Les sous programmes (fonctions et procédures)

Université de Bejaia-2020/2021
Département de Technologie
1ere année Technologie
Module : Informatique2
Présenté par : Mme MAMMERI

➢Exemple introductif

Le programme suivant calcule la combinaison $c = \frac{n!}{k! \times (n-k)!}$ avec *n>=k*

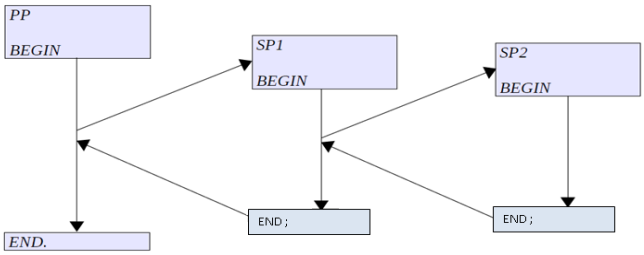
```
Program combinaison;
Uses wincrt ;
i, n,k, f1, f2, f3, c : integer ;
Begin
Write('Donner la valeur  de n et k : ');
   Read(n, k);
f1:=1;
For i:=1 to n do
     f1:=f1*i;

f2:=1;
 For i:=1 to k do
f2:=f2*i;

f3:=1;
 For i:=1 to (n-k) do
f3:=f3*i;

c:=f1 div (f2*f3);
write(c);
End.
```

Remarque : le même traitement est réalisé pour calculer le factoriel d'un nombre entier (redondance d'instructions).

➢ Afin de rendre le programme plus lisible et réduire le nombre d'instructions (moins d'espace mémoire), on fait appel à des fonctions et procédures (sous-programmes).

➢Définition d'un sous-programme (SP)

Un SP (procédure ou fonction) est un programme à l'intérieur d'un autre programme, Il possède la même structure que le programme principal (PP). Il peut être appelé par le PP ou par un autre SP pour réaliser un traitement (ou plusieurs) et retourner un ou plusieurs résultats.



3

➢Structure générale d'un programme Pascal comportant des SPs

Program <Nom_Programme>;
uses wincrt;
<Déclaration des étiquettes éventuelles>;
<Déclaration des constantes éventuelles>;
<Déclaration des types éventuels>;
<Déclaration des variables éventuelles>;
<Déclaration des fonctions>;
<Déclaration des procédures>;
BEGIN
<instruction 1>;
<instruction 2>;
<instruction 3>;
.....
…….
<instruction N>;
END.

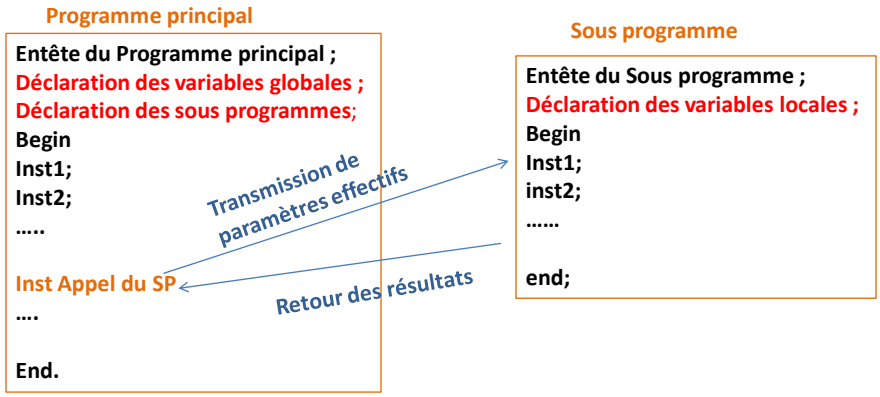
Objets globaux

SP

Corps du PP

4

➢Appel d'un sous programme



5

➢Exécution d'un Programme contenant un Sous Programme

1. On exécute en premier le PP.
2. Quand l'instruction Appel du SP est exécutée, l'exécution du PP sera suspendu. Le SP sera maintenant exécuté jusqu'à end;
   - Le SP reçoit du PP des données (paramètres d'entrés) , et retournes des résultats (paramètres de sorties)au PP.
3. Une fois l'exécution du SP est terminée, la suite du PP sera exécutée.

6

➢Paramètres et variables utilisés

- **Paramètres effectifs** : sont des variables utilisées lors de l'appel à un sous programme.
- **Paramètres formels** : sont des variables définies dans l'entête du sous programme.
- **Variables globales** : sont des variables définies dans le PP, sont accessibles aux SP.
- **Variables locales** : sont des variables définies dans le SP, sont inaccessibles par le PP et par d'autres SP.

