RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

UNIVERSITE LARBI TEBESSI DE TÉBESSA

FACULTÉ DES SCIENCES ET DE LA TECHNOLOGIE

DOMAINE DE FORMATION : **Sciences et Technologies**

Support de Cours :

Informatique 2

Cours, Exercices corrigés et Travaux pratiques

Matière : Informatique 2
Filière : Sciences et Technologie
Niveau : 1ère Année ST Tronc commun

Réalisé par : Dr. Ali Wided

Année universitaire : 2021/2022

Table des matières

Préface

Ce polycopié regroupe des notions de base sur la programmation en langage Pascal. Il constitue un support de cours pour des étudiants n'ayant aucune connaissance en programmation, destiné particulièrement aux étudiants de première année Science et technologie (ST). Il comporte donc des cours fondamentaux, avec des exercices corrigés et travaux pratiques, ces derniers contiennent l'ensemble des travaux traités, en séance de TP, par les étudiants de première année ST. Ces séries des travaux pratiques englobe toutes les notions présentées en cours.

Nous avons choisi le langage Pascal parce que c'est un langage conçu initialement pour des objectifs pédagogiques, facile à apprendre, et proche au langage algorithmique, aussi le programmeur n'a pas donc à se soucier des contraintes imposées par le langage et des spécificités techniques de ce langage. En utilisant Pascal, le programmeur aura l'impression d'écrire un texte en anglais.

L'objectif de ce cours est d'apprendre aux étudiants comment résoudre un problème par un programme, commençant par l'analyse de ce problème, déterminer le procédé la plus efficace pour résoudre ce problème, et enfin exprimer cette méthode en langage Pascal.

Ce polycopié permet un enseignement autonome. Les exercices de chaque chapitre ont une difficulté graduelle. Après avoir compris le cours, l'étudiant est conseillé d'essayer de résoudre les exercices par lui-même avant de voir la correction. L'étudiant ne doit pas oublier qu'un même exercice peut être résolu par différents programmes.

Qu'est-ce que c'est que Pascal ?

Pascal est un langage de programmation impératif qui, conçu pour l'enseignement, se caractérise par une syntaxe claire, rigoureuse et facilitant la structuration des programmes, pascal a été inventé par Niklaus Wirth en 1969 avec l'aide d'un de ses étudiants à l'école polytechnique de ZURICH.

D'où vient le nom Pascal ?

Le nom vient du mathématicien français Blaise Pascal. Blaise Pascal, né le 19 juin 1623 à Clermont (aujourd'hui Clermont-Ferrand) en Auvergne et mort le 19 août 1662 à Paris, est un mathématicien, physicien, inventeur, philosophe, moraliste et théologien français.

Chapitre 1 Les variables Indicées

1. Les tableaux unidimensionnels

1.1. Définition d'un tableau

Un tableau est un ensemble de N variables simples de même type appelées : éléments du tableau. Où chaque élément peut être identifié par sa position dans le tableau et peut être traitée comme une seule variable. Cette position est appelée indice.

L'accès à un élément particulier d'un tableau se fait grâce à l'opération d'indexage notée : Nom du tableau [indice]

Ainsi, $\forall i \in [1, N]$:

- Nom du tableau [i] donne la valeur de l'élément qui se trouve à la position i.

- Nom du tableau [i] n'est pas définie lorsque $i \notin [1, N]$

Exemple :

Si T est le tableau d'entiers suivant :

12	16	8	1	30	4

Alors :

T[2] a pour résultat 16

T[5] a pour résultat 30

T[10] n'est pas définie.

1.2. Déclaration des tableaux

Pour déclarer un tableau, il faut donner :

- Son nom
- Ses bornes : la borne inférieure correspond à l'indice minimal et la borne supérieure correspond à l'indice maximal.
- Le type des éléments le composant.

Syntaxe en Pascal :

Type nom = Array[<indice min> ... <indice max>] Of <type des composants> ;

ou Var nom : Array[<indice min> .. <indice max>] Of <type des composants> ;

Exemple :

Var T: Array [1..10] Of Integer;

Schématiquement, on va représenter la variable T comme suit :

i=1	i=2	i=3	i=4	i=5	i=6	i=7	i=8	i=9	i=10
20	15	80	120	300	79	400	90	500	30

Dans la mémoire centrale, les éléments d'un tableau sont stockés de façon linéaire, dans des zones adjacentes.

1.3. Création d'un tableau (Ecriture dans un tableau)

La création d'un tableau consiste au remplissage des cases le composant. Cela peut s'effectuer par lecture, ou par affectation.

Par exemple, pour remplir le tableau T précédent, on peut faire :

T [1] := 20 ; T [2] := 15 ; ….....T [10] := 30;

Si on devait lire les valeurs, il faudrait écrire :

```
For i: = 1 to 10 do
        Read (T[i]);
```

1.4. Affichage d'un tableau

Afficher un tableau revient à afficher les différents éléments qui le composent. Pour cela, on le parcourt à l'aide d'une boucle et on affiche les éléments un à un.

Exemple

Ecrire un programme qui permet de créer un tableau de type Integer T1 de taille 10 par lecture, et un tableau T2 de même taille en mettant dans T2[i] le double de T1[i], i ∈ {1, .., 10 }. Afficher T2.

```
Program  affichageTableaux;
Type tab = Array[1..10] Of Integer ;
Var T1, T2 : tab ;
i : Integer ;
Begin
{Lecture de T1 }
For i :=1 To 10 Do
Begin
        Write('donner T1[',i,']:');
        Readln(T1[i]);
End;
{création de T2 }
For i :=1 To 10 Do
          T2[i] := 2*T1[i];
{affichage de T2 }
For i: =1 To 10 Do
      Writeln('T2[',i,']=', T2[i]);
End.
```

Remarque :

- Quand on ne connait pas à l'avance le nombre d'éléments exactement, il faut maximiser ce nombre, pour n'utiliser qu'une partie du tableau.
- Il est préférable de définir un TYPE tableau réutilisable, plutôt que de déclarer à chaque fois, en VAR, le tableau en entier.
- Il est interdit de mettre une variable dans l'intervalle de définition du tableau.

1.5. Traitement d'un tableau

Après avoir créé un tableau, on peut y effectuer plusieurs opérations comme le calcul de la somme ou de la moyenne des éléments, la recherche du plus petit ou du plus grand élément du tableau, le test d'appartenance d'un élément au tableau, …etc.

Exemple: on considère un tableau d'entiers T déclaré comme suit :

```
Var T : Array[1 .. n] Of Real ;
```

Somme des éléments d'un tableau

On effectue la somme des éléments du tableau T, le résultat est dans la variable S

```
S := 0 ;
For i :=1 To n Do
        S := S + T[i];
Writeln('la somme des éléments de T est:', S);
```

Minimum d'un tableau

On cherche le minimum des éléments du tableau T, le résultat est dans la variable min :

```
Var min: real;
min := T[1] ; //on met le premier élément dans la variable min
For i :=2 To n Do
If T[i] < min Then
        min := T[i];
Writeln('Le minimum des éléments de T est:', min) ;
```

Test d'appartenance

On cherche si l'élément x appartient à T, le résultat est mis dans la variable booléenne bool :

```
bool := false;
For i:=1 To n Do
If T[i]=x Then bool := true;
If bool then
```

Write(‘x appartient à T’);

Else Write(‘x n’’appartient pas à T’);

On remarque que l’on peut arrêter la recherche si l’on rencontre l’élément x dans T. Pour cela, il faut utiliser une boucle While ou Repeat.

1.6. Ajout d’un élément

On suppose que le tableau T est « rempli » et qu’il reste une case non occupée à la fin (la $n^{ième}$ case).

Si on veut alors ajouter un entier x à la fin du tableau, il suffira d’écrire : T[n] := x

Mais, si on veut ajouter x dans une case dont l’indice k est différent de n, il faudra décaler les éléments T[k], T[k+1], …, T[n-1] vers la droite pour libérer la case d’indice k :

```
For i := n Downto k+1 Do
        T[i] := T[i-1];
            T[k] := x;
```

1.7. Suppression d’un élément

Pour supprimer l’élément se trouvant à la position k de T, on l’écrase en faisant décaler les éléments placés après lui vers la gauche :

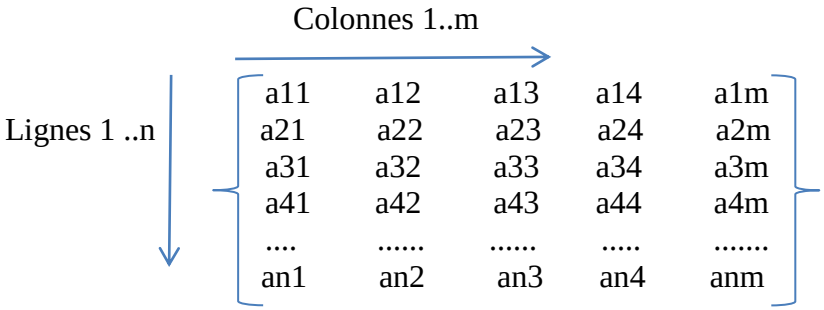
```
For i := k To n Do
        T[i] := T[i+1];
```

Il faut noter qu’après une telle opération, l’élément se trouvant à la dernière position (n) n’est plus significatif.

2. Les tableaux bidimensionnels (les matrices)

2.1. Définition d'une matrice

Les matrices sont des tableaux à deux dimensions. Schématiquement, on va représenter la matrice M comme suit :



Pour accéder à un élément de la matrice, nous écrivons M [i, j] où :

- L'indice i est numéro de la ligne,
- l'indice j est numéro de la colonne.

2.2. Déclaration d'une matrice

Syntaxe en Pascal :

On déclare une matrice à deux dimensions de la façon suivante :

Var identificateur : **Array**[1..M, 1..N] Of type_éléments ;

Remarque :

- Dimension du tableau ci-dessus : 2 (matrice)
- Nombre de cases disponibles : M x N
- Le tableau T est bidimensionnel, tous les éléments T[i,j] sont des variables (i = 1 ... M, j = 1 ... N) de même type
- pour accéder au contenu d'une case on utilise T[i,j].

Exemple

Nous désirons réaliser un programme permettant l'initialisation d'une matrice de dimension 5. Il s'agit donc d'une matrice à 5 colonnes et 5 lignes, ne comportant que des 0, sauf sur sa diagonale où il n'y a que des 1.

Program Matrice ;

Const

MAX_L = 5 ;

MAX_C = 5 ;

Type M = array [1 .. MAX_L, 1 .. MAX_C] of integer ;

Var i, j : Integer ;

MAT : M ;

Begin

for i := 1 to MAX_L **do**

begin

for j := 1 to MAX_C **do**

```
begin
if i = j then
MAT [i, j] := 1
else
MAT [i, j] := 0 ;
write (MAT [i, j]) ;
end ;
writeln ;
end ;
End.
```

Example :

Ecrire un programme qui permet de saisir et d'afficher un tableau M à deux dimensions d'entiers de demensions 2x3.

```
Program SaisiretAfficher;
Var M = array[1..2, 1..3] of integer ;
i,j : integer ;
begin  {saisie de M}
for i :=1 to 2 do
for j :=1 to 3 do
begin
write('donner M[',i,',',j,']:');
readln(M[i,j]);
end;
(* affichage de M *)
for i :=1 to 2 do
begin
for j :=1 to 3 do
write(M[i,j],' ');
writeln;
end; end.
```

3. Exercices Corrigés

3.1. Exercices

Exercice 1:

Écrire un programme PASCAL qui permet Calculer la somme et la moyenne des éléments d'un Tableau V réel de taille N.

Exercice 2:

Écrire un programme PASCAL qui permet de rechercher le plus petit élément dans un tableau réel T ainsi que sa position.

Écrire un programme PASCAL qui permet de rechercher le plus grand élément dans un tableau réel T ainsi que sa position

Exercice 3 :

Soit T un Tableau de type réel de taille N. Écrire un programme PASCAL qui permet de rechercher si une valeur réelle x existe ou non dans le tableau T. Dans le cas ou x existe dans T, on affiche aussi sa position.

Exercice 4 :

Ecrire un programme pascal permettant de faire le tri d'un tableau T de type TAB et de taille n, avec la méthode de tri par sélection.

Ecrire un programme pascal permettant de faire le tri d'un tableau T de type TAB(tableau d'entiers) et de taille n, avec la méthode de tri à bulles.

Ecrire un programme pascal permettant de faire le tri d'un tableau T de type TAB(tableau d'entiers) et de taille n, avec la méthode de tri par insertion.

Exercice 5 :

Soit une matrice A réelle d'ordre NxM.

1 Écrire un programme PASCAL qui calcule la somme et la moyenne des éléments de la matrice A.

2 Écrire un programme PASCAL qui permet de calculer la somme de chaque ligne et le produit de chaque colonne.

Exercice 6:

1 Ecrire un programme Pascal qui permet de calculer la somme de deux matrices A et B contenant 4 lignes et 3 colonnes.

2 Ecrire un programme Pascal qui permet de calculer la transposée d'une matrice A de 2 lignes et 3 colonnes.

3 Ecrire un programme Pascal qui permet de calculer la somme des éléments de la diagonale d'une matrice carrée d'ordre 3.

3.2. Corrigées

Exercice 1:

```
Program exercice_1;
var V : array[1..100] of Real;
i, n : integer ;
Som, Moy : real;
Begin
Write('Donner la taille du tableau : ');
Read(n);
Writeln('Donner les éléments du tableau : ');
For i:=1 to n do
    Read( V[i] );
Som:=0;
For i:=1 to n do
Som:= Som + V[i];
Moy:=Som/n; {La moyenne égale à la somme sur le nombre d'éléments}
Writeln('La somme d'éléments  du tableau est : ', Som);
Writeln('La moyenne d'éléments  du tableau est : ', Moy);
End.
```

Exercice 2:

```
Program exo4_Min;
var T : array[1..100] of Real;
i, n, Pos : integer ;  Min:Real;
Begin
Write('Donner la taille du tableau T : ');
Read(n);
Writeln('Donner les éléments du tableau T : ');
For i:=1 to n do
Read( T[i] );
Min:=T[1]; Pos:=1; {On suppose que le Min est la valeur de la première case}
```

```
For i:=2 to n do {On parcours toutes les autres cases}
If T[i] < Min then {Pour chaque case qui est inférieur au Min}
begin
Min := T[i]; {On met à jour la valeur de Min  }
Pos := i; {On actualise sa position}
end;
Write('Le Min du T est : ',Min,' et sa position est : ', Pos);
End.
Program exo_Max;
var
V : array[1..100] of Real;
i, n, Pos : integer ; Max:Real;
Begin
Write('Donner la taille du vecteur V : ');
Read(n);
Writeln('Donner les composantes du vecteur V : ');
For i:=1 to n do
Read( V[i] );
Max:=V[1]; Pos:=1; {On suppose que le Max est la valeur de la première case}
For i:=2 to n do {On parcours toutes les autres cases}
If V[i] > Max then {Pour chaque case qui est supérieur au Max}
begin
Max := V[i]; {On actualise le Max}
Pos := i; {On actualise sa position}
end;
Write('Le Max du V est : ',Max,' et sa position est : ', Pos);
End.
```

Exercice 3 :

```
Program exo3;
var
T : array[1..100] of Real;
i, n, Pos : integer ; x:Real; found: Boolean;
Begin
Write('Donner la taille du tableau T : ');
Read(n);
Writeln('Donner les éléments du tableau T : ');
```

```
    For i:=1 to n do
    Read( T[i] );
    Write('Donner la valeur de x : ');
    Read(x);
    found:=false; i:=1; {Initialiser found à false}
    While (i<=n) AND (found = false) do
    begin
    If T[i]=x then {Si une case égale à x}
    begin
    found:=true; Pos:=i;
    End;
    i := i+1;
    end;
    If found =true Then
    Write('x existe dans le tableau et sa positon est : ', Pos)
    Else
    Write('x n''existe pas dans le tableau);
    End.
```

Exercice 4 :

```
program Tri_selection;
type A : array[1..100] of integer;
 var
T: A;
n ,i,j, pos_min, z :integer;
begin
write('donner la taille du tableau');
read(n);
For i: =1 to n do
readln(T[i]);
For i:=1 to n-1 do
begin
       pos_ min:=i;
For j:= i+1  to n do
begin
if ( T[j] < T[pos_min])then
     pos_min := j ;
end;
if ( pos_min <> i) then
Begin
        z =T[i];
       T[i]:=T[pos_min];
       T[pos_min]:=z;
 end;
end;
```

```
For i:=1 to n do
writeln(T[i]);
End.
program Tri_Bulles;
type A =array[1..100] of integer;
 var
T: A; n, z ,i,j:integer;
Echange:boolean;
begin
write('donner la taille du tableau');
read(n);
For i: =1 to n do
readln(T[i]);
for i:= n downto 1 do
   for j:= 1 to i-1 do
    if(T[j]> T[j+1]) then
    begin
        z :=T[j];
        T[j]:=T[j+1];
        T[j+1]:=z;
       end;
 For i:=1 to n do
writeln(T[i]);
end.
program Tri_insertion;
type A =array[1..100] of integer;
 var
T: A;  n ,i,j, TMP:integer;
begin
write('donner la taille du tableau');
read(n);
For i: =1 to n do
readln(T[i]);
For i: =2 to n do
begin
      TMP :=T[i]; j := i;
       while (j > 1 and T[j-1] > TMP)   do
          begin
             T[j] := T[j-1];
              j := j-1;
          end;
       T[j] := TMP;
end;
For i:=1 to n do
writeln(T[i]);

End.
```

