Chapitre n°5 :

Les sous programmes

Durée : 16H

Leçon 1 — L'analyse modulaire

I - Introduction

Exemple : Organisation d'une fête.

II - L'analyse modulaire

1. Définition :

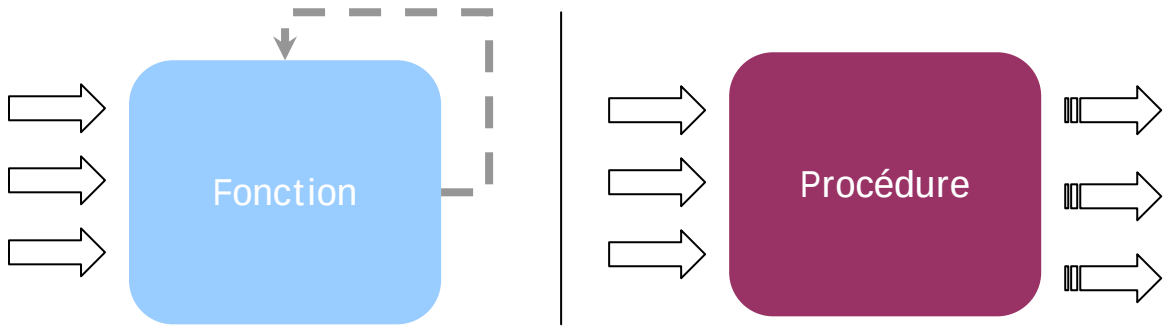
L'analyse modulaire consiste à **diviser** un problème en sous problèmes de **difficultés moindres**. Ces derniers peuvent être divisés à leur tour en d'autres sous problèmes jusqu'à ce qu'on arrive à un niveau abordable de difficulté.

2. Intérêts de l'analyse modulaire :

- Facilité la résolution du problème.
- Permettre la réutilisation d'instructions.
- Organisation du code source.
- Localiser les erreurs plus rapidement.
- Faciliter l'évolution et la mise à jour du programme.

3. Notions de sous-programme:

- Un sous-programme est un ensemble d'instructions, déclaré dans un programme ou dans un sous-programme.
- Sa peut être une procédure ou une fonction.
- Un sous-programme peut être exécuté plusieurs fois grâce à des appels.



Leçon 2

Les fonctions

I – Définition

Une fonction est un sous-programme qui renvoie une seule valeur ayant un type simple, ce type sera celui de la fonction.

Activité 1 :

Ecrire un programme qui détermine puis affiche le nombre de combinaisons de p objets parmi n, n et p sont deux entiers naturels strictement positifs ($n \geq p$).

Exemple : 5! = 120 = 5 * 4 * 3 * 2 * 1

Pré analyse :

- Afficher résultat de la combinaison
- Traitement : La combinaison $C_n^p = \frac{n!}{p! * (n - p)!}$.

  Donc il faut calculer n !, p !, (n-p) !.
- Donner : n et p

a. Analyse :

	Nom : combinaison	
S	**L.D.E**	**O.U**
6	Résultat = écrire ("combinaison de p objets parmi n = ", c)	c
5	c ← f1 div (f2*f3)	
2	f1 ← **FN** fact (n)	f1
3	f2 ← **FN** fact (p)	f2
4	f3 ← **FN** fact (n-p)	f3
1	(n, p) = [ ] Répéter	fact
	n = donnée (''donner un entier n'')	n
	p = donnée (''Donner un entier p'')	p
	jusqu'à ($n \geq p$) et ($p > 0$)	
7	Fin Combinaison	

Tableau de déclaration des objets **globaux**

Objet	Type	Rôle
p, n	Entier	
f1, f2, f3	Entier	
fact	fonction	Calcul factoriel x
c	Entier	Nombre de combinaison

Algorithme :

0) Début Combinaison
1) Répéter
   Ecrire (" Donner un entier n : "), Lire (n)
   Ecrire (" Donner un entier p : "), Lire (p)
   Jusqu'à (n ≥ p) et (p > 0)
2) f1 ← FN fact (n)
3) f2 ← FN fact (p)
4) f3 ← FN fact (n-p)
5) c ← f1 Div (f2 * f3)
6) écrire ("Combinaison de p objets parmi n = ", c)
7) Fin combinaison

b. Analyse de la fonction Fact :

	DEF FN fact (x : entier) : entier	
S	**L.D.E**	**O.U**
	Résultat = fact	
2	fact ← f	f
1	f = [f ←1] Pour c de 1 à *x* faire	c
	f ←f * c	
	Fin Pour	
3	Fin fact	

Tableau de déclaration des objets **Locaux**

Objet	Type	Rôle
c, f	Entier	

Algorithme

0) DEF FN fact (x : entier) : entier
1) f ← 1

   Pour c de 1 à *x* faire

   f ← f * c

   Fin Pour
2) fact ← f
3) Fin fact

c. Pascal (voir fichier : cnp.pas)

II - Appel d'une fonction en Pascal

L'appel de la fonction doit nécessairement apparaître dans une expression.

