Chapitre 5

Les sous programmes

Leçon 1 :

L'analyse modulaire

I- Introduction :

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

II- L'analyse modulaire :

1- Définition :

L'analyse modulaire consiste à diviser un problème en sous problème de difficultés moindres. Ces derniers sont aussi soumis à cette division jusqu'à ce qu'on arrive à un niveau abordable de difficulté.

2- Intérêts :

- Plus d'organisation en séparant les difficultés et les taches.
- S'occuper d'un seul problème à la fois.
- En cas d'erreur la division en module permet de savoir quel module à corriger
- Plus facile à faire évoluer.
- Permet d'éviter la répétition d'un même traitement dans un programme.

3- Notion de sous-programme :

C'est la décomposition modulaire d'un programme en plusieurs sous-programmes. Un sous-programme est appelé aussi une procédure ou une fonction. C'est une portion de texte analogue à un programme, déclaré dans un programme ou dans un sous-programme et dont la partie instruction peut être exécutée plusieurs fois au cours du traitement du programme grâce à des appels.

Exemple d'analyse modulaire : étude de fonction mathématique.

Leçon 2 :

Les procédures

1-Définition :

Les procédures sont des sous-programmes qui peuvent avoir plusieurs résultats.

2-Vocabulaire et syntaxe :

En analyse :

DEF PROC nom (paramètres formels : type)
Résultat=
Traitement
Fin nom

En algorithme :

0) **DEF PROC** nom (paramètres formels : type)
1) Traitement
2) **Fin** Nom

En Pascal :

Procedure nom (paramètres formels : type) ;
Déclaration des variables locales ;
Begin
Traitement ;
End;

Appel de la procédure :

Proc nom_procédure (paramètres effectifs)

L'appel d'une procédure doit se trouver dans une instruction d'appel et ne peut pas être dans une expression comme c'est le cas d'une fonction

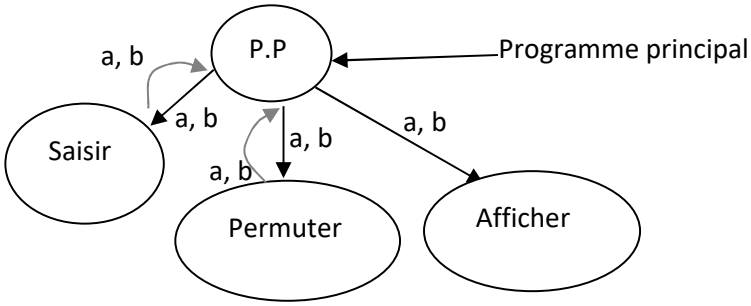
Remarque : Il est possible de définir un sous-programme sans paramètres. La communication avec l'appelant se produit grâce aux ressources (objets) communes partagées entre l'appelé et l'appelant.

Activité :

Ecrire un programme qui permet de saisir deux entiers a et b >0 puis permuter leurs valeurs enfin les afficher.

Corrigé : *[Méthode à partir du Bac 2017 (analyse du PP seulement et les algorithmes et les tableaux de déclaration relatifs aux modules envisagés.)]*

Pour résoudre ce problème on va utiliser les modules suivants : saisir, permuter et afficher



Analyse du programme principal :

Nom : permutation

Résultat= PROC Afficher (a,b)

(a,b)= PROC permuter (a,b)

(a,b)= PROC Saisir (a,b)

Fin Permutation

Tableau de Déclaration de Objet Globaux

Objet	Type/nature	Rôle
a, b	Entier	Entiers à saisir
Afficher, permuter, saisir	Procédures	Afficher, permuter et saisir (a et b)

Algorithmes :

Algorithme de la procédure Afficher :

0) DEF Proc Afficher (a : entier, b :entier)

1) Ecrire ("a=", a, " b=",b)

2) Fin Afficher

Algorithme de la procédure Saisir :

0) DEF PROC Saisir (**VAR** a : entier, **VAR** b :entier)

1) Répéter

Ecrire("a="), lire(A)

Ecrire("b="), lire(b)

Jusqu'à (a>0) et (b>0)

2)Fin Saisir

Algorithme de la procédure permuter :

0) DEF proc permuter (**VAR** a : entier, **VAR** b :entier)

1) c ← a

2) a ← b

3) b ← c

4)Fin permuter

T.D.O locaux

objet	Type/nature	Rôle
c	Entier	Variable intermédiaire

Traduction Pascal :

```
Program permutation;
Uses wincrt ;
Var a,b :integer;        ← objets globaux

Procedure saisir( VAR a : integer ; VAR b :integer);
Begin                    ← passage par variable
        repeat
                writeln('a=') ; readln(a);
                writeln('b=') ; readln(b);
        until (a>0) and (b>0);
End ;
                                  paramètres
                                  Formels
Procedure permuter( VAR x :integer; VAR y:integer);
Var  c: integer;         ← objets locaux
Begin
        C:=x;
        x:=y;
        y:=c;
End ;

Procedure afficher( a : integer ; b :integer);
Begin
        Writeln('a=',a,'b=',b);        passage par valeur
End ;

Begin
        Saisir(a,b);     ← appel d'une procédure
        Permuter(a,b);
        Afficher(a,b);
End.                     Paramètres effectifs
```

Leçon 3 :

Déclaration, accès aux objets et mode de transmission

I-Déclaration et accès aux objets

1- Les objets locaux :

Tous les objets (constantes, types, variables et sous-programme) déclarés dans un sous-programme sont dits locaux à celui-ci.

