$.
   - random donne équiproablement les réels de $[0, 1[$ ie si "r :=random" alors $P(r < a) = a$.
   On simule un tirage par un "if $(r < a)$"
2. Trirages sans remise
Il faut connaître le nombre de boules de chaque type restant et donc mettre en place un compteur pour chaque type de boule.
3. Nombre de
On utilise un compteur initialisé à 0 et incrémenté de 1 à chaque succés.
(C :=0 ;... ; if D=6 then C :=C+1 ;)
4. Moyenne
On calcule la somme des résultats et on divise par le nombre de résultats. Il faut donc une variable S pour la somme et un compteur pour le nombre de résultats, s'il n'est pas connu d'avance.
5. Maximum ou minimum
Pour déterminer le maximum ou le minimum, on initialise avec le premier, et on met à jour à chaque nouveau.
Exemple : max et min en 10 lancers de dés

```
randomize;
D:=random(5)+1; max:=D; min:=D;
for i:=2 to 10 do begin
    D:=random(5)+1;
    if D>max then max:=D;
    if D<min then min:=D;
end;
```

PROBABILITES

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

Exercice 2
Ecrire un programme qui simule une loi hypergéométrique de paramètres 10, 3 et $\frac{3}{10}$.

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

Travaux Pratiques d'Informatique 7

ANNALES CONCOURS

Exercice 1 (ESSEC 2003)

On se propose de simuler informatiquement une variable aléatoire.
On supposera que random(3) fournit au hasard un nombre élément de $\{1, 2, 3\}$ et que random(2) fournit au hasard un élément de $\{1, 2\}$

```
program ESSEC2003;
var ini,y : integer;
begin
   ini:=random(3);
   if ini=3 then y:=random(2) else y:=3 ;
end.
```

On appelle Y le contenu de y après exécution du programme ESSEC2003.
Donner la loi de Y, calculer son espérance $E(Y)$.

Exercice 2 (EDHEC 2012)

On désigne par n un entier naturel supérieur ou égal à 2. On note p un réel de $]0; 1[$ et on pose $q = 1 - p$.
On dispose d'une pièce donnant "Pile" avec la probabilité p et "Face" avec la probabilité q.
On lance cette pièce et on arrête les lancers dans l'une des deux situations suivantes :
• Soit si l'on a obtenu Pile.
• Soit si l'on a obtenu n fois Face.
Pour tout entier naturel k non nul, on note P_k (respectivement F_k l'événement « on obtient Pile (respectivement Face) au k^{e} lancer ».
On note T_n le nombre de lancers effectués, X_n le nombre de "Pile" obtenus et enfin Y_n le nombre de "Face" obtenus.

Compléter les trois instructions manquantes pour que le programme suivant simule l'expérience aléatoire décrite ci-dessus et pour qu'il affiche, dans cet ordre, les valeurs prises par les variables aléatoires T_n , X_n et Y_n, à l'exécution de l'instruction `Writeln (t, x, y) ;`

```
   Program EDHEC _2012 ;
Var n,t,x,y : integer ; p :real ;
Begin
Randomize ; t := 0 ; x : = 0 ; y := 0 ;
Readln(n) ;
While (x = 0) and (t<n) do
begin
------ ;
If random >p then ------ else ------ ;
end ;
Writeln (t, x, y) ;
End.
```

Exercice 3 (EDHEC 2011)

On désigne par n un entier naturel supérieur ou égal à 2.
On dispose de n urnes, numérotées de 1 à n, contenant chacune n boules.
On répète n épreuves, chacune consistant à choisir une urne au hasard et à en extraire une boule au hasard. On suppose que les choix des urnes sont indépendants les uns des autres.
Pour tout i de $\{1, 2, ..., n\}$, on note X_i la variable aléatoire prenant la valeur 1 si l'urne numérotée i contient toujours n boules au bout de ces n épreuves, et qui prend la valeur 0 sinon.

Compléter le programme informatique suivant pour qu'il simule l'expérience décrite au début de cet exercice et affiche les valeurs prises par X_1 et N_1 pour une valeur de n entrée par l'utilisateur.

```
Program edhec_2011;
Var x1, n1, n, k, tirage, hasard : integer;
Begin
    Randomize;
    Writeln('donnez un entier naturel n
         supérieur ou égal à 2');
    Readln(n);
    n1:=0;
    x1:=1;
    For k:=1 to n do begin
        hasard := random(n)+1;
        if hasard = 1 then begin
        x1 :=--------;
        n1:= --------;
    end;
    end;
    Writeln(x1,n1);
End.
```

Exercice 4

Ecrire un programme qui simule une loi hypergéométrique de paramètres 8, 7 et $\frac{1}{2}$.

Exercice 5

Ecrire un programme qui simule une VAR Y telle que Y+1 suive une loi géométrique de paramètre $\frac{1}{2}$.

