CHAPITRE

Chapitre 4

Structures de contrôle itératives

Objectifs :

- Utiliser les structures itératives complètes et les structures itératives à condition d'arrêt pour résoudre des problèmes.

- Savoir choisir la structure itérative adéquate pour résoudre un problème.

- Résoudre des problèmes récurrents.

Plan du chapitre :

Leçon 1

Structures de contrôle iteratives complètes

Objectifs spécifiques :

- Utiliser la structure itérative complète pour résoudre des problèmes.
- Résoudre des problèmes récurrents.

Plan de la leçon :

Leçon 1

Les structures de contrôle iteratives complètes

Les concepts sont dits supérieurs s'ils ont sous eux d'autres concepts, qui par rapport à eux sont des concepts inférieurs.

KANT

I. Rappel et définition

Dans la leçon précédente relative aux structures de contrôle conditionnelles, nous avons permis à un programme de prendre des décisions. Nous allons reprendre dans cette leçon, la structure permettant de faire répéter un traitement par l'ordinateur. Cette structure de contrôle s'appelle structure de contrôle itérative complète. Elle est utilisée pour exprimer une répétition d'un traitement un nombre fini de fois.

I.1 Définition

Une structure de contrôle itérative complète exprime la répétition d'un traitement un nombre de fois connu d'avance.

Nous allons voir dans la suite comment exprimer cette structure.

I.2 Vocabulaire et syntaxe

Au cours de l'analyse du problème, l'objet R a une définition itérative complète. On sait aussi que n est le nombre de répétitions. On l'exprime ainsi :

R = [Initialisation] **Pour** c **de** 1 à n **faire**
Traitement
FinPour

La séquence "Traitement" est composée d'instructions ordonnées par ordre d'exécution. On pourra l'exprimer aussi sous la forme :

R = [Initialisation] **Pour** c **de** 1 à n **faire**
Instruction 1
Instruction 2
:
Instruction p
FinPour

On remarque que R est la répétition de ces p instructions. Le nombre de ces répétitions est égal à n. La variable c a servi de compteur qui est automatiquement incrémenté par défaut de 1 par la définition de la structure elle-même.

L'initialisation délimitée par les crochets et placée devant la structure Pour, comporterait les éventuelles définitions nécessaires au bon fonctionnement de la structure itérative. Elle comporte essentiellement les objets, indépendants du compteur de l'itération.

La forme de la structure aura donc la forme suivante :

R = [Inst1, Inst2, …,Instm] **Pour** c **de** 1 à n **faire**
Instruction 1
Instruction 2
.
.
.
Instruction p
FinPour

NB :
1. Le compteur c est de type scalaire (entier, caractères, booléen, etc.).
2. L'incrémentation de c est automatique et fait progresser au successeur de la valeur en cours.

I.3 Traduction en Pascal

```
PROGRAM nom_programme;
USES WINCRT;
VAR   .
      .
      .
BEGIN
      .
      .
      .
   {Instructions d'initialization}
   FOR i:=1 TO n DO
      BEGIN
           .
           Instructions du traitement à répéter
           .
      END;
END.
```

Activité 1

Écrire un programme qui saisit un entier naturel n suivi de n réels à mettre dans un tableau M. Le programme affiche ensuite toutes les valeurs de M supérieurs à 10.

Analyse

	NOM : Supérieurs_à_10	
S	**L.D.E.**	**O.U.**
2 1 3	**Résultat** = sup_10 sup_10 = [ ] **Pour** i **de** 1 **à** n **faire** [ ]**Si** M[i]>10 **Alors** Ecrire (M[i]) **FinSI** **FinPour** M = [**Lire**("n=",n] **Pour** i **de** 1 **à** n **faire** Lire(M[i]) **FinPour** i = compteur **Fin Supérieurs_à_10**	i n M

Tableau de déclaration des objets

Objet	Type/nature	Rôle
i	Entier	Nombre d'itérations
n	Entier	Compteur
M	Tableau de réels	Contient les n réels

Algorithme

0) Début Supérieurs_à_10
1) [Lire("n=",n] Pour i de 1 à n faire
Lire(M[i])
FinPour
2) Pour i de 1 à n faire
[ ]Si M[i]>10 alors
Ecrire (M[i])
FinSI
FinPour

3) Fin Supérieurs_à_10

Traduction en Pascal

```
PROGRAM superieurs_a_10;
USES WINCRT;
VAR  i, n : INTEGER;
     M : ARRAY[1..100] OF REAL;

BEGIN
 WRITE('n=');READLN(n);
 FOR i:=1 TO n DO
        BEGIN
           READLN(M[i]);
        END;
   FOR i:=1 TO n DO
        BEGIN
             IF (M[i]>=10) THEN
                  BEGIN
                     WRITELN(M[i]);
                  END;
        END;
END.
```

I.4 Parcours décroissant

Parfois, on se trouve contraint de faire une itération avec un comptage décroissant. On exprime cette structure comme suit :

R = [Inst1, Inst2, …,Instm] **Pour** c **de** n **à** 1 (pas = -1) **Faire**
Instruction 1
Instruction 2
.
.
.
Instruction p
FinPour

NB : La décrémentation du compteur est automatique en effet, ce dernier passe au prédécesseur de sa valeur en cours. Ici, le compteur décroît de 1.

La traduction en Pascal de cette structure est de la forme :

```
{Instructions d'initialisation}
FOR i:=n DOWNTO 1 DO
     BEGIN
          ...
          Instructions du traitement à répéter
          ...
     END;
```

Remarque :
Généralement, cette exigence provient du fait que le module à répéter utilise les valeurs du compteur.

Activité 2

On se propose de chercher dans un tableau T de noms, les occurrences d'un nom v donné. On demande d'afficher les indices correspondants à ces occurrences d'une façon décroissante.

Analyse

Pour chercher les occurrences de v dans T et les afficher dans un ordre décroissant, il suffit de le parcourir à partir de la fin.