2- Les objets globaux :

Les objets utilisés dans un sous-programme et non déclarés dans celui-ci sont des objets globaux déclarés ailleurs.

3- Accès aux objets :

Tous les objets locaux d'un sous-programme sont inaccessible :

par le programme principal, par les sous-programmes déclarés au même niveau que le sous-programme considéré, par le sous-programme qui englobent les sous-programmes considérés.

II- Les paramètres et leur mode de transmission :

On distingue deux types de paramètres :

1-Les paramètres formels : qui figurent dans la définition de la procédure.
2-Les paramètres effectifs : qui figures dans l'appel de la procédure et qui sont manipulés par celle-ci.

Remarque : Les paramètres formels et les paramètres effectifs doivent s'accorder de point de vue nombre et ordre et leurs types doivent être identique ou compatible, selon le mode de passage des paramètres.

3-Mode de passage des paramètres :

Il existe 2 modes de passage des paramètres : le mode par valeur et le mode par variable. Pour le cas de fonction, nous définissions seulement le mode par valeur.

1. Mode de passage par valeur :

-Permet au programme appelant de transmettre une valeur au sous-programme appelé.

-Le transfert d'information est effectué dans un seul sens : du programme appelant vers le sous-programme appelé.

-Au moment de l'appel, la valeur du paramètre effectif est copiée dans la variable locale désignée par les paramètres formels correspondants.

Remarque : Toute modification du paramètre formel est sans conséquence sur le paramètre effectif

2. *Mode de passage par variable :*

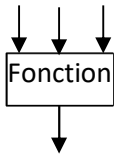
-Le passage de paramètres par variables permet au programme appelant de transmettre une valeur au sous-programme appelé et inversement.

-Dans l'entête de la procédure, on doit précéder les paramètres formels transmis par variable par le mot clé **VAR**.

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

Leçon 4

Les fonctions



1-Définition :

Une fonction est un sous-programme qui renvoie une valeur de type **simple**, ce type sera le **type** de la fonction.

2-Syntaxe :

Déclaration d'une fonction :

En analyse :

DEFFN nom (paramètres formels: type): Résultat Résultat=

Nom ← résultat calculé

Traitement

Fin nom

En algorithme :

0) **DEF FN** nom (paramètres formels: type) : Type_Resultat
1) Traitement
2) Nom ← résultat calculé
3) **Fin Nom**

En Pascal :

Function nom (paramètres formels: type) : Type_Resultat;
Déclaration des variables locales;
Begin
Traitement;
Nom:=RésultatCalculé;
End;

Appel de la fonction :

Analyse

Y←FN f(x) **(x est un paramètre effectif)**

TDO:

objet	type/nature	rôle
f	Fonction	f(x)=.....

Remarques :

1) L'appel de la fonction se fait à l'aide de FN
2) Le préfixe FN est ajouté devant le nom de la fonction que nous avons créé ; ça nous aidera à nous rappeler qu'il faudra analyser.
3) F est l'appelant.
4) Dans l'analyse :

   DEFFN nom f(X:entier) : type_entier

   {x paramètre formel}

- X est déjà déclarés au niveau de l'entête de la fonction.
- F est une fonction ayant un seul paramètre formel x. Il est possible qu'une fonction ait plusieurs paramètres.
- Les variables déclarés dans la fonction sont appelés variables locales à la fonction f.

Une fonction est constituée de trois parties :

1) La partie entête de la fonction ou nous trouvons son nom qui est suivit entre parenthèses des paramètres en entrée et de leur mode de passage, puis du type du résultat.

2) La partie déclaratives ou tous les objets locaux de la fonction sont déclarés.

3) La partie instruction ou nous trouvons les instructions propres à la fonction. Il est bon de rappeler que ces instructions sont exécutées à la demande de l'appelant par une instruction d'appel.

3-Transmission du résultat de la fonction :

Une fonction possède un type, c'est celui du résultat quelle calcule.

On appelle une fonction en utilisant simplement son nom suivi de la liste des paramètres effectifs séparé par des virgule (,)

Un appel de fonction figure obligatoirement dans une expression sous forme suivante :

V ←FN nom de la fonction (liste des paramètres effectifs)

Exemple : y ← FN f(x)

Il doit y avoir nécessairement dans la partie instruction de la fonction au moins une affectation explicite ayant l'identificateur de la fonction à gauche du symbole d'affectation.

4-Définition d'une fonction :

Lors de l'utilisation d'une fonction, il faut :

1) Spécifier le type de la fonction
2) Déclarer, si nécessaire, une variable locale de même type que la fonction (pour faire les calculs intermédiaires)
3) Affecter le résultat de calcul de la fonction au nom de la fonction, obligatoirement, avant la fin du bloc.

5-Mode de passage :

Pour le cas de la fonction, nous définissant seulement le mode de passage par valeur.