Travaux Pratiques d'Informatique 8

REVISIONS : 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 (ECRICOME 2011)

On considère l'application φ défini sur $\mathbb{R}_+$ par :

$$\begin{cases} \varphi(x) = 1 - x^2 \ln(x) & \text{si } x > 0 \\ \varphi(0) = 1 \end{cases}$$

On considère les deux suites $(a_n)_{n\in\mathbb{N}}$ et $(b_n)_{n\in\mathbb{N}}$ définies par :
$a_0 = \sqrt{2}$ et $b_0 = 2$,
$\forall n \geq 0$,
si $\varphi(a_n)\,\varphi\left(\dfrac{a_n + b_n}{2}\right) < 0$
alors $a_{n+1} = a_n$ et $b_{n+1} = \dfrac{a_n + b_n}{2}$
$\forall n \geq 0$, si $\varphi(a_n)\,\varphi\left(\dfrac{a_n + b_n}{2}\right) \geq 0$
alors $a_{n+1} = \dfrac{a_n + b_n}{2}$ et $b_{n+1} = b_n$.

Ecrire un programme en Pascal calculant a_7 et b_7

Exercice 2 (ECRICOME 2010)

On considère l'application φ définie sur $\mathbb{R}_+^*$ par :

$$\forall x \in \mathbb{R}_+^*, \quad \varphi(x) = \ln(x) - \ln(x+1) + \frac{1}{x}$$

1. Déterminer la limite de $\varphi(x)$ lorsque x tend vers 0 par valeurs positives.
2. Déterminer la limite de $\varphi(x)$ lorsque x tend vers $+\infty$.
3. Prouver que φ est strictement monotone sur $\mathbb{R}_+^*$.
4. Dresser le tableau de variation de φ et y faire apparaître les limites de φ en 0^+ et $+\infty$.
5. On rappelle que $\ln(2) \simeq 0,7$ et $\ln(3) \simeq 1,1$.
   Montrer que l'équation $\varphi(x) = 1$ possède une unique solution notée α et que :
   $$\frac{1}{3} < \alpha < \frac{1}{2}$$
6. Proposer un programme en Pascal permettant d'encadrer α dans un intervalle d'amplitude 10^{-2}.

REVISIONS

Exercice 3

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

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

Exercice 4

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

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

Travaux Pratiques d'Informatique 9

ANNALES CONCOURS

Exercice 1 (EML 2009)

On note $f : \mathbb{R} \to \mathbb{R}$ l'application définie, pour tout $x \in \mathbb{R}$, par :

$$f(x) = \begin{cases} \dfrac{x}{e^x - 1} & \text{si } x \neq 0 \\ 1 & \text{si } x = 0 \end{cases}$$

On considère la suite $(u_n)_{n\in\mathbb{N}}$, définie par $u_0 = 1$ et, pour tout $n \in \mathbb{N}$, $u_{n+1} = f(u_n)$.
On sait que f admet un point fixe et un seul, noté α et $\forall n \in \mathbb{N}$

$$|u_n - \alpha| \leq \frac{1}{2^n}(1 - \alpha)$$

Ecrire un programme en Turbo-Pascal qui calcule et affiche le plus petit entier naturel n tel que$|u_n - \alpha| < 10^{-9}$

Exercice 2 (ECRICOME 2009)

On considère l'application φ définie sur $\mathbb{R}^{+*}$ par :

$$\varphi(x) = 2\ln\left(\frac{x}{2}\right) + \frac{1}{x}$$

1. Déterminer la limite de $\varphi(x)$ lorsque x tend vers 0 par valeurs positives.
2. Déterminer la limite de $\varphi(x)$ lorsque x tend vers $+\infty$.
3. Justifier la dérivabilité de φ sur $\mathbb{R}^{+*}$, déterminer sa dérivée.
4. Dresser le tableau de variation de φ, faire apparaître les limites de φ en 0^+ et $+\infty$.
5. On rappelle que $\ln(2) \simeq 0,7$. Montrer l'existence de deux réels positifs α et β tels que :

$$\varphi(\alpha) = \varphi(\beta) = 0$$

6. Proposer un programme en Pascal permettant d'encadrer α dans un intervalle d'amplitude 10^{-2}.

Exercice 3 (EDHEC 2009)

Dans cet exercice, p désigne un réel de $]0;1[$ et on note $q = 1 - p$.
On rappelle que la fonction random renvoie de façon uniforme un réel aléatoire élément de $[0;1[$.
Soit X une VAR, on définit la variable aléatoire T de la façon suivante :
Pour tout ω de Ω tel que $X(\omega)$ est un entier naturel pair, on pose $T(\omega) = \dfrac{X(\omega)}{2}$, et, pour tout ω de Ω tel que $X(\omega)$ est un entier naturel impair, on pose $T(\omega) = \dfrac{1 + X(\omega)}{2}$.
Compléter le programme suivant pour que, d'une part, il simule les lancers d'une pièce donnant "*pile*" avec la probabilité p et calculer la valeur prise par la variable aléatoire X égale au rang du premier "*pile*" obtenu lors de ces lancers (X suit bien la loi géométrique de paramètre p) et pour que, d'autre part, il calcule et affiche la valeur prise par T.

```
Program edhec2009;
Var x,t,lancer:integer;\newline
Begin
    Randomize;
    x:=0;
    Repeat
        lancer:=random;
        x:=.......;
    until(lancer<=p);
    if(x mod 2=0)
    then ........
    else.........;
    Writeln(t);
 End.
```