Exemples :

```
objet := nom_fonction (paramètres effectifs);

Write (nom_fonction (paramètres effectifs));

If nom_fonction (paramètres effectifs) = … then … ;
```

III - Définition d'une fonction en Pascal

```
Function nom_fonction (liste des paramètres formels) : Type du résultat ;
        Var
              {Déclarations des variables locales} ;
        begin
              {Instructions de la fonction} ;
              Nom_fonction := résultat ;
        end;
```

Suite Activité 1 :

Créer un module nommé saisie ( ) qui assure la saisie contrôlée de n et p.

Analyse

	Nom : combinaison (version 2)	
S	**L.D.E**	**O.U**
…	…	…
1	(n, p) = **PROC** saisie (n, p)	saisie
7	Fin combinaison	

T.D.O globaux

Objet	**Type**	**Rôle**
Saisie	procédure	

	DEF PROC saisie (**var** n, p : entier)	
S	L.D.E	O.U
1	Résultat = n, p n, p = [ ] répéter n = donnée (" Donner n : ") p = donnée (" Donner p : ")	
2	jusqu'à (p > 0) et (n ≥ p) Fin saisie	

Algorithme

0) DEF PROC saisie (var n, p : entier)

1) Répéter

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

Ecrire (" Donner p : "), lire (p)

Jusqu'à (p > 0) et (n ≥ p)

2) Fin saisie

{exemple d'un programme construit en modules}	Commentaire

```
Program Combinaison ;
Uses wincrt;

Var
        f1, f2, f3 ,p , n : integer ;
        c : integer ;
```

Déclaration du programme principal

```
Procedure saisie ( var n, p :integer);
Begin
Repeat
        Writeln('donner un entire n ');
        readln(n) ;
        Writeln('donner un entire p');
        readln(p) ;
until (n>=p) and ( p>0);
end;
```

Entête

Le corps de la procédure

Sous programme Appelé

Paramètres formels

```
Function fact (x: integer) : integer ;
Var f, c :integer;
Begin
    f:=1;
 For c :=1  to  n  Do
     Begin
            f:= f * c;
        End;
Fact:=f;
End ;
```

Entête

Déclaration

Le corps de la fonction

Sous programme Appelé

```
Begin
Saisie (n, p);  {Appel}

f1:= fact (n);   {Appel}
f2 := fact (p);   {Appel}
f3 := fact (n-p);  {Appel}

c := f1 div (f2 *f3) ;

writeln ('la combinaison de p objets parmi n est  =  ', c);
End.
```

Paramètres effectifs

Programme principal

Leçon 3 Déclaration, accès aux objets et modes de transmission

I - Déclarations et accès aux objets :

1. Les objets locaux :

Un objet Local est un objet déclaré et connu seulement à l'intérieur d'un sous-programme.

Exemples : c, f dans au niveau de la FN fact.

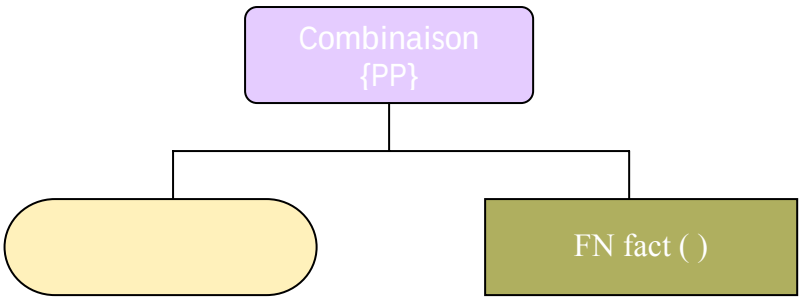
2. Les objets globaux :

Un objet Global est déclaré dans la partie déclarative du programme principal. Cet objet est utilisable par le PP et par les différents autres sous-programmes.

Exemples : f1, f2, f3 au niveau du PP.