Remarques :

- Dans la partie instruction de la fonction, les paramètres formels transmis par valeur ne doivent en aucun cas se trouver à gauche du symbole d'une instruction d'affectation.
- Même si par erreur la valeur du paramètre formel transmis par valeur est modifié dans l'appelé au retour après exécution, les paramètres effectifs gardent la même valeur originale transmise lors de l'appel.
- En effet, au niveau de l'appelé on manipule qu'une copie.
- Un identificateur peut cacher un autre. (Un objet local à un sous-programme a le même nom qu'un objet global (homonyme)).
- L'ordre des sous programmes est important, lorsqu'un sous-programme fait appel à un autre, alors ce dernier doit être définit avant.

Activité 2 :

Écrire un programme qui permet de saisir deux entiers x et y <=9 puis calculer la puissance x^y enfin afficher le résultat.

Corrigé : *[Méthode à partir du Bac 2017 (analyse du PP seulement et les algorithmes et les tableaux de déclaration relatifs aux modules envisagés.)]*

Pour résoudre ce problème on va utiliser les modules suivants :

procédure saisir(x,y)

fonction puissance(x,y)

Analyse du programme principal :

Nom: calcul
Résultat= Ecrire("X à la puissance y =",p)
p ← FN puissance(x,y)
(x,y)=PROC saisir(x,y)
Fin calcul

Tableau de Déclaration des Objets Globaux

Objets	Type/Nature	Rôle
x , y	Entier	Entiers à saisir
p	Entier	Puissance à calculer
Saisir	Procédure	Saisir x et y
puissance	fonction	Calculer la puissance p

Algorithme de la procédure saisir :

```
0)DEF PROC Saisir(var x,y : entier)
1)Répéter
                Ecrire("x= ") , lire(x)
                  Ecrire("y= "), lire(y)
        Jusqu'a (x<=9)  et (y<=9)
2)Fin saisir
```

Algorithme de la fonction puissance :

```
0)DEF FN puissance(x,y:entier):entier
1) [m ← 1] pour i de 1 à y faire
                  m ← m*x
             FinPour
2) puissance ← m
3)Fin puissance
```

Tableau de Déclaration des Objets Locaux

Objets	Type/Nature	Rôle
m	Entier	Puissance à calculer
i	entier	compteur

Programme pascal :

```
Program calcul ;
Uses wincrt ;
Var x , y  :  integer ; p:longint ;
Procedure saisir(var x,y:integer) ;
Begin
Repeat
     Writeln('x= ');readln(x) ;
     Writeln('y= ');readln(y) ;
Until (x<=9) and (y<=9);
End ;
Function puissance(x,y:integer):longint ;
Var i:integer;m:longint ;
Begin
M:=1 ;
For i:=1 to y do
M:=m*x ;
Puissance:=m ;
End ;

Begin
saisir(x,y) ;
P:=puissance(x,y) ;
Writeln('X à la puissance y =',p) ;
End.
```

Remarque :

On peut modifier la procédure saisir comme suit :

```
Procedure saisir(var a:integer) ;
Begin
Repeat
   Writeln('donner un entier ');
   readln(a) ;
Until (a<=9) ;
End ;
```

Puis l'appeler 2 fois dans le programme principal, avec **saisir(x) ;** puis **saisir(y) ;**

Exercice 7 page 121:

Remarque :

```
Randomize ;
For i :=1 to n do
T[i] :=random(100);
```

- Hasard en Pascal `random ;`

  donne un réel au hasard entre 0 et 1(exclu).

- Hasard(n) donne des valeurs au hasard entre 0 et n-1.

- **Remplissage d'un tableau par des valeurs aléatoires entre 0 et 99**

Pour i de 1 à n faire

T[i] ← hasard(100)

FinPour

- **Remplissage avec des valeur entre a et b (a<b) :** x ← hasard(b-a+1) + a

Exemple : valeurs entre 10 et 20 x ← hasard(11)+10

- **Remplissage d'un tableau par des lettres minuscules au hasard :**

a 97 z 122

Pour i de 1 à n faire

T[i] ← **chr(** hasard(26)+97 **)**

FinPour

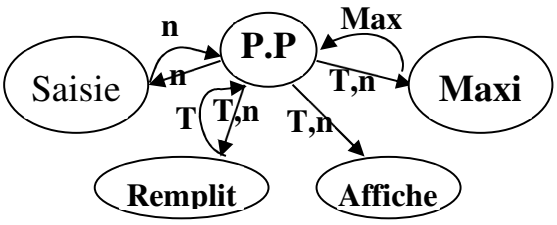
Exercice 1 :

Écrire un programme qui permet de

- Saisir un entier n (entre 2 et 20)
- Remplir un tableau T au hasard par n entiers de valeurs entre 0 et 100
- Afficher les éléments de T
- Afficher le maximum de T

Corrigé : Pour résoudre ce problème on a besoin de définir les modules suivants :

- **procédure Saisie(n)** qui permet de saisir la taille du tableau entre 2 et 20.
- **procédure Remplit(T, n)** permet de remplir T par n entiers au hasard (entre 0 et 100).
- **procédure Affiche(T, n)** permet d'afficher le tableau T.
- **fonction Maxi(T,n)** qui permet de renvoyer le maximum du tableau.