Exercice 4 (HEC 2009)

Dans tout le problème, on considère la suite $(u_n)_{n\in\mathbb{N}}$ définie par $u_0 = 0$, $u_1 = 1$ et la relation pour tout n de $\mathbb{N}$, $u_{n+2} = u_{n+1} + u_n$.
On propose la fonction Pascal suivante :

```
Function f(n : integer) integer ;
var temp,u,v,k : integer ;
Begin
u := 0 ; v :=1 ;
for k :=1 to n-1 do
Begin
temp := _____; v := _____; u := _____;
end ;
f := _____;
end ;
```

Compléter cette fonction aux quatre places signalées par des tirets de façon que la valeur rendue soit u_n.

Travaux Pratiques d'Informatique 10

NOUVEAUTE : RECURSIVITE

Les fonctions et procédure de Pascal peuvent s'appeler elles mêmes,. La récursivité est la possibilité donnée à un sous programme (procédure ou fonction) de s'appeler lui-même. Une procédure ou une fonction qui s'appelle elle-même est dite récursive. La notion informatique de récursivité est intimement liée à celle mathématique de récurrence.
Cela permet de programmer directement la relation de récurrence mathématique.
N.B : l'ordinateur doit garder en mémoire (entasser) tous les résultats intermédiaires. On peut avoir ainsi des débordement de mémoire....

1. Factorielle :
   $0! = 1$ et pour tout $n \geq 1 : n! = n\,(n-1)!$

```
function fact(n:integer):integer;
begin}
if (n<0) then fact:=0;
if n=0 then fact:=1;
if (n>0) then fact:=fact(n-1)*n;}
end;
```

2. Coefficients du binôme :
   - $\binom{n}{p} = 0$ si l'on n'a pas $0 \leqslant p \leqslant n$
   - $\binom{0}{0} = 1$ et $\binom{0}{p} = 0$ si $p \neq 0$
   - $\binom{n}{p} = \binom{n-1}{p-1} + \binom{n-1}{p}$

```
function bin(n,p:integer):integer;
begin
    if not( (0<=p) and (p<=n) )
    then bin:=0;
    if n=0 then begin
        if p=0 then bin:=1 else bin:=0;
    end;
    if (n>0)
    then bin:=bin(n-1,p-1)+bin(n-1;p);
end;
```

Exercice 1 (EDHEC 2006 (récursive))

1. On considère la déclaration de fonction, en Pascal, rédigée de manière récursive :

```
Function f(n : integer) : integer ;
Begin
If (n=0) then f := -----
else f : =-----
end ;
```

   Compléter cette déclaration pour qu'elle renvoie $n!$ lorsqu'on appelle $f(n)$.

2. On considère la déclaration de fonction récursive suivante :

```
Function g (a :real ; n :integer) :real ;
Begin
If (n=0) then g : = 1
else g : = a * g(a,n- 1) ;
end ;
```

   Dire quel est le résultat retourné à l'appel de $g(a, n)$.

3. Proposer un programme (sans écrire la partie déclarative) utilisant ces deux fonctions et permettant d'une part le calcul de la somme $\sum_{k=0}^{n-1} \frac{a^k}{k!} e^{-a}$ et d'autre part, à l'aide du résultat de la question 1a), le calcul et l'affichage du taux de panne à l'instant n d'une variable aléatoire suivant la loi de Poisson de paramètre $a > 0$, lorsque n et a sont entrés au clavier par l'utilisateur (on supposera $n \geq 1$).

4. Compléter la déclaration de fonction suivante pour, qu'elle renvoie la valeur de $\sum_{k=0}^{n-1} \frac{a^k}{k!} e^{-a}$ à l'appel de *sigma(a,n)*.

```
Function sigma(a : real ; n : integer) :
real ;
var k : integer ;
p : real ;
Begin
p : = 1 ; s :=1 ;
For k : = 1 to n-1 do begin p := p*a / k ;
s := ... ; end ;
s := ... ;
sigma : = s ;
end ;
```

Exercice 2

Soit u définie par $u_0 = 1$ et $\forall n \in \mathbb{N}^*$, $u_n = 3u_{n-1} + 1$.

1. Ecrire un programme itératif (for to do) qui calcule et affiche les n premiers termes de la suite.
2. Ecrire un programme utilisant une fonction récursive qui calcule et affiche les n premiers termes de la suite.