Exercice 5 :

```
Program exo5_1;
var
A : array[1..10, 1..10] of Real;
i, j, N, M : integer;
Som, Moy: real;
Begin
Write('Donner les dimensions de la Matrice A : ');
Read(N, M);
Writeln('Donner les composantes de la matrice A : ');
For i:=1 to N do
    For j:=1 to M do
  Read( A[i, j] );
Som:=0; {Initialiser toujours la somme à ZÉRO}
For i:=1 to N do
    For j:=1 to M do
Som := Som + A[i, j];
Moy := Som / (N*M); {La moyenne égale à la somme sur nombre d'éléments}
Write('La somme est : ',Som,' La Moyenne est : ',Moy);
End.
```

```
Program exo5_2;
var A : array[1..10, 1..10] of Real;
i, j, N, M : integer;
Som, Prod : array[1..10] of Real;
Begin
Write('Donner les dimensions de la Matrice A : ');
Read(N, M);
Writeln('Donner les composantes de la matrice A : ');
For i:=1 to N do
    For j:=1 to M do
        Read( A[i, j] );
For i:=1 to N do
begin
Som[i] := 0; {Initialiser la somme de la ligne i à 0}
For j:=1 to M do
            Som[i] := Som[i] + A[i, j];
end;
For j:=1 to M do
begin
Prod[j] := 1; {Initialiser le produit de la colonne j à 1}
For i:=1 to N do
Prod[j] := Prod[j] * A[i, j];
end;
Writeln;
Writeln('La somme des lignes de la matrice A : ');
For i:=1 to N do
Write(Som[i]);
Writeln;
Writeln('Le produit des colonnes de la matrice A : ');
For j:=1 to M do
Write (Prod[j]);
End.
```

Exercice 6:

```
Program Somme_matrices;
Var  A, B, SOM: array[1..4,1..3]of real;
i, j:integer;
Begin
For i:=1 to 4 do
     For j:=1 to 3 do
 Begin
 Read(A[i,j]);
 Read(B[i,j]);
  SOM[i,j]:= A[i,j]+B[i,j];
  end;
For i:=1 to 4 do
  Begin
    For j:=1 to 3 do
        Write(SOM[i,j],' ');
Writeln;
end;
End.
Program transposee_matrice;
Var  A, A_T: array[1..2,1..3]of real;
i, j:integer;
Begin
For i:=1 to 2 do
    For j:=1 to 3 do
Begin
Read(A[i,j]);
    A_T[j,i]:= A[i,j];
end;
For i:=1 to 2 do
Begin
      For j:=1 to 3 do
            Write(A_T[i,j],' ');
Writeln;
end;
End.
Program Somme_diagonale;
Var A: array[1..3,1..3]of integer;
 i, j, som :integer;
Begin
For i:=1 to 3 do
     For j:=1 to 3 do
           Read(A[i,j]);
For i:=1 to 3 do
          som:= som+ A[i,i];
          Write('la somme des éléments de la diagonale = ', som);
End.
```

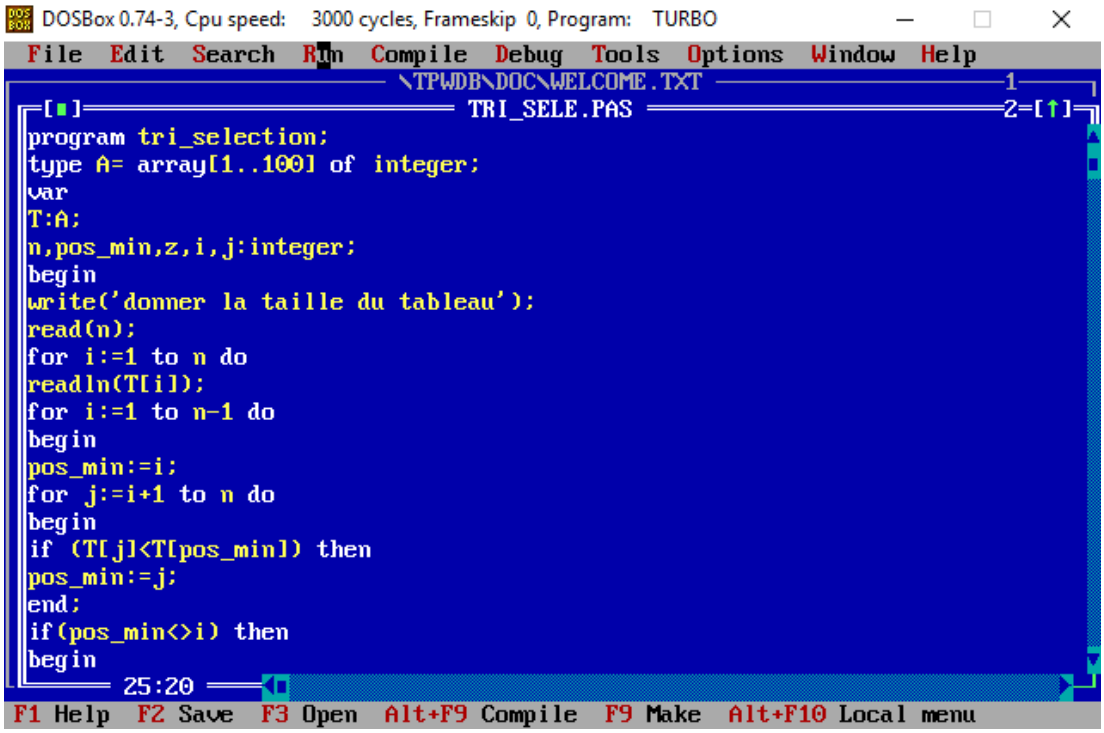
4. Travaux Pratiques

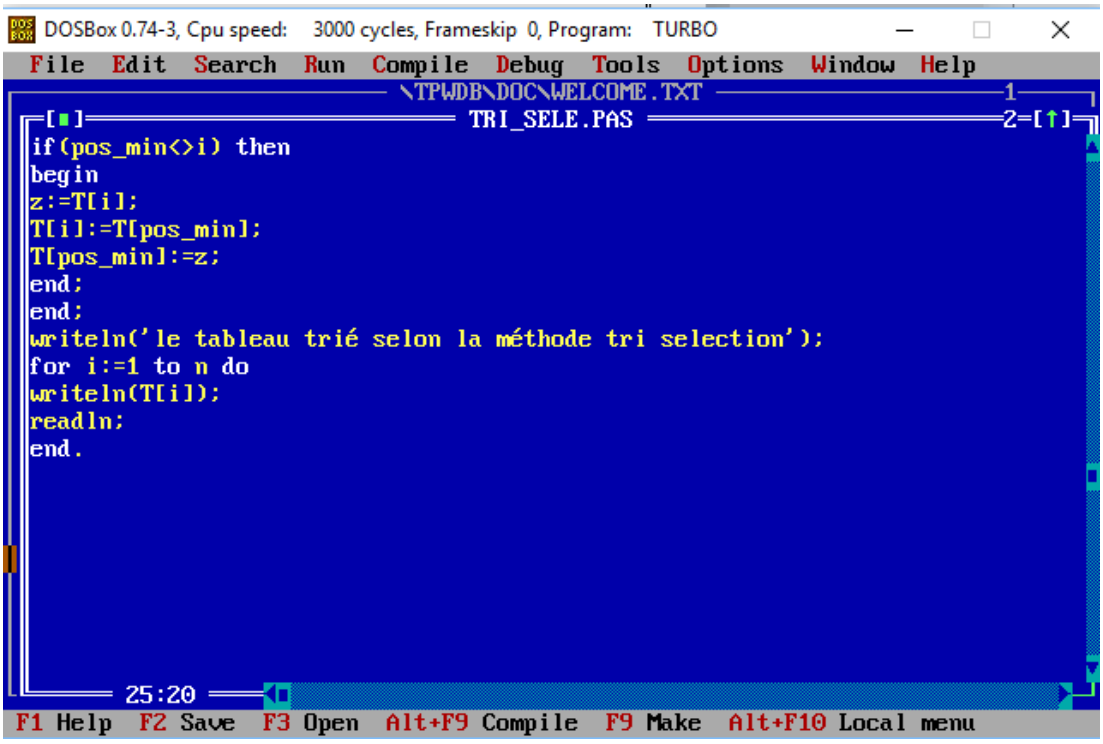
1 Ecrire un programme pascal permettant de faire le tri d'un tableau T de type TAB et de taille n, avec la méthode de tri par sélection.

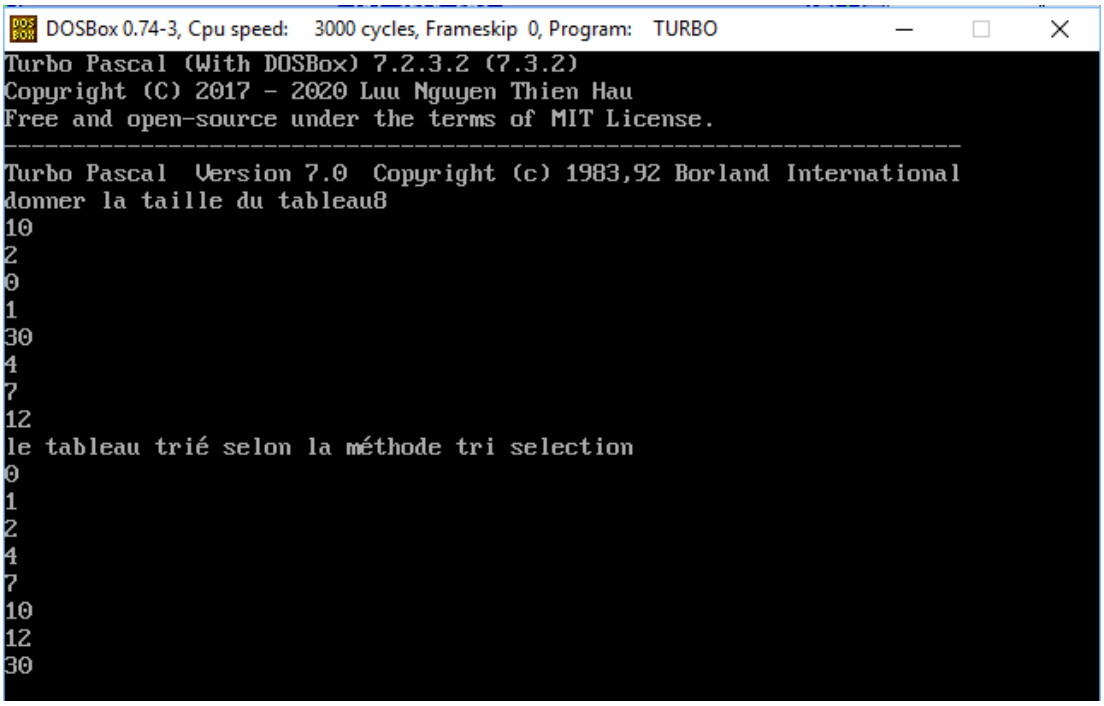
2 Ecrire un programme pascal permettant de faire le tri d'un tableau T de type TAB(tableau d'entiers) et de taille n, avec la méthode de tri à bulles.

3 Ecrire un programme pascal permettant de faire le tri d'un tableau T de type TAB(tableau d'entiers) et de taille n, avec la méthode de tri par insertion.

Tp 1 : la méthode de tri par sélection

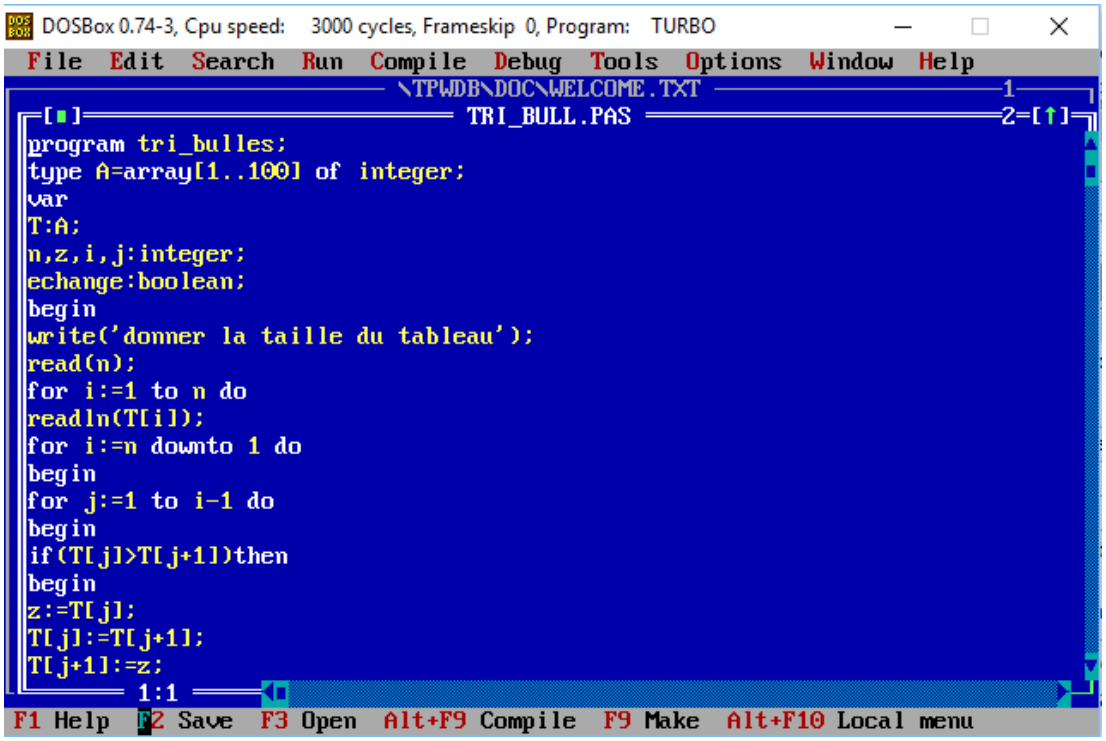


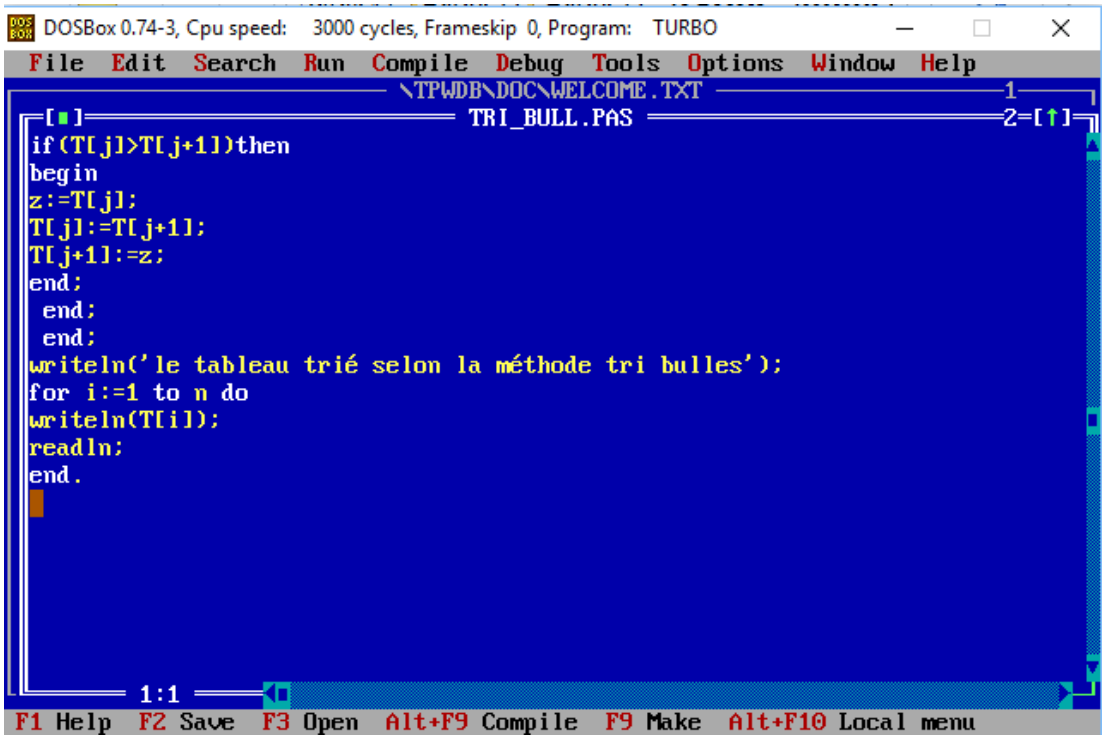

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: TURBO
File Edit Search Run Compile Debug Tools Options Window Help
\TPWDB\DOC\WELCOME.TXT
TRI_SELE.PAS
if(pos_min<>i) then
begin
z:=T[i];
T[i]:=T[pos_min];
T[pos_min]:=z;
end;
end;
writeln('le tableau trié selon la méthode tri selection');
for i:=1 to n do
writeln(T[i]);
readln;
end.
25:20
F1 Help F2 Save F3 Open Alt+F9 Compile F9 Make Alt+F10 Local menu

