LYCEE Secondaire Houmt Souk 📖 DISCIPLINE : Informatique	PROFESSEUR: Mr Bayahi Raouf
Chapitre 5	LES SOUS PROGRAMMES

I. INTRODUCTION

Afin de faciliter la résolution d'un problème complexe et / ou de grande taille, on doit le décomposer en sous problèmes élémentaires, indépendants et de taille réduite.

II. ANALYSE MODULAIRE

II.1 Définition

L'analyse modulaire consiste à diviser un programme en sous-programmes ou en modules.

II.2 Intérêts

- Le programme sera plus lisible, plus facile à documenter, à mettre au point et à maintenir (modifier par la suite).
- On obtient des sous-programmes de tâches claires, précises et explicites.

Un sous programme peut être une fonction ou une procédure.

III. FONCTIONS

III.1 Introduction

Nous avons déjà été amené à utiliser des « fonctions prédéfinies » telles que SQR, ABS, CHR, ORD, COPY,…Cette notion de fonction est très proche de la notion mathématique classique.

III.2 Activité

Ecrire un programme Pascal nommé Calcul_combinaiqon qui saisit deux entiers n et p tel que 1<=p<=n sachant que C(p,n)=n!/(p!*(n-p)!) et appelle une fonction factorielle qui calcule chaque factoriel

Analyse du programme principal

```
3   Résultat = Ecrire (C)
2   C← FN Factorielle (N)/( FN Factorielle (P) *FN Factorielle (N-P))
1   N,P =[] Répéter
        N,P= donnée     ("Taper 2 entiers ")
        Jusqu'à (P>=1)et(N>=P)
4   Fin Calcul_combinaison
```

Tableau de déclaration des objets globaux

Objet	Nature / Type	Rôle
N, P,C	Variables/ Entier	
Factorielle	Fonction	

Algorithme du programme principal

```
0/ Début Calcul_ combinaison
1/  Répéter
      ecrire(«taper 2 entiers »)
      lire(N,P)
    Jusqu'à (P>=1)et(N>=P)

2/ C← FN Factorielle (N)/( FN Factorielle (P)
*FN Factorielle (N-P))
3) Ecrire (C)
4/ Fin Cacul_ combinaison
```

Analyse de la fonction Factorielle

```
    DEF FN Factorielle  (x : entier) : entier
     Résultat = Factorielle
2    Factorielle ← F
1    F= [ F←1] Si x<>0 alors
                Pour i de 1 à x faire
                        F← F*i
                FinPour
            Finsi
3   Fin Factorielle
```

Algorithme de la fonction Factorielle

0/ DEF FN Factorielle (x : entier) : entier
1/ F←1
Si x<>0 alors
Pour i de 1 à x faire
F← F*i
FinPour
Finsi
2/ factorielle ← F
3/ Fin Factorielle

Tableau de déclaration des objets locaux

Objet	Nature / Type	rôle
I	Entier	Compteur
f	Entier	FACTORIELLE X

Traduction en Turbo Pascal

```
PROGRAM Calcul_combinaison;
USES  WINCRT;
VAR
N,P, F : longint ; {F et N sont des variables globales}
C:real;

FUNCTION Factorielle (x: integer): integer;
VAR
  i  :integer ; { i est une variable locale}
BEGIN
    IF x<=1 then F:=1
ELSE
       Begin
              F:=1;
              For  i:=2 to x do
                     Begin
                            F:= F*i;
                     End;
End;
Factorielle:= F;
END;
BEGIN {programme principal}
Repeat
    WRITELN ('Taper un entier p');
    READLN (p);
    WRITELN ('Taper un entier n');
    READLN (n);
Until (p>=1)and(n>p) ;
C:= Factorielle (N)/(Factorielle (P)*Factorielle (N-P));
Writeln (C:5:2) ;
END.
```

III.3 Définition

Une fonction est un sous-programme qui possède un ou plusieurs paramètres (arguments) et qui renvoie **qu'une seule valeur de type simple** (entier, réel, caractère, booléen, chaîne de caractères) au programme appelant.