Exercice 3

Pourquoi les deux fonctions suivantes ne s'arrêtent-t-elles pas ?

```
FUNCTION sansfin1(x :real) :real;
BEGIN
   sansfin1 :=sansfin1(x-1) ;
END ;

FUNCTION sansfin2(x :real) :real
BEGIN
    IF x=0 then sansfin2 :=0
    ELSE sansfin2 :=sansfin2(x) ;
END ;
```

Travaux Pratiques d'Informatique 11

ANNALES CONCOURS

Exercice 1 (EDHEC 2008)

Pour tout couple (p, q) d'entiers naturels, on pose $I(p,q) = \int_0^1 x^p (1-x)^q \, dx$

1. Montrer que : $\forall (p,q) \in \mathbb{N} \times \mathbb{N}$,
$$I(p,q) = \frac{q}{p+1} I(p+1, q-1)$$
2. Déterminer$I(p,0)$.
3. Compléter la déclaration récursive suivante afin qu'elle permette le calcul de $I(p,q)$:

```
Function i(p, q : integer ) : real ;
Begin
     If q = 0 then i :=... else ... ;
End;
```

Exercice 2 (EML 2006)

On note $F : \mathbb{R}^2 \to \mathbb{R}$ l'application définie pour tout $(x,y) \in \mathbb{R}^2$ par :
$$F(x,y) = (x-1)(y-2)(x+y-6)$$
On considère la suite réelle $(u_n)_{n\in\mathbb{N}}$ définie par $u_0 = 4$ et :
$$\forall n \in \mathbb{N},\ u_{n+1} = F(1+u_n, u_n)$$
On a pu montrer dans l'exercice que
$$\forall n \in \mathbb{N} : \quad u_n \geq 4^{n+1}$$
Ecrire un programme en Turbo-Pascal qui calcule et affiche le plus petit entier naturel n tel que $u_n \geq 10^{10}$

Exercice 3 (EDHEC 2005)

Un mobile se déplace sur les points à coordonnées entières d'un axe d'origine O

Au départ, le mobile est à l'origine.

Le mobile se déplace selon la règle suivante : s'il est sur le point d'abscisse k à l'instant n, alors, à l'instant $(n+1)$ il sera sur le point d'abscisse $(k+1)$ avec la probabilité $\frac{1}{3}$ ou sur le point d'abscisse 0 avec la probabilité $\frac{2}{3}$.

Pour tout n de $\mathbb{N}$, on note X_n l'abscisse de ce point à l'instant n et l'on a donc $X_0 = 0$.

Compléter le programme suivant pour qu'il simule l'expérience aléatoire étudiée et affiche la valeur prise par X_n pour une valeur de n entrée par l'utilisateur.

```
Program edhec2005 ;
Var k, n, u, X : integer ;
begin
     Readln(n) ;
     Randomize ;
     X :=0 ;
     For k :=1 to n do
     begin
          u := random(3) ;
          if (u = 2) then X :=..... ;
               else X :=..... ;
          end ;
     Writeln (X) ;
end.
```

Exercice 4 (EDHEC 2007)

On lance une pièce équilibrée et on note Z la variable aléatoire égale au rang du lancer où l'on obtient le premier "pile".

Après cette série de lancers, si Z a pris la valeur k ($k \in \mathbb{N}^*$), on remplit une urne de k boules numérotées 1, 2, $\cdots$, k, puis on extrait au hasard une boule de cette urne.

On note X la variable aléatoire égale au numéro de la boule tirée après la procédure décrite ci-dessus.

On décide de coder l'événement ≪obtenir un "pile"≫ par 1 et l'événement ≪obtenir un "face"≫ par 0.

1. Compléter le programme suivant pour qu'il affiche la valeur prise par Z lors de la première partie de l'expérience décrite ci-dessus.

```
Program edhec_2007 ;
Var z,hasard :integer ;
begin
     randomize ; z :=0 ;
     repeat
        z :=..... ; hasard :=..... ;
until (hasard=1) ;
     writeln(z) ;
end.
```

2. Quelle instruction faut-il rajouter avant la dernière ligne de ce programme pour qu'il simule l'expérience aléatoire décrite dans ce problème et affiche la valeur prise par la variable aléatoire X ?

Exercice 5 (Loi de Pascal)

On effectue des lancers successifs d'un dé honnête.
On note X la variable aléatoire égale au nombre de lancers effectués lors de l'apparition d'un second 6.
Ecrire un programme, utilisant la fonction DE, qui simule cette expérience et qui affiche la valeur de X ainsi que les résultats successifs sous la forme : 1 5 3 6 4 4 1 3 6 ; X=9.
On généralise l'expérience : X désigne maintenant le temps d'attente de l'obtention d'un nième 6. Ecrire une fonction Pasc prenant n comme paramètre, qui simule cette expérience et renvoie la valeur de X.

