

2016

PROGRAMMER AVEC ZAYDOUN

Algorithmique et programmation

Ce livre est destiné aux élèves du 4eme année de l'enseignement
secondaire SECTION SCIENCES DE L'INFORMATIQUE

Ahmed Amine Rassas
Etudiant

Rassas Amine



PREFACE



Zied Chaouch (mort Janvier 5, 2016) est un professeur universitaire d'informatique. Il était très connu dans la ville de Sousse comme professeur d'algorithmique de tutorat pour les élèves des écoles secondaires.

Cette œuvre a pour but d'assembler les travaux du défunt monsieur Zied Chouch, mon cher professeur d'Informatique durant sa carrière d'enseignement en vue de la documentation de ses chef d'œuvres . Plusieurs générations ont profité du savoir qu'il ne cessait de répandre là où il passait. L'amour et le respect ont imprégné les rapports que les apprenants tenaient avec cet terminent savant . Son sérieux et son attachement à son travail étaient non seulement à l'origine du succès de tous ses disciples mais aussi du respect et de reconnaissance de ses collègues.

D'ailleurs , un bon nombre de ses élèves ont poursuivi leurs études universitaires en informatique avec excellence. Humainement il était très proche de ses élèves, allant jusqu'à entretenir des relations amicales avec eux. Il était d'une bonté sans égale et d'une sincérité remarquable.

Pour moi, c'était un père avec ses conseils , un ami avec son humilité , un frère avec son soutien . J'ai tant essayé de lui adresser cette reconnaissance quand il était avec nous , malheureusement , la mort me l'a arraché. tu resteras toujours vivant dans mon cœur et dans ma mémoire. Que votre âme repose en paix.

#ZAYDOUN L'homme qui a passionné toute une génération par l'informatique

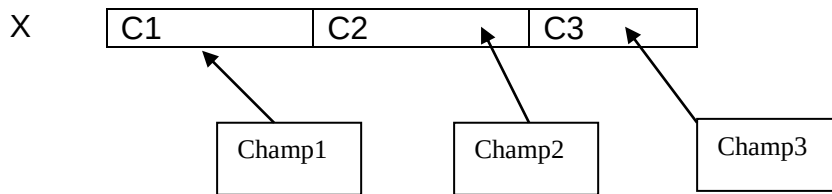
SOMMAIRE

Chapitre 1 : Les enregistrements et les fichiers	3
Chapitre 2 : La récursivité	52
Chapitre 3 : Les algorithmes arithmétiques	67
Chapitre 4 : Les algorithmes de tri & de recherche	78
Chapitre 5 : Les algorithmes récurrents	89
Chapitre 6 : Les algorithmes approximation	105

Chapitre 1 : Les enregistrements et les fichiers

I. Les enregistrements

1) Enregistrement simple :



TDNT

Type
Nom_type = enregsitrement C1 : type C2 :type C3 :type Fin nom_type

Pascal :

Type
Nom_type = record
 C1 :type ;
 C2 :type ;
 C3 :type ;

End ;

Exercice n°1

Déclarer un nouveau type nommé **groupe**, sachant qu'un groupe est caractérisé par un code, un nom, nombre de personne dans le groupe.

TDNT

Type
groupe = enregsitrement cod : entier nom :chaîne [20] nbr : entier Fin groupe

Pascal :

Type
groupe = record
 cod : integer;
 nom : string[20] ;
 nbr : integer;

End ;

Exercice n°2

Déclarer un nouveau type nommé élève , sachant qu'un élève est caractérisé par un CIN, un nom, prénom , âge.

TDNT

Type
eleve = enregistrement CIN, age : entier Nom, prenom : chaîne [20] Fin eleve

Exercice n°3 :

Ecrire une analyse et un programme pascal permettant de saisir deux produit A et B, sachant qu'un produit est caractérisé par un code, libellé et un prix. Afficher le produit le plus cher.

Analyse principale :

Résultat = proc affich(A, B)
 A = proc saisi(A)
 B = proc saisi(B)
 Fin

TDNT

Type
Produit = enregistrement Cod : entier Lib : chaîne Prix : réel Fin produit

TDOG

Nom	Type
A, B	Produit
Saisi	Procédure
Affich	Procédure

Analyse de la procédure saisie

Def proc saisi (var X : produit)
 Résultat = X
 x.Cod=donnée
 x.Lib = donnée
 x.Prix = donnée
 Fin proc saisi

Analyse de la procédure saisie (2ème méthode le plus utilisé) : on utilise avec qu'avec saisi et l'affichage.

Def proc saisi (var X : produit)
 Résultat = X
 Avec X faire
 Cod=donnée
 Lib = donnée
 Prix = donnée
 Fin avec
 Fin proc saisi

Analyse de procédure affich

Def proc affich (A, B :produit)

Résultat = []

si A.prix>B.prix alors écrire (A.cod,A.lib,A.prix)

 sinon si B.prix>B.prix alors écrire (B.cod ,B.lib,B.prix)

 sinon écrire("de même prix");

Fin proc affich

Pascal :

program ex;

uses winCRT;

type

produit = record

 cod:longint;

 lib:string;

 prix:real;

end;

var

a, b:produit;

{***** les modules *****}

procedure saisi(var x:produit);

begin

 with x do

 begin

 write('code: '); readln(cod);

 write('libellé: '); readln(lib);

 write('prix: '); readln(prix);

 end;

end;

procedure affich(a, b:produit);

begin

 if a.prix>b.prix then writeln(a.cod,' ',a.lib,' ',a.prix:0:3)

 else if b.prix>a.prix then writeln(b.cod,' ',b.lib,' ',b.prix:0:3)

 else writeln('de même prix');

end;

{***** PP *****}

begin

 saisi(a);

 saisi(b);

 affich(a,b);

end.

Exercice n°2

Ecrire une analyse et un programme pascal permettant de saisir les informations de deux élèves, sachant qu'un élève est caractérisé par son nom, note1, coefficient 1, note2, coefficient2, afficher le nom de l'élève possédant la meilleure moyenne ainsi que sa moyenne.

Analyse principale :

Résultat = proc affich (X, Y)
Y = proc saisi(Y)
X = proc saisi (X)
Fin ex

TDNT

Type
Eleve= enregistrement
Nom :chaîne
n1,c1,n2,c2 :réel
fin eleve

TDOG

Nom	Type
X , Y	Eleve
Saisi	Procédure
Affich	procédure

Analyse de la procédure saisi

Def proc saisi (var A : eleve)
Résultat = A
Avec A faire
 Nom=donnée
 N1=donnée
 C1= donnée
 N2= donnée
 C2= donnée
Fin avec
Fin proc saisi

Analyse de la procédure affich

Def proc affich(X, Y :eleve)
Résultat = []

Moy1 ← ((x.n1*x.c1)+ (x.n2*x.c2))/(x.c1+x.c2)
Moy2 ← ((y.n1*y.c1)+ (y.n2*y.c2))/(y.c1+y.c2)

Si moy1>moy2 alors écrire(x.nom)
Sinon si moy2>moy1 alors écrire(y.nom)
Sinon écrire("même moyenne")
Fin si
Fin proc affich

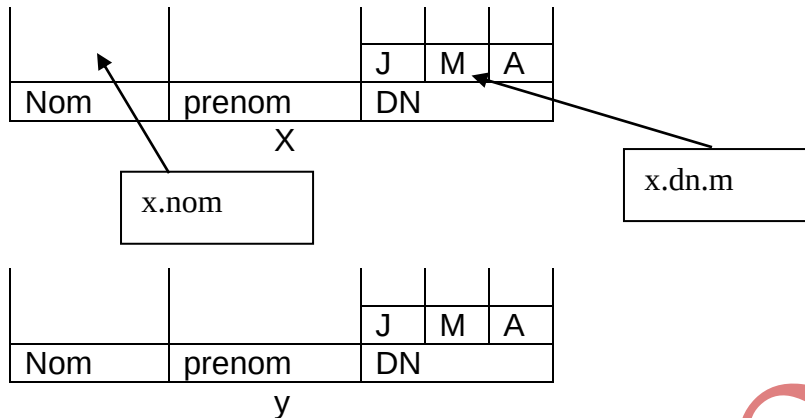
TDOL

Nom	Type
Moy1, moy2	Réel

2) Enregistrement imbriqué

Exercice 2:

Ecrire un programme permettant de saisir deux personnes x et y, sachant qu'une personne est caractérisé par : un nom, un prénom et une date de naissance et afficher la personne la plus jeune.



Analyse principale :

Résultat = proc affich(x, y)

X = proc saisi(x)

Y = proc saisi(y)

Fin ex

TDNT

Type
Date = enregistrement J, m, a : entier Fin date
pers = enregistrement Nom : chaîne Prenom : chaîne Dn : date Fin eleve

TDOG

Nom	Type
X, Y	pers
Saisi	Procédure
Affich	Procédure

Analyse de la procédure saisi

Def proc saisi (var z : eleve)

Résultat = z

Z =

Avec z faire

Nom= donnée
 Prenom=donnée
 Avec dn faire
 J, m, a= donnée
 Fin avec

Fin avec
 Fin proc saisi

Analyse de la procédure affich

Def proc affich (x, y : eleve)

Résultat = []

[]=

Si x.dn.a > y.dn.a alors écrire(x.nom, x.prenom, x.dn.j,"/", x.dn.m,"/", x.dn.a)
 Sinon si x.dn.a < y.dn.a alors écrire(y.nom, y.prenom, y.dn.j,"/", y.dn.m,"/", y.dn.a)
 Sinon si x.dn.m > y.dn.m alors écrire(x.nom, x.prenom, x.dn.j,"/", x.dn.m,"/", x.dn.a)
 Sinon si x.dn.m < y.dn.m alors écrire(y.nom, y.prenom, y.dn.j,"/", y.dn.m,"/", y.dn.a)
 Sinon si x.dn.j > y.dn.j alors écrire(x.nom, x.prenom, x.dn.j,"/", x.dn.m,"/", x.dn.a)
 Sinon si x.dn.j < y.dn.j alors écrire(y.nom, y.prenom, y.dn.j,"/", y.dn.m,"/", y.dn.a)
 Sinon écrire("même age")
 Fin si
 Fin proc affich

3) Tableau d'enregistrement :

Exercice 1 :

Ecrire un programme permettant de saisir un tableau par n produit avec n compris entre 5 et 20, sachant qu'un produit est caractérisé par un code, date de fabrication, prix, afficher le prix le plus cher.

