Chapitre n°4 :

Les structures de contrôle itératives

Durée : 12H

Objectifs du cours :

- Boucle Pour,
- Répéter
- Tant que.

Leçon 1 La structure de contrôle itérative complète

Introduction

Exemple : Afficher sur écran les nombres de la base décimale.

```
Ecrire ("0")                    Pour compteur = 0 à 9 faire
….                                  Ecrire (compteur)
Ecrire ("9")                    Fin Pour
```

Activité 1 : Saisie d'un tableau

Ecrire un programme qui saisit un tableau T de n réel. Saisir l'entier n, on suppose que (n ≤ 100). Indiquer le rang i lors de la $i^{ème}$ saisie au niveau de l'affichage.

Pré analyse

$1^{ère}$ solution	$2^{ème}$ solution
Tr [1] = donnée Tr [2] = donnée ….. Tr [n] = donnée	Penser à utiliser une boucle qui répète automatiquement l'instruction de saisie n fois.

I – Rappel et définition :

1. Définition : (voir livre page 106)
2. Vocabulaire et syntaxe (voir livre page 106, 107)
3. Parcours décroissant (voir livre page 109)

Activité 2 :

Faire l'analyse d'un programme qui permet de calculer la paye hebdomadaire d'un ouvrier.

- La solution doit nous permettre de :
  - Saisir le nombre d'heures travaillées par jour. (Jours ouvrables : Du lundi au Vendredi)
  - Saisir la valeur du taux horaire de payement
- Puis de calculer le nombre total d'heures travaillées de la semaine et d'afficher le montant de la paye.

a. Analyse

	Nom : paye_semaine	
S	**L. D. E**	**O.U**
4	Résultat = écrire (PS)	PS
3	PS ← total * THP	total, THP
2	total = [total ← 0]	
	Pour j **de** lundi **à** vendredi **faire**	j
	H[j] = donnée ("Entrez le nombre d'heure du jour ", ord (j) +1, " : ")	H
	total ← total + H [j]	
	Fin pour	
1	THP = donnée ("Précisez la valeur du taux horaire : ")	
5	j = compteur	
	Fin paye_semaine	

T.D.N.T

Type
jours = (dimanche, lundi, mardi, mercredi, jeudi, vendredi, samedi)
jours_de_travail = lundi .. vendredi
Paye = tableau de 5 entier

T.D.O

Objet	Type	Rôle
j	jour_de_travail	Compteur représentant le jour en cours
total	octet	Total heures de la semaine
THP	Réel	Taux horaire de payement
PS	Réel	Paye de la semaine
H	paye	Tableau des heures travaillées par jour

b. Algorithme

0) Début paye_semaine
1) Ecrire ("Précisez la valeur du taux horaire : "), lire (THP)
2) total ← 0

   Pour j de **lundi** à **vendredi** faire

      écrire ("Entrez le nombre d'heure du jour ", ord(j) +1, " : "), lire (H[j])

      total ← total + H[j]

   Fin pour
3) MP ← total * THP
4) écrire ("La paye de la semaine est : ", PS, " DT")
5) Fin paye_semaine

c. Pascal (voir fichier : paye_sm.pas)

Leçon 2 Les structures de contrôle itératives à condition d'arrêt

I - La structure Répéter … Jusqu'à :

1. Définition

L'arrêt de la répétition de la boucle répéter se fait lorsque la condition d'arrêt est **vraie.** Cette condition est située à la fin de la structure. Par conséquent, le traitement sera exécuté **au moins une fois** quelque soit le résultat de la condition d'arrêt.

2. Vocabulaire et syntaxe :

En algorithmique	En Pascal
= [Initialisation] **Répéter** …… suite d'instructions …… **Jusqu'à** (condition d'arrêt)	{Initialisation}; **Repeat** …… suite d'instructions; …… **Until** (condition d'arrêt) ;

Activité 1 :

Ecrire un programme qui saisit un tableau C de n caractères et qui affiche un message indiquant si un caractère v existe ou non dans C.

a. Analyse :

	Nom : recherche	
S	L.D.E	O.U
	Résultat = Affichage	
3	Affichage = [ ] Si existe_v alors écrire (v, " Existe dans C") Sinon écrire (v, " n'existe pas dans C") Fin si	existe_v
2	existe_v = [v = donnée, i ← 1, existe_v ← faux] Répéter Si C [i] = v alors existe_v ← vrai Fin si Jusqu'à (existe_v) ou (i = n)	v i C n
1	C = [n = donnée] Pour i de 1 à n faire C [i] = Donnée (" Donner la case ", i," : ") Fin pour i = compteur	
4	**Fin** recherche	