	NOM : Occurrences	
S	**L.D.E.**	**O.U.**
	Résultat = Occur	
3	Occur = [ ] **Pour** i **de** n **à** 1 pas=-1 **faire**	i
	[ ]**Si** T[i]=v **Alors**	n
	Ecrire (i , ",")	T
	FinSI	v
	FinPour	
1	T = [**Lire**("n=",n] **Pour** i **de** 1 à n **faire**	
	Lire(T[i])	
	FinPour	
	i = compteur	
2	v = Donnée ("v = ")	
4	**Fin Occrrences**	

Tableau de déclaration des objets

Objet	Type/nature	Rôle
i	Entier	Compteur
n	Entier	Nombre d'itérations
T	Tableau de chaînes de caractères de dimension 1..N_Max	Contient les n noms
N_Max	Constante = 50	Indique la dimension max de T
v	Chaîne de caractères	Le nom à chercher

Algorithme

0) Début Occurrences
1) [Lire("n=",n] Pour i de 1 à n faire
Lire(T[i])
FinPour
2) Lire("v = ",v)
3) Pour i de n à 1 (pas=-1) faire
[ ]Si T[i]=v alors
Ecrire (i, ",")
FinSI
FinPour
4) Fin Occurrences

Traduction en Pascal

```
PROGRAM Occurrences;
USES WINCRT;
VAR  N_Max =50;
VAR  i, n : INTEGER;
        v  : STRING;
T : ARRAY[1..N_Max] OF STRING;

BEGIN
WRITE('n=');READLN(n);
FOR i:=1 TO n DO
 BEGIN
      READLN(T[i]);
 END;
      WRITE('v = ');READLN(v);
  FOR i:=n DOWNTO 1 DO
      BEGIN
         IF (T[i] =v) THEN
               BEGIN
                    WRITELN(i, ',');
               END;
      END;
END.
```

Un cas d'exécution

```
n=6
Bizerte
Tunis
Sousse
Kef
Tunis
Tunis
v = Tunis
6,
5,
2,
```

Cas général :

Il y a des fois où le compteur entre dans le calcul fait par le module à répéter ; en plus les opérations de calcul exigent des valeurs non entières et progressant avec un pas p non entier. L'astuce consiste à chercher par division entière le nombre d'itérations à accomplir et avec une expression généralement linéaire ou affine, revenir au compteur dont on a besoin.

Dans ce cas, la forme générale est :

R = [Inst1, Inst2, …,Instm] **Pour** c de d **à** f (pas = p) **faire**
Instruction 1
Instruction 2
.
.
.
Instruction p
FinPour

NB.
Si p est positif, le parcours est ascendant et si p est négatif, le parcours est descendant.
Le nombre de répétitions est n =1 +E((f - d)/p) et dans ce cas le compteur effectif est c = i*p.
Remarquez que n est toujours positif, c'est le signe de p qui détermine le compteur c.

Traduction en Pascal

```
PROGRAM nom_programme;
USES WINCRT;
VAR
   .
   .
   .

BEGIN
   .
   .
   .
{Instructions d'initialization}
n:=1+ROUND((f-d)/p);
FOR i:=1 TO n DO
        BEGIN
           c:=i*p;
           ...
           Instructions du traitement à répéter
           ...
        END;
END.
```

Activité 3

On se propose de dessiner un cercle par des "*" et pour qu'il soit visible à l'écran, nous allons prendre un rayon égal à R tel que 5<R<12 . Nous rappelons que les équations paramétriques du cercle sont : x(t) = R.cos(t) et y(t) = R.sin(t), t étant un angle mesuré en radians et parcourant l'intervalle [0,2π]. Et pour que ce cercle soit tracé au milieu de l'écran, nous allons procéder à un changement d'origine de la forme suivante : x(t) = 40+R.cos(t) et y(t) = 12+R.sin(t) (l'écran étant de dimension 80x25).
Ecrire un programme qui dessine un cercle avec des "*".

	NOM : Cercle	
S	**L.D.E.**	**O.U.**
	Résultat = cercle	t
2	cercle = [ ] **Pour t de** 0 **à** 2π (pas=p) **faire**	p
	cx=ARRONDI(40+R.COS(t))	cx
	cy=ARRONDI(12+R.SIN(t)	R
	Ecrire(TAB(cx,cy),"*")	cy
	FinPour	
	t = compteur	
	p = 0.01 (Constante)	
1	R = Donnée("Rayon = ")	
3	**Fin cercle**	

Objet	Type/nature	Rôle
t	réel	Angle
cx	Entier	Abscisse du point en cours
cy	Entier	Ordonnée du point en cours
p	Constante réelle = 0.01	Le pas angulaire
pi	Constante réelle = 3.14	Valeur approchée à 0.01 près de π
R	Réel	Le rayon du cercle à tracer

Nous remarquons dans l'analyse faite que le compteur est réel ; ce genre de compteur ne convient pas avec la définition que nous avons présentée précédemment. Nous allons remédier à cette défaillance en calculant d'abord le nombre d'itérations et en utilisant t comme angle auquel on ajoute p à chaque itération.
Le nombre d'itérations est E(2π/p)+1.
De l'analyse précédente, nous obtenons l'algorithme suivant en tenant compte de la légère modification sur le compteur.

Algorithme

0) Début cercle
1) lire(R)
*2) [n←1+ARRONDI(2*pi/p), t← -p] Pour i de 1 à n faire*
t←t+p
cx← ARRONDI((40+R.COS(t))
cy← ARRONDI((12+R.SIN(t))
Ecrire(TAB(cx,cy),"")*
FinPour
3) Fin cercle

Traduction en Pascal

```
PROGRAM cercle;
USES WINCRT;
   CONST pi =3.14;
         p = 0.01;
   VAR   i, n, cx, cy: INTEGER;
         t,R:REAL;
BEGIN
 READLN(R);
 n := 1+ROUND(2*pi/p);
 t:= -p;
 FOR i:=1 TO n DO
   BEGIN
     t:=t+p;
     cx:=40+ ROUND(R*COS(t));
     cy:=12+ ROUND(R*SIN(t));
     GOTOXY(cx, cy);
     WRITE("*");
    END;
END.
```

Exercice corrigé 1

Ecrire un programme qui saisit un mot composé de lettres en majuscules et remplace chacune de ses lettres d'ordre pair par sa symétrique dans la liste de l'alphabet par rapport au milieu. Par exemple, la lettre A sera remplacée par la lettre Z et réciproquement, la lettre J par la lettre Q et réciproquement.
A la fin, le programme affiche la transformée de cette chaîne.

Analyse

Nous tenons à remarquer qu'une chaîne de caractères est vue comme un tableau de caractères. Nous allons exploiter cette propriété dans la résolution du problème.

	NOM : mot_transformé	
S	**L.D.E.**	**O.U.**
2 1 3	**Résultat** = Ecrire("Mot transformé : ",mot_t) mot_t = [Lire("mot = ", mot), mot_t←mot, L ←LONG(mot)] **Pour** c **de** 2 **à** l (pas=2) **faire** code ←ORD(mot[c]) mot_t[c] ← CHR(2*x0 - code +1) **FinPour** **Fin mot_transformé**	mot_t mot L c x0 code

Remarques :

- le mot transformé mot_t est initialisé à mot car il y a que les caractères pairs qui seront modifiés.
- Comme le compteur est automatiquement incrémenté de 1, nous allons utilser un autre compteur alant de 1 à L DIV 2 et c sera égal à 2i.