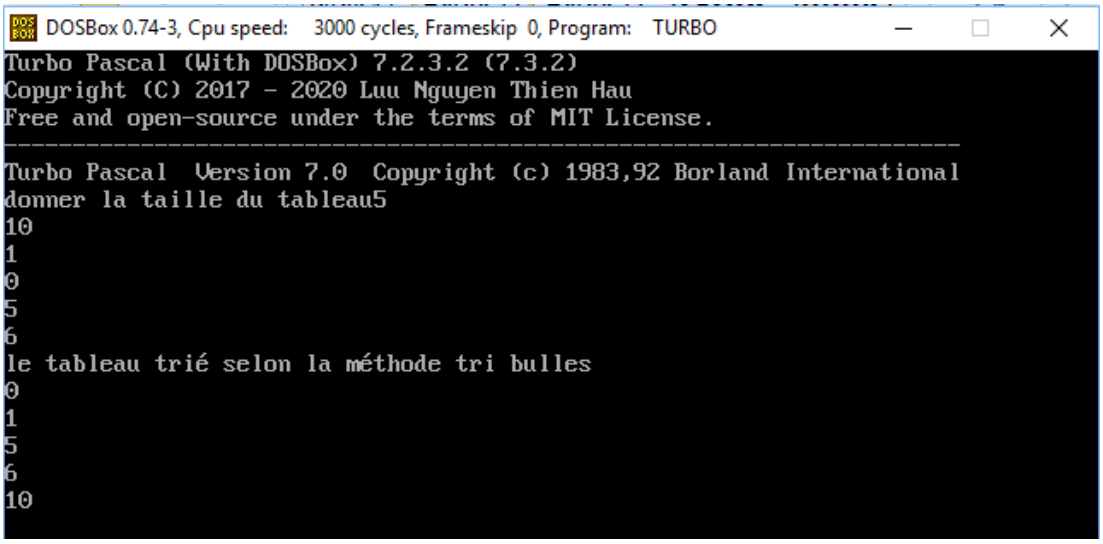

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: TURBO
Turbo Pascal (With DOSBox) 7.2.3.2 (7.3.2)
Copyright (C) 2017 - 2020 Luu Nguyen Thien Hau
Free and open-source under the terms of MIT License.
Turbo Pascal Version 7.0 Copyright (c) 1983,92 Borland International
donner la taille du tableau8
10
2
0
1
30
4
7
12
le tableau trié selon la méthode tri selection
0
1
2
4
7
10
12
30

Travaux Pratiques

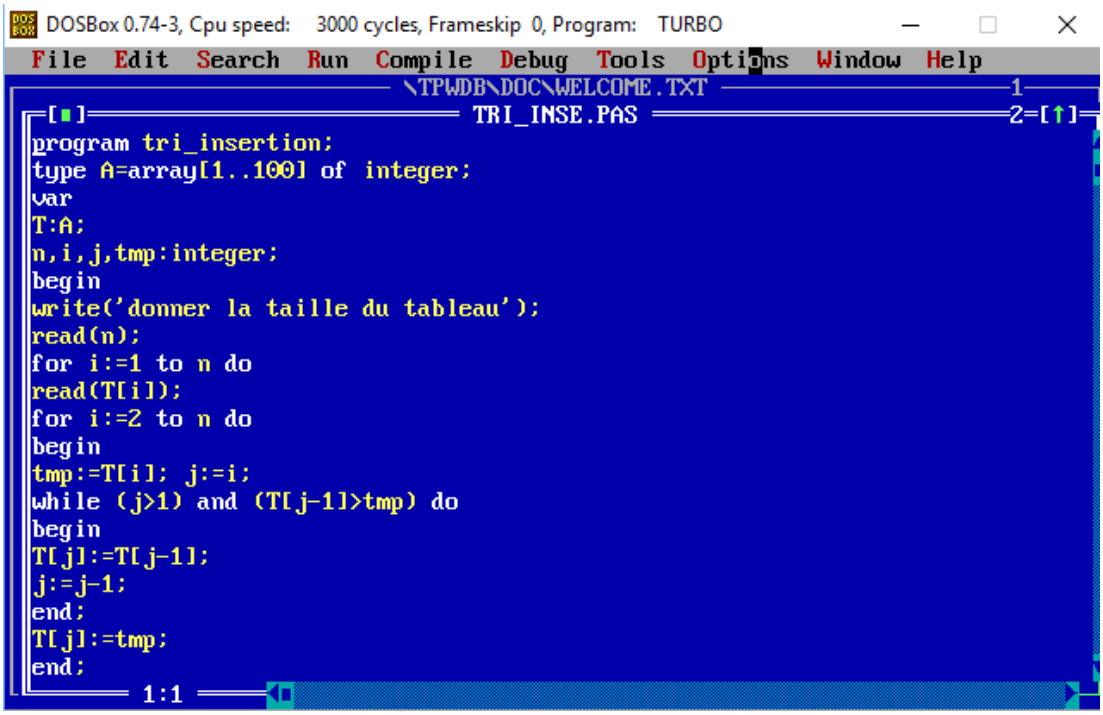
Tp2: la méthode de tri à bulles

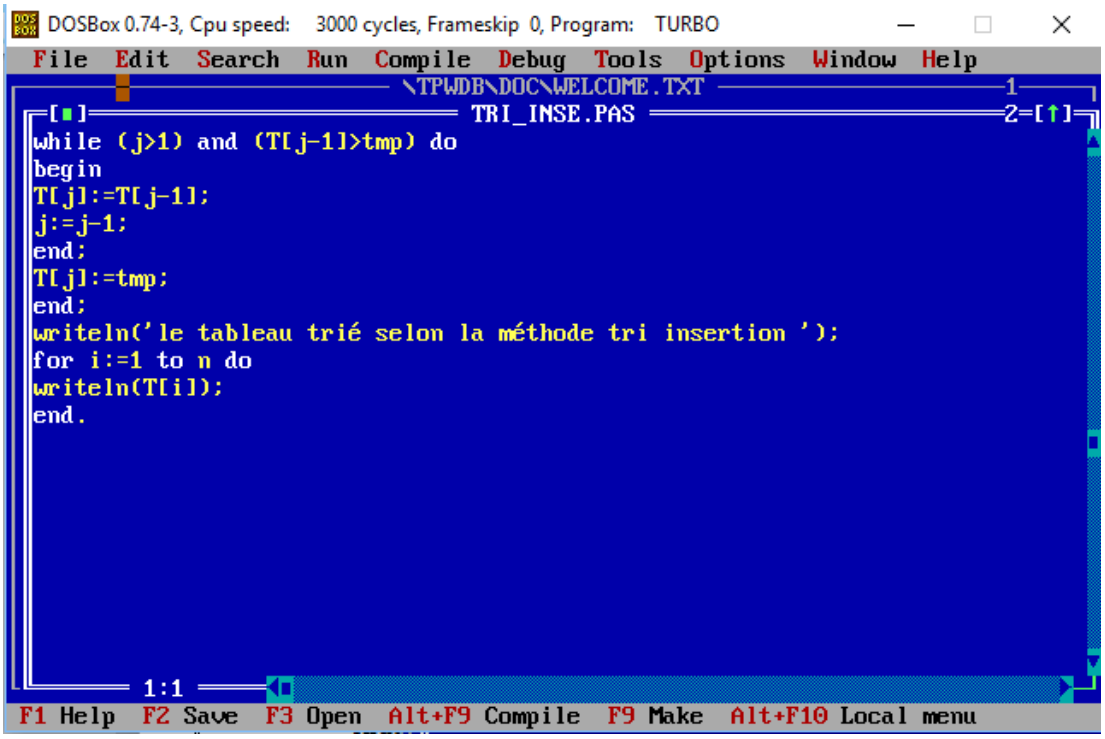


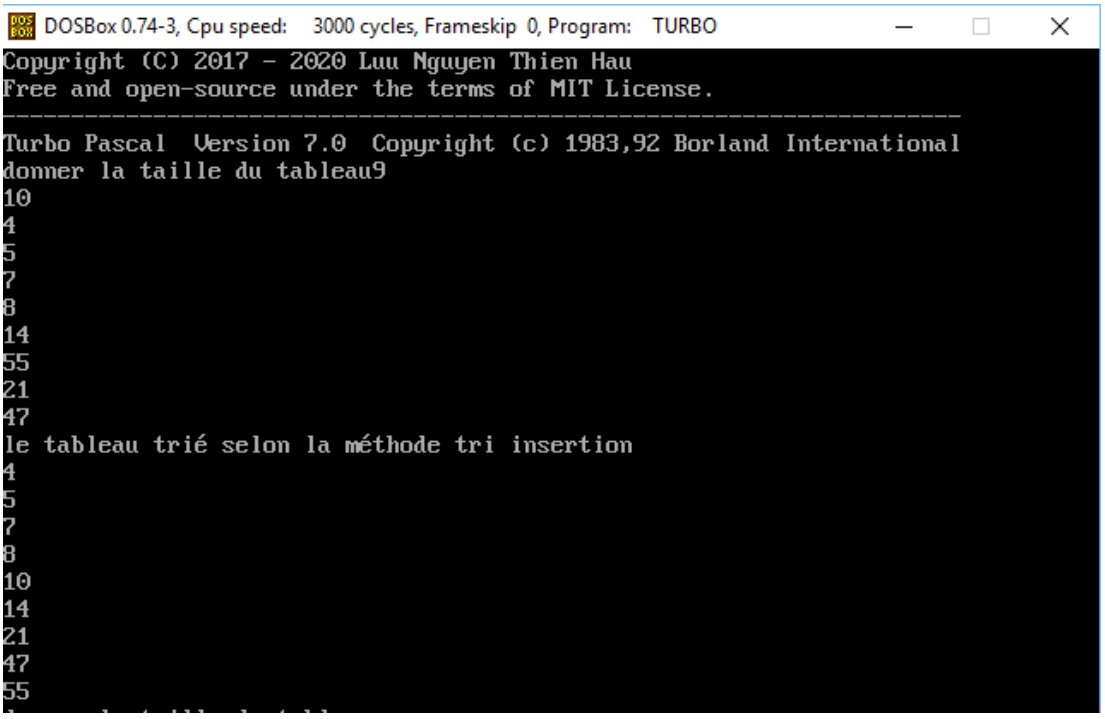




Tp3: la méthode de tri par insertion




DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: TURBO
File Edit Search Run Compile Debug Tools Options Window Help
\TPWDB\DOC\WELCOME.TXT
TRI_INSE.PAS
while (j>1) and (T[j-1]>tmp) do
begin
T[j]:=T[j-1];
j:=j-1;
end;
T[j]:=tmp;
end;
writeln('le tableau trié selon la méthode tri insertion ');
for i:=1 to n do
writeln(T[i]);
end.
1:1
F1 Help F2 Save F3 Open Alt+F9 Compile F9 Make Alt+F10 Local menu


DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: TURBO
Copyright (C) 2017 - 2020 Luu Nguyen Thien Hau
Free and open-source under the terms of MIT License.
Turbo Pascal Version 7.0 Copyright (c) 1983,92 Borland International
donner la taille du tableau9
10
4
5
7
8
14
55
21
47
le tableau trié selon la méthode tri insertion
4
5
7
8
10
14
21
47
55

Chapitre 2 Les fonctions et procédures

(Les sous-programmes)

1. Description d'un sous-programme

La solution d'un problème pratique est complexe et longue. Résoudre ce type de problème en une seule phase peut conduire à un échec ou un algorithme compliqué et illisible, dont l'écriture est assez difficile.

Lorsque la solution est réalisée par un seul algorithme, elle sera longue et lourde et très difficile à concevoir.

Pour résoudre facilement et sûrement un problème, on le décompose en parties plus simples, plus ou moins indépendantes. La solution complète sera composée de ces solutions partielles. Donc, l'algorithme résolvant ce problème est composé de l'ensemble des sous-programmes résolvant les sous problèmes partielles.

2. Type de sous-programme

Un sous- programme peut se présenté sous forme de fonction ou de procédure.

2.1. Une fonction

est un ensemble d'instructions qui forment un sous-programme, Chaque fois qu'on l'appelle elle renvoie au programme appelant une valeur qui est le résultat du traitement effectués par les instructions de la fonction.

Structure d'une fonction :

```
function nom_fonction (paramètres : types_params ) : type_resultat;
    Var  locales : types_locales;
    res : type_resultat;
begin
        { ... dans le corps, on fait ce qu'on veut de res ...}
        { on dit que le résultat est res }
        nom_fonction := res;
end;
```

‹ nom_fonction › est le nom de la fonction.

‹paramètres› les paramètres de la fonction.

‹ locales › les variables définit dans cette partie ne sont utilisables qu'à l'intérieur de la fonction.

‹ le corps › : Cette partie doit contenir au moins une action de retour (renvoie) de la valeur de la fonction : ‹nom_fonction›←‹expression›.

Remarque :

Les paramètres de la fonction sont tous des données. Le résultat est renvoyé dans le nom de la fonction, qui est lui-même une variable de retour.

Exemples :

a- La fonction calculant la factorielle entier donné d'un nombre entier

```
Function facto (x: Integer): Integer;
Var i, r: Integer;
Begin
r := 1 ;
For i := 2 to x Do
        r := r * i ; { r = i! }
facto := r ;
End ;
```

Fonction nous disant si un nombre est multiple de 5 ou non.

```
Function mult5 (x : Integer) : Boolean ;
Begin
r := 1 ;
If x Mod 5 = 0 Then
        mult5 := True ;
Else
        mult5 := False ;
End;
```

2.2. Une procédure

Une procédure est un groupe d'instructions qui ne renvoie pas de valeur après leur exécution. Une fonction, quant à elle, est en réalité une procédure qui retourne une valeur.

Déclaration d'une procédure

La déclaration d'une procédure définit un sous-programme et lui associé un identificateur avec lequel il sera appelé. Elle se fait dans l'entête de l'algorithme, elle consiste à :

- Attribuer un nom à la procédure.
- Définir son entête et son corps.

Syntaxiquement, la déclaration se fait dans l'entête de l'algorithme, comme suit :

```
procedure nom_procedure(paramètres : types_params);
      Var  locales : types_locales;
begin
            <  corps >
end;
```

Exemples :

```
procedure Min ( a, b : integer ;
var  inf : integer );
begin
        if a < b then inf := a
        else inf := b;
end;
```

```
procedure Max ( a, b : integer ;
var  sup : integer );
begin
   if a > b then sup := a
   else sup := b;
end;
```

```
procedure MinMax ( x, y : integer ;
var  p, g : integer );
begin
Min (x, y, p);
Max (x, y, g);
end;
```

```
program exo1;
var u, v, pp, pg : integer; { var
globales }
{ ici proc édures Min, Max et
            MinMax }
begin
     write (’u v ? ’);
     readln (u, v);
     MinMax (u, v, pp, pg);
     writeln (’min ’, pp, ’ max ’, pg);
end;
```

2.3. Procédure vs fonction

Exemple du produit

```
                    PROGRAM exemple;

                    VAR a, b, c, d : real;
PROCEDURE Produit (x, y : real;     |   FUNCTION Produit (x, y : real) :
real;                               |
        var  z : real);             |        var  res : real;
BEGIN                               |   BEGIN
        z := x * y;                 |        res := x * y;
END;                                |        Produit := res;
                                    |    END
                        BEGIN
                    write (’a b ? ’);
                        readln (a, b);
Produit (a, b, c);                  |        c := Produit (a, b);
Produit (a-1, b+1, d);              |        d := Produit (a-1, b+1);
                    writeln (’c = ’, c, ’ d = ’, d);
                        END.
```

3. Mode de passages de paramètres

Il existe deux type de passage de paramètres : par valeur et par référence (par variable ou encore par adresse).

3.1 Passage paramètres par valeur

C'est le mode de transmission par défaut, il y a copie de la valeur, des paramètres effectifs dans les variables locales issues des paramètres formels de la procédure ou de la fonction appelé. Dans ce mode, le contenu des paramètres effectifs ne peut pas être modifié par les instructions de la fonction ou de la procédure. Car nous ne travaillons pas directement avec la variable, mais sur une copie. À la fin de l'exécution du sous-programme la variable conservera sa valeur initiale. Les paramètres dans ce cas sont utilisés comme données.

3.2. Passage paramètres par référence

Ici, il s'agit non plus d'utiliser simplement la valeur de la variable, mais également son emplacement dans la mémoire (d'où l'expression « par adresse »). En fait, le paramètre formel se substitue au paramètre effectif durant le temps d'exécution du sous-programme et à la sortie il lui transmet sa nouvelle valeur. Un tel passage de paramètre se fait par l'utilisation du mot-clé Var.