Analyse du programme principal:

NOM: Maximum
Résultat=Ecrire("Le max est",Max)
Max ← FN Maxi (T,n)
Proc Affiche(T, n)
T=PROC Remplit(T,n)
n=PROC Saisie(n)
FIN Maximum

Tableau de déclaration des nouveaux types:

TYPE
TAB=Tableau de 20 entiers

Tableau de déclaration des objets:T.D.O. Globaux :

Objet	Type/Nature	Rôle
T	TAB	
n, Max	Entier	
Maxi	Fonctions	
Remplit, Saisie, Affiche	Procédures	

Algorithme de la procédure saisie :

0) DEF PROC Saisie(**VAR** nb:entier)
1) Répéter
Ecrire("Donner n: "), lire(nb)
Jusqu'à nb dans [2..20]
2) FIN saisie

Algorithme de la procédure Remplit :

0) DEF PROC Remplit(Var T:TAB, n:entier)
1) Pour i **de** 1 **à** n **faire**
T [i] ← Hazard(101)
Fin pour
2)FIN Remplit

T.D.O.Locaux:

Objet	Type/Nature	Rôle
i	entier	Compteur

Algorithme de la procédure Affiche :

0) DEF PROC Affiche(T:TAB, n:entier)
1) Pour i **de** 1 **à** n **faire**
Ecrire("L'entier n° ",i, " est ",T[i]) ;
Fin pour
3) FIN Affiche

T.D.O.Locaux:

Objet	Type/Nature	Rôle
i	entier	Compteur

Algorithme de la fonction Maxi :

0) DEF FN Maxi(T:TAB, n:entier) : entier
1) Ma←T[1]
2) Pour i **de** 2 **à** n **faire**
Si T[i]>Ma **alors** Ma←T[i] **FinSi**
Fin pour
3) Maxi← Ma
4) FIN Maxi

T.D.O.Locaux :

Objet	Type/Nature	Rôle
i	Entier	Compteur
Ma	entier	Calculer le max

Programme Pascal :

```
program maximum;
uses wincrt;
type  tab=array [1..20] of integer;
var  n,max:integer;  t:tab;
procedure saisie (var n:integer);
begin
repeat
write('donner un entier(entre 2 et 20):');
readln(n);
until (n>=2) and  (n<=20);
end;
procedure remplit(var t:tab;n:integer);
var
i:integer;
begin
RANDOMIZE;
for i:=1 to n do
T[i]:=random(101);
end;
procedure Affiche(t:tab;n:integer);
var i:integer;
begin
for i:=1 to n do
writeln('T[',i, ']= ',T[i]);
end;
function maxi(t:tab;n:integer):integer;
var i,ma:integer;
begin
Ma:=t[1];
for i:=2 to n do
if T[i]>ma then ma:=T[i];
maxi:=ma;
end;
begin
saisie(n);
remplit(t,n);
affiche(t,n);
max:= maxi(t,n);
writeln('le maximum est ',max);
End.
```

Exercice 2:

Écrire un programme qui permet de saisir n entre 2 et 5 puis remplit un tableau T par des valeurs au hasard entre 10 et 99 puis affiche le tableau en ordre inverse enfin calcul et affiche le nombre des entiers pairs dans T.

Corrigé :

```
program pairs;
uses wincrt;
type
     tab=array[1..5]of integer;
var
     n,nb:integer;
     t:tab;

procedure saisir(var n:integer);
begin
repeat
write('n= ');  readln(n);
until n in [2..5];
end;

procedure remplir(var t:tab;n:integer);
var
     i:integer;
begin
for i:=1 to n do
t[i]:=random(90)+10;
end;

procedure afficher(t:tab;n:integer);
var
     i:integer;
begin
for i:=n downto 1 do   writeln(t[i]);
end;
```

```
function calculer(t:tab;n:integer):integer;
var
     nb,i:integer;
begin
nb:=0;
for i:=1 to n do
if t[i] mod 2=0 then nb:=nb+1;
calculer:=nb;
end;

begin
saisir(n);
randomize;
remplir(t,n);
afficher(t,n);
nb:=calculer(t,n);
writeln('le nombre d''entiers pairs est:',nb);
end.
```

Exercice 3:

Écrire un programme qui permet de saisir n entre 3 et 7 puis de remplir un tableau T de taille n par des lettres minuscules au hasard puis affiche le tableau en ordre inverse enfin calcul et affiche le nombre des voyelles dans T

Corrigé :

```
Program Voyelles;
uses wincrt;
type
     tab=array[1..7]of char;
var
     n,nbv:integer;
       t : tab;


procedure saisir(var n:integer);
begin
repeat
write('n= ');  readln(n);
until n in [3..7];
end;
procedure remplir(var t:tab;n:integer);
var i:integer;
begin
for i:=1 to n do
t[i]:=chr(random(26)+97);
end;

procedure afficher(t:tab;n:integer);
var i:integer;
begin
for i:=n downto 1 do   writeln(t[i]);
end;
```

```
function calcul(t:tab;n:integer):integer;
var nbv,i:integer;
begin
nbv:=0;
for i:=1 to n do
if t[i] in ['a','e','y','u','i','o'] then nbv:=nbv+1;
calcul:=nbv;
end;
begin
saisir(n);
randomize;
remplir(t,n);
afficher(t,n);
nbv:=calcul(t,n);
writeln('Le nombre des voyelles est:',nbv);
end.
```

Exercice 4 :

Écrire un programme qui permet de saisir n (doit être n=2 ou n=3) puis de remplir 2 vecteur t1 et t2 de taille n, puis de calculer et d'afficher le produit scalaire de deux vecteurs, en fin afficher si ce deux vecteurs sont orthogonaux.