7

➢Procédure

Une procédure est un SP qui retourne un ou plusieurs résultats

➢ Déclaration

Paramètres formels

Procedure Nom_Procedure (param1:type1; param2:type2; …;paramN:typeN**);** — Entête de la procédure

<Déclaration des constantes>;
<Déclaration des variables>; — Objets locaux
<Déclaration des étiquettes>;
Begin
<instruction 1>;
<instruction 2>; — Corps de la procédure
<instruction 3>;
.....
…….
<instruction N>;

End;

8

➢Paramètres formels d'une procédure

Les paramètres formels d'une procédure peuvent être passés par valeur ou passé par adresse.

❑ **Passage par valeur**

• Il s'agit de recopier la valeur du paramètre effectif dans le paramètre formel correspondant.

• Les paramètres passés par valeur sont des paramètres d'entrés pour la procédure.

• Le changement de la valeur du paramètre formel dans la procédure ne changera pas la valeur du paramètre effectif correspondant dans le PP.

❑ **Passage par adresse (variable)** :

• Il s'agit de recopier l'adresse mémoire du paramètre effectif dans le paramètre formel correspondant.

• Les paramètres passés par adresse sont des paramètres de sorties ou d'entrés/sorties pour la procédure.

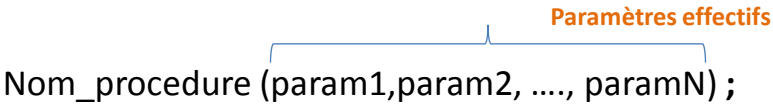
• Les paramètres passés par adresse sont précédés par le mot clé **Var.**

• Le changement de la valeur du paramètre formel dans la procédure changera la valeur du paramètre effectif correspondant dans le PP.

9

➢Appel d'une procédure

une procédure est appelée par le PP ou par un autre SP en respectant la syntaxe suivante :



Remarques

❑ Les paramètres effectifs doivent avoir le même nombre que les paramètres formels.

❑ Le type de chaque paramètre effectif doit être le même que le paramètre formel correspondant.

❑ L'ordre des paramètres effectifs doit correspondre aux paramètres formels

10

➢ *Exemple*

Le programme suivant calcule la combinaison en utilisant une procédure.

$$C = \frac{n!}{k! \times (n-k)!}$$ avec *n>=k*

```
Program combinaison;
Uses wincrt;
Var  k,n,f1,f2,f3,C: integer;
Procedure factoriel ( n: integer;  var  f:  integer);
Var  i: integer;
Begin
 f:=1;
For i:=1 to n do
     f:=f*i;
end;
 Begin
  Read(n, k);
    factoriel (n,f1);
    factoriel( k,f2);
    factoriel( n-k, f3);
 c:=f1 div (f2*f3);
 write(c);
 End.
```

Variables globales

Deux paramètres formels :
n : paramètre d'entré, passé par valeur
f: paramètre de sortie, passé par adresse, précédé par **var**

Variable locale

Deux paramètres effectifs dans chaque appel :
1er appel : n , f1
2eme appel : k, f2
3eme appel : n-k, f3

Déroulement pour n = 5 et k = 3

Variables globales | Procédure factoriel

Instructions	n	k	f1	f2	f3	c	n	i	f	Affichage
Read (n, k) ;	5	3	/	/	/	/				
factoriel (n, f1) ;							~~5~~		/	
f :=1 ;			~~1~~						~~1~~	
For i := 1 **to** n **do**			~~1~~					~~1~~	~~1~~	
f :=f*i ;			~~2~~					~~2~~	~~2~~	
			~~6~~					~~3~~	~~6~~	
			~~24~~					~~4~~	~~24~~	
			120					~~5~~	~~120~~	
factoriel (k, f2) ;							~~3~~		/	
f :=1 ;				~~1~~					~~1~~	
For i := 1 **to** n **do**				~~1~~				~~1~~	~~1~~	
f :=f*i ;				~~2~~				~~2~~	~~2~~	
				6				~~3~~	~~6~~	
factoriel (n-k, f3) ;							2		/	
f :=1 ;					~~1~~				~~1~~	
For i := 1 **to** n **do**					~~1~~			~~1~~	~~1~~	
f :=f*i ;					**2**			2	2	
C:=f1 **div** (f2*f3);						10				
Write (c);										10

➢Fonctions

Une fonction est un SP qui retourne un et un seul résultat (une valeur).

➢Déclaration

Paramètres formels

Function Nom_fonction (param1:type1; param2:type2; …;paramN:typeN) **: type_fonction ;**
<Déclaration des constantes>;
<Déclaration des variables>;
<Déclaration des étiquettes>;
Begin
<instruction 1>;
<instruction 2>;
<instruction 3>;
.....
…….
Nom_fonction := résultat; ⟶ résultat retourné

End;

Type de résultat à retourner

Corps de la fonction

13

➢Paramètres formels d'une fonction

Les paramètres formels d'une fonction sont des paramètres passés par valeur (paramètres d'entrés).