Exemples

Passage par valeur

```
function Puissance (x : real;n:integer) : real;
var p:real;
begin
 p:=1;
 while n>0 do begin
       p:= p * x;
       n:= n - 1;
 end;
 Puissance := p;
end;
var n : integer;
begin
n:=3;
writeln(' Pi au cube =' 
Puissance(3.14159,n));
writeln(n);  {la valeur de n n'a pas modifié}
end.
```

Passage par référence

```
procedure DoubleTheValue (var Value: Integer);
begin
  Value := Value * 2;
end;
```

Dans ce cas, le paramètre est utilisé aussi bien pour passer une valeur à la procédure que pour retourner une nouvelle valeur au code appelant.
Quand vous écrivez :

```
var
  X: Integer;
begin
  X := 10;
  DoubleTheValue (X);
```

la valeur de la variable X devient 20, parce que la procedure utilise une référence vers l'emplacement mémoire d'origine de X, en affectant sa valeur initiale.

4. Exercices Corrigés

4.1. Exercices

Exercice 01 (Somme des n premiers carrés)

Ecrire une fonction qui prend en paramètre un entier n et qui calcule la somme S des n premiers carrés : $S = 1^2 + 2^2 + 3^2 + \ldots + n^2$

Par exemple, pour n valant 4, S vaut 1 + 4 + 9 + 16 = 30.

Exercice 02

Soit montypetab un nouveau type défini de la façon suivante :

type montypetab = array[1..100] of integer; Soit tab une variable globale de type montypetab.

a) Ecrire une procédure sans paramètre qui demande à l'utilisateur la valeur de toutes les cases du tableau, procedure remplissage.

b) Ecrire une fonction qui prend comme paramètre un entier n et qui compte le nombre de valeurs du tableau superieur à n, function comptesupn

c) Ecrire une fonction sans paramètre qui calcule la somme des valeurs du tableau,

d) Ecrire une fonction sans paramètre qui calcule la moyenne des valeurs du tableau.

e) Ecrire le bloc principal du programme pour que l'utilisateur soit amené à remplir le tableau, et que soit affiché ensuite le nombre de valeurs égales à 0 ainsi que la moyenne des valeurs. On utilisera au mieux les procédures et fonctions déjà écrites.

Exercice 03

1. On considère le programme suivant :

a) Quel est le résultat à l'écran ?

b) Que se passe-t-il si on enlève "var" devant la déclaration du paramètre res ?

c) Que se passe-t-il si on remplace n par x dans toute la procédure tcaf ?

d) Que se passe-t-il si on remplace tcaf (10,x) par tcaf(10,res) ?

```
program surprise;
var x : real;
procedure tcaf ( n : integer; var res : real );
var i : integer;
begin
res := 1;
for i := 1 to n do
        res := res * i;
end;
begin
tcaf (10, x);
writeln ('Le résultat est ', x);
end.
```

Exercice 04

On considère le programme incomplet suivant :

Compléter le programme de façon cohérente :

1) ajouter les procédures demandevaleurs, trouvemaxetmin et affichemaxminmoy

2) ajouter la fonction calculemoyenne,.

```
program incomplet;
var t : array[1..10] of real;
max, min, moy : real;
begin
    demandevaleurs;
    moy := calculemoyenne;
    trouvemaxetmin ( max, min );
    affichemaxminmoy (max, min, moy );
End.
```

Exercice 05:
Ecrire un programme pascal qui permet de calculer

$$\binom{n}{k}=\frac{n!*k!}{(n-k)!}$$

Exercice 06
Ecrire un programme Pascal qui appelle une procédure qui fait l'échange du contenu de deux variables entières.

Exercice 07
On considère le programme incomplet suivant :

```
program triangle;
const minx = 0; maxx = 20;
miny = 0; maxy = 10;
type tabpoint = array[1..2] of real; { pour les 2 coordonnées d'un point }
var p1, p2, p3 : tabpoint ;
numeropoint : integer;
propriete : string;
begin
writeln('Entrez les coordonnées du premier point');
p1[1] := demandevaleur (minx, maxx); p1[2] := demandevaleur(miny, maxy);
writeln('Entrez les coordonnées du deuxième point');
p2[1] := demandevaleur (minx, maxx); p2[2] := demandevaleur(miny, maxy);
writeln('Entrez les coordonnées du troisième point');
p3[1] := demandevaleur (minx, maxx); p3[2] := demandevaleur(miny, maxy);
propriete := 'quelconque';
if (estisocele ( p1, p2, p3 ) then propriete := 'isocèle';
if (estequilateral (p1, p2, p3) then propriete := 'équilatéral';
affichetriangle ( propriete );
end.
```

a) Ecrire la fonction demandevaleur qui demande la valeur d'une coordonnée à l'utilisateur et vérifie que celle-ci est comprise dans l'intervalle dont les bornes sont données par les paramètres de la fonction.
b) Ecrire la fonction booléenne estisocele qui prend la valeur true si et seulement si le triangle formé par les 3 points passÈs en paramËtres est isocèle. Il est sans doute opportun d'écrire une autre fonction qui calcule la distance entre 2 points.
c) Ecrire la fonction booléenne estequilateral qui prend la valeur true si et seulement si le triangle formé par les 3 points passés en paramètres est équilatéral.

d) Ecrire la procédure affichetriangle qui affiche à l'écran un triangle composé d'*, respectant la convention suivante :

- il y a 1 seule étoile sur la première ligne.
- il y a toujours 1 étoile dans la première colonne de chaque ligne.
- il y a toujours 7 lignes.
- si le triangle est quelconque, il y a cinq étoiles de plus à chaque ligne.
- si le triangle est isocèle, il y a deux étoiles de plus à chaque ligne.
- si le triangle est équilatéral, il y a 4 étoiles de plus à chaque ligne jusqu'à la ligne 4, puis 4 étoiles de moins à chaque ligne jusqu'à la dernière ligne.

4.2. Corrigées

Exercice 01

```
program Somme_n_premiers_carres;
fuction Somme(n:integer) : integer;
VAR S,i : integer;
begin
 S:=0;
for i := 1 to n do
        begin
           S := S + i*i;
           Somme:=S;
        End;
END;
var nb : integer; { Variable du programme principal }
 begin  { Programme principal }
writeln('Entrer n');
 readln(nb);
writeln('S vaut ',Somme(nb));
END.
```

Exercice 02

```
a)procedure remplissage;
var i : integer;
begin
```

```
for i:=1 to 100 do
        begin
        writeln('Entrez la ', i,' ème valeur du tableau');
        readln( tab[i] );
end;
end;
b)function comptesupn ( n : integer ) : integer;
var i, cpt : integer;
begin
cpt := 0;
for i:=1 to 100 do
            begin
            if tab[i] > n then
                cpt := cpt + 1;
                comptesupn := cpt;
            end;
end;
c)function somme : integer;
var i, s : integer;
begin
s := 0;
for i:=1 to 100 do
          begin
          s := s + tab[i];
          somme := s;
         end;
end;
d)function moyenne : real;
begin
        moyenne := somme / 100;
end;
e) { Bloc principal }
begin
remplisage;
writeln ('Nombre de valeurs supérieures à 10 : ', comptesupn (10) );
```

```
writeln ('Moyenne : ', moyenne  );
end;
```

Exercice 03

a) Il s'agit du résultat de factorielle 10. On voit ici que le passage par référence permet de
renvoyer un résultat comme le ferait une fonction factorielle.

b) Si on enlève "var", la valeur de x n'est pas changée et comme aucune valeur ne lui a été
donnée, le résultat est imprévisible.

c) Cela ne change rien, le x de la procédure n'est pas confondu avec le x déclaré en variable
globale. Dans le "for i:=1 to x do", c'est donc le x déclaré en paramètre qui est pris en compte.

d) Il y a une erreur de compilation car res n'a pas été déclaré en variable globale et n'est donc
pas connu à cet endroit du programme.

Exercice 04

```
program complet
var t : array[1..10] of real;
max, min, moy : real;
procedure demandevaleurs;
var i : integer;
begin
for i:=1 to 10 do
        begin
            write('Entrez la valeur numéro ', i, ' : ');
            readln ( t[i] );
        end;
end;
function calculemoyenne : real;
var i : integer; somme : real;
begin
```

```
somme := t[1];
for i:=2 to 10 do
                begin
                somme := somme + t[i];
                 calculemoyenne := somme / 10;
                 end;
end;
procedure trouvemaxetmin ( var maximum : real; var minimum : real );
var i : integer;
begin
maximum := t[1]; minimum := t[1];

for i:=2 to 10 do
          begin
               if (t[i] > maximum) then maximum := t[i];
               if (t[i] < minimum) then minimum := t[i];
            end;
end;
procedure affichemaxminmoy (max : real; min : real; moyenne : real);
begin
writeln('Le maximum est ', max);
writeln('Le minimum est ', min);
writeln('La moyenne est ', moy);
end;
begin
demandevaleurs;
moy := calculemoyenne;
trouvemaxetmin ( max, min );
affichemaxminmoy (max, min, moy );
end.
```

Exercice 05:

```
PROGRAM factoriel;
VAR  k,n:INTEGER;  fact: real;
FUNCTION factorielle (N : INTEGER) : INTEGER ;
VAR i,F : INTEGER ;
```

```
BEGIN
F:=1;
FOR i:=1 TO N DO
F:=F*i;
factorielle :=F;
END;
BEGIN
write('Entrer deux entiers n et k');
readln(n,k);
fact:=( factorielle(n)*factorielle(k))/factorielle(n-k);
write(fact);
END.
```

Exercice 06:

```
PROGRAM echange_version1;
VAR x1, x2, x3: integer;
PROCEDURE echange;
BEGIN
x3:= x1;
x1:=x2;
x2:=x3;
End;
BEGIN { programme principal}
x1:= 10;
x2:= 4;
x3:= 15;
Writeln('valeurs de x1 et x2 avant appel de la procédure échange:');
Writeln('x1= ', x1,' x2= ',x2,' x3= ', x3);
echange;
Writeln('valeur après appel de la procédure échange:');
Writeln('x1= ', x1,' x2= ',x2,' x3= ', x3);
END.
PROGRAM echange_version2;
VAR  x1, x2, x3: integer;
PROCEDURE echange;
var  aux: integer;
BEGIN
aux:= x1;
x1:=x2;
x2:=aux;
End;
BEGIN { programme principal}
x1:= 10;
x2:= 4;
x3:= 15;
Writeln('valeurs de x1 et x2 avant appel de la procédure échange:');
```

```
Writeln('x1= ', x1,' x2= ',x2,' x3= ', x3);
echange;
Writeln('valeur après appel de la procédure échange:');
Writeln('x1= ', x1,' x2= ',x2,' x3= ', x3);
END.
```

Exercice 07:

a)

```
function demandevaleur (inf : integer; sup : integer) : real;
var x : real;
begin
writeln('Entrez une valeur entre ',inf:2,' et ', sup:2); readln(x);
while (x < inf) or (x > sup) do
begin
writeln('Cette valeur n"est pas bonne, recommencez !'); readln(x);
end;
demandevaleur := x;
end;
```

b)

```
function distance ( x1, y1, x2, y2 : real ) : real;
begin
distance := sqrt ( sqr(x2-x1) + sqr(y2-y1) );
end;
function estisocele (p1, p2, p3 : tabpoint) : boolean;
var d1, d2, d3 : real;
begin
d1 := distance ( p1[1], p1[2], p2[1], p2[2] );
d2 := distance ( p2[1], p1[2], p3[1], p3[2] );
d3 := distance ( p1[1], p1[2], p3[1], p3[2] );
if (d1 = d2) or (d1 = d3) or (d2 = d3) then estisocele := true
else estisocele := false;
end;
```

c)

```
function estequilateral (p1, p2, p3 : tabpoint) : boolean;
var d1, d2, d3 : real;
begin
d1 := distance ( p1[1], p1[2], p2[1], p2[2] );
d2 := distance ( p2[1], p1[2], p3[1], p3[2] );
d3 := distance ( p1[1], p1[2], p3[1], p3[2] );
if (d1 = d2) and (d1 = d3) then estequilateral := true
else estequilateral := false;
end;
```

d)

```
procedure affichetriangle ( ppte : string );
var ligne, colonne, ajout, nbcol : integer;
begin
writeln;
if (ppte = 'quelconque') then ajout := 5;
if (ppte = 'isocèle') then ajout := 2;
```

```
if (ppte <> 'équilatéral')
then begin { On traite les 2 premiers cas ensembles }
nbcol := 1;
for ligne := 1 to 7 do
begin
for colonne :=1 to nbcol do
write ('*');
nbcol := nbcol + ajout;
writeln;
end;
end
else
begin { Si c'est équilatéral, solution pas plus bête qu'une autre ...}
writeln('*');
writeln ('*****');
writeln ('*********');
writeln ('*************');
writeln ('*********');
writeln ('*****');
writeln ('*');
end;
end;
```

5. Travaux Pratiques

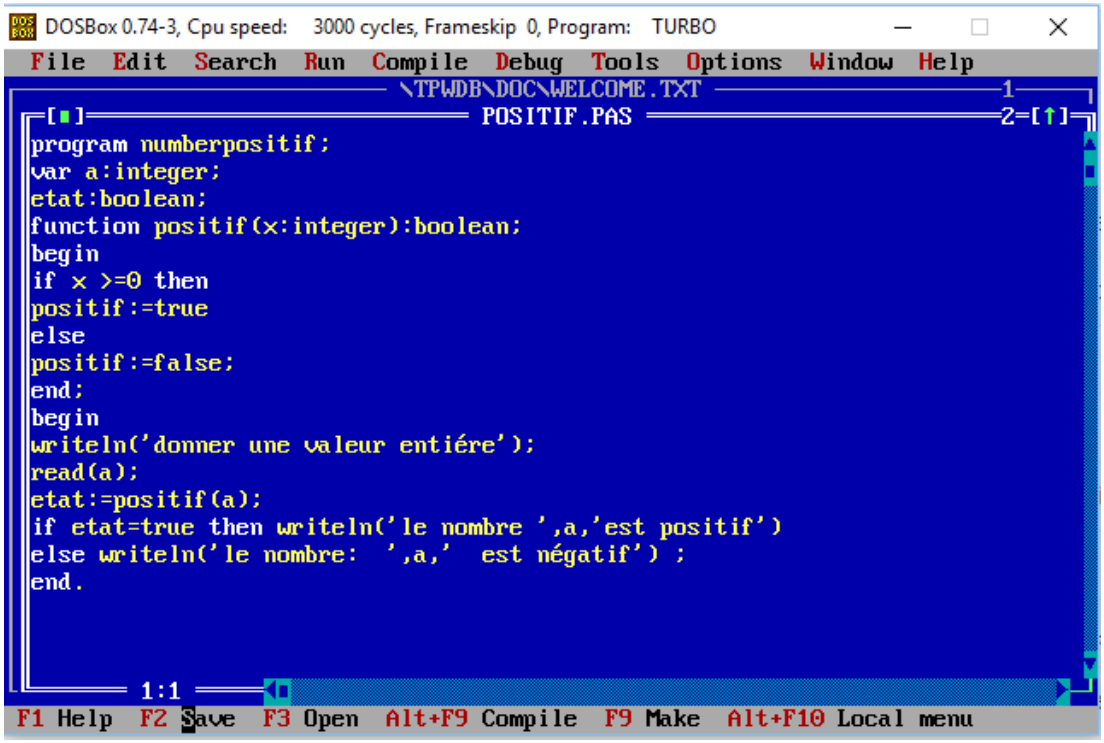
TP1:

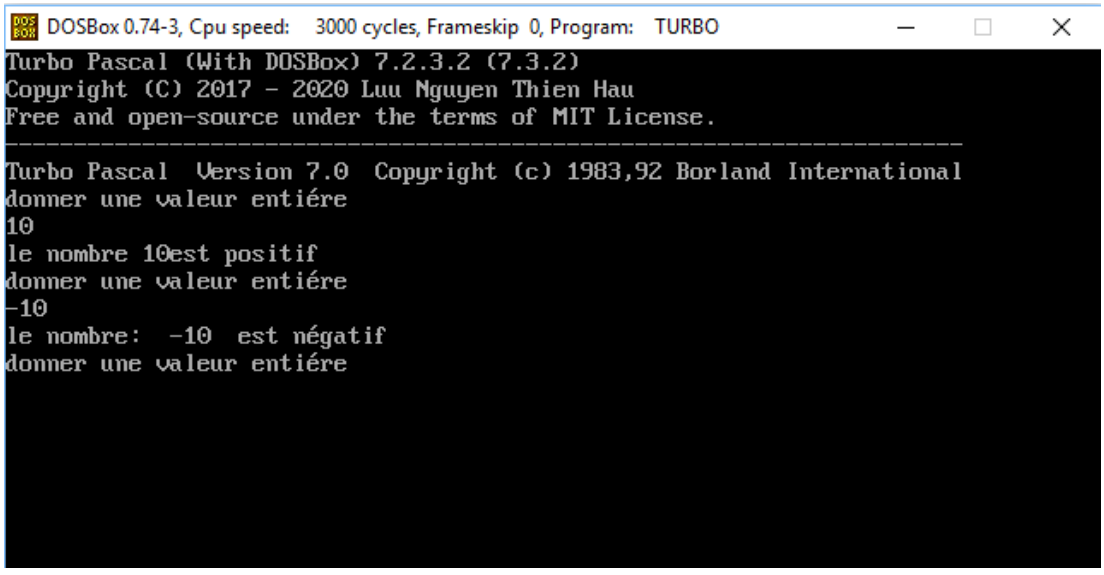
Ecrire un programme Pascal qui contient une fonction qui teste si un nombre est positif ou négatif