On rappelle que : pour t1(x1,y1) et t2(x2,y2) est ps=x1*x2+y1*y2

(deux vecteur sont orthogonaux si leurs produit scalaire est égale à zéro)

Corrigé :

```
program produit;
uses wincrt;
type tab=array[1..3] of integer;
var t1,t2:tab;
     ps,n:integer;
procedure saisir (var n:integer);
begin
repeat
write('n= '); readln(n);
until (n=2) or (n=3);
end;
```

```
procedure remplir(var t:tab; n:integer);
var i:integer;
begin
for i:=1 to n do
begin
write('donner l''élement n°',i,'= ');  readln(t[i]);
end;
end;
function calculer_ps(T1:tab;T2:tab;n:integer):integer;
var ps,i:integer;
begin
ps:=0;
for i:=1 to n do
ps:=ps+T1[i]*T2[i];
calculer_ps:=ps;
end;
procedure afficher_dec(ps:integer);
begin
 if ps=0 then writeln('T1 et T2 sont orthogonaux')
 else writeln('T1 et T2 ne sont pas orthogonaux');
end;
begin
 saisir(n);
 remplir(t1,n);
 remplir(t2,n);
 ps:=calculer_ps(t1,t2,n);
 writeln('Le produit scalaire est',ps);
 afficher_dec(ps);
end.
```

Exercice 5 :

Écrire un programme qui permet de remplir un tableau T de taille n par des entiers positifs, puis extraire les entiers pairs dans un tableau T1 et les entiers impaires dans un tableau T2 , enfin afficher les tableaux T1 et T2.

Corrigé : *[Méthode à partir du Bac 2017 (analyse du PP seulement et les algorithmes et les tableaux de déclaration relatifs aux modules envisagés.)]*

Analyse du programme principal:

NOM: Parité
Résultat= Proc Afficher(T2,n2)
Proc Afficher(T1,n1)
(T1,n1,T2,n2) = Proc Extraire(T,n,T1,n1,T2,n2)
(T,n)=Proc Saisir(T,n)
FIN Parité

Tableau de déclaration des nouveaux types:

TYPE
TAB=Tableau de 10 entiers

Tableau de déclaration des objets: T.D.O. Globaux :

Objet	Type/Nature	Rôle
T	TAB	Tableau à saisir
T1	TAB	Tableau des pairs
T2	TAB	Tableau des impairs
N	Entier	Taille du tableau T
N1	Entier	Taille du tableau T1
N2	Entier	Taille du tableau T2
Saisir	Procédure	Saisir T et n
Extraire	Procédure	Extraire T1 et T2
Afficher	Procédure	Afficher un tableau

Algorithme de la procédure saisir :

0) DEF PROC Saisir(**VAR** n:TAB, **VAR** n:entier)
1) Ecrire("Donner n: "), lire(nb)
2) Pour i de 1 à n faire
Répéter
Ecrire("Donner l'élément n°",i), lire(T[i])
Jusqu'à
FinPour
3) FIN saisir

T.D.O.Locaux:

Objet	Type/Nature	Rôle
i	entier	Compteur

Algorithme de la procédure Afficher :

0) DEF PROC Afficher(T:TAB, n:entier)
1) Pour i **de** 1 **à** n **faire**
Ecrire("L'entier n° ",i, " est ",T[i]) ;
Fin pour
3) FIN Affiche

T.D.O.Locaux:

Objet	Type/Nature	Rôle
i	entier	Compteur

Algorithme de la procédure extraire :

0) DEF Proc Extraire(T:TAB, n:entier, T1:TAB,j:entier n,T2:TAB,k:entier)
1) j←0
2) k←0
3) Pour i **de** 1 **à** n **faire**
Si T[i] mod 2 =0 **alors**
j←0
T1[j]←T[i]
Sinon
K←0
T1[j]←T[i]
FinSi
Fin pour
4) FIN Extraire

T.D.O.Locaux :

Objet	Type/Nature	Rôle
i	Entier	Compteur

Programme Pascal :

```
program Ex5parite;
uses wincrt;
type  tab=array[1..10]of integer;
var   t,t1,t2:tab;
      n,n1,n2:integer;

procedure saisir(var t:tab;var n:integer);
var i:integer;
begin
writeln('donner la taille du tableau');
readln(n);
    for i:=1 to n do
    repeat
 writeln('Donner l''élément n°',i);
 readln(t[i]);
    until t[i]>0;
end;

procedure extraire(t:tab;n:integer;
var t1:tab;var j:integer;  var t2:tab;var k:integer  );
  var i:integer;
begin
j:=0; k:=0;
for i:=1 to n do
if t[i] mod 2=0 then begin
                     j:=j+1;
                     t1[j]:=t[i];
                     end
                else begin
                     k:=k+1;
                     t2[k]:=t[i];
                     end;
end;
```

```
procedure afficher(a:tab;b:integer);
var i:integer;
begin
    for i:=1 to b do
writeln('l''élément n°',i,'est:', a[i]);
end;

begin
saisir(t,n);
extraire(t,n,t1,n1,t2,n2);
writeln('Le tableau des pairs: ');
afficher(t1,n1);
writeln('le tableau des impairs:');
afficher(t2,n2);
end.
```