Analyse principale :

Résultat = proc affich (t, n)
 T, n = proc saisi(t, n)
 Fin ex

TDNT

Type
Date = enregistrement J, m, a :entier Fin date
Prod= enregistrement Cod :entier Prix :réel Df :date Fin prod
Tab = tableau [1..20] de produit

TDOG

Nom	Type
T	Tab
N	Entier

Saisi
Affich

Procédure
Procédure

Analyse de la procédure saisi

```
Def proc saisi (var t :tab ; var n :entier)
Résultat = t, n
T= []
Pour i de 1 à n faire
  Avec t[i] faire
    Code = donnée
    Prix= donnée
    Avec df faire
      J, m, a,= donnée
    Fin avec
  Fin avec
Fin pour

Répéter
  N = donnée
Jusqu'à n dans [5..20]
Fin proc saisi
```

TDOL

Nom	Type
I	Entier

Analyse de la procédure affich

```
Def proc affich(t :tab ; n :entier)
Résultat = [ ]
Max ← t[1].prix
Pour i de 2 à n faire
  Si t[i].prix > max alors max ← t[i].prix
Fin si
Fin pour
TDOL
```

Nom	Type
I	Entier
Max	Prod

Pascal :

```
program ex;
uses winCRT;

type
date= record
  j, m, a:integer;
end;

prod= record
  cod:integer;
  prix:real;
  df:date;
```

```

end;

tab = array[1..20] of prod;

var
t:tab;
n:integer;

{***** les modules *****)}

procedure saisi(var t:tab ; var n:integer);
var
i:integer;
begin

repeat
    writeln('donner n ');
    readln(n);
until n in [5..20];

for i:=1 to n do
begin
    with t[i] do
    begin
        writeln('code: '); readln(cod);
        writeln('prix: '); readln(prix);
        with df do
        begin
            writeln('date de fabrication ');
            readln(j, m, a);
        end;
    end;
end;
end;

procedure affich(t:tab ; n:integer);
var
max:real;
i:integer;
begin
    max:=t[1].prix;

    for i:=2 to n do
    begin
        if t[i].prix>max then max:=t[i].prix;
    end;

    writeln(max:0:3);
end;

{***** PP *****)}

begin
    saisi(t, n);

```

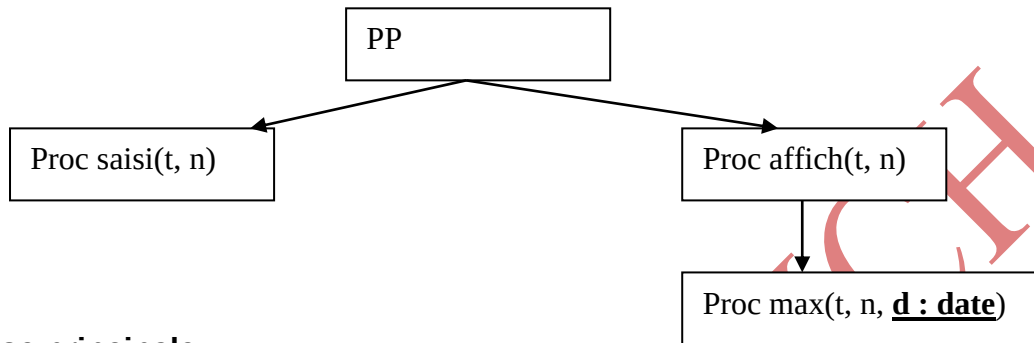
```

    affich(t, n);
end.

```

Exercice 2 :

Ecrire un programme permettant de saisir un tableau contenant les informations de n élèves, avec n compris entre 3 et 30, sachant qu'un élève est caractérisé par un nom, date de naissance (j, m, a) et classe. Afficher l'élèves le plus âgé si il existe plusieurs élèves de même âge que le maximum on doit les afficher.



Analyse principale :

Résultat = proc affich(t, n)
Proc saisi(t, n)

Fin ex

TDNT

Type
date= enregistrement J, m, a :entier Fin date
eleve = enregistrement Nom, clas :chaîne Dn :date Fin eleve
Tab = tableau [1..30] d'eleve

TDOG

Nom	Type
T	Tab
N	Entier
Saisi, Affich	Procédure

Analyse de la procédure saisi

Def proc saisi (var t :tab ; var n :entier)

Résultat = t, n

T= []

Pour i de 1 à n faire

Avec t[i] faire

Nom = donnée

Clas= donnée

Avec dn faire

J, m, a = donnée
 Fin avec
 Fin avec
 Fin pour

Répéter
 N = donnée
 Jusqu'à n dans [3..20]
 Fin proc saisi

Analyse de la procédure affich

```
Def proc affich (t :tab ; n :entier)
Résultat = [ ]
[ ] = proc max(t, n, d)
  Pour i de 1 à n faire
    Si (t[i].dn.a=d.a) et (t[i].dn.m=d.m) et (t[i].dn.j=d.j) alors
      Ecrire(t[i].nom)
    Fin si
  Fin pour
Fin proc affich
```

TDOL

Nom	Type
Max	Procédure
I	Entier
D	date

Analyse de la procédure max

```
Def proc max(t :tab ; n :entier ; var dmax :ty_date)
Résultat = dmax

Dmax ← t[1].dn
Pour i de 2 à n faire
  Si t[i].dn.a < dmax.a alors dmax ← t[i].dn
  Sinon si (t[i].dn.a = dmax.a) et (t[i].dn.m < dmax.m) alors dmax ← t[i].dn
  Sinon si (t[i].dn.a = dmax.a) et (t[i].dn.m = dmax.m) et (t[i].dn.j < dmax.j) alors
    dmax ← t[i].dn
  Fin si
Fin pour
Fin proc max
```

TDOL

Nom	Type
i	Entier

Pascal :

```
program ex;
uses winCRT;

type
ty_date = record
```

```

        j, m, a:integer;
end;

ty_elev=record
        nom, clas:string[20];
        dn:ty_date;
end;

tab=array[1..20] of ty_elev;
var
t:tab;
n:integer;

{***** les modules *****)}

procedure saisi(var t:tab ; var n:integer);
var
i:integer;
begin
    repeat
        writeln('donner la taille du tableau ');
        readln(n);
    until n in [3..20];

    for i:=1 to n do
        begin
            with t[i] do
                begin
                    write('nom: ');readln(nom);
                    write('classe: '); readln(clas);
                    with dn do
                        begin
                            write('date de naissance: ');
                            readln(j, m, a);
                            end;
                        end;
                    end;
                end;
            end;

procedure max(t:tab ; n :integer ; var dmax :ty_date);
var
i:integer;
begin
    Dmax:=t[1].dn;

    for i:=2 to n do
        begin
            if t[i].dn.a<dmax.a then dmax:=t[i].dn
            else if (t[i].dn.a=dmax.a) and (t[i].dn.m<dmax.m) then dmax:=t[i].dn
            else if (t[i].dn.a=dmax.a) and (t[i].dn.m=dmax.m) and (t[i].dn.j<dmax.j) then
                dmax:=t[i].dn;
            end;
        end;
    end;

```

```

procedure affich (t:tab; n:integer);
var
dmax:ty_date;
i:integer;
begin
    max(t, n, dmax);

    for i:=1 to n do
    begin
        if (dmax.a=t[i].dn.a) and (dmax.m=t[i].dn.m) and (dmax.j=t[i].dn.j)
        then writeln(t[i].nom, ' ', t[i].dn.j, '/', t[i].dn.m, '/', t[i].dn.a, ' ', t[i].clas);
        end;
    end;
end;

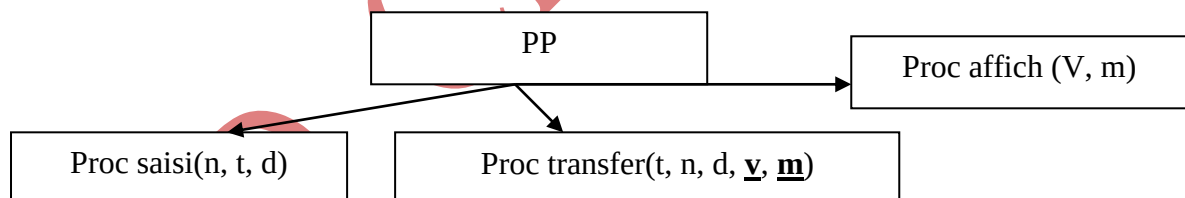
{***** PP *****}

begin
    saisi(t, n);
    affich(t, n);
end.

```

Exercice n°3

Saisir un tableau T par n médicament avec n compris entre 4 et 20, sachant qu'un médicament est caractérisé par un code, libelle et date d'expiration, saisir une date quelconque et transférer dans un tableau V que les médicaments périmés par rapport à la date saisie puis afficher le tableau V.



Analyse principale :

Résultat= proc affich(v, m)
 V, m = proc transfer(t, n, d, v, m)
 T, n, d =proc saisi(t, n, d)

Fin ex

TDNT

Type
Date= enregistrement J, m, a : entier Fin date
Med= enregistrement Cod :entier Lib : chaîne Dexp : date

Fin med

Tab = tableau [1..20] de med

TDO

Nom	Type
T, V	Tab
M, n	Entier
D	Date
Saisi, affich, tranfer	Procédure

Analyse de la procédure saisi

Def proc saisi(var t :tab ; var n :entier ; var d :date)

Résultat = t, n, d

Avec d faire

J, m, a= donnée

Fin avec

Pour i de 1 à n faire

Avec t[i] faire

Cod= donnée

Lib=donnée

Avec dexp faire

J, m, a= donnée

Fin avec

Fin avec

Fin pour

Répéter

N = donnée

Jusqu'à n dans [4..20]

TDOL

Nom	Type
I	Entier

Analyse de la porcdure affich

Def proc affich(v :tab ; m :entier)

Résultat = []

Pour i de 1 à m faire

Avec t[i] faire

Ecrire(cod, lib, dexp.j, dexp.m, dexp.a)

Fin avec

Fin pour

Fin proc affich

TDOL

Nom	Type	Rôle
I	Entier	Compteur

Analyse de la procédure transfer

Def proc transfer (t :tab ; n :entier ; d :date ; var v :tab ; var m :entier)

Résultat = v, m

m ← 0

Pour i de 1 à n faire

Si (t[i].dexp.a < d.a) ou (t[i].dexp.a = d.a et t[i].dexp.m < d.m) ou (t[i].dexp.a = d.a et t[i].dexp.m = d.m et t[i].dexp.<= d.j) alors

m ← m+1

v[m] ← t[i]