T.D.N.T

Type
TC = tableau de 50 caractères

T.D.O

Objet	Type	Rôle
C	TC	Tableau de caractères
i	Entier	Compteur
existe_v	Booléen	
v	caractère	L'élément recherché
n	Entier	Nombre de case utilisées

b. Algorithme :

0) Début recherche

1) Ecrire ("Donner n : "), lire (n)

Pour i de 1 à n faire

C [i] = Donnée (" Donner la case ", i," : ")

Fin pour

2) Ecrire ("Donner v : "), lire (v)
i ← 1
existe_v ← faux

Répéter

Si C [i] = v alors existe_v ← vrai
Fin si

Jusqu'à (existe_v) ou (i = n)

3) Si existe_v alors écrire (v, " Existe dans C")

Sinon écrire (v, " n'existe pas dans C")

Fin si

4) Fin recherche

c. Pascal (voir fichier : rech_c.pas)

II - La structure Tant que … Faire :

Activité 2 :

Faire une analyse qui permet de déterminer le PGCD de 2 entiers a et b saisis par l'utilisateur.

a. Analyse :

	Nom : pgcd_différence	
S	L.D.E	O.U
4	Résultat = écrire (" Le plus grand diviseur commun est: ", a)	a
3	a = [ ] Tant que a ≠ b faire Si a < b alors a ← a – b Sinon b ← b – a Fin Si Fin Tant que	v i C
1	a = donnée ("Donner a : ")	n
2	b = donnée ("Donner b : ")	
5	**Fin** pgcd_différence	

T.D.O

Objet	Type	Rôle
a	Entier	
b	Entier	

b. Algorithme :

0) Début pgcd_différence
1) Ecrire ("Donner a : "), lire (a)
2) Ecrire ("Donner b : "), lire (b)
3) Tant que a ≠ b faire

   Si a < b alors a ← a – b
   Sinon b ← b – a
   Fin Si
4) écrire (" Le plus grand diviseur commun est: ", a)
5) Fin pgcd_différence

c. Pascal (voir fichier : pgcd_dif.pas)

- Méthode d'Euclide

Algorithme

0) Fonction **pgcd_euclide** (a, b : entier) : entier
1) Tant que b ≠ 0 faire
   r ← a mod b
   a ← b
   b ← r
   Fin tant que
2) pgcd_euclide ← a
3) Fin **pgcd_euclide**

T.D.O Locaux

Objet	Type	Rôle
r	Entier	Reste de la division Euclidienne

1. Définition

Le traitement n'est exécuté que lorsque la condition est vérifiée. La condition peut ne pas être vérifiée dés la première exécution de la boucle tant que, dans ce cas **le traitement ne sera jamais exécuté**.

2. Vocabulaire et syntaxe

En algorithmique	En Pascal

= [Initialisations]
Tant que (Condition d'arrêt) faire
……
suite d'instructions à répéter
……
Fin Tant que
	Initialisations ;
	while (condition d'arrêt) do
	Begin
	…… ;
	suite d'instructions;
	…… ;
	End;

Activité 3 :

Faire l'analyse d'un programme qui affiche la division entière de 2 entiers sans utiliser la fonction prédéfinie DIV.

a. Analyse :

	Nom : division2	
S	L.D.E	O.U
4	Résultat = écrire ("La division est : ", i)	i
3	i = [i ← 0] **Tant que** (r1 ≥ r2) **Faire**	r1
	r1 ← r1 – r2	r2
	i ← i + 1	
	Fin Tant que	
1	r1 = Donnée ("Donner le premier réel")	
2	r2 = Donnée ("Donner le deuxième réel")	
5	Fin division2	

T.D.O :

Objet	**Type**	**Rôle**
r1, r2	Entiers	
i	Entier	

b. Algorithme :

0) Début division2
1) Ecrire ("Donner le premier réel"), lire (r1)
2) Ecrire ("Donner le deuxième réel"), lire (r2)
3) i ← 0

   Tant que (r1 ≥ r2) Faire

   r1 ← r1 – r2
   i ← i + 1

   Fin Tant que
4) écrire ("La division est : ", i)
5) Fin division2

c. Pascal (voir fichier : div_ent.pas)

- Version 2

Faire l'analyse d'un programme qui affiche la division réelle de 2 entiers sans utiliser l'opérateur /.