Exercice 6 :

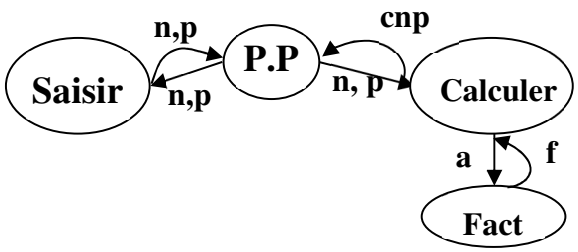
Écrire un programme qui permet de saisir deux entiers n et p (tq 1≤p≤n) et de calculer puis afficher le nombre de combinaison de p éléments parmi n : CNP

sachant que: $C_n^p = \frac{n!}{p!*(n-p)!}$

Corrigé :

Pour résoudre ce problème on besoin de définir les modules suivants :

- **procédure Saisir(n, p)** qui permet de saisir n et p.
- **fonction Calculer(n, p)** permet de calculer la CNP.
- **fonction fact(a)** permet de calculer la factoriel de a.



Analyse du programme principal:

NOM: Combinaison

Résultat=Ecrire("La CNP est",CNP)

CNP ← FN Calculer (n,p)

(n,p)=Proc Saisir(n, p)

FIN Combinaison

Tableau de déclaration des objets:T.D.O. Globaux :

Objet	Type/Nature	Rôle
Calculer	Fonctions	
n, p,CNP	Entier	
Saisir	Procédure	

Algorithme de la procédure saisir:

0) **DEF PROC Saisir(VAR n,p:entier)**
1) Ecrire(" n= "),lire(n)
2) **Répéter**

   Ecrire("p = "),lire(p)

   Jusqu'à p dans [1..n]

3) **FIN saisir**

Algorithme de la fonction Calculer:

0) **DEF FN Calculer (n,p : entier) : entier**
1) **C←** FN Fact (n) **DIV** (FN Fact (p) * FN Fact(n-p))
2) **Calculer ← C**
3) **FIN Calculer**

T.D.O.Locaux:

Objet	Type/Nature	Rôle
C	entier	

Algorithme de la fonction Fact :

0) **DEF FN Fact(a:entier) : entier**
1) **[f ←1] Pour** i **de** 2 **à** n **faire**

   f ← f*i

   Fin Pour

2) **Fact ← f**
3) **FIN Fact**

T.D.O.Locaux:

Objet	Type/Nature	Rôle
f , i	entier	

Programme Pascal :

```
Program combinaison;
uses  wincrt;
var n,p,cnp:integer;

procedure saisir(var n,p:integer);
begin
write('n= '); Readln(n);
     repeat
          write('p= '); Readln(p);
     until (p in [1..n] );
end;

function fact(a:integer):word;
var i:integer;f:word;
begin
f:=1;
for i:=2 to a do  f:=f*i;
fact:=f;
end;

function calculer(n,p:integer):integer;
 begin
calculer:=fact(n) div (fact(n-p)*fact(p));
 end;

begin
saisir(n,p);
cnp:=calculer(n,p);
write('La CNP est ',cnp);
end.
```

Exercice 7:

Écrire un programme qui permet de Saisir un réel x entre 0 et 1, puis calculer puis afficher la valeur approché VA de exp(x) avec la formule suivante: $e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \ldots$

NB: Faire le calcul jusqu'à $\frac{x^i}{i!}$ <=0,001

Corrigé :

```
program exponentiel;
uses wincrt;
var x,VA :real;
procedure saisir(var x:real);
begin
     repeat
Writeln('donner un réel entre 0 et 1:');
readln(x);
     until (x>=0) and (x<=1);
end;

function fact (a:integer):word;
var i:integer; f:word;
begin
f:=1;   for i:=2 to a do
          f:=f*i;
fact:=f;
end;

function power(x:real;n:integer):real;
var i:integer;  p:real;
begin
 p:=1;   for i:=1 to n do
            p:=p*x;
power:=p;
end;
```

```
function expo(x:real):real;
var i:integer; s,d,k:real;
begin
s:=1;i:=0;
repeat
     i:=i+1;
     d:=power(x,i)/fact(i);
     s:=s+d;
until d<=0.0001 ;
expo:=s;
end;
 begin
saisir(x); VA := expo(x) ;
writeln('valeur apporoché:',VA:10:8);
writeln('calcul directe  :',exp(x):10:8);
 End.
```

Répéter le calcul jusqu'à:

terme (d) soit <=0.0001

Exercice 8 :

Écrire un programme qui permet de saisir une chaîne Ch composée uniquement de chiffres et de lettres puis extraire les lettres dans une chaîne **CH1** et les chiffres dans une chaîne **CH2**.