Tableau de déclaration des objets

Objet	Type/nature	Rôle
mot_t	Chaîne de caractères	Le mot transformé
mot	Chaîne de caractères Entier	Le mot initial
L	Entier	La longueur du mot
c	Entier	Compteur
i	Entier	Compteur
code	Entier	Code du caractère en cours
x0	Constante x0=77	Ordre ASCII de "M" milieu de "A".."Z"

Algorithme

0) Début mot_transformé
1) [Lire("mot = ", mot), mot_t←mot, L ←LONG(mot)] Pour i de 1 à L DIV 2 faire
c←2.i
code← mot[c]
*mot_t[c]← CHR(2*x0 - code+1)*
FinPour
2) Ecrire("Mot transformé : ",mot_t)
3) Fin mot_transformé

Traduction en Pascal

```
PROGRAM mot_transformé;
USES WINCRT;
   CONST x0=77 ;
   VAR i, L, c, code : INTEGER;
        mot,mot_t :STRING;

BEGIN
WRITE('mot = '); READLN(mot);
Mot_t:=mot;
L:= LENGTH(mot);
FOR i:=1 TO (L DIV 2) DO
        BEGIN
          c:=2*i;
          code:=mot[c];
          mot_t[c]:=CHR(2*x0-code+1);
        END;
WRITELN('Mot transformé : ',mot_t);
END.
```

Un cas d'exécution

```
mot = INFORMATIQUE
Mot transformé : IMFLRNAGIJUV
```

II. Les itérations complètes récurrentes

Souvent, certains résultats sont définis d'une façon récurrente. Dans de tels cas, le résultat se forme au fur et à mesure et à une étape donnée, il dépend d'un certain nombre de résultats précédents. Une telle structure est dite itérative récurrente. En fait, les résultats intermédiaires sont gérés par une relation de récurrence. Si cette relation lie deux éléments successifs, on dit que la récurrence est d'ordre 1, si elle lie trois éléments successifs, on dit qu'elle est d'ordre 2, etc.

Activité 13

Ecrire un programme qui calcule la factorielle d'un entier naturel n et l'affiche à l'écran.

Analyse

Remarquons que le calcul de la factorielle a une définition itérative complète et à l'étape i, la factorielle est égale à sa valeur à l'étape i-1 multipliée par i.

	NOM : Factorielle	
S	**L.D.E.**	**O.U.**
3 2 1 4	**Résultat** = Ecrire (n,"! = ", fact) fact = [fact ← 1] **Pour** i **de** 1 **à** n **faire** fact ← fact * i **FinPour** n = DONNEE("n = ") **Fin Factorielle**	n i fact

Objet	Type/nature	Rôle
n	Entier	Le nombre dont on veut calculer la factorielle
fact	Entier	la valeur de n!
i	Entier	Un compteur

Algorithme

0) Début Factorielle
1) Lire ("n = ", n)
2) [fact←1] Pour i de 1 à n Faire
*fact ← fact * I*
FinPour
5) Ecrire(n,"! = ",fact)
6) Fin Factorielle

Traduction en Pascal

```
PROGRAM Factorielle;
USES WINCRT;
VAR n, I, fact :INTEGER;

BEGIN
  WRITE('n = '); READLN(n);
  fact := 1;
  FOR I:=1 TO n DO
      BEGIN
        fact :=fact*I;
      END;
  WRITELN(n,'! = ',fact);
END.
```

Un cas d'exécution

```
n = 5
5! = 120
```

Exercice corrigé 2

n individus passent un examen composé de 3 épreuves de coefficients respectifs 2, 2 et 4. Après la correction, on se propose de faire le calcul des moyennes, la moyenne générale par épreuve, la moyenne générale de toutes les épreuves pondérées et le nombre d'admis sachant qu'un candidat est déclaré admis s'il obtient une moyenne supérieure ou égale à 10 .

Analyse

Pour résoudre ce problème, nous allons choisir quatre tableaux, un pour chacune des trois épreuves (N1, N2, N3) et un dernier pour la moyenne (M).

	NOM : Les_admis	
S	**L.D.E.**	**O.U.**
5	**Résultat** = (Ecrire ("Nombre d'admis : ", n_ad), "Moyenne Générale = ", MG)	n_ad MG
4	n_ad = [n_ad ← 0] **Pour** i **de** 1 **à** n **faire** [ ] **Si** (M[i]e10) **alors** n_ad ← n_ad +1 **FinSi** **FinPour**	i n M
3	MG ← TOT_M / n	
2	Tot_M = [Tot_M ← 0] **Pour** i **de** 1 **à** n **faire** Tot_M ← Tot_M +M[i] **FinPour**	TOT_M N1 N2
1	M = [Lire("n=",n)] **Pour** i **de** 1 à n **faire** Lire("Note 1 =",N1[i]) Lire("Note 2 =",N2[i]) Lire("Note 3 =",N3[i]) M[i] ← (2 * N1[i]+2 * N2[i]+4 * N3[i])/8 **FinPour**	N3
6	**Fin Les_admis**	

Tableau de déclaration d'un nouveau type

Type
Vecteur : tableau [1..50] de réels

Objet	Type/nature	Rôle
n_ad	Entier	Nombre d'admis
MG	Réel	Moyenne générale
i	Entier	Compteur
n	Entier	Nombre de candidats
M	Vecteur	Tableau comportant les moyennes des cadidats
TOT_M	Réel	Le total des moyennes
N1	Vecteur	Le tableau de la 1ère note
N2	Vecteur	Le tableau de la 2ème note
N3	Vecteur	Le tableau de la 3ème note

Algorithme

0) Début Les_admis
1) Lire ("n = ", n) Pour i de 1 à n faire
Lire("Note 1 =",N1[i])
Lire("Note 2 =",N2[i])
Lire("Note 3 =",N3[i])
*M[i] ← (2 * N1[i]+2 * N2[i]+4 * N3[i])/8*
FinPour
2) [Tot_M←0] Pour i de 1 à n faire
Tot_M ← Tot_M +M[i]
FinPour
3) MG ←TOT_M/n
4) [n_ad ← 0] Pour i de 1 à n faire
Si (M[i]e10) alors
n_ad ← n_ad +1
FinSi
FinPour
5) Ecrire ("Nombre d'admis : ", n_ad), "Moyenne Générale = ", MG)
6) Fin Les_admis