➢Appel d'une fonction

Paramètres effectifs

Id_variable := Nom_fonction (param1,param2, …., paramN) ;

Ou bien

Id_variable := Expression contenant la fonction ;

Remarques

Id_variable et type_fonction sont de même type .

Les paramètres effectifs dans l'instruction d'appel à une fonction peuvent être des valeurs, des variables ou des expressions.

14

Exemple

Le programme suivant calcule la combinaison $C = \frac{n!}{k! \times (n-k)!}$ avec *n>=k* en utilisant une fonction.

```
Program combinaison;
Uses wincrt;
Var  k,n,f1,f2,f3,C: integer;
Function factoriel ( n: integer) :  integer;
Var  i, f: integer;
Begin
f:=1;
For i:=1 to n do
     f:=f*i;
factoriel := f ;
end;
 Begin
  Read(n, k);
   f1 :=  factoriel (n);
   f2:= factoriel( k);
   f3:= factoriel( n-k);
 c:=f1 div (f2*f3);
 write(c);
 End.
```

Variables globales

n : paramètre formel d'entré, passé par valeur

Variables locales

Le type de la fonction et la variable résultat f est le même

Retour du résultat

Un paramètre effectif dans chaque instruction d'appel :
1er appel : n , Le résultat est retourné dans f1.
2eme appel : k ,Le résultat est retourné dans f2.
3eme appel : n-k, Le résultat est retourné dans f3.

Remarque : on peut faire directement
C:=factoriel(n) **div** (factoriel(k)*factoriel(n-k));
sans déclarer les variables f1, f2 et f3.

15

Déroulement pour n = 5 et k = 3

Instructions	n	k	f1	f2	f3	c	n	i	f	Affichage
Read (n, k) ;	5	3								
f1 :=factoriel (n) ;							5			
f :=1 ;									1	
For i := 1 **to** n **do** f :=f*i ;								1	1	
								2	2	
								3	6	
								4	24	
								5	120	
factoriel:= f ;			120				factoriel :=120			
f2 :=factoriel (k) ;							3			
f :=1 ;									1	
For i := 1 **to** n **do** f :=f*i ;								1	1	
								2	2	
								3	6	
factoriel :=f ;				6			factoriel :=6			
f3 :=factoriel (n-k) ;							2			
f :=1 ;									1	
For i := 1 **to** n **do** f :=f*i ;								1	1	
								2	2	
factoriel:=f;					2		factoriel :=2			
C:=f1 **div** (f2*f3);						10				
Write (c);										10

16

➢Type tableau dans un SP

Dans un SP (fonction ou procédure), on ne déclare pas un tableau de cette manière :

Procedure exemple (T : array[1..50] of real) ; erreur de déclaration

➢ **La Solution :**
Dans la partie déclaration des variables globales, on déclare un tableau comme suit :

```
Type tab=array[1..50] of real;
 var T: tab;
```

Puis, dans l'entête du SP, on déclare le tableau comme suit :

Procedure exemple (T : **tab**) ;

17

Exercices

1. Ecrire un programme qui calcule le minimum d'un tableau en utilisant une fonction.

```
Program exercice ;
Uses wincrt ;
Type tab=array [1..50] of real ;
Var  T:tab;
    i, n : integer;
    min : real;

Function  minimum  (n:integer ;  T:tab) :  real;
Var  mini : real;
begin
mini:= T[1];
for  i := 2 to n do
  if T[i]<mini  then
     mini:=t[i];
minimum:=mini ;
end;

Begin
Read(n );
for  i : = 1  to n do
  read( T[i]) ;
min := minimum (n , T);
write (' le minimum=', min) ;
end.
```

18

2. Ecrire un programme qui calcule le minimum d'un tableau et sa position en utilisant une procédure.

Remarque : pour ce problème, on ne peut pas utiliser une fonction, par ce que, on a deux résultats à calculer: le minimum et sa position

```
Program exercice ;
Uses wincrt ;
Type tab=array [1..50] of real ;
Var  T:tab;
 P,   i, n : integer;
    min : real;

procedure  calcul  (n:integer ;  T:tab ; var p: integer ;  var min : real ) ;
begin
min:= T[1];
p:=1;
for  i := 2 to n do
  if T[i]<min  then
     begin
     min:=t[i];
    p:=i;
    end;
end;

Begin
Read(n );
for  i : = 1  to n do
  read( T[i]) ;
calcul(n , T, p, min);
write (' le minimum=', min, 'sa position = ', p) ;
end.
```

19