Corrigé :

Analyse du programme Principal :

Nom : ex8

Résultat=Ecrire ("Lettres=",ch1),Ecrire("Chiffres =",ch2)

(ch1,ch2)=Proc extraire (ch,ch1,ch2)

Ch=Proc saisir(ch)

Fin ex8

Tableau de Déclaration des Objets globaux

Objet	Type/Nature	Rôle
Ch, ch1,ch2 Saisir, extraire	Chaine de caractère Procédure	

Algorithme de la procédure saisir :

0) **DEF PROC saisir(var ch :chaine)**
1) [ ] **Répéter**

Ch=donnée("ch=")

Jusqu'a verif(ch)=vrai

2) **Fin** saisir

Tableau de Déclaration des Objets locaux

Objet	Type/Nature	Rôle
verif	Fonction	

Algorithme de la fonction verif :

0) **DEF FN verif(ch :chaine) : booléen**
1) [ok←vrai , i←0]**Répéter**

i←i+1

Si Non(ch[i] dans ["0.."9", "a".."z","A".."Z"])

alors ok ← faux

Finsi

Jusqu'à (i=long(ch) ou (ok=faux)

2) verif ← ok
3) **Fin** verif

Tableau de Déclaration des Objets locaux

Objet	Type/Nature	Rôle
i ok	Entier booléen	

Algorithme de la procédure extraire :

0) **DEF PROC extraire(ch :chaine, Var lettres,chiffres :chaine)**
1) [lettres← "" , chiffres ←""] **Pour i de 1 à long(ch) faire**

Si ch[i] dans ["0".."9"] **alors** chiffres ←chiffres+ch[i]

Sinon Lettres←lettres+ch[i]

FinSi

FinPour

2) **Fin** extraire

Tableau de Déclaration des Objets locaux

Objet	Type/Nature	Rôle
i	Entier	

Traduction en Pascal :

```
program ex8;
uses wincrt;
var   ch,ch1,ch2:string;

function verif(ch:string):boolean;
var    i:integer;      ok:boolean;
begin
i:=0;   ok:=true;
repeat
i:=i+1;
if NOT( ch[i] in ['0'..'9', 'a'..'z', 'A'..'Z']) then
ok:=false;
until (i=length(ch)) or (ok=false);
verif:=ok;
end;

procedure saisir(var ch :string) ;
begin
repeat
write('ch='); readln(ch);
until verif(ch);        { ou until verif(ch)=true ; }
end ;
```

```
procedure extraire(ch:string; var lettres,chiffres:
string);
var i:integer;
begin
lettres:=''; chiffres:='';
for i:=1 to length(ch) do
if ch[i] in ['0'..'9'] then chiffres:=chiffres+ch[i]
else lettres:=lettres+ch[i];
end;
begin
saisir(ch) ;
extraire(ch,ch1,ch2);
writeln('lettres=',ch1);  writeln('chiffres=',ch2);
end.
```

Exercice n°9 :

Écrire un programme qui inverse une chaîne de caractères de longueur < 10 caractères, puis indique si c'est un palindrome. *(s'écrivant de la même façon de gauche à droite et de droite à gauche . Exemples : radar, elle,…) .*

Corrigé :

Analyse du programme principal

Nom : palindrome

Résultat= **Si** ch=ch1 **alors**

Ecrire("Palindrome")

Sinon Ecrire("Non palindrome")

Finsi

ch1 ← FN inverse(ch)

PROC saisir(ch)

Fin Palindrome

T.D.O globaux

Objet	Type/Nature	Rôle
ch,ch1 Saisir Inverse	Chaîne Procédure Fonction	

Algorithme de la procédure saisir :

0) DEF PROC Saisir (var ch :string)

1) **Répéter**

Ecrire("Donner une chaîne")

Lire (ch)

Jusqu'à long(ch)<10

2)Fin Saisir

Algorithme de la fonction Inverse :

0) DEF FN Inverse(ch :chaîne) : Chaîne

1) [ch2←""]**Pour** i **de** long(ch) **à** 1 (pas=-1) **faire**

Ch2 ←ch2 +ch[i]

FinPour

2) Inverse ← ch2

3) Fin Inverse

T.D.O Locaux

Objet	Type/Nature	Rôle
Ch2	Chaîne	
i	entier	

Traduction en pascal :

```
program palindrome;
uses wincrt;
var ch, ch1:string;

procedure saisir(var ch:string);
begin
    repeat
     writeln('Donner un mot ');
     readln(ch);
    until length(ch)<10;
 end;
function inverse(ch:string):string;
var ch2:string;
      i:integer;
```

```
begin
    ch2:='';
  for i:=length(ch) downto 1 do
ch2:=ch2+ch[i];
inverse:=ch2;
end;
begin
    saisir(ch);
ch1:=inverse(ch);
   if ch=ch1 then writeln('Palindrome')
                      else writeln('non palindrome');
end.
```

Exercice 10:

Écrire un programme permettant d'enlever les caractères qui se répètent dans une chaîne ch. Exemples :ch="sciences" → "scien" "programmation" → "progamtin"

Corrigé :