3. Niveaux des sous programme :

On peut organiser les appels des sous-programmes en une arborescence hiérarchique. Cette arborescence s'étend de la racine (Identificateur du PP) jusqu'au dernier niveau de nœuds (sous-programme) imbriqués.



4. Accès aux objets

La porté de l'objet défini les possibilités d'accès à ce dernier à partir de différents endroits du programme. Exemple, un objet déclaré dans un sous-programme n'est pas accessible à partir du programme principal. Par contre, un objet global est accessible à partir d'un sous-programme.

Cas général : voir Définition page 162.

Attention ! : voir livre page 164.

II - Les paramètres et leurs modes de transmission :

1. Les paramètres formels et les paramètres effectifs :

- Les paramètres formels figurent dans l'entête de la définition d'un sous-programme.
- Les paramètres effectifs figurent dans l'appel d'un sous-programme.
- Les paramètres **effectifs** et les paramètres **formels** doivent s'accorder du point de vue **nombre** et **ordre**.
- Leurs **types** doivent être **identiques** ou compatibles.
- Leurs **identificateurs** peuvent être **différents.**

2. Mode de passage de paramètres :

La substitution des paramètres effectifs aux paramètres formels s'appelle passage de paramètre. Il s'agit en fait d'un transfert de données entre le PP et le sous programme appelé.

Il existe deux modes de passage de paramètres :

- Le passage de paramètres par **valeur**
  a. Le paramètre formel n'est pas précédé par le mot **VAR.**
  b. La communication est à sens unique (Appelant → appelé).
  c. La valeur du paramètre ne change pas.
  d. Les paramètres d'une fonction sont passés par valeur.

- Le passage de paramètres par **variable**
  a. Le paramètre formel est précédé par le mot VAR.
  b. La communication est à double sens (Appelant ↔ appelé).
  c. La valeur du paramètre peut se changer.

- ## Exercice de révision Chapitre 5 L1, L2, L3

Enoncé :

T étant un tableau d'entiers contenant au maximum 20 éléments. Faire l'analyse d'un programme qui permet de :

1. Saisir n.
2. Remplir le tableau T par des valeurs aléatoires comprises entre 0 et 50.
3. D'afficher la position de la case contenant la valeur minimum. Pour cela, vous devez prévoir le module suivant : pos_min ( ).

Remarques :

- Préciser la nature du module pos_min ( ).
- Compléter ses paramètres.

a. Analyse :

	Nom : recherche_minimum	
S	**L.D.E**	**O.U**
3	Résultat = écrire ("La position du minimum est : ", FN pos_min (T, n))	T
2	T = [ ] Pour i de 1 à n faire	n
	T[i] ← Aléa (51)	pos_min
	Fin pour	
1	n = [ ] Répéter	
	n = donnée ("donner un entier n : ")	
	jusqu'à (n ≥ 1) et (n ≤ 20)	
4	i = compteur	
	Fin recherche_minimum	

T.D.N.T

Type
Aléatoire = tableau de 20 entiers

T.D.O globaux

Objet	Type	Rôle
n	Entier	Taille du tableau
T	Aléatoire	Tableau d'entiers
pos_min	fonction	Recherche du minimum

Algorithme :

0) Début recherche_minimum
1) Répéter
   Ecrire (" Donner un entier n : "), Lire (n)
   jusqu'à (n ≥ 1) et (n ≤ 20)
2) Pour i de 1 à n faire
   T[i] ← Aléa (51)
   Fin pour
3) écrire ("La position du minimum est : ", FN pos_min (T, n))
4) Fin recherche_minimum

b. Analyse de la fonction pos_min :

	DEF FN pos_min (T : Aléatoire, n : entier) : entier	
S	**L.D.E**	**O.U**
2 1 3	Résultat = pos_min pos_min ← pm pm = [pm ← 1] pour i de 2 à n faire Si T [i] < T[pm] alors pm ← i Fin si Fin pour i = compteur Fin min	pm i

T.D.O **Locaux**

Objet	Type	Rôle
pm	Entier	Position du minimum
i	Entier	compteur

Algorithme :

0) DEF FN pos_min (T : Aléatoire, n : entier) : entier
1) pm ← 1
   Pour i de 2 à n faire
   Si T [i] < T[pm] alors pm ← i
   Fin si
   Fin pour
2) pos_min ← pm
3) Fin pos_min

c. Pascal (voir fichier : pos_min.pas)

Leçon 4

Les procédures

I – Définition

Une procédure est un sous-programme qui peut produire zéro ou plusieurs résultats.