Traduction en Pascal

```
PROGRAM Les_admis;
USES WINCRT;
TYPE vecteur = ARRAY[1..50] OF REAL;
VAR n_ad, n, i, fact :INTEGER;
MG, TOT_M : REAL;
N1, N2, N3, M  : vecteur;
BEGIN
   WRITE('n = ');(READLN(n);
      FOR i:=1 TO n DO
```

```
    BEGIN
         WRITE('Note 1 = ');READLN(N1[i]);
         WRITE('Note 2 = ');READLN(N2[i]);
         WRITE('Note 3 = ');READLN(N3[i]);
         M[i]:=(2* N1[i]+2* N2[i]+4*N3[i])/8;
       END;
    TOT_M:=0;
    FOR i:=1 TO n DO
         BEGIN
             TOT_M:=TOT_M+M[i];
         END;
      MG := TOT_M / n;
      n_ad:=0;
    FOR i:=1 TO n DO
         BEGIN
              IF (M[i]>=10) THEN
                BEGIN
                   n_ad:=n_ad+1;
                END;
         END;
    WRITELN('Nombre d''admis : ',n_ad),'Moyenne générale = ',MG);
END.
```

Un cas d'exécution

```
n = 3
Note 1 = 12
Note 2 = 13
Note 3 = 14
Note 1 = 6
Note 2 = 8
Note 3 = 9
Note 1 = 11
Note 2 = 15
Note 3 = 18
Nombre d'admis : 2   Moyenne générale = 12.25
```

Retenons

- Une itération complète consiste en la répétition d'un traitement un nombre de fois connu d'avance.
- Dans une itération complète, le compteur est incrémenté ou décrémenté automatiquement.
- Une itération est dite récurrente si le traitement à l'étape i dépend des étapes qui le précèdent. En général, les résultats intermédiaires sont gérés par une relation de récurrence. En fait, on s'intéresse au résultat final de l'itération.

Exercices

Exercice 1

Écrire un programme qui défile le nom et le prénom de l'utilisateur de la droite vers la gauche sur la ligne du milieu de l'écran.

Exercice 2

Écrire un programme qui fait tourner des étoiles sur les quatre bords de l'écran. C'est en fait une simulation d'un jeu de lumière

Exercice 3

Écrire un programme qui dessine sur deux lignes de l'écran, deux caractères "*" sachant que l'un se déplace de la gauche vers la droite et l'autre se déplace de la droite vers la gauche.

Exercice 4

Reprendre le programme de l'exercice 1 et transformer le défilement de telle façon que la chaîne formée du nom et du prénom rentre par la droite et sort par la gauche. Temporiser ce défilement en provoquant un ralentissement sensible.

Exercice 5

Écrire un programme qui permet de trouver puis d'afficher tous les entiers à deux chiffres ayant le chiffre des unités divisible par le chiffre des dizaines.
Exemple : 24 vérifie cette propriété, en effet, 2 divise 4.

Exercice 6

Écrire un programme qui affiche tous les entiers positifs de deux chiffres tels que la somme de ses deux chiffres divise leur produit.

Exercice 7

Écrire un programme qui saisit un entier naturel n non nul et supérieur à 20 puis remplit un tableau R par n réels aléatoires compris entre 0 et 20 en utilisant la fonction HAZARD préprogrammée. Ensuite, Il calcule la moyenne arithmétique de ces n réels et affiche les éléments de R supérieurs ou égaux à MG ainsi que leur nombre, il en fait de même pour les éléments de R inférieurs à MG.

Exercice 8

Écrire un programme Pascal intitulé **OCCURENCE** qui permet de saisir une chaîne de caractères CH puis d'afficher les occurrences des voyelles qui figurent dans CH.

Exemple :
Si CH ='LYCEE 25 juillet'
Le programme OCCURENCE affichera les résultats suivants :

L'occurence de 'E' est 3
L'occurence de 'Y' est 1
L'occurence de 'U' est 1
L'occurence de 'I' est 1

Remarque : la recherche de l'occurrence ne fait pas de distinction entre les voyelles majuscules et minuscules.

Exercice 9

Écrire un programme qui saisit un texte en français et détermine les fréquences des voyelles et les met dans un tableau de 6 éléments. Le programme affiche ensuite la fréquence de chacune des voyelles.
Le texte pourra comporter des lettres en majuscules ou en minuscules ainsi que les caractères accentués.

Exercice 10

On dispose de deux tableaux T1 et T2 contenant respectivement n et m entiers positifs et non nuls.
On désire chercher dans T2 tous les diviseurs d'un élément donné de T1.

Exemple :

T1	23	15	10	277	300	34
	1	2	3	4	5	6
T2	3	6	5	1		

Si indice = 2 alors 3, 5 et 1 seront affichés à l'écran.

Écrire un programme Pascal qui permet de saisir les deux tableaux T1 et T2 et l'indice d'un élément p de T1 puis d'afficher à l'écran tous les diviseurs de p figurant dans T2.

Exercice 11

Soit un tableau T1 de n éléments ($1 \leq n \leq 100$) . Les éléments de T1 sont des entiers naturels de trois chiffres.
On se propose de remplir un tableau T2 de la façon suivante :
T2[i] est égal à la somme des carrés des chiffres de T1[i].

Exemple :

Si T1[i] = 254 alors T2[i] = 22 + 52 + 42 = 45

Écrire un programme Pascal qui permet de saisir les éléments de T1, de remplir puis d'afficher le tableau T2.

Exercice 12

Écrire un programme Pascal qui saisit un tableau A de n chaînes de caractères, cherche et affiche la longueur de la chaîne la plus longue puis toutes les chaînes ayant cette longueur.

Leçon 2

Structures de contrôle iteratives à conditions d'arrêt

Objectifs spécifiques:

- Utiliser la structure itérative à condition d'arrêt pour résoudre des problèmes.
- Résoudre des problèmes récurrents basés sur des itérations à condition d'arrêt.

Plan de la leçon

Leçon 2

Les structures de contrôle iteratives à condition d'arrêt

Ce n'est pas assez de faire des pas qui doivent un jour conduire au but, chaque pas doit être lui même un but en même temps qu'il nous porte en avant.

GOETHE

I. Introduction

Nous revenons dans cette leçon à une structure de contrôle permettant à l'ordinateur de répéter un traitement donné. Nous avons vu le cas des répétitions dont le nombre est connu d'avance. Nous l'avons appelé : structure itérative complète. Cependant, plusieurs problèmes nécessitent les répétitions d'un traitement dont on ne connaît pas le nombre. C'est une condition qui doit gérer l'arrêt des répétitions.
Nous allons vous présenter des activités auxquelles vous êtes relativement familiarisées.

Activité 1

Une personne en possession d'une carte interbancaire (CIB) se présente à un distributeur pour retirer de l'argent. Cette personne a des doutes concernant son code secret. Le système ne lui permet que trois essais au maximum.
Ecrire les étapes du programme permettant la lecture du code secret.