Tp2:

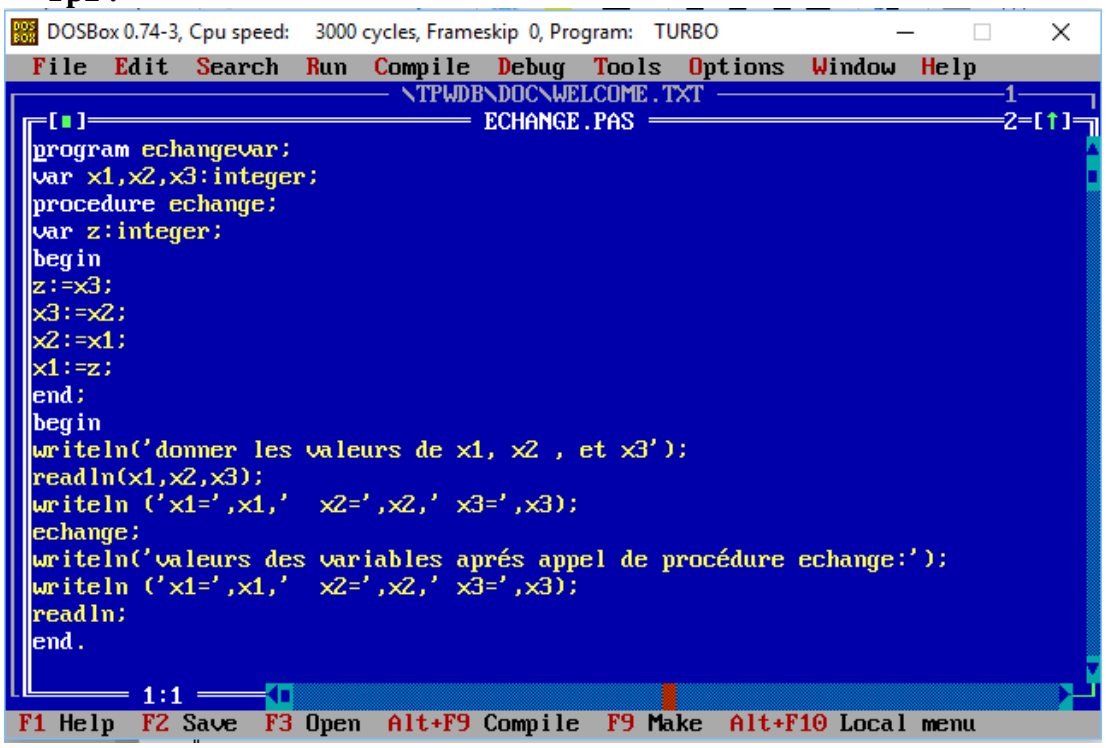
Ecrire un programme Pascal qui appelle une procédure qui fait l'échange du contenu de deux variables entières.

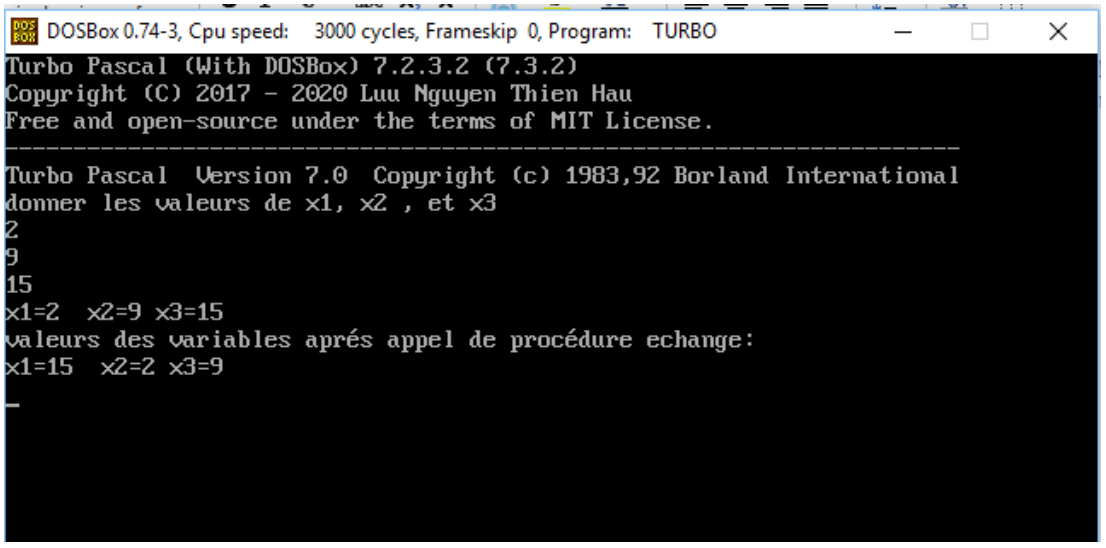
Tp1:





Tp2 :


DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: TURBO
File Edit Search Run Compile Debug Tools Options Window Help
\TPWDB\DOC\WELCOME.TXT
ECHANGE.PAS
program echangevar;
var x1,x2,x3:integer;
procedure echange;
var z:integer;
begin
z:=x3;
x3:=x2;
x2:=x1;
x1:=z;
end;
begin
writeln('donner les valeurs de x1, x2 , et x3');
readln(x1,x2,x3);
writeln ('x1=',x1,' x2=',x2,' x3=',x3);
echange;
writeln('valeurs des variables aprés appel de procédure echange:');
writeln ('x1=',x1,' x2=',x2,' x3=',x3);
readln;
end.
1:1
F1 Help F2 Save F3 Open Alt+F9 Compile F9 Make Alt+F10 Local menu


DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: TURBO
Turbo Pascal (With DOSBox) 7.2.3.2 (7.3.2)
Copyright (C) 2017 - 2020 Luu Nguyen Thien Hau
Free and open-source under the terms of MIT License.
Turbo Pascal Version 7.0 Copyright (c) 1983,92 Borland International
donner les valeurs de x1, x2 , et x3
2
9
15
x1=2 x2=9 x3=15
valeurs des variables aprés appel de procédure echange:
x1=15 x2=2 x3=9

Chapitre 3 : Les enregistrements et fichiers

1. Structure de données hétérogénes

Imaginons que l'on veuille afficher les notes d'une promotion par ordre croissant avec les noms et prénoms de chaque étudiant On va donc utiliser trois tableaux (pour stocker les noms, les prénoms, et les notes).

Noms	Prénoms	Notes
hawam	donia	12
tartar	ahmed	14
Nasri	solaf	15
mahdi	sofiane	18

Problème

Lorsque l'on va trier le tableau des notes il faut aussi modifier les tableaux « noms » et « prénoms »

1.1. Structure d'un enregistrement

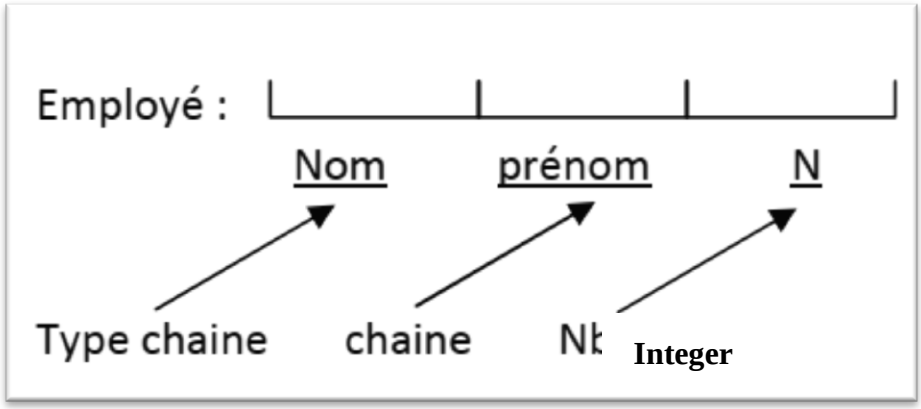
Un enregistrement est un ensemble quantique d'éléments n'ayant pas nécessairement le même type contrairement aux tableaux où tous les éléments doivent être de même type. On utilise la notion d'enregistrement lorsqu'on désire stocker plusieurs informations relatives à un même objet.

Exemple :

Supposant que l'on veuille stocker les informations suivantes : le nom, le prénom, le nombre d'enfants relatives à un employé. En pascal, ceci se traduit comme suit :

```
Personne = Record
Nom : string ;
Prénom : string ;                    Var  Employé : Personne ;
N: integer ;
End;
```

Remarques : La déclaration Personne = Record… sert à définir un type nommé Personne composé de trois éléments : le nom de type chaine, le prénom de type chaine, et le nombre d'enfants de type integer. Cette définition de type n'implique aucunement une réservation de cellule en mémoire centrale de l'ordinateur. La déclaration var Employé : Personne sert à déclarer une variable nommée Employé de type Personne, cette instruction implique une réservation en mémoire centrale d'un espace mémoire nommé Employé est organisé de la manière représentée sur la figure ci-dessus.



Les éléments : nom, prénom et N du type enregistrement Personne sont appelés des champs.

1.2. Syntaxe d'un enregistrement

Type nom du nouveau type **= Record**

Liste 1 de champs ;

Liste 2 de champs ;

.

.

Liste n de champs ;

End ;

Exemples :

```
type  étudiant = record
Nom : string;
Prénom : string;
Notes : real ;
End;
```

```
Type  point = record
x : integer;
y : integer;
End;
```

Ces types peuvent ensuite être utilisés pour définir des variables.

Exemples :

Var etudiant1, étudiant 2 : étudiant;

tabetudiant : array [1..100] of étudiant;

p1, p2, p3 : point;

quadrilatère : array [1..4] of point;

1.3. Syntaxe générale de l'utilisation d'un record :

Pour donner une valeur à un "champ" du record, on écrit un **.** entre le nom de la variable et le nom du champ : exemple

Nomdevariable**.**nomduchamp := expression

Pour utiliser la valeur d'un champ dans une expression, c'est pareil, on utilise un **.** entre le nom de la variable et le nom du champ. Reprenons les exemples suivants :

Exemple1 :

```
Type formulaire = Record
nom : string[20] ;
age :integer ;
sexe : char;
nb_enfants : integer;
End ;
Var personne : formulaire ;
BEGIN
personne.nom := 'mohammed' ;
personne.age := 40 ;
personne.sexe := 'M' ;
personne.nb_enfants := 3 ;
END.
```

Exemple 2 :

```
Program DATE ;
TYPE DAT =Record
Jour, mois, annee :integer ;
END ;
VAR DT :DAT ;
Begin
DT.jour :=7 ;
DT .mois :=7 ;
DT.annee :=2006;
Writlen(DT.jour, '/', DT.mois, '/', DT.annee) ;
Readln ;
END.
```

1.3. Type intervalle

Ce type décrit une suite de valeurs naturellement classées (nombres entiers ou caractères) limitée de part et d'autre par deux bornes constantes extrêmes.Pour le déclarer, on n'a qu'à appeler :

Type nom=valinf..valsup ;

Exemple :

Type mois=1 .. 12; {intervalle mois}
Type jour=1.. 31; {intervalle jour}
Type lettre_majus='A' .. 'Z' {intervalle alphabet majuscule }
Type lettre_minus='a'.. 'z' {intervalle alphabet minuscule }
Type chiffre=0..9 ;{intervalle chiffre}
Var m: mois;
j: jour;
Mlettre : lettre_majus;
lettre : lettre_minus ;
c: chiffre;

1.4. Enregistrement dans un enregistrement :

Il est possible en Pascal de définir un type enregistrement à l'intérieur d'un enregistrement.

Exemple :

Supposant que l'on veuille stocker la date d'embauche et la date de naissance dans la variable Employé, voici le résultat :

Type chaine = packed **array** [1..10] of char;
Nbenf = 1..10 ;
Date = **Record**
Jour : 1..31 ;
Mois : 1..12 ;
Année : 2000.. 2100 ;
End;
Personne = **Record**
Nom, Prénom : **chaine** ;
N : Nbenf ;
Dnais, Drecr : **Date** ;
End;

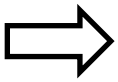
1.5. L'instruction "WITH"

L'instruction WITH est une instruction de factorisation. Considérant la variable Employé de type Personne où Personne défini comme suit :

Type chaine = packed **array** [1..10] of char;
Nbenf = 1..10 ;
Personne = **Record**
Nom, Prénom : chaine ;
N : Nbenf ;
End;
Var Employé : Personne ;

Considérons la séquence d'instruction suivante qui consiste à afficher le contenu de chaque champ de la variable 'Employé'.

Writeln (Employé.Nom) ;
Writeln (Employé.Prénom) ;
Writeln (Employé.N) ;



WITH Employé DO
Begin
Writeln (Nom) ;
Writeln (Prénom) ;
Writeln (N) ;
End:

Nous remarquons que dans chaque instruction d'affichage le nom 'Employé' figure. Il existe une écriture plus simplifiée mettant en facteur le nom de la variable 'Employé', on utilise l'instruction WITH.

Syntaxe de l'instruction WITH:

WITH Nom_variable **DO**
Instruction
Begin
Séquence d'instructions
End ;

La partie située après le mot clé DO s'appelle le corps de l'instruction WITH. Lorsque le corps de l'instruction WITH est une séquence d'instructions, on doit délimiter cette dernière par les deux mots clés Begin et End.

1.6. Les variantes dans les enregistrements

Certains champs d'un enregistrement peuvent être différents suivant la valeur d'un des champs en utilisant la structure Case... Of dans la déclaration de l'enregistrement.

Exemple :

```
Type Statut = (Célibataire, Marié, Divorcé, Veuf);
Personne = Record
Nom : string[20];          ← Partie figée
Case situation : Statut Of
Célibataire : ();
Marié : (enfants : 0..10);
Divorcé, Veuf : (enfants : 0..10; remarié : boolean)   Partie non figée
End ;
End ;
```

Dans cette exemple, suivant la situation de la personne, si c'est marié alors l'enregistrement Personne aura le champ enfants, si c'est divorcé ou veuf alors en plus du champ enfants on aura un autre champ booléen qui est remarié.

Les champs situés dans la partie figée feront partie de la structure de l'enregistrement à tout moment. Les champs situés dans la partie non figée sont soumis à une condition, ils feront partie de la structure de l'enregistrement selon cette condition.

Exemple 1 :

Soit le type temps_t défini comme suit :

```
TYPE heure_t = 0..23;

minute_t = 0..59;
seconde_t = 0..59;
temps_t = Record
        h : heure_t;
        m : minute_t;
        s : seconde_t;
End;
```

Soit les variables t : temps_t, et ns : integer.

1. Ecrire une procédure écriture(t) qui affiche à l'écran les valeurs des champs de t sous la forme h:m:s sans retour à la ligne.
2. Ecrire une procédure lecture(t) qui lit au clavier les valeurs à mémoriser dans les champs de t.
3. Ecrire une procédure calc_ns(t,ns) qui renvoie le nombre de secondes ns écoulées depuis 0 :0 :0.
4. Faire un programme qui lit t1 et t2 et qui dit si t1 est < ou = ou > ` a t2, en passant

5. par la conversion en secondes. Le programme utilise les procédures précédentes.

Exemple 2 :

Dans cet exercice Nous avons au plus 500 étudiants et pour chaque étudiant nous voulons avoir les informations suivantes : Le matricule, le nom et prénom, la section, le groupe, les notes de ses 3 Emd, de ses 3 tp, de son projet, sa moyenne.

Il faut donc déclarer la structure convenable pour ce problème. Puis construire 2 procédures : le premier (INFOGENE) qui permet de rentrer toutes les informations concernant un étudiant et toutes ses notes et le second (MOYALGO) qui calcule sa moyenne.

```
Type  etudiant = RECORD
matricule :integer;
nom,prenom:string[30];
section:'a'..'d';
groupe:1..12;
emd: array[1..3] of real;
tp: array[1..3] of real;
prj, moy:real;
rang: integer;
END;
promo = array[1..500] of etudiant;
```

2. Les fichiers

2.1. Notion de fichier

Un fichier est une structure de données, toutes de même type. L'accès à un élément (une donnée) du fichier peut se faire :

La notion d'identifiant de fichier:

- nom externe : c'est le nom du fichier pour le système de fichier du système d'exploitation considéré, par exemple : mon_fichier.dat
- nom interne : c'est le nom du fichier connu par le programme, il est déclaré par le programme, par exemple : Fichier_de_données

⇒ nécessité d'associer le fichier physique à sa représentation interne :

commande Assign

- le « repère » de position, Il permet de savoir où l'on se trouve dans un fichier lorsque l'on accède à celui-ci dans l'ordre de ses enregistrements

(accès séquentiel). C'est aussi lui qui détermine le résultat des opérations telles que : Eof ou Eoln.

Déclaration de fichier en PASCAL :

<Nom Fichier> : FILE OF <Type_Enregistrement> ;

<Nom Fichier> : TEXT ;

<Nom Fichier> : FILE ;

2.2. Les modes d'accès aux fichiers

Les méthodes par lesquelles on lit ou on écrit un enregistrement d'un fichier sont appelées les modes d'accès.