Activité 2 :

Soit T un tableau de n caractères ($2 \leq n \leq 20$). On propose d'écrire un programme qui saisit n et T, en suite il appel une procédure Occurrence qui cherche le nombre de caractères alphabétiques et le nombres de chiffres dans T.

a. Analyse :

	Nom : calcul_caractères	
S	**L.D.E**	**O.U**
4	Résultat = écrire ("Le nombre de caractère alphabétique est : ", L, "Le nombre de chiffre est : ", C)	L, C
3	L, C = PROC compter (T, n, L, C)	T, n compter
2	T = PROC remplir (T, n)	remplir
1	n = PROC saisie (n)	saisie
5	Fin calcul_caractères	

T.D.N.T

Type
Tab_c = tableau de 20 caractères

T.D.O globaux

Objet	Type	Rôle
n	Entier	Taille du tableau
T	Tab_c	Tableau de caractères
L, C	Entier	Les compteurs alphabétiques, chiffres
Compter	Procédure	Calcul de L et de C
Remplir	Procédure	Remplissage de T
Saisie	Procédure	Saisie contrôlée de n

Algorithme :

0) Début calcul_caractères
1) PROC saisie (n)
2) PROC remplir (T, n)
3) PROC compter (T, n, L, C)
4) écrire ("Le nombre de caractère alphabétique est : ", L, "Le nombre de chiffre est : ", C)
5) Fin calcul_caractères

b. Analyse de la procédure saisie :

	DEF PROC saisie (var n : entier)	
S	**L.D.E**	**O.U**
1	Résultat = n n = [ ] répéter n = donnée ("Donner le nombre de case du tableau : ") jusqu'à (n ≤ 20) et (n ≥ 2)	
2	Fin saisie	

Algorithme :

0) DEF PROC **saisie** (var n : entier)
1) Répéter
   Ecrire ("Donner le nombre de case du tableau : "), Lire(n)
   Jusqu'à (n ≤ 20) et (n ≥ 2)
2) Fin **saisie**

c. Analyse de la procédure remplir :

	DEF PROC remplir (var T: tab_c, n : entier)	
S	**L.D.E**	**O.U**
1	Résultat = T T = [ ] pour i de 1 à n faire T[i] = donnée ("donner la case n° ", i " : ") Fin pour i = compteur	i
2	Fin remplir	

Algorithme :

0) DEF PROC **remplir** (var T: tab_c, n : entier)
1) Pour i de 1 à n faire
   Ecrire ("donner la case n° ", i " : "), Lire (T[i])
   Fin pour
2) Fin **remplir**

T.D.O **Locaux**

Objet	Type	Rôle
i	Entier	compteur

d. Analyse de la procédure compter :

	DEF PROC compter (T: tab_c, n : entier, var L, C : entier)	
S	**L.D.E**	**O.U**
1	Résultat = L, C L, C = [L ← 0, C ← 0] pour i de 1 à n faire Si Majus (T[i]) dans ["A".."Z"] alors L ← L + 1 Sinon Si T[i] dans ["0".."9"] alors C ← C + 1 Fin si Fin pour i = compteur	i
2	Fin compter	

Algorithme :

0) DEF PROC **compter** (T: tab_c, n : entier, var L, C : entier)
1) L ← 0
   C ← 0
   Pour i de 1 à n faire
   Si Majus (T[i]) dans ["A".."Z"] alors L ← L + 1
   Sinon Si T[i] dans ["0".."9"] alors C ← C + 1
   Fin si
   Fin pour
2) Fin **compter**

T.D.O **Locaux**

Objet	Type	Rôle
i	Entier	compteur

e. Pascal (voir fichier : calc_car.pas)

II – Les procédures non paramétrées

Créer en pascal une procédure nommée cercle. Cette dernière dessine au centre de la fenêtre d'exécution un cercle d'étoile de rayon 10 (caractères).

Principe :

- Le cercle sera constitué d'étoile. Les coordonnées des étoiles seront calculées selon les formules suivantes :
- Pour un angle a donnée, $x = \cos(a) \times rayon + 40$

$$y = \sin(a) \times rayon + 12$$

- On vous rappel que les coordonnées du centre de la fenêtre d'exécution sont (40,12).
- Les angles calculés varient entre 0 et 2π avec un pas constant.

Solution :

```
procedure cercle;
const
    r=10;
    p=0.1;
var
  a: real;
begin
    a:=0;
    repeat
        gotoxy(round(cos(a)*r)+40,round(sin(a)*r)+12);
        write('°');
        a:=a +p;
    until (a > 2*pi);
end;
```

Pascal (voir fichier : cercle.pas)