Réponse :

En effet, dès que la personne aura mis sa carte dans le distributeur, il va entrer dans une boucle qui va lui permettre la saisie de son code. Si la première saisie est fausse, le distributeur va lui donner une seconde chance. Si après cette chance, le code saisi est encore erroné, le distributeur donne la troisième et la dernière chance, après quoi, la carte sera confisquée si le code n'est pas encore correct.
Nous pouvons formuler cette action par :

```
[Entrer la carte] Répéter
                      Incrémenter l'essai
                      Saisir le code
                  Jusqu'à (code correct) ou (essai=3)
```

Activité 2

Vous êtes dans une station de transport public à attendre le moyen de transport désiré. Ce dernier arrive, le convoyeur procède à faire monter les voyageurs avec la condition qu'aucun ne doit voyager debout.
Écrivez l'action faite par le convoyeur.

Réponse :
Dans ce cas et en arrivant à une station, le convoyeur avertit le chauffeur d'ouvrir les portes s'il reste des places vacantes et commence l'action itérative suivante. Tant qu'il y a encore des places, faire monter les voyageurs.
Cette opération pourra être formulée comme suit :

```
[Arrêter le bus] Tant Que (il y a des places vacantes) faire
                     Monter voyageur
                 Fin
```

Vous avez vu que dans chacune des deux activités, nous avons utilisé une structure itérative particulière. Ce sont les deux formulations traduisant la structure itérative à condition d'arrêt.

Définition

On appelle structure de contrôle itérative à condition d'arrêt l'action qui consiste à répéter un traitement donné et que l'arrêt est géré par une condition.
Il existe deux formulations pour traduire une telle structure :
La structure Répéter …. Jusqu'à …
et la structure : Tant Que … Répéter ….

II. La structure Répéter … Jusqu'à …

II.1. Vocabulaire et syntaxe

Les expressions que nous allons présenter sont utilisées dans l'analyse et dans l'algorithme.

Formulation 1 : La structure Répéter … Jusqu'à …
On suppose que R est un objet dont la définition est itérative à condition d'arrêt et que la formulation 1 s'adapte le mieux, nous traduisons ce fait par :

```
R = [Initialisation ]   Répéter
                            Traitement
                        Jusqu'à (Condition darrêt)
```

Remarques :

- La définition de R commence par une initialisation éventuelle. Elle comporte les définitions nécessaires à la marche du processus d'itérations.
- "Traitement" est le jeu d'instructions à répéter.
- "Condition d'arrêt" est une expression booléenne supervisée directement ou indirectement par le traitement répété pour la faire passer à l'état vraie afin que l'itération s'arrête.

Nous pouvons écrire plus en détail cette formulation :

R = [Inst1, Inst2, …,Instm] **répéter**
Instruction 1
Instruction 2
.
.
.
Instruction p
Jusqu'à (Condition d'arrêt)

II.2. Traduction en Pascal

```
PROGRAM nom_programme;
USES WINCRT;
VAR   .
      .
      .
BEGIN
  .
  .
  .
  {Instructions d'initialisation}
  REPEAT
      ...
      Instructions du traitement à répéter
      ...
  UNTIL (Condition d'arrêt);
  ...
END.
```

Activité 3

Écrire un programme qui cherche si une valeur v existe dans un tableau B. B est composé de n réels. v est une valeur réelle donnée. Le programme affichera un message de la forme :
"v existe dans B"
ou "v n'existe pas dans B".

Analyse

Nous allons procéder à un parcours du tableau B en affectant une variable booléenne qu'on nommera v_y_est. Cette itération à condition d'arrêt s'arrête soit quand on aura trouvé v soit qu'on aura fini de parcourir tout le tableau B.

	NOM : Recherche	
S	**L.D.E.**	**O.U.**
4	**Résultat = Écrire**(v, décision, " dans le tableau")	décision
3	décision = [décision ← " n'existe pas"] **Si** v_y_est **alors**	v_y_est
	décision ← " existe"	
	FinSI	
2	v_y_est = [Lire("v = ", v),i← 0] **Répéter**	i
	i ← i+1	B
	v_y_est ← (B[i]=v)	v
	Jusqu'à (v_y_est) OR (i=n)	n
1	B = [**Lire**("n = ", n)] **Pour** i **de** 1 à n **faire**	
	Lire(B[i])	
	FinPour	
5	**Fin Recherche**	

Tableau de déclaration des objets

Objet	Type/nature	Rôle
décision	Chaîne de caractères	Comporte la bonne expression
v_y_est	Booléen	Variable logique passant à vrai si v est dans B
v	Réel	Nombre à chercher dans le tableau
i	Entier	Compteur
B	Tableau de réels	Tableau comportant les n réels
n	Entier	Le nombre d'éléments de B

Algorithme

0) Début Recherche
1) [Lire("n=",n] Pour i de 1 à n faire
Lire(B[i])
FinPour

2) [Lire("v = ",v),i← 0] Répéter
i ← i+1
v_y_est ← (B[i]=v)
Jusqu'à (v_y_est) OR (i=n)
3) [décision ← " n'existe pas"] Si v_y_est alors
décision ← " existe"
FinSI
4) Écrire (v, décision, " dans le tableau")
5) Fin Recherche

Traduction en Pascal

```
PROGRAM Recherche;
USES WINCRT;
VAR i, n : INTEGER;
     decision : STRING;
     v_y_est : BOOLEAN;
     v : REAL;
     B : ARRAY[1..100] OF REAL;

BEGIN
  WRITE('n=');READLN(n);
  FOR i:=1 TO n DO
        BEGIN
             READLN(B[i]);
        END;
  WRITE('v = ');READLN(v);
  i:=0;
  REPEAT
    i:=i+1;
    v_y_est:=(B[i]=v)
  UNTIL (v_y_est) OR (i=n);

  decision :='n''existe pas';

  IF v_y_est  THEN
    BEGIN
      decision:=  'existe';
    END;
   WRITELN(v:5:3,'  ', decision, ' dans le tableau');
END.
```

Un cas d'exécution

```
n=5
1
5.89
6
44
124.10
v = 44
44.000  existe dans le tableau
```

Nous allons voir dans l'activité qui suit une application très fréquente de la formulation "Répéter … Jusqu'à …". En effet, elle s'adapte le mieux dans les cas de saisie de données assujetties à des conditions de contrôle.

Activité 4

Écrire un programme qui cherche la dernière position d'une valeur v dans un tableau B de n réels. L'entier n est compris entre 20 et 30. Tous les réels de B ainsi que v sont dans l'intervalle [0,20].

Analyse

Le problème ressemble étroitement à celui de l'activité 3 seulement les données sont astreintes à des conditions ; en plus, la recherche de la dernière valeur de v dans B nous oblige à faire le parcours de B en commençant par la fin pour éviter un parcours total.