III.4 Déclaration d'une fonction

- ***En analyse***

DEF FN nom_fonction (pf1 :type_pf1 ;pf2,pf3 :type_pf_2_3 ;…) :type_fonction
Résultat = nom_fonction
Nom_fonction← résultat
……
Fin nom_fonction

- ***En algorithme***

0) DEF FN nom_fonction (pf1 :type_f1 ;pf2,pf3 :type_pf_2_3 ;…) :type_fonction
1) ……
2) ……
n-1) nom_fonction ← résultat
n) Fin nom_fonction

- ***En pascal***

```
Program nom_p_p ;
Uses wincrt ;
Const          {Déclaration des constantes globales}
Type           {Déclaration des nouveaux types globaux}
Var            {Déclaration des variables globales}
{Déclaration de la fonction}
FUNCTION nom_fonction (pf1 :type_pf1 ;pf2,pf3 :type_pf_2_3 ;…) :type_fonction ;
        Const          {Déclaration des constantes locales}
        Type           {Déclaration des nouveaux types locaux}
        Var            {Déclaration des variables locales}
        Begin
                Instructions de la fonction ;
                Nom_fonction :=résultat ;
        End ;
{programme principal}
begin
        instructions  du pp
        appel de la fonction ;
end.
```

Remarque : En Pascal, la déclaration d'une fonction se fait après la déclaration des variables globales du programme principal.

III.5 Appel d'une fonction

L'appel d'une fonction se fait par son nom entre parenthèses liste de ses arguments effectifs.

L'appel d'une fonction permet d'exécuter les instructions de la fonction en substituant les paramètres effectifs aux paramètres formels.

Une fonction peut être appelée à partir du programme principal ou d'une autre fonction ou d'une procédure ou à partir d'elle même (notion de récursivité).

Une fonction peut être appelée dans une action d'affectation ou de sortie ou dans une structure conditionnelle ou itérative.

En analyse et en algorithme	***En pascal***
Variable ← FN nom_fonction (pe1,pe2,…,pen)	Variable := nom_fonction (pe1,pe2,….,pen)
Ecrire (FN nom_fonction(pe1,pe2,…,pen))	Writeln(nom_fonction (pe1,pe2,…,pen)
Si FN nom_fonction (pe1,pe2,…,pen) opr valeur alors	IF nom_fonction (pe1,pe2,…,pen) opérateur valeur then
Pour i de 1 à n faire variable← FN nom_fonction(pe1,pe2,…,pen)	**for i :=1 to n do** variable := nom_fonction(pe1,pe2,…,pen)

Remarques :

1. Au niveau de l'analyse et de l'algorithme et dans l'instruction d'appel, il faut ajouter devant le nom de la fonction le préfixe FN.
2. L'objet d'appel d'une fonction est de calculer une valeur et non pas de modifier les valeurs des variables ; donc ***le passage des paramètres*** entre le programme appelant et une fonction est toujours ***par valeur***.

III.6 Paramètres effectifs et formels

A/ Paramètres effectifs

Se sont des paramètres qui figurent dans l'instruction d'appel de la fonction ou de la procédure. Ces paramètres seront substitués aux paramètres formels au moment de l'appel de la fonction ou de la procédure.

B/ Paramètres formels

Se sont des paramètres qui figurent dans l'entête de la fonction ou de la procédure. Ces paramètres seront utilisés par les instructions de la fonction ou de la procédure.

Remarques :

1. Les paramètres effectifs et formels doivent s'accorder de point de vue nombre et ordre.
2. Leurs types doivent être identiques ou compatibles.

III.7 Objets globaux et locaux

A/ Objets globaux

Un objet global (constantes, nouveaux types, variables,…) est un objet déclaré dans la partie déclarative des objets du programme principal, utilisé par ce dernier et / ou par toutes les fonctions et les procédures qui y sont déclarées.

B/ Objets locaux

Un objet local (constantes, nouveaux types, variables,…) est un objet déclaré dans la partie déclarative des objets de la fonction ou de la procédure et utilisé seulement par celle-ci.

III.8 Application

Ecrire une fonction qui calcule x^y avec (x>=0 et y>=0).