Fin si

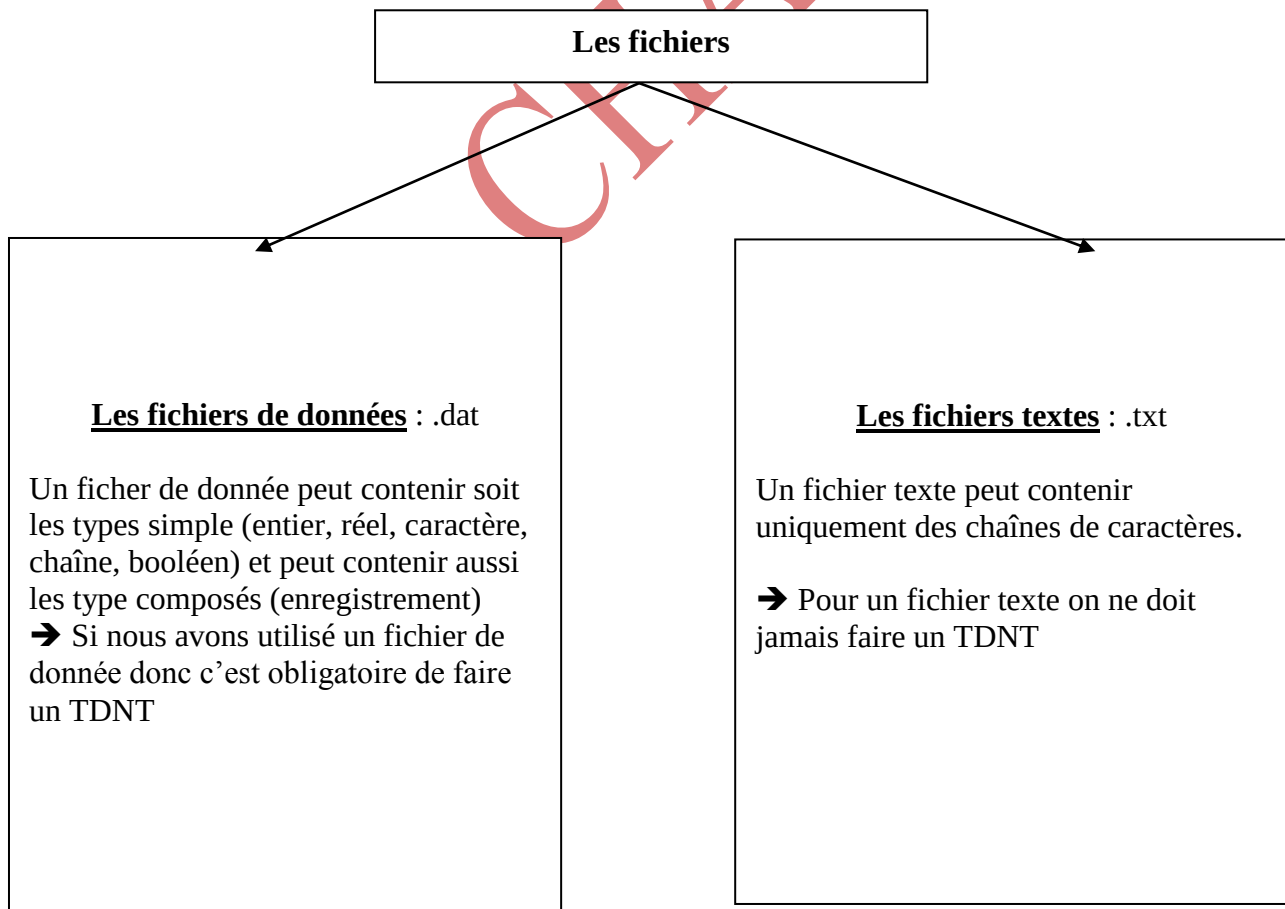
Fin pour

Fin proc transfer

TDOL

Nom	Type	Rôle
I	Entier	Compteur

II. Les natures des fichiers



1) Les

données (.dat) :

	Bloc n°0
	1
	2
	3
	4

fichiers de

Le bloc peut contenir des types simples et des types composé

Le fichier s'appelle F et F de type Fich, donc on doit faire TDNT de Fich, comme le tableau il s'appelle T et de type TAB donc on doit faire TDNT de TAB

TDNT (syntaxe générale)

Type
Nom_type = fichier de type de contenu du fichier
Exemple :
Fich = fichier de réel

TDO

Nom	Type	Rôle
F	Fich	Fichier de réel

Pascal :

Type
Fich= **file of** real ;

Var
F:fich;

Les procédures prédéfinies sur les fichiers de données

Analyse – algorithme	Pascal
Associer (nom_logique, nom_physique) Exemple; Associer (f,"c:\bac\khaled.dat")	assign(f,'c:\bac\khaled.dat');
Recréer(f) Permet de créer un fichier dans le disque dur :	rewrite(f);
Ouvrir(f) Ouvrir un fichier f et se pointer au bloc zéro	reset(f);
Fermer(f) Je quitte le fichier f	close(f);
Lire_nl(f, x) Lire à partir du fichier f	readln(f, x);
Ecrire_nl(f, x) Ecrire dans un fichier f	writeln (f, x);
Fin_fichier (f) Indique la fin du fichier : résultat (vrai ou faux)	eof(f);
Taille_fichier(f) Retourne la taille du fichier → nombre de bloc dans le fichier	filesize(f)
Pointer (f, p) Pointer directement au bloc p	seek(f, p)
Position(f) Indique la position du pointeur dans le fichier f	filepos(f)
Tronquer(f) Supprime tous les blocs situés au dessous du pointeur ainsi que le bloc où il se positionne le pointeur	truncate(f);
. Effacer(f) supprimer le fichier à partir du disque dur	Erase(f)

Activité n°1 :

Saisir un fichier F situé sur "c:\bac 2015\youssef.dat" par 10 entiers, puis afficher le contenu du fichier f.

Analyse principale :

Résultat = proc affich(f)
F= Proc saisi (f)
Associer (f,"d:\bac 2014\ hamdi.dat")
Recréer (f)
Fin

TDNT

Type
Fich = fichier d'entier

TDOG

Nom	Type	Rôle
F	Fich	Fichier
Saisi	Procédure	Procédure saisi
Affich	Procédure	Procédure affichage

Analyse de la procédure saisi

Def proc saisi(var f :fich)
Résultat = f

Ouvrir(f)

Pour i de 1 à 10 faire
X= donnée
Ecrire(f, x)
Fin pour

Fermer(f)

TDOL

Nom	Type	Rôle
X	Entier	Intermédiaire
I	Entier	Compteur

Analyse de la procédure affich

Def proc affich (var f :fich)
Résultat = []

Ouvrir(f)

Pour i de 1 à 10 faire
Lire(f, x)
Ecrire(x)

Fin pour

Fermer(f)

Fin proc affich

TDOL

Nom	Type
I	Entier
X	Entier

Pascal :

```
program ex;  
uses wincrt;
```

```
type  
fich= file of integer;  
var  
f:fich;
```

```
{***** les modules *****}
```

```
procedure saisi (var f:fich);  
var  
i, x:integer;  
begin  
reset(f);
```

```
for i:=1 to 10 do  
begin  
writeln('donner un entier ');  
readln(x);  
write(f, x);  
end;
```

```
close(f);  
end;
```

```
procedure affich(var f:fich);  
var  
x:integer;  
begin  
reset(f);  
while(not(eof(f))) do  
begin  
read(f, x);  
writeln(x);  
end;
```

```
close(f);  
end;
```

```
{***** PP *****}
```

```

begin
  assign(f, 'c:\bac 2015\youssef.dat');
  rewrite(f);

  saisi(f);
  affich(f);
  reset(f);
  writeln('le nombre de bloc est ', filesize(f));
end.

```

Activité n°2:

Saisir un fichier F contenant N élèves avec n compris entre 4 et 10, sachant qu'un élève est composé se : CIN , nom, date de naissance, classe, puis afficher le fichier F

TDNT

Type
Date = enregistrement J, m, a :entier Fin date
Eleve = enregistrement Cin :entier long Nom :chaîne Dn : date Class : chaîne Fin elev
Fich = fichier d'eleve

Brouillon :

- saisir le fichier
- afficher le fichier

Analyse principale :

```

Résultat = proc affich(f)
F=      Proc saisi (f)
Associer (f,"d:\bac 2014\ nefissa.dat")
Recréer (f)
Fin

```

TDNT

Type
Date = enregistrement J, m, a :entier Fin date
Eleve = enregistrement

Cin :entier long
Nom :chaîne
Dn : date
Class : chaîne

Fin elev

Fich = fichier d'eleve

TDOG

Nom	Type	Rôle
F	Fich	Fichier
Saisi	Procédure	Procédure saisie
Affich	Procédure	Procédure affichage

Analyse de la procédure saisie

Def proc saisie(var f :fich)
Résultat = f

Ouvrir(f)
Pour i de 1 à 4 faire
 Avec x faire
 Cin= donnée
 Nom = donnée
 Avec dn faire
 J, m, a = donnée
 Fin avec
 Clas= donnée
 Fin avec

Ecrire(f, x)
Fin pour
Fermer(f)

Analyse de la procédure affichage

Def proc affichage (var f :fich)
Résultat = []

Ouvrir(f)
Pour i de 1 à 4 faire
 Lire(f, x)
 Ecrire(x.cin, x.nom, x.dn.j, x.dn.m, x.dn.a, x.clas)
Fin pour

Fermer(f)
Fin proc affichage

TDOL

Nom	Type
I	Entier
X	Eleve

Pascal :

```
program ex;  
uses wincrt;
```

```
type  
dat=record  
    j, m, a:integer;  
end;
```

```
elev=record  
    cin:integer;  
    nom:string[10];  
    dn: dat;  
    clas:integer;  
end;
```

```
fich= file of elev;
```

```
var  
f:fich;
```

```
{***** les modules *****}
```

```
procedure saisi(var f:fich);
```

```
var  
i, n:integer;  
x:elev;  
begin  
    repeat  
        writeln('donner n');  
        readln(n);  
    until n in [4..10];
```

```
    reset(f);  
    for i:=1 to n do  
    begin  
        with x do  
        begin  
            write('cin: ');readln(cin);  
            write('nom: '); readln(nom);  
            with dn do  
            begin  
                write('date de naissance'); readln(j, m, a);  
            end;
```

```
            write('classe: '); readln(clas);  
        end;
```

```
        write(f, x);  
    end;
```

```
    close(f);  
end;
```

```

procedure affich(var f:fich);
var
  x:elev;
begin
  reset(f);

  while (not(eof(f))) do
  begin
    read(f, x);
    writeln(x.cin, ' ', x.nom, ' ', x.dn.j, '/', x.dn.m, '/', x.dn.a, ' ', x.clas) ;
  end;

  close(f);
end;

{***** PP *****)
begin
  assign(f, 'c:\bac 2015\amine.dat');
  rewrite(f);

  saisi(f);
  affich(f);
end.

```

Activité n°3:

Saisi un fichier F situé sur « d:\bac 2014\feriel.dat », contenant 5 entiers de quatre chiffres, afficher les entiers dont la somme de ces chiffres donnera un entier d'un seul chiffres.

Exemple : 2004 → 2+0+0+4 = 6 donc 2004 sera afficher.

Brouillon :

- saisir fichier
- calculer et afficher à partir du fichier

Pascal :

```

program ex3;
uses winCRT;

type
  Fich=file of integer;

var F:Fich;

{***** les modules *****)

procedure saisi(var F:Fich);
var
  i, x:integer;
Begin
  reset(F);

```

```

for i:=1 to 5 do
begin
  repeat
    writeln('donner un entier');
    readln(x);
    until (x>=1000) and (x<=9999);

    write(f, x);
  end;

  close(F);
End;

procedure affich(var f:fich);
var
  x, a, e, s, i:integer;
  ch:string;
begin
  reset(f);

  while not(eof(f)) do
  begin
    read(f, x);

    str(x, ch);

    s:=0;
    for i:=1 to length(ch) do
    begin
      val(ch[i], a, e);
      s:=s+a;
    end;

    if s<10 then writeln(x);
  end;

  close(f);
end;

{***** PP *****}

BEGIN
  Assign(F,'d:\bac 2015\Feriel.dat');
  rewrite(F);
  saisi(F);
  affich(F);
END.