	NOM : cherche_dernier	
S	**L.D.E.**	**O.U.**
6	**Résultat = Écrire** (v, décision, " dans le tableau")	décision
5	décision = [décision ← " n'existe pas"] **Si** dp≠0 **alors**	dp
	STR(dp,s)	s
	décision ← " a une dernière place " + s + " "	
	FinSI	i
4	dp = [i← n+1, dp ← 0] **Répéter**	n
	[i ← i-1] **Si** (B[i]=v) **alors**	
	Dp ← i	B
	Jusqu'à (dp≠0) OR (i=1)	
3	v = [ ] **Repéter**	
	Lire("v = ", v)	
	Jusqu'à (v>=0) et (v<=20)	
2	B = [ ] **Pour** i **de** 1 **à** n **faire**	
	[ ]**Répéter**	
	Lire(B[i])	
	Jusqu'à (B[i]>=0) et (B[i]<=20)	
	FinPour	
1	n = [ ] **Repéter**	
	Lire("n = ", n)	
	Jusqu'à n dans [20..30]	
7	**Fin cherche_dernier**	

Table de déclaration des objets

Objet	Type/nature	Rôle
décision	Chaîne de caractères	Comporte la bonne expression
dp	Entier	Dernière position de v dans B
s	Chaîne de caractères	Conversion de dp en chaîne
i	Entier	Compteur
n	Entier	Le nombre d'éléments de B
B	Tableau de réels	Tableau comportant les n réels

Algorithme

```
0) Début cherche_dernier
1) Répéter
        Lire("n=",n)
   Jusqu'à n dans [20..30]
 2) Pour i de 1 à n faire
        Répéter
              Lire(B[i])
        Jusqu'à (B[i]>=0) et (B[i]<=20)
   FinPour
3) Répéter
        Lire("v=",v)
   Jusqu'à (v>=0) et (v<=20)
4) [i←n+1, dp ← 0] Répéter
                        [i ← i-1] Si  (B[i]=v) alors
                                    dp← i
                                FinSi
                    Jusqu'à (dp≠0) OR (i=1)
5)   [décision ← " n'existe pas"] Si dp≠0  alors
                                    STR(dp,s)
                                    décision ← " a une dernière place  " + s +" "
                                FinSI
6) Écrire (v, décision, " dans le tableau")
7) Fin cherche_dernier
```

Traduction en Pascal

```
PROGRAM Recherche;
USES WINCRT;
VAR  i, n,dp : INTEGER;
     decision,s : STRING;
     v : REAL;
     B : ARRAY[1..100] OF REAL;
```

```
BEGIN

  REPEAT
     WRITE('n=');READLN(n);
  UNTIL n IN [20..30];

  FOR i:=1 TO n DO
     BEGIN
       REPEAT
       WRITE('   ');READ(B[i]);
       UNTIL (B[i]>=0 )AND(B[i]<=20);
     END;

  REPEAT
     WRITE('v=');READLN(v);
  UNTIL (v>=0 )AND(v<=20);
   dp:=0;
   i:=n+1;

  REPEAT
   i:=i-1;
     IF B[i]=v THEN
                dp:=i;
  UNTIL(dp<>0) OR (i=1);

   decision:=' n''existe pas';

  IF dp<>0  THEN
     BEGIN
       STR(dp,s);
       decision:=' a une dernière place ' +s+ ' ';
     END;

  WRITELN(v:5:3,'  ',decision, ' dans le tableau');
END.
```

Un cas d'exécution

```
n=20
  7
  6
  4
  8
  12
  15
  17
  5
  20
  14
  12
  9
  7
  6
  4
  3
  4
  2
  11
  13
v=4
4.000   a une dernière place 17  dans le tableau
```

Commentaires :

Vous avez certainement remarqué la différence entre les solutions des deux activités. En effet, pour la recherche de v dans le tableau B, nous avons utilisé une variable booléenne dans la solution du 1er problème, cette option n'est plus adéquate pour le second, nous avons besoin d'une donnée précise en l'occurrence la dernière position de v dans le tableau B.
Nous avons utilisé une fonction de la bibliothèque STR et une expression de concaténation que nous avons formulé par l'opérateur additif +. Bien sûr, cette formulation n'est pas unique, nous pouvons garder l'affichage de dp comme un entier.

II.3. Les problèmes récurrents

Nous avons vu comment sont définis les problèmes récurrents et comment les résoudre avec une itération complète. Il en est de même pour les itérations à condition d'arrêt. En effet, le résultat final se forme au fur et à mesure jusqu'à ce qu'une condition d'arrêt devienne vraie. Dans l'activité suivante, nous vous proposons un problème classique où nous allons utiliser la structure itérative à condition d'arrêt dans un problème récurrent.

Activité 5

On se propose de chercher et d'afficher la moyenne arithmétique d'une suite de réels. A priori, on ne connaît pas le nombre d'éléments de cette liste, par contre, on sait que sa fin sera marquée par le nombre négatif -1. Ce dernier n'entre pas dans le calcul de la moyenne. Le caractère récurrent réside bien entendu dans le calcul de la moyenne.

Analyse

	NOM : Moyenne	
S	**L.D.E.**	**O.U.**
3	**Résultat = Écrire** ("Moyenne = ", mg)	mg
2	mg = total/nbre	total
1	(total, nbre) = [total← 0, nbre← 0] **Répéter**	nbre
	[Lire("x= ",x)]**Si** (x>0) alors	x
	total ← total + x	
	nbre ← nbre +1	
	FinSi	
	Jusqu'à (x= -1)	
4	**Fin Moyenne**	

Table de déclaration des objets

Objet	Type/nature	Rôle
mg	Réel	La moyenne arithmétique des réels
total	Réel	Le total des réels
nbre	Entier	Le nombre de réels positifs
x	Réel	Variable servant à la saisie des réels

Algorithme

0) Début Moyenne
1) [total← 0, nbre← 0] Répéter
[Lire("x= ",x)]Si (x>0) alors
total ← total + x
nbre ← nbre +1
FinSi
Jusqu'à (x=-1)
2) mg = total/nbre
3) Écrire ("Moyenne = ", mg)
4) Fin Moyenne

Traduction en Pascal

```
PROGRAM Moyenne;
USES WINCRT;
VAR     nbre : INTEGER;
        mg, total, x : REAL;

BEGIN
  total:=0;
  nbre:=0;
  REPEAT
     WRITE('x=');READLN(x);
     IF (x>0) THEN
        BEGIN
          total:=total + x;
          nbre:=nbre + 1
        END;
  UNTIL (x=-1);
        mg:=total/nbre;
  WRITE('Moyenne = ',mg:2:);
END.
```

Un cas d'exécution

```
x=3
x=5
x=7
x=-87
x=7
x=-1
Moyenne = 5.50
```

III. La structure Tant Que … Faire …

III.1 Vocabulaire et syntaxe

On suppose que l'objet R a une structure itérative à condition d'arrêt et que nous avons opté pour la formulation "Tant Que … Faire…"

R = [Initialisation] **Tant Que** Not(Arrêt) **Faire**
Traitement
FinTantQue

Remarques :

- "Traitement" est le jeu d'instructions à répéter.
- "Not(Arrêt)" est une expression booléenne supervisée directement ou indirectement par le traitement répété pour la faire passer à l'état vrai afin que l'itération s'arrête.
- "FinTantQue" sert à délimiter les instructions qui vont former "Traitement". Remarquons que pour la 1ère formulation "Répéter …jusqu'à …", la délimitation de "Traitement" est faite naturellement par les deux termes de la structure "Répéter" et "Jusqu'à"
- La partie "Initialisation" comportera les éventuelles définitions qui serviront au processus itératif.