Travaux Pratiques d'Informatique 12

NOUVEAUTE : SIMULATION DE VAR CONTINUES

La fonction de répartition d'une variable aléatoire réelle X est définie par

$$F_X(x) = P(X \leq x)$$

Une variable aléatoire réelle X suit la loi exponentielle de paramètre $\lambda \in \mathbb{R}_+^*$ si X a pour densité

$$f : x \mapsto \begin{cases} 0 & \text{si } x < 0 \\ \lambda e^{-\lambda x} & \text{si } x \geq 0 \end{cases}$$

1. En utilisant la fonction random écrire un programme simulant une variable aléatoire suivant une loi uniforme sur l'intervalle $[a; b]$.
2. Soit X une variable aléatoire de loi uniforme sur l'intervalle $[0; 1[$. On souhaite déterminer la loi de la variable $Y = -\ln(1 - X)$.
   - Déterminer la fonction de répartition $F_Y : x \mapsto P(Y \leq x)$ de la variable Y.
   - Déterminer la fonction de répartition d'une variable aléatoire suivant la loi exponentielle de paramètre 1.
   - Conclure

   Ecrire un programme qui simule une variable aléatoire suivant une loi exponentielle de paramètre 1.
3. Déterminer plus généralement un programme simulant la loi exponentielle de paramètre $\lambda \in \mathbb{R}_+^*$.
4. On souhaite simuler une loi Gamma de paramètre b et m : . On sait que la loi Gamma de paramètre b et m est la loi de la somme de m variables aléatoires indépendantes suivant une loi exponentielle de paramètre $\frac{1}{b}$. Ecrire un programme qui simule une variable aléatoire suivant une loi Gamma de paramètres b et m.

Exercice 1 (EML 2012)

Soit$a \in \mathbb{R}_+^*$. On considère l'application $g_a : \mathbb{R} \to \mathbb{R}$ définie, pour tout $x \in \mathbb{R}$, par :

$$g_a(x) = \begin{cases} 0 & \text{si } x \leq 0 \\ \dfrac{x}{a^2} e^{-\frac{x^2}{2a^2}} & \text{si } x > 0 \end{cases}$$

On considère une variable aléatoire X admettant g_a comme densité.

1. Déterminer la fonction de répartition de la variable aléatoire X.
2. On considère une variable aléatoire U suivant la loi uniforme sur l'intervalle, $]0; 1]$. Montrer que la variable aléatoire $Z = a\sqrt{-2\ln(U)}$ suit la même loi que la variable aléatoire X
3. En déduire un programme en langage Pascal, utilisant le générateur aléatoire Pascal, simulant la variable aléatoire X, le réel a strictement positif étant entré par l'utilisateur.

Travaux Pratiques d'Informatique 13

ANNALES CONCOURS

Exercice 1 (EML 2007)

On considère l'application $f : \mathbb{R} \to \mathbb{R}$ définie pour tout nombre réel x par :

$$\begin{cases} f(x) = e^{-x} & \text{si } x > 0 \\ f(x) = 0 & \text{si } x \leq 0 \end{cases}$$

f est une densité de probabilité. On considère une variable aléatoire X admettant f pour densité.

On définit la variable aléatoire discrète Y à valeurs dans $\mathbb{N}$ de la façon suivante :

- ⋆ l'événement $(Y = 0)$ est égal l'événement $(X < 1)$
- ⋆ pour tout nombre entier strictement positif n, l'événement $(Y = n)$ est égal à l'événement $(n \leq X < n + 1)$.

$Y + 1$ suit une loi géométrique dont on précisera le paramètre.

Recopier et compléter le programme ci-dessous pour qu'il simule la variable aléatoire Y

```
program eml2007 ;
   var y :integer ; u :real ;
   begin
        randomize ;
        u :=random ; y :=··· ;
        while... do
        ... ... ...
        writeln('y vaut ', y) ;
   end.
```

Exercice 2 (EML 2010)

On note $f : \mathbb{R} \to \mathbb{R}$ l'application de classe C^2, définie, pour tout $x \in \mathbb{R}$, par

$$f(x) = x - \ln\left(1 + x^2\right)$$

On considère la suite $(u_n)_{n\geq 0}$ définie par $u_0 = 1$ et :

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

La suite $(u_n)_{n\geq 0}$ est décroissante, converge vers 0.

Écrire un programme en Turbo-Pascal qui calcule et affiche un entier n tel que $u_n \leq 10^{-3}$.