```

2) Les fichiers textes (.txt) :

Analyse – algorithmie	Pascal
Associer (nom_logique, nom_physique)	assign(f,"c:\bac\khaled.dat");
Associer (f,"c:\bac\khaled.dat")	

Recréer(f) Permet de créer un fichier dans le c: Ouvrir(f) Ouvrir un fichier f et se pointer au bloc zéro Fermer(f) Je quitte le fichier f Lire_nl(f, ch) Lire à partir du fichier f Ecrire_nl(f, ch) Ecrire dans un fichier f Fin_fichier Indique la fin du fichier Effacer(f) supprimer le fichier à partir du disque dur ajouter(f) elle se pointe à la fin de fichier pour ajouter un texte	rewrite(f); reset(f); close(f); readln(f, ch); writeln(f, ch); eof(f); Erase(f) Append(f)
---	---

Série N°1

Exercice n°1 :

Ecrire une analyse permettant de saisir un fichier texte contenant des phrase alphabétique et des espace, sur « c:\bac 2015\essai.txt », la fin de la saisie se termine lorsqu'on tape « fin », sachant que fin ne sera pas écrite dans le fichier,

- Afficher le fichier.
- Supprimer toutes les phrases contenant le mot « ballon ».
- Indiquer le nombre de ligne dans le fichier.

Analyse principale :

Résultat = écrire(fn nombre(f))
 Proc supp(f)
 Proc affich(f)

Proc saisi(f)
 Recréer(f)
 Associer(f, « c:\bac 2013\essai.txt »)

Fin

TDOG

Nom	Type
F	Text
Supp, affich, saisi	Procédure
Nombre	Fonction

Analyse de la procédure saisi

Def proc saisi (var f :text)
 Résultat = f
 Ouvrir (f)
 Répéter
 Ch = donnée

Si ch<> "fin" alors

 Ajouter(f)
 écrire_nl(f, ch)

Fin si
 Jusqu'à ch="fin"
 Femer(f)
 Fin proc saisi

TDOL

Nom	Type
Ch	Chaîne

Analyse de la procédure affich

Def proc affich(var f :text)
 Résultat = []
 Ouvrir(f)
 Tant que (non(fin_fichier(f)) faire
 Lire_nl(f, ch)
 Ecrire(ch)
 Fin tant que
 Femer(f)
 Fin proc affich

TDOL

Nom	Type
Ch	Chaîne

Analyse de la procédure supp

Def proc supp(var f :text)

Résultat = f
 Ouvrir(f)
 Associer(f1, "c:\bac 2013\essai1.txt")
 Recréer(f1)

Tant que non (fin_fichier(f)) faire
 Lire_nl(f, ch)
 Si pos("ballon", ch) = 0 alors
 Ajouter(f1)
 Ecrire_nl(f1, ch)
 Fin si
 Fin tant que

Recréer(f)
 Tant que (non(fin_fichier(f1))) faire
 Lire_nl(f1, ch)
 Ajouter(f)
 Ecrire_nl(f, ch)
 Fin tant que
 Effacer(f1)
 Fermer(f)
 Fin proc supp

TDOL

Nom	Type
Ch	Chaîne
F1	Text

Analyse de la fonction nombre

Def fn nombre(var f :text) : entier
 Résultat = nombre ←s
 [s←0, Ouvrir(f)]
 Tant que (non(fin_fichier(f)) faire
 Lire_nl(f, ch)
 S←S+1
 Fin tant que
 Fermer(f)
 Fin fn nombre

TDOL

Nom	Type
S	Entier
Ch	Chaîne

Pascal :

```
program ex;
uses winCRT;

var
f:text;
```

```
{***** Les Modules *****}
```

```
function alpha(ch:string):boolean;  
var  
  s, i:integer;  
begin  
  s:=0;  
  for i:=1 to length(ch) do  
    begin  
      if (upcase(ch[i]) in ['A'..'Z']) or (ch[i]=' ') then s:=s+1;  
    end;  
  end;
```

```
  alpha:=s=length(ch);  
end;
```

```
procedure saisi(var f:text);  
var  
  ch:string;  
begin  
  reset(f);  
  
  repeat  
    writeln('donner une chaîne');  
    readln(ch);  
  
    if (alpha(ch)=true) and (ch<>'fin') then  
      begin  
        append(f);  
        writeln(f, ch);  
      end;  
  until ch='fin';
```

```
  close(f);  
end;
```

```
procedure affich(var f:text);  
var  
  ch:string;  
begin  
  reset(f);  
  
  while(not(eof(f))) do  
    begin  
      readln(f, ch);  
      writeln(ch);  
    end;
```

```
  close(f);  
end;
```

```
procedure supp(var f:text);  
var  
  ch:string;
```

```

f2:text;
begin
  assign(f2,'c:\bac 2015\provisoir.text');
  rewrite(f2);
  reset(f);

  while (not(eof(f))) do
  begin
    readln(f, ch);
    if pos('ballon',ch)=0 then
      begin
        append(f2);
        writeln(f2, ch);
      end;

  end;

  rewrite(f);
  reset(f2);
  while(not(eof(f2))) do
  begin
    readln(f2, ch);
    append(f);
    writeln(f, ch);
  end;
  close(f);
  { erase(f2); }

end;

procedure nb_ligne(var f:text);
var
  s:integer;
  ch:string;
begin
  reset(f);

  s:=0;
  while(not(eof(f))) do
  begin
    readln(f, ch);
    s:=s+1;
  end;

  close(f);

  writeln('le nombre de bloc est ', s);
end;

{***** PP *****)}

begin
  assign(f,'c:\bac 2015\exxai.txt');
  {ewrite(f);

```

```

saisi(f);}
affich(f);
writeln('*****');

supp(f);
affich(f);

nb_ligne(f);
end.

```

Exercice n°2 :

Soit un fichier F situé sur « c:\bac 2013\essai2.dat » contenant des médicaments, sachant qu'un médicament est caractérisé par un code, libellé, date d'expiration, prix, puis transférer le contenu f dans un fichier F1 situé « c:\bac 2013\essai2.txt ». Afficher le fichier F1.

Analyse principale :

```

Résultat = proc affich(f1)
  F1= Proc transfer(f, f1)
  Recréer(f1)
  Recréer(f)
  Associer(f1, "c:\bac 2013\essai2.txt")
  Associer(f, "c:\bac 2013\essai2.dat")
Fin ex
710

```

TDNT

Type
Ty_dat= enregistrement J, m, a :entier Fin ty_dat
Ty_med= enregistrement Cod :entier long Lib :chaîne Dexp : ty_dat Prix :réel Fin ty_med
Fich = fichier de ty_med

TDOG

Nom	Type
F	Fich
F1	Text
Affich	Procédure
Transfer	Procédure

Analyse de la procédure transfer

```

Def proc transfer (var f :fich ; var f1 :text)
Résultat = f1
Ouvrir(f)

```

```

Ouvrir(f1)
Tant que non (fin _fichier(f)) faire
  Lire(f, x)
  Convch(x.cod, ch_cod)
  Convch(x.dex.j, chj)
  Convch(x.dex.m, chm)
  Convch(x.dex.a, cha)
  Convch(x.prix, chprix)
  Ch ← ch_cod+ " "+x.lib+ " "+chj+ "/" + chm+ "/" + cha+ " "+ch_prix
Ajouter(f1)
Écrire_nl (f1, ch)
Fin tant que
Fermer(f1)
Fermer(f)
Fin proc transfer

```

TDOL

Nom	Type
Chj, cha, chm, ch, ch_cod, ch_prix	Chaîne
X	Ty_med

Analyse de la procédure affich

```

Def proc affich(var f :text)
Résultat = [ ]
Ouvrir(f)
Tant que (non(fin _fichier(f)) faire
  Lire_nl(f, ch)
  Ecrire(ch)
Fin tant que
Femer(f)
Fin proc affich

```

TDOL

Nom	Type
Ch	Chaîne

```

program ex;
uses wincrt;

```

```

type
ty_dat=record
  j, m, a:integer;
end;

```

```

ty_med= record
  cod:longint;
  lib:string[20];
  dexp:ty_dat;
  prix:real;
end;

```

```

fich= file of ty_med;

```

```
var  
f:fich;  
f1:text;
```

```
{***** Les Modules *****}
```

```
procedure saisi(var f:fich);
```

```
var  
x:ty_med;  
i, n:integer;
```

```
begin
```

```
  writeln('donner le nombre de médicament à saisir');  
  readln(n);
```

```
  reset(f);
```

```
  for i:=1 to n do
```

```
  begin
```

```
    with x do
```

```
    begin
```

```
      write('code: '); readln(cod);
```

```
      write('libellé: '); readln(lib);
```

```
      with dexp do
```

```
      begin
```

```
        writeln('donner la date ');
```

```
        readln(j, m, a);
```

```
      end;
```

```
      write('prix: '); readln(prix);
```

```
    end;
```

```
    write(f, x);
```

```
  end;
```

```
  close(f);
```

```
end;
```

```
procedure transfer(var f:fich; var f1:text);
```

```
var
```

```
ch,chj, chm, cha, chprix, chcod:string;
```

```
x:ty_med;
```

```
begin
```

```
  reset(f);
```

```
  reset(f1);
```

```
  while not(eof(f)) do
```

```
  begin
```

```
    read(f, x);
```

```
    str(x.cod, chcod);
```

```
    str(x.dexp.j, chj);
```

```
    str(x.dexp.m, chm);
```

```
    str(x.dexp.a, cha);
```

```
    str(x.prix, chprix);
```

```
  Ch:=chcod+' '+x.lib+' '+chj+'/'+ chm+'/'+ cha+' '+chprix;
```

```
        append(f1);
        writeln(f1, ch);
    end;

    close(f);
    close(f1);
end;

procedure affich(var f:text);
var
    ch:string;

begin
    reset(f);

    while not(eof(f)) do
    begin
        readln(f, ch);
        writeln(ch);
    end;

    close(f);
end;

{***** PP *****}

begin
    assign(f,'c:\bac 2013\essai2.dat');
    rewrite(f);

    assign(f1,'c:\bac 2013\essai2.txt');
    rewrite(f1);

    saisi(f);
    transfer(f, f1);
    affich(f1);
end.
```

Série N°2

Exercice n°1 :

Soit un fichier F situé « d:\bac 2015\location.dat » qui contient n voitures, avec n compris entre 4 et 20, sachant qu'une voiture est caractérisé par matricule (MAT), marque (MARQ), carburant (FUEL), date de location (DL) et la date prévue de retour (DR).