Remarque importante :

Dans les deux formulations, il faut faire attention aux deux cas d'entrée et de sortie de la boucle et de faire les bonnes initialisations et les bons emplacements des instructions relatives à l'avancement dans l'itération. Ceci est souvent une source d'erreurs.

Nous pouvons écrire plus en détail cette formulation :

R = [Inst1, Inst2, …,Instm] **Tant Que** Not(arrêt) **Faire**
Instruction 1
Instruction 2
.
.
.
Instruction p
FinTantQue

III.2 Traduction en Pascal

```
PROGRAM nom_programme;
USES WINCRT;
VAR      .
         .
         .
BEGIN
  .
  .
  .
  {Instructions d'initialisation}
  WHILE Condition DO
   BEGIN
        ...
        Instructions du traitement à répéter
        ...
   END;
   .
   .
   .
END.
```

Activité 6

On se propose de déterminer le PGCD de deux entiers naturels m et n.
Le principe de la recherche repose sur les propriétés suivantes :
le PGCD de m et de 0 est m et tout diviseur de m et n est aussi diviseur du reste de m par n.

Donc on va pouvoir diminuer progressivement les valeurs des couples m et n jusqu'à arriver à annuler l'un des termes. Remarquons que cette méthode de reste converge rapidement vers le résultat c'est-à-dire que si on garde m plus grand que n, le PGCD des deux entiers sera égal à m dès que n devient 0.

Remarque :
De la propriété "tout diviseur de m et n " divise leur différence et dont on pourra faire facilement une démonstration comme pour le reste, nous déduisons un autre procédé de calcul de PGCD en utilisant la différence. Généralement, la méthode de la différence converge vers 0 plus lentement que la méthode du reste. Un bon exercice : prenez quelques couples d'entiers et comparez les deux procédés.

Analyse

	NOM : calcul_pgcg	
S	**L.D.E.**	**O.U.**
3	**Résultat = Écrire** ("PGCD(",m,", ",n,") =", pgcd)	pgcd
2	pgcd = [pgcd ←m] **Si** (m=0) **alors**	m
	pgcd ← n	n
	FinSi	
1	(mf,nf) = [m=Donnée("m = "),n=Donnée("n = ")]	
	Tant Que (m * n≠0) **Faire**	
	[ ] **Si** (m>n) **alors**	
	m ← m MOD n	
	sinon	
	n ← n MOD m	
	FinSi	
	FinTantQue	
4	**Fin calcul_pgcd**	

Tableau de déclaration des objets

Objet	Type/nature	Rôle
pgcd	Entier	Le PGCD de m et n
m	Entier	Le premier entier
n	Entier	Le second entier

Algorithme

0) Début calcul_pgcd
1) [Lire("m = ",m),Lire("n = ", n)]
*Tant Que (m*n ≠0) Faire*
[ ] Si (m>n) alors
m ← m MOD n
sinon
n ← n MOD m
FinSi
FinTantQue
2) [pgcd ←m] Si (m=0) alors
pgcd ← n
FinSi
3) Écrire ("PGCD(",m,", ",n,") =", pgcd)
4) Fin calcul_pgcd

Traduction en Pascal

```
PROGRAM calcul_pgcd;
USES WINCRT;
VAR  m, n, pgcd : INTEGER;

BEGIN
  WRITE('m = '); READLN(m);
  WRITE('n = '); READLN(n);
  WHILE (m*n<>0) DO
  BEGIN
    IF (m>n) THEN
       BEGIN
          m:= m MOD n;
       END
    ELSE
       BEGIN
          n:= n MOD m;
       END;
    END;
  pgcd:=m;
  IF (m=0) THEN
    BEGIN
       pgcd:=n;
    END;
  WRITELN('PGCD(',m,',',n,') = ',pgcd);
END.
```

Un cas d'exécution

```
m = 25
n = 15
PGCD(0,5) = 5
```

Remarque :

Dans ce problème, nous avons adopté la formulation tant que car il est possible de ne pas traiter l'objet d'itération. Référerez-vous à la deuxième activité introductrice où le convoyeur est contraint de ne pas faire monter des voyageurs s'il n'y a plus de places assises.
En effet dans cette deuxième formulation (tant que), la condition est vérifiée avant d'attaquer le traitement à répéter, contrairement à la première formulation (répeter.. jusqu'à) où la condition est vérifiée après avoir exécuté une fois le traitement.
Attaquons maintenant une activité mathématique d'approximation.

Activité 7

En mathématiques, en fait, on sait résoudre peu d'équations. Par contre, il existe des théorèmes forts permettant d'assurer l'existence et parfois l'unicité d'une solution à une équation donnée. Nous en citons le théorème des valeurs intermédiaires. Exemple, dans l'intervalle [0,1], l'équation $e^{-x} - x = 0$, a une solution unique. Elle est mise en évidence en traçant les deux courbes d'équations respectives $y=e^{-x}$ et $y=x$. On se propose d'écrire un programme permettant de trouver et d'afficher une valeur approchée de cette solution à une erreur e près.

Analyse

On définit la fonction f telle que $f(x)=e^{-x} - x$ sur l'intervalle [a,b]=[0,1] (a=0 et b=1). Nous allons utiliser la méthode de dichotomie pour déterminer cette solution. Elle consiste à calculer l'image du milieu de l'intervalle en cours [a,b], on compare son signe avec f(a) en utilisant leur produit. Si ce dernier est nul, c'est en fait, l'image du milieu qui l'est et dans ce cas le problème est fini, nous avons trouvé la solution. S'il est négatif, la solution est entre a et c, il suffit de remplacer b par c? S'il est positif, la solution est dans l'autre moitié, il faut donc remplacer a par c. On continue ce travail tant que la différence en valeur absolue de a et b est supérieur à 2*e. La solution cherchée sera égale au milieu de a et b.