Exercice 3 (EML 2011)

On considère l'application

$$f :]0; +\infty[\to \mathbb{R}, \quad x \longmapsto f(x) = (x + \ln x)\, e^{x-1}$$

On considère la suite réelle $(u_n)_{n\in\mathbb{N}}$ définie par $u_0 = 2$ et, pour tout $n \in \mathbb{N}$, $u_{n+1} = f(u_n)$. On peut établir par récurrence :

$$\forall n \in \mathbb{N}, \quad u_n \geq e^n$$

1. Quelle est la limite de u_n lorsque l'entier n tend vers l'infini ?
2. Écrire un programme en Turbo-Pascal qui calcule et affiche le plus petit entier naturel n tel que $u_n \geq 10^{20}$.

Travaux Pratiques d'Informatique 14

ANNALES CONCOURS

Exercice 1 (ESSEC 2011)

Une question que se pose un joueur de cartes est de savoir combien de fois il est nécessaire de battre les cartes pour que le paquet soit convenablement mélangé. Ce problème décrit un procédé très élémentaire pour mélanger les cartes et propose de répondre alors à cette question.

Considérons un jeu de N cartes numérotées de C_1 à C_N et disposées en un paquet sur une table. Un joueur bat les cartes et repose le paquet sur la table. Le résultat du mélange est une permutation de ces N cartes.

Notations et Rappel : on note S_N l'ensemble des permutations possibles pour ce paquet de N cartes et on rappelle que $card(S_N) = N!$

On se place dans un espace probabilisé $(\Omega, \mathcal{A}, \mathbf{P})$ avec $\Omega = S_N$, $\mathcal{A} = \mathcal{P}(S_N)$ l'ensemble des parties de S_N et $\mathbf{P}$ l'équiprobabilité sur Ω. Pour toute variable aléatoire X on notera $E(X)$ et $V(X)$ l'espérance et la variance de X lorsqu'elles existent.

On considère qu'un paquet est convenablement mélangé lorsque toutes les permutations sont équiprobables, c'est-à-dire lorsque pour toute permutation σ de S_N la probabilité que le tas de cartes se trouve dans la configuration σ vaut $1/N!$

Vocabulaire et notations :

Une carte située au sommet de la pile est dite *en position n° 1*, celle qui se trouve immédiatement en dessous est dite *en position n ° 2*, etc. Ainsi une carte située *en position n° N* désigne la carte située en bas de la pile. On prendra garde à bien distinguer la position d'une carte dans le paquet du numéro qu'elle porte.

Partons d'un tas de cartes rangées initialement dans l'ordre suivant : pour tout i élément de $[[1, N]]$, la carte C_i se trouve en position i. Ainsi, à l'instant initial, la carte C_1 se trouve sur le dessus du paquet alors que C_N se trouve donc tout en dessous du paquet.

Pour k élément de $[[1, N]]$, on appelle *insertion* à la k-ième place l'opération qui consiste à prendre la carte située au-dessus du paquet et à l'insérer entre la k-ième et la $(k+1)$-ième place. Une insertion à la première place ne change pas l'ordre des cartes. Une insertion à la N-ième place consiste à faire glisser la carte située au-dessus du paquet pour la mettre sous le paquet.

Le battage par insertions du jeu de cartes consiste à effectuer une suite d'insertions aléatoires, en choisissant, à chaque instant, au hasard uniformément dans $\{l, \cdots, N\}$ la place à laquelle l'insertion a lieu, indépendamment des insertions précédentes. Les instants successifs d'insertions seront notées 1,2, ... n, ... , l'instant initial est $n = 0$.

Notations. Nous notons :

- T_1 le premier instant où la carte située sur le dessus du paquet est glissée en dernière position, c'est-à-dire le premier instant où la carte C_N se trouve remontée de la position N à la position $N-1$,
- T_2 le premier instant où la carte C_N se trouve remontée en position $N-2$,
- et plus généralement, pour i dans $[[1, N-1]]$, T_i le premier instant où la carte C_N atteint la position $N-i$.
- On posera également $\Delta_1 = T_1$ et $\forall i \in [[2, N-1]]$, $\Delta_i = T_i - T_{i-1}$.
- Enfin, on notera $T = T_{N-1} + 1$.

On admet que les conditions de l'expérience permettent de faire l'hypothèse que les variables aléatoires $(\Delta_i)_{i \in [[1, N-1]]}$ sont indépendantes.

Description d'un exemple. Dans le tableau ci-dessous, nous décrivons les résultats d'une expérience faite sur un paquet de $N = 4$ cartes. La première ligne du tableau indique les instants n, la deuxième ligne indique les positions d'insertions , et dans la dernière ligne figure la configuration du paquet à l'instant n.

		instant	n	0	1	2	3	4	5	6	7
	insertion	en place	k		3	2	4	1	3	4	2
Configuration		position	1	C_1	C_2	C_3	C_2	C_2	C_1	C_4	C_2
du		position	2	C_2	C_3	C_2	C_1	C_1	C_4	C_2	C_4
paquet		position	3	C_3	C_1	C_1	C_4	C_4	C_2	C_3	C_3
		position	4	C_4	C_4	C_4	C_3	C_3	C_3	C_1	C_1

Pour cette experience, on a les resultats $T_1(\omega) = 3$, $T_2(\omega) = 5$ et $T_3(\omega) = 6$ et $T(\omega) = 7$.