Travail demandé : (objectif parcours du tableau et le transfère du fichier au tableau)

- afficher le contenu du fichier.
- Transférer à partir du fichier F dans un tableau T que les voitures « Diesel »
- Afficher les numéros du blocs des voitures essence.
- Supprimer le bloc qui contient la 1ère voiture « diesel »
- Ajouter à la fin du fichier une voiture

Brouillon :

- afficher le contenu du fichier. → proc affich1
- Transférer à partir du fichier F dans un tableau T que les voitures « Diesel »
→ transfer
- Afficher les numéros du blocs des voitures essence. → affich 2
- Supprimer le bloc qui contient la 1ère voiture « diesel » → supp
- Ajouter à la fin du fichier une voiture → ajout

Analyse :

Résultat = proc ajout(f)
Proc supp(f)
Proc affih2(f)
Transfer(f, t, n)
Proc affich1(f)
Associer(f, "d:\bac 2015\feriel.dat")

Fin ex

TDNT

Type
Date= enregistrement J, m, a : entier Fin date
Voit = enregistrement Mat, marq, fuel :chaîne DR, DL :date Fin voit
Fich= fichier de voit
Tab = tableau [1..20] de voit

TDOG

Nom	Type
F	Fich
T	Tab
N	Entier
Ajout, supp, affich1, affich2, transfer	Procédure

Analyse de la procedure affich

```

Def proc affich1(var f :fich)
Résultat = [ ]
Ouvrir(f)
Tant que non(fin_fichier(f)) faire
  Lire(f, x)
  Avec x faire
    Ecrire(marq, fuel, mat, DL.j, DL.m, DL.a, DR.j, DR.m, DR.a)
  Fin avec
Fermer(f)
Fin proc affich

```

TDOL

Nom	Type
X	Voit

Analyse de la procédure transfer

```

Def proc transfer(var f :fich ; var t :tab ; var n :entier)
Résultat = t, n
Ouvrir(f)
N ← 0
Tant que (non(fin_fichier(f))) faire
  Lire(f, x)
  Si x.fuel="diesel" alors
    N ← N+1
    T[N] ← x
  Fin si
Fin tant que
Fermer(f)
Fin proc transfer

```

TDOL

Nom	Type
X	Voit

Analyse de la procédure affich2

```

Def proc affich2(var f :fich)
Résultat = [ ]
Ouvrir(f)
Tant que (non(fin_fichier(f))) faire
    Lire(f, x)
    Si x.fuel="essence " alors écrire(positin(f)-1)
    Fin si
Fin tant que
Fermer(f)
Fin proc transfer

```

TDOL

Nom	Type
X	Voit

Analyse de la procédure supp

```

Def proc supp(var f:fich)
Résultat = f
Ouvrir(f)
Repeater
    Lire(f, x)
    Jusqu'à (x.fuel="diesel")
P ← position (f)-1

Pour i de (p+1) à taille_fichier(f)-1 faire
    Pointer(f, i)
    Lire(f, x)
    Pointer(f, i-1)
    Écrire(f, x)
Fin pour

Pointer(f, taille_fichier(f)-1)
Tronquer (f)
Fin proc supp

```

TDOL

Nom	Type
X	Voit
l, p	Entier

Analyse de la procédure ajout

```

Def proc ajout(var f :fich)
Résultat = f
Avec x faire
    Mat = donnée
    Fuel = donnée

```

Marq = donnée
 Avec DL faire
 J, m, a : entier
 Fin avec
 Avec DR faire
 J, m, a : entier
 Fin avec
 Fin avec

Ouvrir(f)
 Pointer(f, taille_fichier(f))
 Ecrire(f, x)
 Fermer(f)
 Fin proc ajout

Exercice n°2 :

Une agence de voyage désire informatisé la gestion des clients, la base est enregistré dans un fichier "d:\bac 2015\voyage.dat" elle contient, CIN de client, NOM de client, destination et date de départ.

Travail demandé :

- Supprimer un client à partir du fichier sachant que CIN est donnée.
- Transférer dans un fichier "d:\bac 2015\travel.dat" que CIN et les noms des clients d'une destination donnée.
- Ajouter un client à la fin du fichier "d:\bac 2015\voyage.dat".
- Supprimer tous les enregistrement à partir du fichier "d:\bac 2015\voyage.dat" d'une destination donnée

Correction :

Résultat = []
 F1= proc supp2(f1)
 F1= proc ajout(f1)
 F2= proc transfer(f1, f2)
 F1 = proc supp1(F1)
 Associer (F1, ".....\ voyage.dat")
 Associer (F2, ".....\ travel.dat")
 Récréer(f2)
 Fin ex

TDNT

Type
Date = enregistrement J, m, a : entier Fin date
Voy= enregistrement CIN : entier NOM, dest : chaîne DD : date Fin voy
Info = enregistrement

CIN : entier
 NOM : chaîne
 Fin info
 Fich1= fichier de voy
 Fich2 = fichier de info

TDOG

Nom	Type
F1	Fich1
F2	Fich2
Supp, transfer, ajout	Procédure

Analyse principale :

Résultat = []
 F1= proc supp2(f1)
 F1= proc ajout(f1)
 F2= proc transfer(f1, f2)
 F1 = proc supp1(F1)
 Associer (F1,".....\ voyage.dat")
 Associer (F2,".....\ travel.dat")
 Récréer(f2)
 Fin ex

TDNT

Type
Date = enregistrement J, m, a : entier Fin date
Voy= enregistrement CIN : entier NOM, dest : chaîne DD : date Fin voy
Info = enregistrement CIN : entier NOM : chaîne Fin info
Fich1= fichier de voy Fich2 = fichier de info

TDOG

Nom	Type
F1	Fich1
F2	Fich2
Supp, transfer, ajout	Procédure

Analyse de la proc supp → recherche puis tu supprime

Def proc supp(var F1 :fich)

Résultat = F1

Ouvrir(f1)

CI =donnée

Repeater

Lire(f1, x)

Jusqu'à fin_fichier(f1) ou (x.CIN = CI)

Si (X.CIN = CI) alors p ← position(f1)-1

Pour i de (p+1) à taille_fichier(f1)-1 faire

Pointer(f1, i)

Lire(f1,y)

Pointer(f1, i-1)

Ecrire(f1, y)

Fin pour

Pointer(f1, taille_fichier(1)-1)

Tronquer (f1)

Fin si

Fermer(f1)

TDOL

Nom	Type
CI	Entier
Y, X	Voy
I, p	Entier

Analyse de la procédure transfer

Def proc transfer(var f1 :fich1 ; var f2 :fich2)

Résultat = f2

Destination = donnée

Ouvrir(f1)

Ouvrir(f2)

Tant que non(fin_fichier(f1)) faire

Lire(f1, x)

Si x.dest= Destination alors y.CIN ← X.CIN

y.NOM ← X.NOM

écrire(f2, y)

fin si

fin tant que

fermer(f1)

fermer(f2)

fin proc transfer

TDOL

Nom	Type
Destination	Chaîne
X	Voy
Y	Info

Analyse de la procédure ajout

```
def proc ajout(var f1 : fich1)
résultat = F1
avec x faire
  CIN = donnée
  NOM = donnée
  Avec DD faire
    J, m, a= donnée
  Fin avec
  Dest= donnée
Fin avec

Ouvrir(f1)
Pointer(f1, taille_fichier(f1))
Ecrire(f1, x)
Fin proc ajout
```

TDOL

Nom	Type
X	Voy

Analyse de la procédure supp2

```
Def proc supp2( var F1 :fich)
Résultat = F1
Ouvrir(f1)

Destination =donnée
répéter
Repeater
  Lire(f1, x)
  Jusqu'à fin_fichier(f1) ou (x.dest = destination)

Si (X.dest = destination ) alors p ← position(f1)-1
  Pour i de (p+1) à taille_fichier(f1)-1 faire
    Pointer(f1, i)
    Lire(f1,y)
    Pointer(f1, i-1)
    Ecrire(f1, y)
  Fin pour
  Pointer(f1, taille_fichier(1)-1)
  Tronquer (f1)

Fin si
Jusqu'à fin_fichier(f1)
Fermer(f1)
```

TDOL

Nom	Type
Cl	Entier
Y, X	Voy
l, p	Entier

Exercice n°3 :

Une école désire informatiser sa base de données qui est sauvegardé dans un fichier F (le fichier est supposé saisi « c:\bac 2013\base.dat », sachant que le fichier contient l'information suivante d'un élève.

Code, nom, moyenne, date de naissance, classe.

- transférer dans un fichier F1 situé sur « c:\bac 2013\admis.dat » que les élèves admis.
- Transférer à partir de F dans un tableau T que les noms et les moyennes des élèves redoublant.
- Afficher à partir du fichier F1 que le meilleur élève.
- Mettre à jours le fichier F pour l'année scolaire prochaine dans un fichier F2 sauvegardé sous le nom « c:\bac 2013\ ajour.dat »

Analyse principale

Résultat =

```
F2 =   proc ajour(f, f2)
      proc affich(f1)
T, n =   proc transfer2(f, t, n)
F1 =   proc transfer1(f, f1)
```

```
Associer(f,"c:\bac 2013\base.dat)
Recréer(f)
```

```
Associer(f1,"c:\bac 2013\base.dat)
Recréer(f1)
Fin ex
```

TDNT

Type
Ty_dat = enregsitrement J, m, a :entier Fin ty_dat
Ty_elev1 = enregistrement Cod :entier Nom : chaîne Moy :réel Dn : ty_dat Classe : entier Fin ty_elev1
Ty_elev2 = enregistrement Nom :chaîne Moy :réel Fin ty_elev2
Fich = fichier de ty_elev1 Tab = tableau [1..20] de ty_elev2

TDOG

Nom	Type
-----	------

F1, f T N Transfer1, tranfer2, affich, ajour	Fich Tab Entier Procédure
---	------------------------------------

Analyse de la procédure tarnsfer1

```

Def proc transfer1(var f, f1 :fich)
Résultat = f1
Ouvrir(f)
Ouvrir(f1)

Tant que non(finfichier(f)) faire
  Lire(f, x)
  Si x.moy>= 10 alors écrire(f1, x)
  Fin si
Fin tant que
Fermer(f)
Fermer(f1)
Fin proc transfer1

```

TDOL

Nom	Type
X	Ty_elev1

Analyse de la procédure transfer2

```

Def proc transfer2(var f :fich ; var t :tab ; var n :entier)
Résultat = t, n

Ouvrir(f)
N←0

Tant que non (fin_fichier(f)) faire
  Lire(f, x)
  Si x.moy<10 alors
    N←N+1
    T[n].nom←x.nom
    T[n].moy←x.moy
  Fin si
Fin tant que
Fermer(f)
Fin proc transfer2

```

TDOL

Nom	Type
X	Ty_elev1

Analyse de la procédure max

```

Def proc affich (var f1 :fich)
Résultat = écrire (max.cod, max.nom, max.dn.j, max.dn.m, max.dn.a, max.moy,
max.classe)

```

Ouvrir(f1)
Lire(f1, max)

Tant que (non(fin_fichier(f)) faire
 Lire(f, x)
 Si max.moy<x.moy alors max ← x
 Fin si
Fin tant que
Fermer(f1)
Fin proc affich

TDOL

Nom	Type
X	Ty_elev1

Analyse de la procédure ajour

Def proc ajour(var f, f2 :ty_fich)
R  ultat = f2
Ouvrir(f)
Ouvrir(f2)
Tant que non(fin_fichier(f)) faire
 Lire(f, x)
 Si (x.class<>4) et (x.moy>=10) alors x.class ← x.class+1
 x.moy ← 0
   crire(f2, x)
 Sinon si x.moy<10 alors x.moy ← 0
 Ecrire(f2, x)
 Fin si
Fin tant que
Fermer(f)
Fermer(f2)
Fin proc ajour

Analyse de la proc  dure max

Def proc max (var f1 :ty_fich)
R  ultat =   crire(max.nom, max.cod, max.dn.j , max.dn.m, max.dn.a, max.moy,
max.class)
[ouvrir(f1), lire(f1, max)]
Tant que non (fin_fichier(f1)) faire
 Lire(f1, x)
 Si x.moy>max.moy alors max ← x
 Fin si
Fin tant que
Fermer(f1)
Fin proc max

TDOL

Nom	Type
X, max	Ty_elev

Exercice n  4 (PROBLEME FICHIER dat+txt) :

Soit les deux fichiers suivants « c:\class.dat » et « c:\lycee.dat », sachant que le fichier class.dat contient les différentes classe du lycée, et le fichier lycee.dat, contient les informations d'un élève tel que : la classe, nom, moyenne.