- La méthode d'accès que l'on veut utiliser doit être spécifiée au moment de l'ouverture du fichier. Un même fichier peut être accédé par des méthodes différentes selon son organisation qui elle a été définie au moment de sa création.
- Les différentes méthodes d'accès :
  - Accès séquentiel : enregistrements traités en séquence
  - Accès direct : accès direct par le numéro d'enregistrement
  - Accès indexé : accès par l'ordre des clés d'accès

Remarque : on ne peut utiliser que la méthode d'accès séquentielle avec une organisation de fichier séquentielle (Turbo Pascal permet également la méthode d'accès direct).

PASCAL permet de créer et de manipuler 3 types de fichiers.

- Les Fichiers à 'accès directe' de type FILE OF.
- Les Fichiers à accès séquentiel de type TEXT
- Les Fichiers 'sans type' de type FILE.

Les fichiers de type FILE OF sont constitués d'enregistrements de même type, alors que les fichiers TEXT admettent des enregistrements de type différents.

2.3. Lecture et écriture dans un fichier

Lecture des éléments (consulter le fichier)

On lit un élément par la procédure

read(<variable de type fichier>, <nom de variable de type de base>)

La valeur de l'élément qui est désigné par le pointeur est copiée dans la variable et le pointeur se déplace vers l'élément suivant. Quand le pointeur du fichier arrive à la fin du fichier (la fin du fichier est atteinte) et qu'il n'y a pas un autre élément à lire, on dit qu'on est en fin de fichier. Il y a une fonction à résultat booléen - eof(f), qui est vrai si et seulement si la fin de fichier a été atteinte.

Construction d'un fichier

On écrit (ajoute) un élément avec:

write(<variable de type fichier>, <valeur de type de base>)

La valeur s'écrit à la fin du fichier, c'est à dire que quand on écrit, on est toujours en fin de fichier. Le pointeur se déplace à la fin et est prêt à écrire un nouvel élément.

Fermeture d'un fichier

Quand on a fini à utiliser le fichier f, on doit le fermer avec Close(f)

Puis le fichier peut être ouvert encore une fois, soit avec un nouveau nom, soit avec le nom existant.

2.4. Les fichiers de type FILE OF

Exemple : Soit la séquence PASCAL suivante :

Type **ETUDIANT** = RECORD

Nom : string[20] ;

Prenom : string[20] ;

Age : integer ;

End ;

Var F1 : **FILE** OF ETUDIANT ; F2 : **FILE** OF Char ;

F3: FILE OF Real;

On a ainsi défini trois fichiers:

- le fichier F1 est constitué d'enregistrement de type ETUDIANT
- le fichier F2 est constitué de caractères
- le fichier F3 comporte des réels.

RESET (Fichier)

Procédure qui ouvre un fichier déjà existant et le prépare pour une lecture ou une écriture. Le pointeur de fichier est positionné sur le premier enregistrement du fichier.

READ, WRITE, READLN, WRITELN

Lecture ou écriture sur un fichier de type FILE OF.

Exemple

READ(Fichier, Var1,Var2, …,Varn);
WRITE(Fichier, Var1,Var2,…,Varn);

SEEK (Fichier, N)

Procédure permettant de positionner le pointeur de Fichier sur l'enregistrement Numéro N.

FILEPOS(Fichier)

Fonction qui retourne la position du pointeur du fichier spécifié.

FILESIZE (Fichier)

Fonction qui retourne le nombre total d'enregistrements du fichier.

REWRITE(Fichier) :

Procédure qui crée le fichier spécifié s'il n'existait pas. Dans le cas où il existait, il sera détruit. (Ouverture du fichier en écriture)

CLOSE(Fichier)

Procédure qui ferme le fichier.

EOF(Fichier)

Fonction qui retourne TRUE si la fin de fichier est rencontrée sinon retourne FALSE.

ASSIGN(Fichier, 'Nom_Physique') :

Procédure permettant d'associer le nom logique du fichier 'Fichier' au nom physique du fichier.

2.5. Les fichiers de texte

Les fichiers de texte sont des cas particuliers de fichiers. Un fichier de texte est formé d'éléments bien connus : les caractères. Chacun a déjà manipulé de tels fichiers : un programme (Pascal ou autre) est en fait un fichier de texte!

- Les caractères contenus dans un fichier de texte sont organisés en lignes, chacune terminée par une marque de fin de ligne. Après la dernière ligne, le fichier se termine par une marque de fin de fichier.
- Tout ce qui a été dit est valable pour les fichiers de texte. Précisons simplement qu'un fichier est un fichier de texte s'il est déclaré au moyen du type prédéfini text .

Remarque : Un fichier de type text n'est généralement pas équivalent à un fichier de "type" : file of char qui, lui, ne possède pas une structure de lignes.

Déclaration d'un fichier de texte :

f_text : text;

text est un fichier de type texte, mot réservé en Pascal, file of n'est pas nécessaire avec les fichiers textes.

Indicateur de fin de ligne dans le fichier :

Eoln (< File >); {Valeur booléenne, True / False }

Opérations d'ouverture / fermeture :

read (< File >, < variable > {, < variable > });

{lit une chaîne dans le fichier < File > }

readln (< File >, < variable > {, < variable > });

{ même opération et lit le caractère /eoln }

write (< File >, < variable > {, < variable > });

{ écrit une chaîne dans le fichier < File > }

writeln (< File >, < variable > {, < variable > });

{ même opération et écrit le caractère /eoln }

page (< File>); { écrit une marque de saut de page dans le fichier à ne pas utiliser avec output }

- **Lecture d'un fichier texte, 2 options**

  Un fichier texte (étant une suite séquentielle de caractères) peut se lire caractère par caractère, ou ligne par ligne.Le choix dépendra du programme à réaliser, généralement on lit ligne par ligne.

- **Partie déclarative des deux programmes**

```
exemple : text; {fichier texte }
caractere : char; {caractère simple }
chaine_caracteres : string; { chaîne de caractères }
```

Version lecture caractère par caractère

```
while not eof ( exemple ) do
begin
while not eoln ( exemple ) do
begin
read ( exemple,caractere ); { traiter le caractère lu }
end;
readln ( exemple );
End;
```

Version lecture ligne par ligne

```
While not eof ( exemple ) do
begin
readln ( exemple, chaine_caracteres ); { traiter la chaîne lue}
End;
```

Exemple sur les fichiers de texte : copie de fichier

Copie de fichier texte avec lecture par caractère

```
var f_entree,f_sortie : text;
ch : char;
begin
assign (f_entree,'In.dat');{ assignations}
assign (f_sortie,'Out.dat');
reset (f_entree); {fichier ouvert en lecture }
rewrite (f_sortie); { fichier ouvert en écriture }
while not eof (f_entree) do { boucle sur le fichier }
begin
while not eoln (f_entree) do {boucle sur la ligne }
begin
read (f_entree,ch); { lecture d'un caractère }
write (f_sortie,ch); { écriture d'un caractère }
end;
readln (f_entree); {passage à la ligne, entrée}
writeln (f_sortie); { passage à la ligne, sortie }
end;
close (f_entree);{ fermeture}
close (f_sortie); {fermeture des 2 fichiers}
end.
```

Exemples:

Exemple 1 : Ouverture et fermeture des fichiers

```
ASSIGN(F1,'CLIENT.DAT') ;
REWRITE(F1) ; {création et ouverture du fichier F1 en écriture}
… … …
{ initialisation du fichier avec READ/WRITE}
```

```
… … …
CLOSE(F1) ; {Fermeture du fichier}
RESET(F1); {Ouverture du fichier en lecture/écriture}
… … …
{Manipulation du fichier READ,WRITE,SEEK etc… }
CLOSE(F1) ; {fermeture du fichier}
```

Exemple 2 : Notes des TPs des étudiants

```
PROGRAM RESULTATS ;
TYPE INDIVIDU = RECORD
Nom : string[20] ;
Prenom : string[20] ;
Tp1 : real ;
Tp2 : real ;
End ;
VAR ETUDIANT : array[1..100] OF INDIVIDU ;
Moy : real ;
I,N :integer ;
Fin : Boolean ;
Fichier1 : FILE OF INDIVIDU ;
BEGIN
ASSIGN(Fichier1,'D:\eleves.PAS') ;
Rewrite(Fichier1) ;
Writeln('SAISIE DES DONNES DANS Fichier1') ;
Fin := False ; I :=1 ;
WHILE NOT Fin DO
BEGIN
Writeln('Appuyer sur ENTREE Pour Terminer la saisie') ;
Write('LE NOM : ') ;READLN(ETUDIANT[I].Nom) ;
If (ETUDIANT[I].Nom = '') then
goto 1 ;
Write('LE PRENOM : ') ;READLN(ETUDIANT[I].Prenom) ;
Write('NOTE DE TP1 : ') ;READLN(ETUDIANT[I].Tp1) ;
Write('NOTE DE TP2 : ') ;READLN(ETUDIANT[I].Tp2) ;
Write(Fichier1,ETUDIANT[I]) ;
I :=I+1 ;
End ;
```

```
1 : Close(Fichier1) ;
writeln('LECTURE DU FICHIER ET CALCULE DE LA MOYENNE DES TP') ;
{ les infos dont on a besoin se trouvent déjà dans le tableau ETUDIANT}
RESET(Fichier1) ;
I :=1 ;
WHILE NOT EOF(Fichier1) DO
BEGIN
READ(Fichier1,ETUDIANT[I]) ; I :=I+1 ;
End ;
Writeln('AFFICHAGE DES RESULTATS : APPUYER SUR UNE TOUCHE') ;
Readln;Readln;
N := I1 ;
FOR I := 1 TO N DO
BEGIN
MOY := (ETUDIANT[I].Tp1 + ETUDIANT[I].Tp2)/2 ;
write(ETUDIANT[I].Nom) ; write(ETUDIANT[I].Prenom) ;
write(ETUDIANT[I].Tp1:6 :2); {avec 2 chiffres après la virgule}
write(ETUDIANT[I].Tp2:6 :2) ; write(MOY:6 :2) ; writeln;
End ;
CLOSE(Fichier1) ;
Readln ;Readln ;
END.
```

Exemple 3 : Nombres complexes

Soit un tableau T contenant N nombres complexes, écrire un programme qui conserve le tableau dans un fichier ‘complex.pas’ et recherche l’élément du tableau le plus grand en module.

```
PROGRAM maximum;
USES WINCRT;
TYPE complex = RECORD
X: real ; Y: real
End ;
VAR T: array [1..100] OF complex ;
I,N,Pos :integer ;
max :real;
F1 : FILE OF complex ;
```

```
BEGIN
ASSIGN(F1,'D:\complex.PAS') ; Rewrite(F1) ;
Write('INTRODUIRE N ') ; READ(N) ;
Writeln('SAISIE DES DONNES DANS F1') ;
For I :=1 to N do
BEGIN
Write('Partie reelle : ') ;read(T[I].X) ;
Write('Partie imaginaire : ') ;read(T[I].Y) ;
Write(F1,T[I]) ;
End ;
Writeln('LECTURE DU FICHIER ') ;
RESET(F1) ;
I :=1 ;
WHILE NOT EOF(F1) DO
BEGIN
READ(F1,T[I]) ;
I :=I+1 ;
End ;
Writeln('AFFICHAGE DES RESULTATS : APPUYER SUR UNE TOUCHE') ;
Readln;Readln;
N:=I1;
{Recherche du plus grand en module }
Max := SQRT(SQR(T[1].X)+SQR(T[1].Y));
FOR I := 2 TO N DO
BEGIN
If Max < SQRT(SQR(T[I].X)+SQR(T[I].Y))Then
Begin
Max := SQRT(SQR(T[I].X)+SQR(T[I].Y));
Pos := I ;
End ; End;
Write(' Le PLUS GRAND EN MODULE EST : ');
writeln(T[Pos].X,' ', T[Pos].Y) ;
write(' Sa POSITION EST : ', Pos) ;
CLOSE(F1) ;
Readln ;Readln ;
END.
```

3. Exercices Corrigés

3.1. Exercices

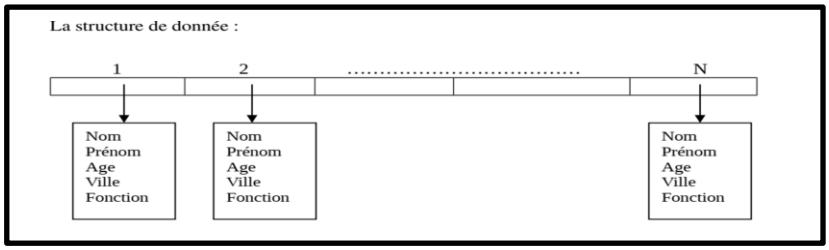
Exercice 01 :

Ecrire un programme qui demande les coordonnées d'un point à l'utilisateur et qui calcule et affiche la distance entre deux points A (x,y) et B(x,y).

On rappelle que la distance AB entre deux points $A(x_A, y_A)$ et $B(x_B, y_B)$ est égale à : $$AB=\sqrt{(x_B-x_A)^2+(y_B-y_A)^2}$$

Exercice 02 :

Considérons une base de données comportant des individus décrits par les informations suivantes :



- Nom
- Prénom
- Age
- Ville
- Fonction

On veut organiser ces informations de telle sorte qu'on puisse sélectionner les individus ayant un certain âge, ou habitant une certaine ville ou ayant un fonction donnée etc… De façon générale gérer la base de données. Proposer une structure de donnée.

On propose: T : array[1..100] of Individu;

Exercice 03

On veut définir le type etudiant contenant les informations d'un étudiant

– le prénom,
– le nom,
– l'âge.

On supposera que les noms et les prénoms ne comportent pas plus de 31 caractères et que les âges sont compris entre 1 et 120 ans.

-Déclarez le type etudiant

-Écrire une procédure SaisirEtudiant qui prend en paramétre un etudiant et qui affecte les trois champs de cet etudiant avec des valeurs saisies au clavier par l'utilisateur.

-Écrire une procédure AfficherEtudiant qui prend en paramétre un etudiant et qui affiche ses trois champs comme dans l'exemple suivant :

Nom = belabass; Prénom = yahia; Âge = 66 ans

-Testez vos procédures dans votre programme principal.

-On propose de faire des groupes d'étudiants (disons des groupes de 5 étudiants) qui s'entendent bien afin qu'ils travaillent ensemble de façon privilégiée. Déclarez le type groupe d'un tableau d'etudiant et écrivez deux procédures, l'une pour affecter les étudiants d'un groupe et l'autre pour les afficher. Testez vos procédures dans votre programme principal.

-Deux prédicats sur les groupes d'étudiants De façon informelle, un prédicat est une fonction qui permet de tester si un article ou une collection d'articles vérifie une propriété donnée. Ici par exemple, on vous demande un prédicat booléen (i.e. la fonction répond vrai si la propriété est vérifiée et faux sinon) permettant à l'utilisateur de tester s'il y a un étudiant d'un âge de son choix dans un groupe. De même, faites un prédicat permettant à l'utilisateur de tester s'il y a au moins une occurrence d'un étudiant du nom de son choix dans un groupe. Vous devez tester vos deux prédicats.

Exercice 04

Ecrire un programme Pascal qui permet de :

- Créer un fichier contenant les informations suivantes :
  - numéro d'identification (entier)
  - nom (chaîne de 20 caractères)
  - moyenne (réel).
- Visualiser la liste des étudiants.
- Visualiser un étudiant par numéro d'identification.

- Visualiser la liste des étudiants par ordre décroissant des moyennes obtenues.

Exercice 05

Ecrire un programme Pascal permettant de stocker la liste des étudiants du fichier **etudiant** dans un deuxième fichier **etudiant2**, après avoir trié les étudiants par ordre croissant de leurs moyennes.

Exercice 06

Ecrire un programme Pascal permettant d'insérer un étudiant dans le fichier **etudiant2** , tout en gardant l'ordre croissant des moyennes obtenues.

Exercice 07

Ecrire un programme Pascal permettant de supprimer ou de modifier les informations d'un étudiant dont le numéro est lu à partir du clavier : Il s'agit donc de la modification ou la suppression d'un enregistrement du fichier etudiant créé précédemment. Si le numéro saisi n'existe pas, on doit le signaler.

Exercice 08

Ecrire un programme Pascal permettant de copier un fichier texte **texte.txt** dans un deuxième fichier **essai.txt**, ensuite de fusionner ces deux fichiers dans un troisième nommé texteglob.txt.