On considère un jeu de $N = 32$ cartes.

MODÉLISATION : On définit en PASCAL le `TYPE Paquet=ARRAY[1 .. 32] OF INTEGER;` Le paquet de 32 cartes est représenté par une variable `Jeu` de `TYPE Paquet` rempli initialement d'entiers entre 1 et 32; donc, initialement, `Jeu[i]` contient i, c'est à dire que la carte C_i est en position i. Au cours des insertions, `Jeu [i]` désigne le numéro de la carte en position numéro i. Par exemple, `Jeu [i] =10` signifie que la carte C_{10} est en position i.

On indique à la fin de cette question un extrait de programme à compléter en suivant les questions suivantes :

1. Écrire la procédure `Init` permettant de définir une variable `Jeu` correspondant à la configuration initiale du paquet de cartes.
2. Compléter la procédure `Insertion` qui simule une opération d'insertion. On rappelle que la fonction `RANDOM(32)` permet de tirer un nombre entier au hasard dans l'intervalle $[[0, 31]]$.
3. Que fait la fonction `T` ?
4. Écrire le programme principal permettant de calculer et d'afficher la moyenne des valeurs prises par la fonction `T` sur 100 expériences et compléter la ligne de déclaration de variables.

Extrait du programme.

```
PROGRAM ESSEC2011;
TYPE Paquet=ARRAY[1 .. 32] OF INTEGER;
VAR Jeu : Paquet;
    .............(à compléter)

PROCEDURE Init( ............. )

PROCEDURE Insertion(VAR Jeu :Paquet);
VAR i,k,cartedessus :INTEGER;
BEGIN
    k := ....... (position où on va insérer la carte du dessus)
    cartedessus :=Jeu[1];
    IF k>1 THEN FOR i :=1 TO k-1 DO Jeu [i] := ....
Jeu[k] := ....
END;

FUNCTION T(Jeu :Paquet) :INTEGER;
VAR n :INTEGER;
BEGIN
    Init(Jeu);
    n :=0;
    WHILE Jeu[1]<>32 DO
    BEGIN
        Insertion(Jeu);
        n :=n+1
    END;
    T :=n
END;

BEGIN {programme principal}

END.
```

Travaux pratiques d'informatiques : Corrections

Exercice 1 (EML 2012)

On considère l'application $g_a : \mathbb{R} \to \mathbb{R}$ définie, pour tout $x \in \mathbb{R}$, par :

$$g_a(x) = \begin{cases} 0 & \text{si } x \leq 0 \\ \dfrac{x}{a^2} e^{-\frac{x^2}{2a^2}} & \text{si } x > 0 \end{cases}$$

1. g_a est continue par morceaux sur $\mathbb{R}$ et positive.
   $\int_{-\infty}^{0} g_a(x)\,dx = 0$ et

   $$\int_0^{+\infty} \frac{x}{a^2} e^{-\frac{x^2}{2a^2}} dx = \frac{1}{a^2} I_1 = a^2$$

   donc $\int_{-\infty}^{+\infty} g_a(t)\,dt$ converge et vaut 1

   Conclusion : $\boxed{g_a \text{ est une densité de probabilité}}$

   On considère une variable aléatoire X admettant g_a comme densité.

2. Soit G la fonction de répartition de X. On a pour tout $t \in \mathbb{R}$: $G(t) = \int_{-\infty}^{t} g_a(x)\,dx$
   - si $t \leq 0$: $G(t) = \displaystyle\int_{-\infty}^{t} 0dx = 0$
   - si $t \geq 0$: $G(t) = \displaystyle\int_{-\infty}^{0} 0dx + \int_0^t \frac{x}{a^2} e^{-\frac{x^2}{2a^2}} dx = \left[-e^{-\frac{x^2}{2a^2}}\right]_0^t = 1 - e^{-\frac{t^2}{2a^2}}$

3. On considère une variable aléatoire U suivant la loi uniforme sur l'intervalle, $]0;1]$.

   Soit f la densité d'une loi $\mathcal{U}_{]0;1]}$: $f(x) = \begin{cases} 1 & \text{si } x \in]0;1] \\ 0 & \text{sinon} \end{cases}$ et F sa fonction de répartition.