On suppose que les fichiers classe.dat et résultat.dat sont saisis

- Transférer la classe et la moyenne maximale pour chaque classe dans un tableau T.
- Chercher les informations des l'élève contenant la moyenne maximale dans le lycée.
- Transférer dans un fichier textes toutes amies de classe de l'élève possédant la moyenne maximale.

Analyse principal:

Résultat =

```
Proc trans_text(t[n], fl, fT)
Proc rech (t[n], fl)
Proc tri (t, n)
Trans_tab (fc, fl, t, n)
Associer (fc," c:\class.dat ")
Associer (fl," c:\résultat.dat ")
Associer (fT,".....txt")
```

Fin ex

TDNT

Type
Eleve=enregistrement Class, nom :chaîne Moy :réel Fin elev Info = enregistrement class : chaîne Moy :réel Fin info Fich_ly=fichier de eleve Fich_cl=fichier de classe Tab= tableau de n info

TDO

Nom	Type
Fl	Fich_ly
FT	Texte
Fc	Fich_cl
T	Tab
N	Entier
Trans_text	Procédure
Tarns_tab	Procédure
Rech	Procédure
Tri	Procédure

Analyse de la procédure Trans tab

```

Def proc trans_tab (var fl :fich_ly ; var fc : fich_cl ; var t :tab ; var n :entier)
Résultat = t, n
T= []
[Ouvrir(fc), n←0]
Pour i de 0 à taille_fichier(fc)-1 faire
  Pointer(fc, i)
  Lire(fc,x)
  Ouvrir(fL)

  Répéter
    Lire(fL, y)
  Jusqu'à y.class=x

  Max←y
  Tant que non(fin_fichier(fL)) faire
    Lire(fL, y)
    Si (y.class=x) et (max.moy<y.moy) alors max←y
  Fin tant que

  N←n+1
  T[n].moy ← max.moy
  T[n].class ← max.class

Fin pour
Fermer(fL)
Fermer(fc)
Fin proc trans_tab

```

TDOL

Nom	Type
I	Entier
X	Chaîne
Y	Eleve
Max	Eleve

Analyse de la procédure tri bull

```

Def proc tri ( var t :tab ; n :entier)
Résultat = t
Répéter
  Test ← faux
  Pour i de 1 à n-1 faire
    Si t[i].moy>t[i+1].moy alors
      Aux←t[i]
      T[i]←t[i+1]
      T[i+1]←aux
      Test←vrai
    Fin si
  Fin pour
Jusqu'à test = faux

```

TDOL

Nom	Type
Aux	Info
I	Entier

Test	Booléen
------	---------

Analyse de recherche

```

Def proc rech (t[n] :info ; var fL :fich_ly)
Résultat = Ecrire(x.class, x.nom, x.moy)
[ouvrir(fL)]
Répéter
    Lire(fL,x)
Jusqu'à x.moy=t[n].moy
Fermer(fL)
Fin proc rech

```

TDOL

Nom	Type
x	eleve

Analyse de la procédure trans_text

Méthode avec append= ajouter

```

Def proc trans_text(t[n] :info ; var fL :fich_ly ; var ft :text)
Résultat = ft
Ft= [recréer(ft), ouvrir(fL)]
    Tant que non(fin_fichier(fL)) faire
        Lire(fL, x)
        Si x.class=t[n].class alors
            Convch(x.moy, chmoy)
            Ch ← x.class+x.nom+chmoy
            Ajouter(ft)
            Ecrire_nl(ft,ch)
        Fin si
    Fin tant que
Fermer(ft)
Fermer(fL)
Fin proc trans_text

```

TDOL

Nom	Type
Ch, chmoy	Chaîne
X	Eleve

Analyse de la procédure trans_text

Méthode sans append= ajouter

```

Def proc trans_text(t[n] :info ; var fr :fich_elev ft :text)
Résultat = ft
Ft= [
recréer(ft), ouvrir(fr)]
    Tant que non(fin_fichier(fr)) faire
        Lire(fr, x)
        Si x.class=t[n].class alors
            Convch(x.moy, chmoy)
            Ch ← x.class+x.nom+chmoy

```

Ecrire_nl(ft,ch)

Fin si
Fin tant que
Fermer(ft)
Fermer(fr)
Fin proc tarns_text

TDOL

Nom	Type
Ch	Chaîne
X	Eleve

Chapitre 2 : La récursivité

I. Syntaxe générale sur les procédures récursifs

1) Pour

Def proc nom (var P1 : type ; P2 : type)
Résultat = nom
Pour i de 1 à fin faire
Traitement
Fin pour
Fin proc nom

Réursive

Def proc nom (var P1 : type ; P2 : type ; i : entier)
Résultat = nom
Si i <= fin alors
 Traitement
 Appel proc nom(P1, P2, i+1)
Fin si
Fin proc nom

2) Répéter

```
Def proc nom (var P1 : type ; P2 : type)
Résultat = nom
Répéter
Traitement
Jusqu'à (condition)
Fin proc nom
```

Réursive

```
Def proc nom (var P1 : type ; P2 : type)
Résultat = nom
Si non(condition) alors
    Traitement
    Appel proc nom (P1, P2)
Fin proc nom
```

3) Tant que

```
Def proc nom (var P1 : type ; P2 : type)
Résultat = nom
Tant que (condition ) faire
    Traitement
Fin tan que
Fin proc nom
```

Réursive

```
Def proc nom (var P1 : type ; P2 : type)
Résultat = nom
Si (condition ) alors
    Traitement
    Appel proc nom(P1, P2)
Fin si
Fin proc nom
```

II. Syntaxe générale sur les fonctions récursifs

1) La fonction retourne le type : entier, réel, chaîne, caractère

```
Def fn nom ( ..... ) : entier, réel, caractère, chaîne
Résultat = nom
Si (cas particulier) alors nom ← ?
Sinon nom ← valeur (+ - * / div mod ) nom(.....)
Fin si
Fin fn nom
```

2) La fonction retourne le type : booléen

```
Def fn nom ( ..... ) : booléen
Résultat = nom
Si (cas particulier vrai ) alors nom ← vrai
```

```

Sinon Si (cas particulier faux) alors nom ← faux
Sinon
  Traitement
  nom ← nom(...)
Fin si
Fin fn nom

```

III. Recherche séquentielle et dichotmique récursifs

1) Recherche séquentielle

```

Def fn rech_seq(x :entier ; t :tab ; n :entier) :booleén
  Résultat = rech_seq
  Si t[n]=x alors rech_seq←vrai
  Sinon Si n=0 alors rech_seq←faux
  Sinon rech_seq ← rech_seq(x, t, n-1)
  Fin si
Fin fn rech_seq

```

2) Recherche dichotmique

```

Def fn rech_dicho(x :entier ; t :tab ; n, deb , fin :entier) :booleén
  Résultat = rech_dicho
  M ← (deb+fin) div 2
  Si t[m]=x alors rech_dicho← vrai
  Sinon si deb>fin alors rech_dicho ←faux
  Sinon
    Si x>t[m] alors deb←m+1
    Sinon fin ←m-1
    Fin si
    Rech_dicho← rech_dicho (x, t, n, deb, fin)
  Fin si

```

Série N°3

Exercice n°1 :

Ecrire une analyse récursive permettant de saisir un entier $n > 2$.

Normal :

```
Def proc saisi(var n :entier)
Résultat = n
Répéter
    N = donnée
Jusqu'à n>2
Fin proc saisi
```

Réursive :

```
Def proc saisi(var n :entier )
Résultat = n
Si n<= 2 alors
    N= donnée
    Proc saisi(n)

Fin si
Fin proc saisi
```

Pascal :

```
program ex;
uses winCRT;

var
n:integer;
```

```

procedure saisi(var n:integer);
begin
  if n<=2 then
    begin
      writeln('donner n ');
      readln(n);
      saisi(n);
    end;
end;

begin
  saisi(n);
  writeln(n);
end.

```

Exercice n°2 :

Ecrire une analyse d'une procédure récursive permettant saisir un tableau par n entier (supposant que n est déjà saisi)

Normal :

```

Def proc rempli(var t :tab ; n :entier)
Résultat = t

```

```

Pour i de 1 à n faire
  T[i] = donnée
Fin pour

```

```

Fin proc rempli

```

récursive :

```

Def proc rempli(var t :tab ; n, i :entier)
Résultat = t

```

```

Si i<= n alors
  T[i] = donnée
  Proc Rempli(t, n , i+1)

```

```

Fin si

```

```

Fin proc rempli

```

Exercice n°3 :

Ecrire une analyse d'une procédure récursive permettant d'afficher un tableau par n réel.

récursive :

```

Def proc affich (var t :tab ; n, i :entier)

```

Résultat = t

Si $i \leq n$ alors

 Ecrire($t[i]$)

 Proc affich($t, n, i+1$)

Fin si

Fin proc affich

Exercice n°4 :

Ecrire une analyse d'une procédure récursive permettant d'éclater un tableau en deux tableaux T_p (valeur paire) et T_i (valeur impaire).

Normal :

Def proc eclat(t : tab ; n : entier ; var tp, ti : tab ; var Np, Ni : entier)

Résultat = tp, ti, ni, np

$Ni \leftarrow 0, np \leftarrow 0$

Pour i de 1 à n faire

 Si $t[i] \bmod 2 = 0$ alors

$Np \leftarrow Np + 1$

$tp[np] \leftarrow t[i]$

 Sinon

$Ni \leftarrow Ni + 1$

$ti[Ni] \leftarrow t[i]$

Fin si

Fin pour

Réursive :

Def proc eclat(t : tab ; n : entier ; var tp, ti : tab ; var Np, Ni : entier ; i : entier)

Résultat = tp, ti, ni, np

Si $i \leq n$ alors

 Si $t[i] \bmod 2 = 0$ alors

$Np \leftarrow Np + 1$

$tp[np] \leftarrow t[i]$

 Sinon

$Ni \leftarrow Ni + 1$

$ti[Ni] \leftarrow t[i]$

Fin si

 Proc ecalat($t, n, tp, ti, np, ni, i+1$)

Fin si

Série N°4

Exercice n°1 :

Saisir un tableau par n entier avec n compris entre 5 et 50, afficher les valeurs impaires du tableau.