Pascal (voir fichier : div_reel.pas)

- Structure itérative complète avec pas différent de 1 :

Activité 4

En utilisant une boucle pour afficher sur écran les nombres impairs compris entre 1 et 9.

Analyse		Pascal

[c ← 1]
Tant que c < =9 faire
Ecrire (c)
c ← c + 2
Fin Tant que
	Pas = 2
	Nombre d'itération = 5
		c := 1;
		while c <= 9 do
		begin
		write (c : 5);
		c:= c + 2;
		end;

Analyse	
[ ] Pour c de 1 à 9 (pas = 2) faire Ecrire (c) Fin pour	Pas = 2 Nombre d'itération = 1 + E (9 – 1) / 2 = 5

Pascal

```
n := 1 + Round ((9 – 1) / 2);
for i := 1 to n  do
begin
      c := i * 2;
      write (c – 1 : 5);
end;
```

Pascal (voir fichier : pas_pour.pas)

Cas général : voir livre (page 111)

- Les itérations complètes récurrentes :

Une **récurrence** est une relation entre plusieurs termes successifs d'une suite, qui permet de calculer le terme d'indice le plus élevé en fonction des autres. Se fonctionnement nécessite l'initialisation des premiers termes de la suite.

Exemples :

Calcul récurent d'ordre 1

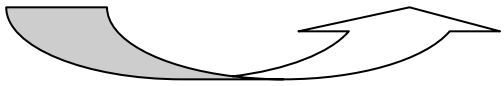
1. Somme des n premiers nombre entiers :

$$\begin{cases} U_0 = 0 \\ U_n = U_{n-1} + n \end{cases}$$

- Fonctionnement :

  Pour n = 5,

$U_5 = U_4 + 5$ $U_5 = 10 + 5 = 15$
$U_4 = U_3 + 4$ $U_4 = 6 + 4 = 10$
$U_3 = U_2 + 3$ $U_3 = 3 + 3 = 6$
$U_2 = U_1 + 2$ $U_2 = 1 + 2 = 3$
$U_1 = U_0 + 1$ $U_1 = 0 + 1 = 1$
$U_0 = 0$



Algorithme

0) Début **somme_n**
1) Lire (n)
2) cumul ← 0
3) Pour i de 1 à n faire
   cumul ← Cumul + i
   Fin pour
4) Ecrire (cumul)
5) Fin somme_n

T.D.O

Objet	**Type**	**Rôle**
i	Entier	Compteur
Cumul	Entier	Somme des n premier entiers

2. Factorielle n :

$$\begin{cases} U_0 = 1 \\ U_n = U_{n-1} \times n \end{cases}$$

3. Calcul de la puissance :

$$\begin{cases} n^0 = 1 \\ n^p = n^{p-1} \times n \end{cases}$$

Calcul récurent d'ordre 2

1. Suite de Fibonacci :

$$\begin{cases} U_1 = 1 \\ U_2 = 2 \\ U_n = U_{n-1} + U_{n-2} \quad (n \geq 3) \end{cases}$$

- $V_n = U_n / U_{n-1}$ $(n \geq 2)$

  La suite (V_n) tend vers une limite appelée nombre d'or.

Correction exercice 12 - page 142.

a. Analyse :

	Nom : recherche_x0	
S	L.D.E	O.U
3	Résultat = écrire ("F admet un minimum entre ", x0, " et ", x)	x0, x
2	x0, x = [x ← pas, x0 ← x, fx0 ← x0 +1 + (1/x0)]	pas fx0
	Répéter x ← x + pas fx ← x + 1 + (1 / x)	fx
	Si fx0 > fx **alors** x0 ← x fx0 ← fx	
	Fin Si	
	Jusqu'à (x > 4) ou (fx0 < fx)	
1	pas = Donnée ("Donner le pas : ")	
4	**Fin** recherche_x0	

T.D.O :

Objet	**Type**	**Rôle**
x	réel	
x0	Réel	
fx	réel	
fx0	réel	
Pas	réel	

b. Algorithme :

0) Début **recherche_x0**

1) Ecrire ("Donner le pas : "), lire (pas)

2) x ← pas

   x0 ← x

   fx0 ← x0 +1 + (1/x0)

   Répéter

   x ← x + pas

   fx ← x + 1 + (1 / x)

   Si fx0 > fx **alors** x0 ← x

   fx0 ← fx

   Fin Si

   Jusqu'à (x > 4) ou (fx0 < fx)

3) Ecrire ("F admet un minimum entre ", x0, " et ", x)

4) Fin **recherche_x0**

Pascal (voir fichier : min_f3.pas)