3.2. Corrigées

Exercice 01

```
program distance;
TYPE point = record
        x : real;
        y : real;
end;
VAR   A: point;  B: point; dist : real;
begin
Writeln('Entrer les coordonnées du point A');
Readln(A.x);
Readln(A.y);
Writeln('Entrer les coordonnées du point B');
```

```
Readln(B.x);
Readln(B.y);
dist:=sqrt(sqr(B.x-A.x)+sqr(B.y-A.y));
writeln('La distance entre A et B : ',dist);
end.
```

Exercice 2 :

```
PROGRAM LireEcrire;
Type Individu = RECORD
Nom, Prénom : string[20] ;
Age : integer ;
Ville, Fonction : string[25] ;
End;
Var T : array[1..1000] of Individu;
I,N : integer ;
BEGIN
Writeln('Introduire le nombre d'' individus : ') ;
readln (N) { N est le nombre d'individus }
Writeln('Introduire Nom,Prénom,Age,Ville, Fonction de chaque Individu ') ;
For I := 1 to N do
begin
Readln (T[I].Nom);
Readln(T[I].Prenom);
Readln(T[I].Age);
Readln(T[I].Ville);
Readln(T[I].Fonction);
End;
For I := 1 to N do
begin
Write(T[I].Nom);
Write(T[I].Prenom);
Write(T[I].Age);
Write(T[I].Ville);
Write(T[I].Fonction);
Writeln;
end;
END.
```

Exercice 3 :

```
program étudiants;
const
AgeMin = 1; AgeMax = 120;
taille = 31; nbAmis = 5;
{déclaration du type d'enregistrement pour un étudiant}
```

```
type etudiant = record
nom : string[taille]; {champ pour le nom de l'étudiant}
prenom : string[taille]; {champ pour le prénom d'un étudiant}
age : AgeMin..AgeMax; {champ pour l'âge d'un étudiant}
end;
{déclaration du type d'un groupe qui est un tableau de 5 étudiants}
Type groupe = array[1..nbAmis] of etudiant;
var copains : groupe;
temp : etudiant;
cible : 1..120;
find : boolean;
{procédure pour la saisie d'un étudiant}
procedure SaisieEtudiant(var Ami : etudiant);
begin
Write( ' Saisie du nom : '); ReadLn(Ami.nom);
Write( ' Saisie du prénom : '); ReadLn(Ami.prenom);
Write( ' Saisie de l ' ' âge : '); ReadLn(Ami.age)
end;
{procédure pour l'affichage d'un étudiant}
procedure AfficheEtudiant(Ami : etudiant);
begin
Write( ' Prénom = ');
Write(Ami.prenom);
Write( '  Nom = ');
Write(Ami.nom);
Write( ' Âge = ');
WriteLn(Ami.age)
end;
{procédure pour la saisie d'un groupe}
procedure SaisieGroupe(var mesAmis : groupe);
var i : 1..nbAmis;
begin
WriteLn( ' Saisie d ' un groupe : ');
```

```
for i:=1 to nbAmis do
 begin
Write( ' Saisie de  l'étudiant  numéro ');
Write(i); WriteLn( ' : ');
SaisieEtudiant(mesAmis[i])
End;
end;
{procédure pour l'affichage d'un groupe}
procedure AfficheGroupe(mesAmis : groupe);
var i : 1..nbAmis;
begin
WriteLn( ' affiche  d ' un groupe : ');
for i:=1 to nbAmis do
AfficheEtudiant(mesAmis[i])
end;
{fonction pour tester si il y a un étudiant d'un age donné dans le groupe}
function TestParAge(mesAmis : groupe; ageCible : integer) : boolean;
var i : 1..nbAmis;
resultat : boolean;
begin
resultat := false;
for i:= 1 to nbAmis do
 begin
if mesAmis[i].age = ageCible then
resultat := true
end;
TestParAge :=resultat;
end; { TestParAge }
begin
temp.nom := ' hawam ';
temp.prenom := ' ahmed ';
temp.age := 22;
SaisieEtudiant(temp);
AfficheEtudiant(temp);
SaisieGroupe(copains);
```

```
AfficheGroupe(copains);
Write( ' saisir l' âge recherché : ');
ReadLn(cible);
find := TestParAge(copains, cible);
if (find) then
WriteLn( ' I l y a au moins un étudiant ayant l' âge requis . ')
else
WriteLn( ' I l n ' ' y a pas d ' étudiant ayant l ' âge requis . ')
end.
```

Exercice 04

```
program scolarite ;
function menu : integer; var
choix : integer ;
begin
writeln('1. Création du fichier.');
writeln('2. Visualiser la liste des étudiants.'); writeln('3. Visualiser un étudiant par
numéro.');
writeln('4. Visualiser la liste par ordre décroissant des moyennes.'); writeln('5.
Quitter l''application.');
writeln('Entrez votre choix :'); readln(choix);
menu := choix; end;
type etudiant = record
num_id : integer ;
 nom : string[20] ;
moyenne : real ;
end;
var etud : file of etudiant ; e : etudiant ;
reponse : char ;
choix, num, nb, i, j : integer ; etd : array[1..10] of etudiant ;
begin
choix := menu ;
repeat
case choix of
1 : begin { Création du fichier }
assign(etud, 'etudiant');
rewrite(etud);
repeat
writeln('Donnez le numéro de l''étudiant :'); readln(e.num_id);
writeln('Donnez le nom de l''étudiant :'); readln(e.nom);
writeln('Donnez la moyenne de l''étudiant :'); readln(e.moyenne);
write(etud, e); writeln('Autre saisie ? o/n'); readln(reponse);
 until reponse = 'n';
close(etud);
end;
2: begin {Visualiser la liste des étudiants }
assign(etud, 'etudiant');
```

```
reset(etud);
while not eof(etud) do
begin
read(etud, e);
writeln('Numéro = ', e.num_id, ', Nom : ', e.nom, ', Moyenne : ', e.moyenne);
end;
close(etud);
end;
3: begin {Consulter un étudiant par un numéro }
writeln('Donnez le numéro de l''étudiant :'); readln(num);
assign(etud, 'etudiant'); reset(etud);
while not eof(etud) do
 begin
 read(etud, e);
if (e.num_id = num) then
writeln('Numéro = ', e.num_id, ', Nom : ', e.nom, ', Moyenne : ', e.moyenne); end;
close(etud);
 end;
4: begin {Visualiser la liste des étudiants par ordre décroissant des moyennes }
assign(etud, 'etudiant');
reset(etud); nb := 1;
{On stocke les enregistrements du fichier dans un tableau d'enregistrements pour le
tri }
while not eof(etud) do
 begin
 read(etud, etd[nb]);
nb := nb + 1 ;
end;
close(etud);
{Tri du tableau par ordre décroissant des moyennes }
for i := 1 to nb − 2 do
for j := i+1 to nb − 1 do
if etd[j].moyenne > etd[i]. moyenne then
 begin
 e := etd[i]; etd[i] := etd[j]; etd[j] := e ; end;
{Affichage du tableau après tri }
 for i := 1 to nb − 1 do
with etd[i] do writeln(num_id, ' ', nom, ' ', moyenne);
end;
5: begin end
else writeln('Choix inexistant. Recommencer.');
 end;
if choix <> 5 then choix := menu; until choix = 5 ;
end.
```

Exercices Corrigés

Exercice 05

```
program Tri_fichier ;
 type etudiant = record
 num_id : integer ;
 nom : string[20] ;
moyenne : real ;
 end;
var etud : file of etudiant ; e : etudiant ;
nb, i, j : integer ;
etd : array[1..10] of etudiant ;
begin
assign(etud, 'etudiant'); reset(etud);
nb := 1;
{ On stocke les enregistrements du fichier etudiant dans un tableau d'enregistrements
pour le tri }
while not eof(etud) do
begin
read(etud, etd[nb]);
nb := nb + 1 ;
 end; close(etud);
{ Tri du tableau par ordre croissant des moyennes }
 for i := 1 to nb − 2 do for j := i+1 to nb − 1 do
if etd[j].moyenne < etd[i]. moyenne then
 begin
 e := etd[i];
etd[i] := etd[j]; etd[j] := e ;
 end;
{Stockage du tableau dans le fichier etudiant2 }
assign(etud, 'etudiant2');
rewrite(etud);
for i := 1 to nb − 1 do write(etud, etd[i]); close(etud);
{Visualiser la liste des étudiants à partir du fichier etudiant2 }
 writeln('Liste des étudiants dans le fichier etudiant2 :'); assign(etud, 'etudiant2');
reset(etud);
while not eof(etud) do
begin
read(etud, e);
writeln('Numéro = ', e.num_id, ', Nom : ', e.nom, ', Moyenne : ', e.moyenne);
end; close(etud);
end.
```

Exercice 06

```
program Insertion_fichier ;
 type etudiant = record
num_id : integer ;
nom : string[20] ;
 moyenne : real ;
end;
var etud : file of etudiant ; e : etudiant ;
nb, i, j : integer ;
```

```
etd : array[1..10] of etudiant ; b : boolean ;
begin
{Visualiser la liste des étudiants à partir du fichier etudiant2 avant l'insertion }
writeln('Liste des étudiants dans le fichier etudiant2 avant insertion :');
assign(etud, 'etudiant2'); reset(etud);
while not eof(etud) do
begin
 read(etud, e);
writeln('Numéro = ', e.num_id, ', Nom : ', e.nom, ', Moyenne : ', e.moyenne);
end;
close(etud);
{On stocke les enregistrements du fichier etudiant2 dans un tableau
d'enregistrements }
assign(etud, 'etudiant2'); reset(etud);
nb := 1;
while not eof(etud) do
begin
 read(etud, etd[nb]);
nb := nb + 1 ;
end; close(etud);
{Lire les informations de l'étudiant à inserer }
writeln('Saisir l''étudiant à inserer :'); writeln('Donnez le numéro de l''étudiant :');
readln(e.num_id);
writeln('Donnez le nom de l''étudiant :'); readln(e.nom);
writeln('Donnez la moyenne de l''étudiant :'); readln(e.moyenne);
{Insertion de l'étudiant dans le tableau } i := 1; b := false;
while (i <= nb − 1) and NOT b do
begin
if (e.moyenne < etd[i].moyenne) then b := true ; if NOT b then i := i + 1 ;
end;
for j := nb downto i+1 do etd[j] := etd[j−1]; etd[i] := e;
{ Stockage du tableau dans le fichier etudiant2 }
assign(etud, 'etudiant2');
rewrite(etud);
for i := 1 to nb do write(etud, etd[i]); close(etud);
{Visualiser la liste des étudiants à partir du fichier etudiant2 après l'insertion}
writeln('Liste des étudiants dans le fichier etudiant2 après l''insertion :');
assign(etud, 'etudiant2'); reset(etud);
while not eof(etud) do
 begin
 read(etud, e);
writeln('Numéro = ', e.num_id, ', Nom : ', e.nom, ', Moyenne : ', e.moyenne);
end; close(etud); end.
```

Exercice 07

```
program scolarite ;

function menu : integer;

var choix : integer ;
```

```
begin
writeln('1. Visualiser la liste des étudiants.');
writeln('2. Modifier un enregistrement.');
writeln('3. Supprimer un enregistrement.');
writeln('4. Quitter l''application.'); writeln('Entrez votre choix :');
readln(choix); menu := choix;
 end;
type etudiant = record
num_id : integer ;
nom : string[20] ;
moyenne : real ;
end;
var etud : file of etudiant ; e : etudiant ;
reponse : char ;
choix, num, nb, i, j : integer ; etd : array[1..20] of etudiant ; exist : boolean;
begin
choix := menu ;
repeat
case choix of
1: begin {Visualiser la liste des étudiants }
 assign(etud, 'etudiant');
reset(etud);
while not eof(etud) do begin read(etud, e);
writeln('Numéro = ', e.num_id, ', Nom : ', e.nom, ', Moyenne : ', e.moyenne);
end;
close(etud);
end;
2: begin {Modification d'un enregistrement }
assign(etud, 'etudiant');
reset(etud); nb := 1;
{On stocke les enregistrements du fichier dans un tableau d'enregistrements pour le
tri }
while not eof(etud) do
 begin
 read(etud, etd[nb]);
nb := nb + 1 ; end; close(etud);
writeln('Donnez le numéro de l''étudiant à modifier :'); readln(num);
i := 1;
exist := false;
while (i <= nb−1) and not exist do
begin
if etd[i].num_id = num then exist := true ; if not exist then i := i + 1 ;
end;
if exist then
begin
writeln('Saisir les nouvelles informations :'); writeln('Donnez le numéro de l''étudiant
:'); readln(e.num_id);
writeln('Donnez le nom de l''étudiant :'); readln(e.nom);
writeln('Donnez la moyenne de l''étudiant :'); readln(e.moyenne);
etd[i] := e;
{Stockage du tableau dans le fichier etudiant }
assign(etud, 'etudiant');
```

```
rewrite(etud);
for i := 1 to nb−1 do write(etud, etd[i]); close(etud);
end
else writeln('Il n"existe pas un étudiant avec le num : ', num);
end;
3: begin { Suppression d'un enregistrement }
 assign(etud, 'etudiant');
reset(etud); nb := 1;
{ On stocke les enregistrements du fichier dans un tableau d'enregistrements pour le
tri }
while not eof(etud) do
 begin
 read(etud, etd[nb]);
nb := nb + 1 ;
end; close(etud);
writeln('Donnez le numéro de l"étudiant à supprimer :'); readln(num);
i := 1;
exist := false;
while (i <= nb−1) and not exist do
 begin
if etd[i].num_id = num then exist := true ; if not exist then i := i + 1 ;
end;
if exist then
begin
for j := i to nb−2 do etd[j] := etd[j+1];
{Stockage du tableau dans le fichier etudiant }
 assign(etud, 'etudiant');
rewrite(etud);
for i := 1 to nb−2 do write(etud, etd[i]); close(etud);
end
else writeln('Il n"existe pas un étudiant avec le num : ', num);
end;
4: begin end
else writeln('Choix inexistant. Recommencer.');
 end;
if choix <> 4 then choix := menu; until choix = 4 ;
end.
```

Exercice 08

```
program concat_copier_fichier ;

var fichier1, fichier2, fichier3 : text ; ligne : string[80] ;
begin
assign(fichier1, 'texte.txt'); M$I−}
reset(fichier1);
if (ioresult = 0) then
 begin
{Copier le fichier texte.txt dans essai.txt }
assign(fichier2, 'essai.txt'); rewrite(fichier2);
while not eof(fichier1) do
 begin readln(fichier1, ligne); writeln(fichier2, ligne);
end; close(fichier1); close(fichier2);
```

```
{Concaténer texte.txt et essai.txt dans texteglob.txt }
assign(fichier1, 'texte.txt');
reset(fichier1); assign(fichier2, 'essai.txt');
reset(fichier2);
assign(fichier3, 'texteglob.txt'); rewrite(fichier3);
while not eof(fichier1) do
 begin readln(fichier1, ligne); writeln(fichier3, ligne);
end; close(fichier1);
while not eof(fichier2) do
begin readln(fichier2, ligne); writeln(fichier3, ligne);
end; close(fichier2); close(fichier3);
writeln('Le contenu du fichier texte.txt :'); assign(fichier1, 'texte.txt');
reset(fichier1);
while not eof(fichier1) do
 begin readln(fichier1, ligne); writeln(ligne);
end; close(fichier1);
writeln('Le contenu du fichier essai.txt :'); assign(fichier2,   'essai.txt');
reset(fichier2);
while not eof(fichier2) do
 begin readln(fichier2, ligne); writeln(ligne);
end; close(fichier2);
writeln('Le contenu du fichier texteglob.txt :'); assign(fichier3,   'texteglob.txt');
reset(fichier3);
while not eof(fichier3) do begin readln(fichier3, ligne); writeln(ligne);
end; close(fichier2);
 end
else writeln('Fichier texte.txt inexistant.');
end.
```