Brouillon :

- saisir n
- remplir tableau
- afficher les impaires

<u>Itératif</u>	<u>Réursive</u>
<u>Saisi n</u> Def proc saisi(var n :entier) Résultat = n Répéter N = donnée Jusqu'à n dans [5..50] Fin proc saisi	<u>Saisi n</u> Def proc saisi(var n :entier) Résultat = saisi Si non(n dans[5..50]) alors N = donnée Proc saisi(n) Fin si Fin proc saisi
<u>Remplir tableau</u> Def proc rempli (n :entier ; var t :tab) Résultat = t Pour i de 1 à n faire T[i]= donnée Fin pour Fin proc rempli	<u>Remplir tableau</u> Def proc rempli (n :entier ; var t :tab ; i :entier) Résultat = rempli Si i <= n alors T[i]= donnée Proc rempli(n, t, i+1) Fin si Fin proc rempli
<u>Affichage des impaires</u> Def proc affich_imp(t :tab ; n :entier) Résultat = [] Pour i de 1 à n faire Si t[i] mod 2 <> 0 alors écrire(t[i])	<u>Affichage des impaires</u> Def proc affich_imp(t :tab ; n :entier ; i :entier) Résultat = affich_imp Si i <= n alors

```

Fin si
Fin pour
Fin proc affich_imp

```

```

Si t[i] mod 2 <> 0 alors écrire(t[i])
Fin si
Affich_imp(t, n, i+1)
Fin si
Fin proc affich_imp

```

Exercice n°2 :

Saisir un tableau par n chaîne non vide avec n compris entre 5 et 20, afficher les voyelles de chaque chaîne.

Brouillon :

- saisi n
- remplir tableau
- affich

Itératif	Réursive
<u>Saisi n</u> <pre> Def proc saisi(var n :entier) Résultat = n Répéter N = donnée Jusqu'à n dans [5..20] Fin proc saisi </pre>	<u>Saisi n</u> <pre> Def proc saisi(var n :entier) Résultat = saisi Si non(n dans[5..20]) alors N = donnée Proc saisi(n) Fin si Fin proc saisi </pre>
<u>Remplir tableau</u> <pre> Def proc rempli (n :entier ; var t :tab) Résultat = t Pour i de 1 à n faire Répéter T[i]= donnée Jusqu'à t[i]<>"" Fin pour Fin proc rempli </pre>	<u>Remplir tableau</u> <pre> Def proc rempli (n :entier ; var t :tab ; i :entier) Résultat = rempli Si i <= n alors T[i]= donnée Si t[i]=" " alors proc rempli(n, t, i) sinon Proc rempli(n, t, i+1) Fin si Fin si Fin proc rempli </pre>
<u>Affichage des voyelles</u> <pre> Def proc affich(t :tab ; n :entier) Résultat = [] Pour i de 1 à n faire Pour j de 1 à long(t[i]) faire Si Majus(t[i][j]) dans ["liste des voyelle"] alors écrire(t[i][j]) </pre>	<u>Affichage des voyelles</u> <pre> Def proc affich(t :tab ; n, i, j :entier) Résultat = affich Si i<=n alors Si j<= long(t[i]) alors Si majus(t[i][j]) dans ["liste des voyelle"] alors écrire (t[i][j]) </pre>

```

Fin si
Fin pour
Fin pour
Fin proc affich

```

```

Fin si
Affich(t, n, i, j+1)
Fin si
Affich(t, n, i, 1)
Fin si
Fin proc affich

```

Série N°5

Exercice n°1 (exercice n°6 page 91) :

Proposer une analyse et déduisez l'algorithme d'une fonction récursif appelé contigu, qui détermine si une chaîne comporte deux caractères contigus identiques :

Contigus (« elle ») : à pour résultat vrai.
 Contigus (« fonction ») : à pour résultat faux.
 Contigus (« b ») : à pour résultat faux.
 Contigus (« youssef ») à pour résultat vrai

```

Def fn contigu(ch :chaîne) : booleén
Résultat = contigu
Si long(ch) dans [0, 1] alors contigu ← faux
Sinon si ch[1]= ch[2] alors contigu ← vrai
Sinon
  Delete(ch, 1, 1)
  Contigu← fn contigu(ch)
Fin si
Fin fn contigu

```

Pascal :

```

program ex;
uses wincrt;

var
ch:string;

{***** Les Modules *****}

function contigue(ch:string):boolean;
begin
  if ch[1]=ch[2] then contigue:=true
  else if length(ch) in [0,1] then contigue:=false
  else
    begin
      delete(ch, 1, 1);
      contigue:=contigue(ch);
    end;
end;

```

```
end;
```

```
{***** PP *****}
```

```
begin
```

```
  writeln('donner une chaîne');
```

```
  readln(ch);
```

```
  writeln(contigue(ch));
```

```
end.
```

Exercice n°2 :

Ecrire un algorithme récursif permettant de vérifier si une chaîne saisie est tautogramme ou non.

On appelle tautogramme ssi tous les mots qui forment la chaîne commencent par la même lettre (on suppose que la chaîne ne commence pas par une espace et ne se termine pas par une espace et on ne trouve pas deux espaces consécutifs).

Exemple :

Le lion lape le lait lentement.

Correction :

```
Def fn tauto (ch : chaîne) : booléen
```

```
Résultat = tauto
```

```
Si pos("-",ch)=0 alors tauto ← vrai
```

```
Sinon si ch[1]<> ch[pos("-",ch)+1] alors tauto ← faux
```

```
Sinon
```

```
  P ← pos("-",ch)
```

```
  Efface(ch, 1, p)
```

```
  Tauto ← fn tauto(ch)
```

```
Fin si
```

```
Fin fn tauto
```

Pascal :

```
program ex;
```

```
uses winCRT;
```

```
var
```

```
ch:string;
```

```
{***** Les Modules *****}
```

```
function tauto(ch:string):boolean;
```

```
var
```

```
p:integer;
```

```
begin
```

```
  if pos(' ', ch)=0 then tauto:=true
```

```
  else if ch[1]<>ch[pos(' ', ch)+1] then tauto:=false
```

```
  else
```

```
    begin
```

```
      p:=pos(' ',ch);
```

```
      delete(ch, 1, p);
```

```

        tauto:=tauto(ch);
    end;

end;

{***** PP *****)
begin
    writeln('donner une chaîne');
    readln(ch);

    writeln(tauto(ch));
end.

```

Correction :

```

program ex;
uses wincrt;

type
    tab = array[1..20] of integer;

var
    t:tab ;
    n,i:integer;

{***** Les Modules *****)

procedure saisi(var t:tab ; i, n:integer);
begin
    if i<=n then
        begin
            writeln('donner t['i,']');
            readln(t[i]);
            saisi(t, i+1, n);
        end;
    end;

procedure affich (t:tab ; i, n:integer);
begin
    if i<=n then
        begin
            writeln(t[i]);
            affich (t, i+1, n);
        end;
    end;

{***** PP *****)

begin
    n:=5;
    saisi(t, 1, n);
    affich(t, 1, n);

```

end.

Exercice n°3 :

Saisir un tableau par n entier puis afficher les éléments des tableaux.

Pascal :

```
program ex;
uses wincrt;

type
tab = array[1..20] of integer;

var
t:tab ;
n:integer;

{***** Les Modules *****)

procedure saisi(var n:integer);
begin
    writeln('donner n ');
    readln(n);
end;

procedure rempli(var t:tab ; n, i:integer);
begin
    if i<=n then
    begin
        writeln('donner t[',i,']');
        readln(t[i]);
        rempli(t, n, i+1);
    end;
end;

procedure affich (t:tab ; n, i:integer);
begin
    if i<=n then
    begin
        writeln(t[i]);
        affich(t, n, i+1)
    end;
end;

{***** PP *****)

begin
saisi(n);
rempli(t, n, 1);
affich(t, n, 1);
end.
```

Exercice n°4 :

Ecrire une procédure récursive permettant d'afficher les voyelles à partir d'une chaîne.

ch = "Khaled"
On affiche : A e

Pascal :

```
program ex;  
uses wincrt;
```

```
var  
ch:string;
```

```
{***** Les Modules *****}
```

```
procedure affich(ch:string; i:integer);  
begin
```

```
    if i<=length(ch) then
```

```
        begin
```

```
            if upcase(ch[i]) in ['A','E','Y','U','I','O'] then writeln(ch[i]);  
            affich(ch, i+1);  
        end;
```

```
end;
```

```
{***** PP *****}
```

```
begin  
writeln('donner une chaîne');  
readln(ch);
```

```
affich(ch, 1);  
end.
```

Exercice n°5 :

Ecrire une fonction récursive permettant de supprimer les espaces superflus à partir d'une chaîne quelconque.

Analyse :

```
Def fn supp(ch :chaîne) : chaîne  
Résultat = supp  
Si pos(" _ _ ",ch)=0 alors supp ← ch  
Sinon  
    Delete(ch, pos(" _ _ ",ch), 1)  
    Supp ← supp(ch)  
Fin si  
Fin supp
```

Pascal :

```
program ex;  
uses wincrt;
```

```

var
ch:string;

{***** Les Modules *****)

function supp(ch:string):string;
begin
  if pos(' ',ch)=0 then supp:=ch
  else
    begin
      delete(ch, pos(' ',ch), 1);
      supp:=supp(ch);
    end;
end;

end;

{***** PP *****)

begin

writeln('donner une chaîne ');
readln(ch);

writeln(supp(ch));
end.

```

Exercice n°6 :

Ecrire une procédure récursive permettant d'éclater un tableau T de N entiers en deux tableaux Tp (contient les entiers paires) et Ti (contient les entiers impaires).

Correction :

```

Def proc eclat (t : tab ; n :entier ; var Tp :tab ; var p :entier ; var Ti :tab ; var imp :entier ;
i :entier)
Résultat = Tp, Ti, p, Imp
Si i<= n alors
  Si t[i] mod 2 = 0 alors
    P←P+1
    Tp[p]←T[i]
  Sinon
    Imp←Imp+1
    Ti[imp]←t[i]
  Fin si
  Eclat(t, n, tp, p, ti, imp, i+1)
Fin si
Fin proc ecalt

```

Exercice n°7 :

Ecrire une Fonction récursive permettant de faire la somme d'un tableau T de N entiers.

Correction :

```

Def fn somme(t :tab ; n :entier) :entier
Résultat = somme

```

```
Si n =0 alors somme ← 0
Sinon somme ← t[n] + somme(t, n-1)
Fin si
Fin fn somme
```

ZIED CHAOUCH

Chapitre 3 : Les algorithmes arithmétiques

I. PPCM récursive :

```

program ex;
uses wincrt;
var
a, b:integer;

{***** les modules *****}

procedure saisi(var x:integer);
begin
  repeat
    writeln('donner un entier ');
    readln(x);
  until x>0;
end;

function ppcm (a, b, max:integer):integer;
var
maxi, min:integer;
begin
  if a<b then
    begin
      min:=a;
      maxi:=b;
    end
  else
    begin
      min:=b;
      maxi:=a;
    end;

  max:=maxi+max;

  if (max mod min <>0) then
    begin
      ppcm:=ppcm(a, b, max);
    end
  else ppcm:=max;
end;

{***** pp *****}

begin
  saisi(a);
  saisi(b);
  writeln(ppcm(a, b, 0));
end.

```

II. PGCD récursive

```

Def fn PGCD (A, B :entier) :entier
Résultat = PGCD
Si A= B alors PGCD ← A
Sinon
  Si A>B alors PGCD ← PGCD(A-B, B)
  Sinon PGCD ← PGCD(A, B-A)

```

```

    Fin si
Fin si
Fin fn PGCD

```

III. Nombre premier Récursive

```

program ex;
uses wincrt;
var
a, b:integer;

{***** les modules *****)}

procedure saisi(var x:integer);
begin
    repeat
        writeln('donner un entier ');
        readln(x);
    until x>0;
end;

function premier (x, i:integer):boolean;
begin
    if i>= x div 2 then premier:=true
    else if x mod i=0 then premier :=false
    else premier:=premier(x, i+1);
end;

{***** pp *****)}

begin
    saisi(a);
    writeln(premier(a, 2));
end.