   Soit enfin H la fonction de répartition de $Z = a\sqrt{-2\ln(U)}$

   Pour tout $x \in \mathbb{R}$:

   $$\begin{aligned} H(x) &= \mathrm{P}\left(a\sqrt{-2\ln(U)} \leq x\right) \\ &= \mathrm{P}\left(\sqrt{-2\ln(U)} \leq \frac{x}{a}\right) \text{ car } a > 0 \\ &= \mathrm{P}\left(-2\ln(U) \leq \left(\frac{x}{a}\right)^2\right) \\ &= \mathrm{P}\left(\ln(U) \geq -\frac{1}{2}\left(\frac{x}{a}\right)^2\right) \\ &= \mathrm{P}\left(U \geq \exp\left(-\frac{1}{2}\left(\frac{x}{a}\right)^2\right)\right) \\ &= 1 - F\left(\exp\left(-\frac{1}{2}\left(\frac{x}{a}\right)^2\right)\right) \end{aligned}$$

   Comme U est à densité, F est continue sur $\mathbb{R}$ et C^1 sur $\mathbb{R}\setminus\{0;1\}$

   Donc H est continue sur $\mathbb{R}$ et C^1 là où $\exp\left(-\frac{1}{2}\left(\frac{x}{a}\right)^2\right) \neq 0$ et $\neq 1$ i.e. en $x \neq 0$

   Donc Z est à densité et une densité de Z est :

   $$h(x) = -F'\left(\exp\left(-\frac{1}{2}\left(\frac{x}{a}\right)^2\right)\right) \exp\left(-\frac{1}{2}\left(\frac{x}{a}\right)^2\right) \frac{-x}{a^2}$$

   et comme $\exp\left(-\frac{1}{2}\left(\frac{x}{a}\right)^2\right) \in]0;1]$ pour tout $x \in \mathbb{R}$ alors

   $$h(x) = \exp\left(-\frac{1}{2}\left(\frac{x}{a}\right)^2\right) \frac{x}{a^2} = g_a(x)$$

   Conclusion : $\boxed{Z \text{ suit la même loi que } X}$

4. La loi uniforme $\mathcal{U}_{]0;1]}$ est simulée par `U :=random ;` et `Z :=a*sqrt(-2*ln(U))` simulera donc la loi de X

```
program simula ;
var a,U,Z :real ;
begin
     writeln('a,') ;readln(a) ;randomize ;
     U :=random ;
     Z :=a*sqrt(-2*ln(U) ;
     writeln(Z) ;
end.
```

Exercice 2 (EML 2007)

On considère l'application $f : \mathbb{R} \to \mathbb{R}$ définie pour tout nombre réel x par :

$$\begin{cases} f(x) = e^{-x} & \text{si } x > 0 \\ f(x) = 0 & \text{si } x \leq 0 \end{cases}$$

f est la densité d'une loi exponentielle de paramètre 1.

Conclusion : $\boxed{f \text{ est une densité de probabilité.}}$

On considère une variable aléatoire X admettant f pour densité.

On définit la variable aléatoire discrète Y à valeurs dans $\mathbb{N}$ de la façon suivante :

- ⋆ l'événement $(Y = 0)$ est égal l'événement $(X < 1)$
- ⋆ pour tout nombre entier strictement positif n, l'événement $(Y = n)$ est égal à l'événement $(n \leq X < n+1)$.

Pour $n = 0$ on a $\mathrm{P}(Y = 0) = \mathrm{P}(X < 1) = \int_{-\infty}^{1} f(t)\,dt = \int_0^1 e^{-t}dt = \left[-e^{-t}\right]_0^1 = 1 - e^{-1}$.

Pour n entier strictement positif :

$$\begin{aligned} \mathrm{P}(Y = n) &= \mathrm{P}(n \leq X < n+1) \\ &= \int_n^{n+1} e^{-t}dt \text{ car } n \leq n+1 \\ &= \left[-e^{-t}\right]_n^{n+1} \\ &= -e^{-(n+1)} + e^{-n} \\ &= e^{-n}\left(1 - \frac{1}{e}\right) \end{aligned}$$

Conclusion :

$$\boxed{\forall n : \mathrm{P}(Y=n) = \left(1-\frac{1}{e}\right)e^{-n}}$$

Les valeurs de $Y+1$ sont $\mathbb{N}^*$

Et pour $n \in \mathbb{N}^* : (Y+1=n) = (Y=n-1)$ et

$$\begin{aligned}\mathrm{P}(Y+1=n) &= \left(1-\frac{1}{e}\right)e^{-n+1} \\ &= \left(\frac{1}{e}\right)^{n-1}\left(1-\frac{1}{e}\right)\end{aligned}$$

Conclusion : $\boxed{Y+1 \hookrightarrow \mathcal{G}\left(1-\frac{1}{e}\right)}$

$Y+1$, loi géométrique est celle du premier succès. Donc Y est le nombre d'expériences **avant** le premier succès dans une suite d'expériences indépendantes dont la probabilité de succès est $1-\frac{1}{e}$.

On continue tant que l'on a échec (probabilité $\frac{1}{e} = e^{-1}$)

```
program eml2007 ;
   var y :integer ; u :real ;
   begin
        randomize ;
        u :=random ; y :=0 ;
        while u<exp(-1) do
        begin y :=y+1 ; u :=random end ;
        writeln('y vaut ', y) ;
   end.
```