Analyse du programme principal

```
3   Résultat = Ecrire (x, " à la puissance ", y, " = ", FN puissance (x,y))
1   X = Répéter
        X= donnée      ("X>=0")
        Jusqu'à (x >=0)
2   y = Répéter
        y= donnée      ("y>=0")
    Jusqu'à (y>=0)
4   Fin Calcul_puissance
```

Tableau de déclaration des objets globaux

Objet	Nature / Type	rôle
X,y	Variables /Entier	
puissance	Fonction/entier	

Algorithme du programme principal

```
0/ Début Calcul_puissance
1/  Répéter
        Ecrire ("X>=0"), Lire (x)
   Jusqu'à (x>=0)
2/  Répéter
        Ecrire ("y>=0"), Lire (y)
   Jusqu'à (y>=0)
3) Ecrire (x, " à la puissance y = ", FN puissnace (x,y))
4/ Fin Calcul_puissance
```

Analyse de la fonction puissance

```
        Résultat = puissance
2       puissance ← p
1       p= [p←1] Pour i de 1 à b faire
                        p← p*a
                 FinPour
3       Fin puissance
```

Tableau de déclaration des objets locaux

Objet	Nature / Type	Role
I	Compteur / Entier	
p	Variable/entier	

Algorithme de la fonction Factorielle

0/ DEF FN puissance (a,b : entier) : entier
1/ [ ] [p←1] Pour i de 1 à b faire
p← p*a
FinPour
2/ puissance ← p
3/ Fin puissance

Traduction en Pascal

```
program calcul_puissance;
uses wincrt;
var
  x,y:integer;
function puissance (a,b:integer):integer;
var
  I,p:integer;
begin
        p:= 1;
        for i:= 1 to b do
          begin
          p:= p*a;
          end;
        puissance := p;
end;
begin
   repeat
        write('x>=0');
        readln(x);
   until (x>=0);
   repeat
        write('y>=0');
        readln(y);
   until (y>=0);
   write(x, ' à la puissance ', y,' = ', puissance (x,y));
end.
```

IV. LES PROCÉDURES

IV.1 Activité

Ecrire un programme Pascal nommé permutation qui permet de permuter deux entiers x et y puis affiche le résultat.

IV.3 Procédure non paramétrée ou simple (1ére solution)

Analyse du programme principal

4 **Résultat** = Ecrire (" x= ",x "et" ,"y =",y)
3 X,y= proc permut
1 X= donnée ("X=")
2 y= donnée ("y=")
5 **Fin** permutation

Tableau de déclaration des objets globaux

Objet	Nature / Type	Role
X, y	Variables /Entier	
permut	Procédure	

Algorithme du programme principal

0/ Début permutation
1/ Ecrire ("X="), Lire (x)
2/ Ecrire ("y="), Lire (y)
3/ proc permut
4) Ecrire (" x= ",x "et" ,"y =",y)
5/ Fin permutation

Analyse de la procédure permut

Résultat = x,y
2 x←y
3 y←z
1 z←x
4 **Fin** permut

Tableau de déclaration des objets locaux

Objet	Nature / Type	Role

z	variable / Entier	

Algorithme de la procédure permut

0/ DEF PROC permut
1/ z←x
2/x←y
3/ Y←z
4/ Fin permut

Traduction en Pascal

```
program permutation;
uses wincrt;
var
  x,y:integer;
procedure permut;
var
  z:integer;
begin
        z:=x;
        x:=y;
        y:=z;
        end;
begin
write('x=');
readln(x);
write('y=');
readln(y);
permut;
write('x= ',x ,' et ' ,'y =',y);
end.
```

IV.4 Procédure paramétrée (2ème solution)

Analyse du programme principal

4 **Résultat** = Ecrire (" x= ",x "et" ,"y =",y)
3 X,y= proc permut (x,y)
1 X= donnée ("X=")
2 y= donnée ("y=")
5 **Fin** permutation

Tableau de déclaration des objets globaux

Objet	Nature / Type	rôle
X, y	Variables /Entier	
permut	Procédure	

0/ Début permutation
1/ Ecrire ("X="), Lire (x)
2/ Ecrire ("y="), Lire (y)
3/ proc permut (x,y)
4) Ecrire (" x= ",x "et" ,"y =",y)
5/ Fin permutation