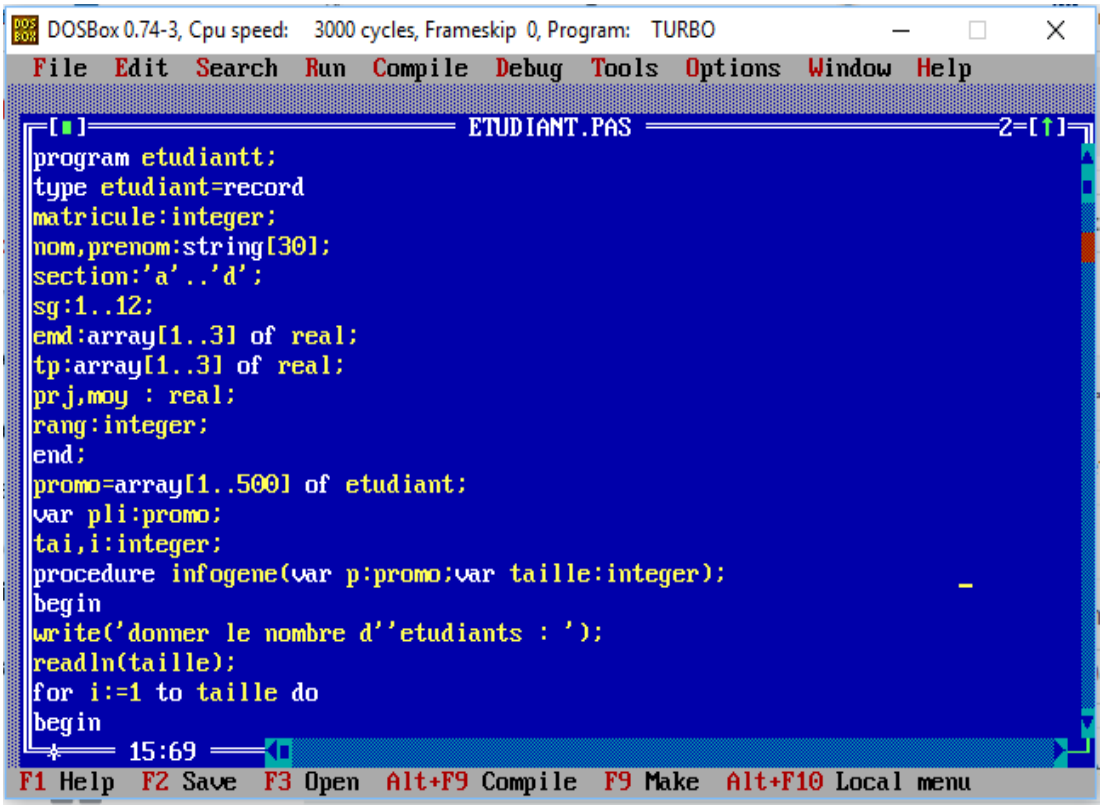
4. Travaux Pratiques

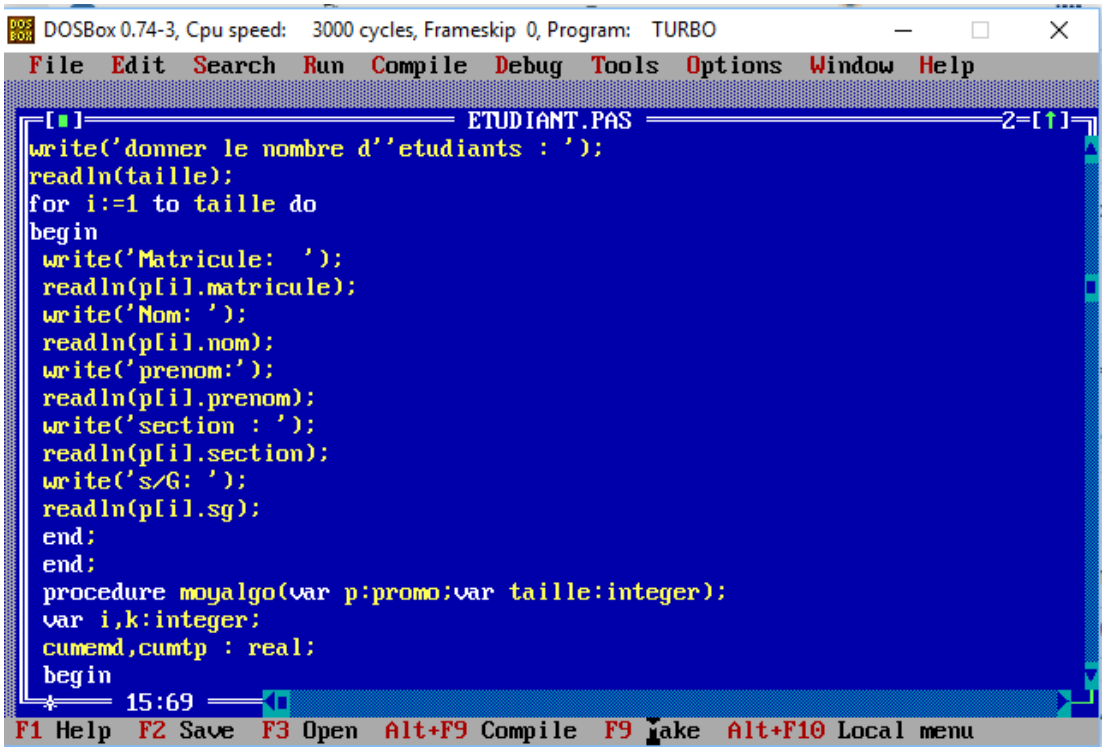
Tp:

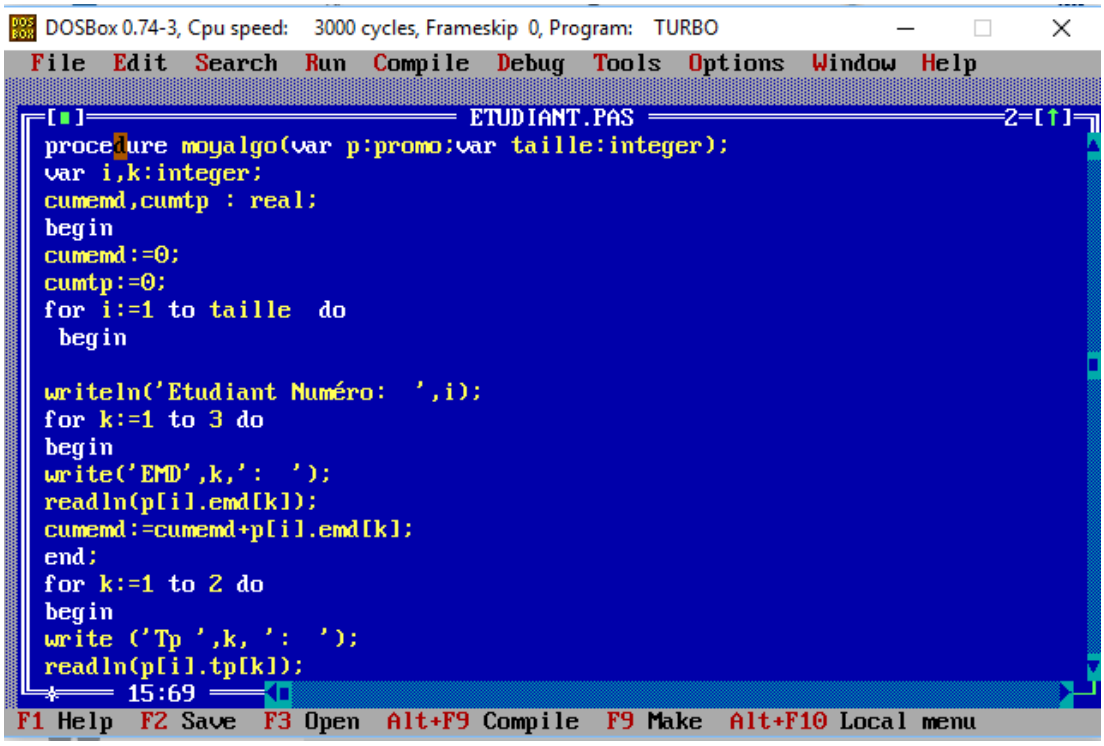
Nous avons au plus 500 étudiants et pour chaque étudiant nous voulons avoir les informations suivantes : Le matricule, le nom et prénom, la section, le groupe, les notes de ses 3 Emd, de ses 3 tp, de son projet, sa moyenne et son rang.

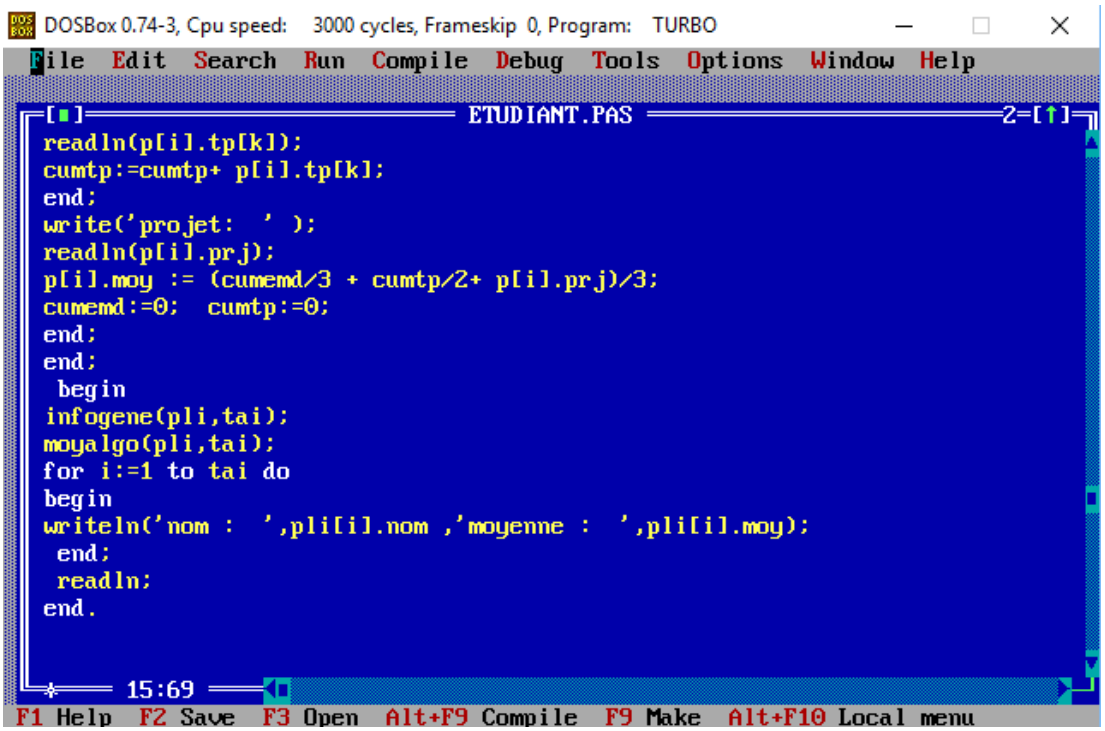
Il faut donc déclarer la structure convenable pour ce problème. Puis construire 2 modules : le premier (INFOGENE) qui permet de rentrer toutes les informations concernant un étudiant et toutes ses notes en algorithmique et le second (MOYALGO) qui calcule sa moyenne.

Solution: le programme :

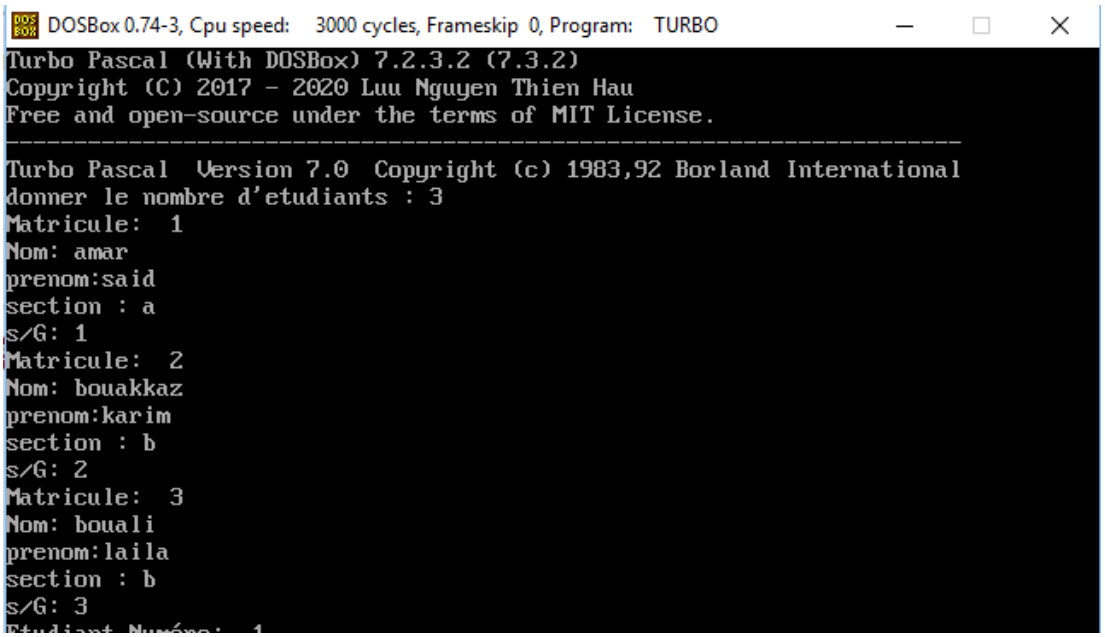


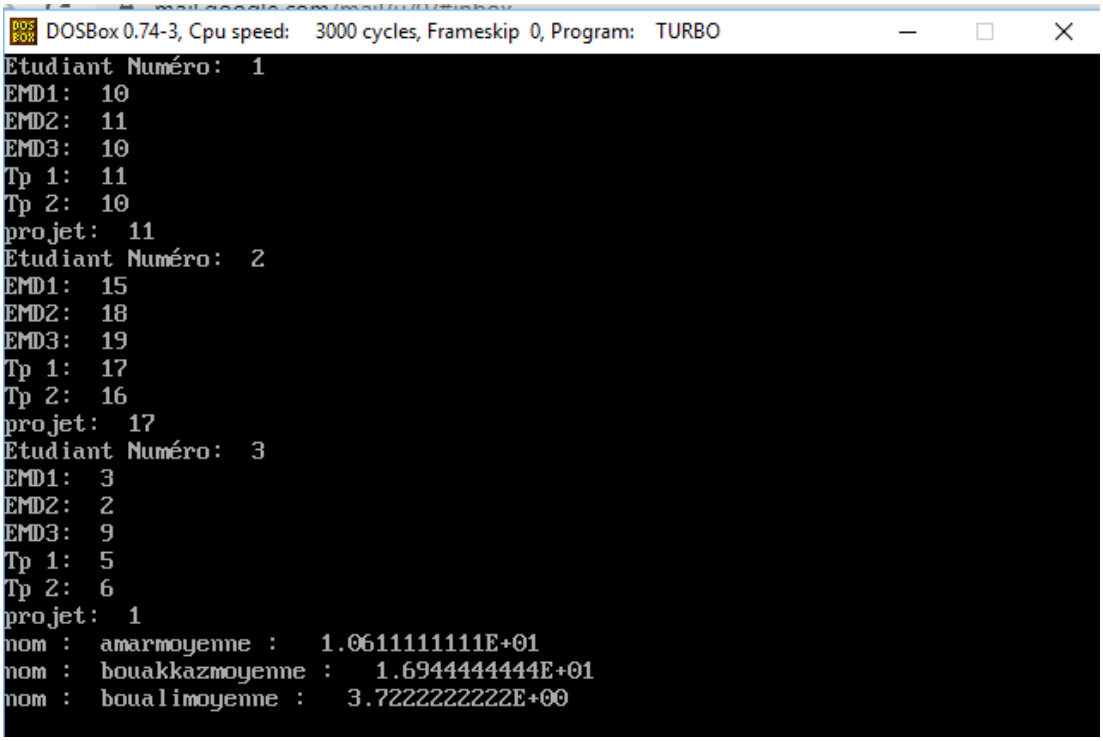

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: TURBO
File Edit Search Run Compile Debug Tools Options Window Help
ETUDIANT.PAS
write('donner le nombre d''etudiants : ');
readln(taille);
for i:=1 to taille do
begin
write('Matricule: ');
readln(p[i].matricule);
write('Nom: ');
readln(p[i].nom);
write('prenom:');
readln(p[i].prenom);
write('section : ');
readln(p[i].section);
write('s/G: ');
readln(p[i].sg);
end;
end;
procedure moyalgo(var p:promo;var taille:integer);
var i,k:integer;
cumemd,cumtp : real;
begin
15:69
F1 Help F2 Save F3 Open Alt+F9 Compile F9 Make Alt+F10 Local menu


DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: TURBO
File Edit Search Run Compile Debug Tools Options Window Help
ETUDIANT.PAS
procedure moyalgo(var p:promo;var taille:integer);
var i,k:integer;
cumemd,cumtp : real;
begin
cumemd:=0;
cumtp:=0;
for i:=1 to taille do
begin
writeln('Etudiant Numéro: ',i);
for k:=1 to 3 do
begin
write('EMD',k,': ');
readln(p[i].emd[k]);
cumemd:=cumemd+p[i].emd[k];
end;
for k:=1 to 2 do
begin
write ('Tp ',k, ': ');
readln(p[i].tp[k]);
15:69
F1 Help F2 Save F3 Open Alt+F9 Compile F9 Make Alt+F10 Local menu



L'exécution du programme :




DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: TURBO
Etudiant Numéro: 1
EMD1: 10
EMD2: 11
EMD3: 10
Tp 1: 11
Tp 2: 10
projet: 11
Etudiant Numéro: 2
EMD1: 15
EMD2: 18
EMD3: 19
Tp 1: 17
Tp 2: 16
projet: 17
Etudiant Numéro: 3
EMD1: 3
EMD2: 2
EMD3: 9
Tp 1: 5
Tp 2: 6
projet: 1
nom : amarmoyenne : 1.0611111111E+01
nom : bouakkazmoyenne : 1.6944444444E+01
nom : boualimoyenne : 3.7222222222E+00

Références

[1] Jean MAYSONNAVE, « Introduction à l'algorithmique générale et numérique - DEUG Sciences : Résumés de cours », Edition Masson, 1996.

[2] Ives GRANJON, « Tavaux dirigés - Informatique : Algorithmique en Pascal et en langage C - DEUG Sciences : Résumés de cours », Edition Dunod, 1999.

[3] Salah FENNI, « Exercices en Turbo Pascal », Edition Chebba, 2000.

[4] Olivier LECARME, « Pascal : Langages d'écriture de systèmes », Techniques de l'Ingénieur, traité Informatique, H 2260.

[5] Adeline CREPIEUX, « Introduction à l'informatique et à la programmation », Cours de DEUG U1, Université de la méditerranée, 2002.

[6] Hugo ETIEVANT, « Cours de Turbo Pascal 7.0 : Le cours aux 100 exemples », 2004.

[7] Philipe TRIGANO, Dominique LENNE, « Algorithmique et programmation », Université de Technologie de Compiègne, 2008.

[8] Christophe DARMANGEAT, « Algorithmique et programmation pour nonmatheux : Cours complet », Université Paris 7, 2008.