	NOM : calcul_racine	
S	**L.D.E.**	**O.U.**
2	**Résultat = Écrire** ("La racine est : ", (a+b)/2, " à ", e, " près.")	a
1	(a,b) = [a←0, b←1, Lire("e = ",e)]	b
	TantQue (|a-b| >2*e) **Faire**	e
	[c← (a+b)/2, p← (exp(-a)-a)(exp(-c)-c)] **Si** (p=0) **alors**	c
	a←c	p
	b←c	
	Sinon Si (p<0) **alors**	
	b←c	
	Sinon	
	a←c	
	FinSi	
	FinTantQue	
3	**Fin calcul_racine**	

Tableau de déclaration des objets

Objet	Type/nature	Rôle
a	Réel	La borne inf de l'intervalle de recherche
b	Réel	La borne sup de l'intervalle de recherche
e	Réel	La valeur de l'erreur
c	Réel	La valeur du milieu de [a,b]
p	Réel	Comporte le produit de f(a).f(c)

Algorithme

0) Début calcul_racine
1) [a← 0, b← 1,Lire("e = ",e]
TantQue (|a-b| >2.e) Faire
[c← (a+b)/2, p← (exp(-a)-a)(exp(-c)-c)] Si (p=0) alors
a←c
b←c
Sinon Si (p<0) alors
b←c
Sinon
a←c
FinSi
FinTantQue
3) Écrire ("La racine est : ", (a+b)/2, " à ", e, " près.")
4) Fin calcul_racine

Traduction en Pascal

```
PROGRAM calcul_racine;
USES WINCRT;
VAR     a, b, e, c,p : REAL;

BEGIN
  a:=0;
  b:=1;
  WRITE('e = '); READLN(e);
  WHILE (ABS(a-b)>2*e) DO
    BEGIN
      c:=(a+b)/2;
      p:=(EXP(-a)-a)*(EXP(-c)-c);
      IF p=0 THEN
                BEGIN
                  a:= c;
                  b:=c;
                END
              ELSE IF (p<0) THEN
                                BEGIN
                                     b:=c;
                                END
                              ELSE
                                BEGIN
                                     a:=c;
                                END;
    END;
  WRITELN('La racine est :',(a+b)/2:6:3,'à',e:6:3,' près.');
END.
```

Un cas d'exécution

```
e = 0.001
La racine est : 0.567 à  0.001 près.
```

Retenons

-Il existe deux formulations pour traduire une structure itérative à condition d'arrêt:

- La structure Répéter … Jusqu'à …
- La structure Tant Que … Faire …

- Dans la structure "Répéter …Jusqu'à …" , le traitement à répéter est exécuté au moins une fois car la condition d'arrêt est testé à posteriori.
- Dans la structure "Tant Que … Faire …. " le traitement à répéter peut ne pas être exécuté car la condition d'arrêt est testée à priori. En effet, dans le cas où elle est vraie dès le début, le traitement n'est pas exécuté.
- Dans l'une ou l'autre formulation, il est conseillé de vérifier à la main l'exécution pour s'assurer du bon fonctionnement de la boucle et surtout des cas limites. Il est important de bien initialiser la structure et de bien positionner les instructions à l'intérieur de la boucle.

Exercices

Exercice 1

Donner la différence entre les deux formulations traduisant une structure itérative à condition d'arrêt.

Exercice 2

Quand il s'agit de faire un contrôle lors de la saisie d'une donnée, pourquoi il est préférable d'opter pour la structure "Répéter .. Jusqu'à…"

Exercice 3

Analyser puis déduire un algorithme et le programme Pascal qui permet de chercher si une valeur v existe dans un tableau. Dans l'affirmative, le programme affiche la dernière et l'avant dernière place si elle existe.

Exercice 4

Écrire un programme qui saisit un mot en français puis il affiche la première lettre double dans l'écriture de ce mot.

Exemple :

ch = "programme"

Le programme affiche la lettre m.

Exercice 5

Écrire un programme qui saisie une phrase en français puis affiche le nombre de mots de cette phrase.

Exercice 6

Écrire un programme qui calcule le PGCD de deux entiers en utilisant la méthode de la différence.

Exercice 7

Le PPCM de deux entiers est le plus petit commun multiple des deux entiers m et n. Pour calculer le PPCM de m et n, on pourra utiliser la formule suivante :

PPCM(m,n) = m.n/PGCD(m,n)

Exercice 8

Deux entiers naturels non nuls m et n sont premiers entre eux si et seulement si ils ont un PGCD égal à 1.

Exemple :

Pour **m = 7** et **n = 4**, PGCD (7,4) = 1 donc **7** et **4** sont premiers entre eux.

En utilisant cette propriété, écrire un programme permettant de saisir deux entiers naturels non nuls m et n et d'afficher s'ils sont premiers entre eux ou non.

Exercice 9

Écrire un programme qui saisit un mot et d'en extraire la première partie du mot comportant plus de voyelles que de consonnes et de l'afficher.

Exercice 10

On dispose de deux tableaux T1 et T2 contenant respectivement n et m entiers positifs et non nuls.

On désire chercher si les éléments de T1 sont tous dans T2 ou inversement. On dira que "T1 est inclus dans T2" ou "T2 est inclus dans T1" ou "T1 et T2 non comparables"

Analyser puis déduire un algorithme et le programme Pascal qui saisit m et n et remplit deux tableaux T1 et T2 respectivement par m et n réels. Ensuite étudie la relation entre T1et T2 comme indiqué plus haut et affiche le résultat.

Exercice 11

Soit l'algorithme suivant :

```
0) Début inconnu
1) Écrire (" n = "); Lire (n)
2)   Pour i de 1 à n Répéter
        Répéter
           Lire(T[i])
        Jusqu'à T[i] Dans [0..9]
   FinPour
3) Écrire ("La plus longue croissante de T mesure ",plc)
4) Fin Inconnu
```

Questions :

1. Traduire cet algorithme en Pascal.
2. Dans **l'action 1**, ajouter les contrôles sur la saisie pour que **n** vérifie la condition suivante : 5 ≤ n ≤ 50
3. Écrire une séquence d'instructions permettant de déterminer plc, la longueur de la plus longue séquence croissante d'éléments du tableau.

Exemple : Pour le tableau T suivant :

La séquence à chercher doit retourner la valeur 3 dans ce cas.

T	2	4	3	**1**	**5**	**7**	2	8	**4**	**8**	**9**	**0**
	1	2	3	4	5	6	7	8	9	10	12	13

Exercice 12

Nous savons par son graphique qu'une fonction donnée f admet un maximum en x_0 dans un intervalle [a,b]. On se propose de calculer une valeur approchée de x_0 à e près où e est une erreur fixée d'avance. Pour ce faire, on calcule l'image f(x) pour une suite de x à un pas égal à 1/2 e jusqu'à ce que f change de monotonie. Les deux dernières valeurs de x encadrent convenablement le x_0 recherché.

Analyser puis déduire un algorithme et le programme Pascal qui détermine à e près (0.01 par exemple) une valeur approchée de x_0 où la fonction suivante f admet un minimum.

f(x) = x+1 +1/x sur l'intervalle]0,4]