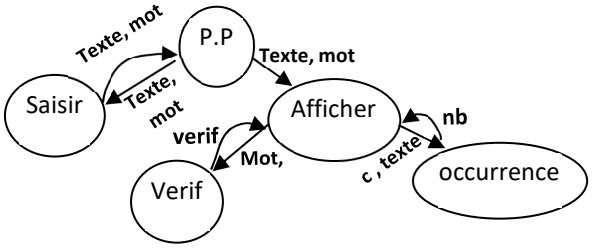
```
program car_double;
uses wincrt;
var ch:string;
 { supp_car: supprime toutes les occurrences d'un caractère d'indice i dans
une sous chaine de ch (entre l'indice i+1 et  length(ch) )}
   function supp_car(ch:string;i:integer) : string;
       var c:char; ch1,ch2:string;
    begin
        c:=ch[i];
        ch1:=copy(ch,1,i);
        ch2:=copy(ch,i+1,length(ch)-i);
       repeat
            delete(ch2,pos(c,ch2),1);
       until pos(c,ch2)=0;
        supp_car:=ch1+ch2;
    end;
{sup_double : supprime toutes les occurrences de tous les caractères de  ch}
  function sup_double(ch:string) : string;
   var  i:integer; ch3:string ;
```

```
        begin  ch3:=ch ;
        i:=0;
      repeat
          i:=i+1;
             ch3:= supp_car(ch3,i);
      until (i=length(ch3)) ;
    sup_double:=ch3;
        end;
begin
 writeln('Donner une chaine '); readln(ch);
 writeln( sup_double(ch) );
end.
```

Exercice 11

Écrire un programme qui demande à l'utilisateur un texte et un mot, puis affiche pour chaque lettre du mot, le nombre d'occurrences de cette lettre dans le texte de départ

Corrigé :



Pour résoudre ce problème on va définir les modules suivants :

- procédure Saisir(texte,mot)
- procédure afficher(texte,mot)
- fonction verif(mot,indice)
- fonction occurrence(c,texte)

Analyse du programme principal :

Nom:nblettres

Résultat=PROC afficher(texte,mot)

PROC saisir(texte,mot)

Fin nblettres

T.D.O.Globaux

Objet	Type/nature	Rôle
Texte, mot	Chaîne	
Saisir	Procédure	Saisir texte et mot
afficher	Procédure	Affiche le nbre occurrence de chaque lettre de mot dans texte

Algorithme de la procédure saisir :

0) DEF PROC saisir (var texte,mot:chaîne)
1) Ecrire(''Donner un mot''),lire(mot)
2) Ecrire(''Donner un texte'') , lire(texte)
3) Fin Saisir

Algorithme de la procédure Afficher :

0) DEF PROC Afficher(texte, mot:chaîne)
1) Pour i de 1 à long(mot) faire
   si verif(mot,i) alors Écrire(''Lettre= '',mot[i])
   nb ← occurrence(mot[i],texte)
   Écrire(nb,'' fois'')
   FinSi
   FinPour
2) Fin Afficher

T.D.O.Locaux

Objet	Type/nature	Rôle
i , nb	entier	
Verif	Fonction	Vérifier pour un caractère d'indice i de mot, si on n'a pas encore afficher le nbre d'occurrences.
occurrence	Fonction	Calcul le nbre d'occurrence d'un caractère c dans un texte

Algorithme de la fonction Verif :

0) DEF FN Verif(mot:chaîne, indice:entier) : booléen
1) mot2 ← sous chaîne(mot,1,indice-1)
2) **si** pos(mot[indice],mot2)=0 **alors** verif ← vrai
   Sinon verif ← faux
   FinSi
3) Fin Verif

T.D.O.Locaux

Objet	Type/nature	Rôle
Mot2	Chaîne	

Algorithme de la fonction Occurrence:

0) DEF FN Occurrence(c:caractère,texte:chaîne) : entier
1) [nb ← 0] pour j de 1 à long(texte) faire
   si texte[j]=c alors nb ← nb+1 FinSi
   FinPour
2) Occurrence ← nb
3) Fin Occurrence

T.D.O.Locaux

Objet	Type/nature	Rôle
Nb,j	entier	

Programme Pascal

```
program nblettres;
uses wincrt;
var  texte,mot : string;
procedure saisir(var texte, mot:string);
begin
    writeln('Donner un texte');    readln(texte);
    writeln('Donner un mot') ;    readln (mot);
end;
function verif(mot:string; indice:integer):boolean;
var mot2:string;
begin
    mot2:=copy(mot,1,indice-1);
    if pos(mot[indice],mot2)=0 then verif:=true
                          else verif:=false;
end;
function occurrence(c:char;texte:string):integer;
var  nb,j:integer;
begin  nb:=0;
   for j:=1 to length(texte) do
          if texte[j]=c then nb:=nb+1;
occurrence:=nb;
end;
procedure afficher(texte,mot:string);
var      nb,i : integer;
begin
for i:=1 to length(mot) do
    if verif(mot,i) then     begin
                    writeln('Lettre= ',mot[i]);
                    nb:=occurrence(mot[i],texte);
                    writeln(nb,' fois');
                          end;
end;
begin
saisir(texte,mot);
afficher(texte,mot);
end.
```

Ou

Verif(mot,i)=true