Analyse de la procédure permut

Résultat = a,b
2 a←b
3 b←z
1 z←a
4 **Fin** permut

Tableau de déclaration des objets locaux

Objet	Nature / Type	rôle
z	variable / Entier	

Algorithme de la procédure permut

0/ DEF PROC permut (a,b :entier)
1/ z←a
2/a←b
3/ b←z
4/ Fin permut

Traduction en Pascal

```
program permutation;
uses wincrt;
var
  x,y:integer;
procedure permut(var a,b:integer);
var
  z:integer;
begin
        z:=a;
        a:=b;
        b:=z;
        end;
begin
write('x=');
readln(x);
write('y=');
```

```
readln(y);
permut(x,y);
write('x= ',x ,' et ' ,'y =',y);
end.
```

A/ Définitions

Une procédure est une suite d'instructions décrivant une ou plusieurs actions à laquelle on accorde un nom qui devient lui-même, en quelque sorte une nouvelle instruction.

Une procédure peut avoir 0 paramètre (procédure non paramétrée) ou 1 ou plusieurs paramètres (procédure paramétrée) et peut retourner 0 ou 1ou plusieurs résultats au programme appelant.

Une procédure paramétrée utilise des paramètres pour faire passer des informations entre le programme appelant et la procédure appelée.

B/ Appel d'une procédure paramétrée

L'appel de la procédure permet d'exécuter les instructions de la procédure en substituant les paramètres affectifs aux paramètres formels.

L'appel d'une procédure se fait par son nom entre parenthèses liste de ses paramètres effectifs.

En analyse et en algorithme	*En pascal*
PROC nom_de_la_procédure(pe1,pe2,…,pen)	nom_de_la_procédure(pe1,pe2,…,pen)

Remarque :

1. Au niveau de l'analyse et de l'algorithme et dans l'instruction d'appel, il faut ajouter devant le nom de la procédure le préfixe PROC.

C/ Déclaration d'une procédure paramétrée

- ***En analyse***

  DEF PROC nom_procédure (pf1 :type_pf1 ;pf2,pf3 :type_pf_2_3 ;…)
  Résultat = …….
  …….
  …….
  Fin nom_fonction

- ***En algorithme***

  0) DEF PROC nom_procédure (pf1 :type_pf1 ;pf2,pf3 :type_pf_2_3 ;…)
     Actions de la procédure
  n) Fin nom_procédure

- ***En pascal***

```
Program nom_p_p ;
Uses wincrt ;
Const          {Déclaration des constantes globales}
Type           {Déclaration des nouveaux types globaux}
Var            {Déclaration des variables globales}
{Déclaration de la procédure}
Procedure  nom_procédure (pf1 :type_pf1 ;pf2,pf3 :type_pf_2_3 ;…) ;
        Const          {Déclaration des constantes locales}
        Type           {Déclaration des nouveaux types locaux}
        Var            {Déclaration des variables locales}


        Begin
                actions de la procédure ;
        End ;
{programme principal}
begin
        instructions  du pp
        appel de la procédure ;
end.
```

Remarques :

- En Pascal, la déclaration d'une procédure se fait après la déclaration des variables globales du programme principal.
- Les variables déclarées au niveau du programme principal sont appelées ***variables globales***.
- Les variables déclarées au niveau de la procédure sont appelées ***variables locales***.

D/ Passage par valeur et par variable

- ***Passage par valeur***

Il permet au programme appelant de transmettre une valeur au programme appelé (fonction ou procédure). Cette valeur du paramètre effectif sera affectée au paramètre formel au moment de l'appel de la fonction ou de la procédure.

Toute modification du paramètre formel est sans conséquent sur la valeur du paramètre effectif.

Le transfert d'information s'effectue dans ***un seul sens*** :

P.appelant ⟶ P.appelé

- ***Passage par variable***

Il permet au programme appelant de transmettre une ou plusieurs valeurs à la procédure et vice-versa au moment de l'appel.

Toute modification du paramètre formel entraîne automatiquement la modification de la valeur du paramètre effectif.

Le transfert d'information s'effectue dans deux sens :

P.appelant ⟷ Procédure

Remarque : Au moment de l'appel des procédures, on ne spécifie pas le mode de transmission. Ce dernier n'est spécifié qu'au niveau de l'entête de la procédure concernée en ajoutant le mot ***VAR*** juste avant l'argument transmis par adresse.