```

IV. Décomposition en facteur premier

1) Décomposition en facteur premier Récursive :

```

program ex;
uses wincrt;
var
a, b:integer;

{***** les modules *****)}

procedure decomp_fact(n, i:integer);
begin
    if n<>1 then
        begin
            if n mod i =0 then
                begin
                    writeln(i);

```

```

                decomp_fact(n div i, i)
                end
                else decomp_fact(n, i+1);
            end;
        end;

{***** pp *****}

begin
    readln(a);
    decomp_fact(a, 2);
end.

```

2) Décomposition en facteur premier sous forme d'une chaîne

```

program ex;
uses wincrt;
var
    n:integer;

{***** les modules *****}

function decomp_fact(n:integer):string;
var
    i:integer;
    ch, ch:string;
begin
    i:=2;
    ch:="";
    repeat
        if n mod i =0 then
            begin
                str(i, chi);
                ch:=ch+chi+'*';
                n:=n div i;
            end
            else i:=i+1;
        until n=1;

        delete(ch, length(ch), 1);
        decomp_fact:=ch;
    end;

{***** PP *****}

begin
    writeln('donner n ');
    readln(n);
    writeln(decomp_fact(n));
end.

```

V. Combinaison

Def fn fact (n :entier) :entier

Résultat = fact ← F

F ← 1

Pour i de 1 à n faire

F ← F*i

Fin pour

Fin fn fact

Def fn comb(n, p :entier) ;réel

Résultat = comb ← fact(n) / (fact(p)*fact(n-p))

Fin fn comb

VI. Arrangement

Def fn arrag(n, p :entier) :entier

Résultat = A

A ← 1

Pour i de N à (n-p+1) faire

A ← A*i

Fin pour

Fin fn arrag

VII. Divisibilité

1) Par 2

On appelle un nombre divisible par 2 ssi le nombre des unités appartient (0, 2, 4, 6, 8)

Def fn div_2 (n ::entier) :booleén

Résultat = div_2 ← test

Si ch[long(ch)] in ['0', '2', '4', '6', '8'] alors test ← vrai

Sinon test ← faux

Fin si

Convch(n, ch)

Fin fn div_2

TDOL

Nom	Type
Test	Booleén
Ch	Chaîne

2) Par 3

un nombre est divisible par 3 lorsque la somme de ses chiffres est un nombre multiple de 3 et on fait le même travail jusqu'on obtient un nombre d'un seul chiffre
exemples:

8 433 (8+4+3+3 = 18)

18 (1+8) = 9

Donc si le nombre appartient (3, 6, 9) donc divisible par 3

Def fn div_3 (n ::entier) :booleén

Résutat = div_3 ← test

Si n dans [3, 6, 9] alors test ← vrai

Sinon test ← faux

Fin si

Repeter

Convch(n, ch)

n ← 0

Pour i de 1 à long(ch) faire

Valeur(ch[i], x, e)

n ← n+x

Fin pour

Jusqu'à n < 10

TDOL

Nom	Type
Test	Booleén
Ch	Chaîne

Pascal :

program ex;

uses wincrt;

var

n:longint;

{***** les modules *****}

function div_3(n:longint):boolean;

var

ch:string;

i, x:longint;

e:integer;

begin

repeat

str(n, ch);

n:=0;

for i:=1 to length(ch) do

begin

val(ch[i], x, e);

n:=n+x;

end;

until n in [0..9];

if n in [3, 6, 9] then div_3:=true

else div_3:=false;

end;

{***** pp *****}

begin

```
writeln('donner un entier ');
readln(n);
writeln(div_3(n));
end.
```

3) Par 4

1288 → multiplier les dizaines par 2 et on ajoute les unités, je répète le travail jusqu'à n devient <10

88 → $(8*2)+8 = 24$

$24 = (2*2)+4 = 8$ donc 8 appartient $[0, 4, 8]$ donc divisible.

```
function div_4(n:longint):boolean;
var
ch, ch1:string;
i, x, d, u:longint;
e:integer;
```

```
begin
```

```
repeat
str(n, ch);
ch1:=copy(ch,length(ch)-1, 2);
```

```
val(ch1[1], d, e);
val(ch1[2], u, e);
n:=(d*2)+u;
until n <10 ;
```

```
if n in [0, 4, 8] then div_4:=true
else div_4:=false;
end;
```

4) Par 5

```
Def fn div_5 (n : entier) :booleén
Résutat = div_5 ← test
```

```
Convch(n, ch)
Si ch[long(ch)] dans ['0', '5'] alors test←vrai
Sinon test←faux
Fin si
```

```
Fin fn div_5
```

TDOL

Nom	Type
Test	Booleén
Ch	Chaîne

5) Par 6

Un nombre est divisible par 6 s'il est divisible à la fois par 2 (pour le n initiale) ET par 3.

Pascal :

```
program ex;  
uses wincrt;  
var  
n:integer;
```

```
{***** les modules *****}
```

```
function div_6(n:integer):boolean;  
var  
ch:string;  
test:boolean;  
x,y,e,i:integer;  
begin
```

```
  x:=n;
```

```
  repeat  
    str(n,ch);  
    n:=0;  
    for i:=1 to length(ch)do  
      begin  
        val(ch[i],y,e);  
        n:=n+y;  
      end;
```

```
  until n in [0..9];
```

```
  str(x, ch);
```

```
  if (n in[3,6,9]) and ((ch[length(ch)]) in ['0','2','4','6','8'])then test:=true  
  else test:=false;
```

```
  div_6:=test;
```

```
end;
```

```
{***** PP *****}
```

```
begin  
writeln('donner un nombre n ');  
readln(n);  
writeln(div_6(n));  
end.
```

6) Par 7

Un entier n est divisible par 7 si et seulement si le nombre obtenu en retranchant le double de son chiffre des unités au nombre formé des autres chiffres est divisible par 7

Exemple :1

$N = 2034$

$203 - (4 \times 2) = 195$; $19 - (5 \times 2) = 9$ donc 2034 n'est pas divisible par 7

Exemple :2

17976

$1797 - 12 = 1785$. $178 - 10 = 168$. $16 - 16 = 0$. 17'976 est bien divisible par 7. pcq on obtient un seul chiffre qui est égale à 0 ou 7

Def fn div_7(n :entier) :booleén
Résultat = div_7 $\leftarrow$ test

Repéter
Convch(n, ch)
Valeur(ch[long(ch)], u, e)
Efface(ch, long(ch), 1)
Valeur(ch, x, e)
 $N \leftarrow \text{abs}(x - (2 \times u))$
Jusqu'à $n < 10$

Si n dans $[0, 7]$ alors test $\leftarrow$ vrai
Sinon test $\leftarrow$ faux
Fin si
Fin fn div_7

7) Par 9

Un entier n est divisible par 9 si et seulement si la somme de ses chiffres jusqu'à obtenir un entier d'un seul chiffres doit égale à 9

Def fn div_9 (n :entier) :booleén
Résultat = div_9 $\leftarrow$ test

Si $n=9$ alors test $\leftarrow$ vrai
Sinon test $\leftarrow$ faux
Fin si

Repeter
Convch(n, ch)
 $n \leftarrow 0$
Pour i de 1 à long(ch) faire
Valeur(ch[i], x, e)
 $n \leftarrow n + x$
Fin pour
Jusqu'à $n < 10$

TDOL

Nom	Type
Test	Booleén
Ch	Chaîne

8) Par 10

Un entier a est divisible par 10 si et seulement si son chiffre des unités est 0.

Def fn div_10 (n :: entier) : booleén
 Résultat = div_10 ← test

Si ch[long(ch)] = "0" alors test ← vrai
 Sinon test ← faux
 Fin si

Convch(n, ch)
 Fin fn div_10

TDOL

Nom	Type
Test	Booleén
Ch	Chaîne

9) Par 11

Un entier a est divisible par 11 si et seulement si la différence entre la somme de ses chiffres de rangs pairs et la somme de ses chiffres de rangs impairs égale à 0 ou 11

N = 2846151

2	8	4	6	1	5	1
1	2	3	4	5	6	7

Somme des indices paires = 8+6+5=19

Somme des indices impaires = 2+4+1+1=8

Abs(19-8) = 11

Def fn div11(n : entier long) : booleén
 Résultat = div1 ← test

S1 ← 0

S2 ← 0

Convch(n, ch)

Pour l de 1 à long(ch) faire

Valeur(ch[l], x, e)

Si l mod 2 = 0 alors s2 ← s2 + x

Sinon s1 ← s1 + x

Fin si

Fin pour

Si abs(s1-s2) dans [0, 11] alors test ← vrai
Sinon test ← faux
Fin si
Fin fn div11

TDOL

Nom	Type
Ch	Chaîne
S1, s2, i	Entier
X, e	Entier

10) Par 13

Un entier n est divisible par 13 si et seulement si le nombre obtenu en additionnant le quadruple de son chiffre des unités au nombre formé des autres chiffres, on refait le même travail jusqu'on obtient un nombre ≤ 39 , si le nombre obtenu dans [13, 26, 39] donc il est divisible par 13

$$7631 = 763 + (4 \cdot 1) = 767$$

$$767 = 76 + (4 \cdot 7) = 104$$

$$104 = 10 + (4 \cdot 4) = 26 \text{ donc divisible par 13}$$

Def fn div_13(n :entier long) ;booleén
Résultat = div 13 ← test

Repete
Convch(n, ch)
Valeur(ch[long(ch)], u, e)
Efface(ch, long(ch), 1)
Valeur(ch, x, e)
 $n \leftarrow x + (4 \cdot u)$
Jusqu'à $n \leq 39$

Si n dans [13, 26, 39] alors test ← vrai
Sinon test ← faux
Fin si
Fin fn div_13

Chapitre 4 : Les algorithmes de tri

I. Tri fusion

1) Tri fusion : les entiers, les réels, les caractères et sur les chaînes par ordre alphabétique

Analyse de la procédure tri_fusion

Def proc tri_fusion(t1, t2 :tab ; n1, n2 :entier ; var t :tab ; var n :entier)

Résultat = t, n

I ← 1

J ← 1

Répéter

 n ← n + 1

 si t1[i] < t2[j] alors t[n] ← T1[i]
 i ← i + 1

 sinon

 t[n] ← T2[j]
 j ← j + 1

 fin si

Jusqu'à (i > n1) et (j > n2)

Si i > n1 alors

 Pour c de j à n2 faire

 N ← N + 1

 T[N] ← T2[c]

 Fin pour

Sinon

 Pour c de j à n1 faire

 N ← N + 1

T[N] ← T1[c]
Fin pour

Fin si

TDOL

Nom	Type
I, j, c	Entier

2) Tri fusion : suivant la longueur de la chaîne

Analyse de la procédure tri fusion

Def proc tri_fusion(t1, t2 :tab ; n1, n2 :entier ; var t :tab ; var n :entier)

Résultat = t, n

I ← 1

J ← 1

Répéter

n ← n + 1

si long(t1[i]) < long(t2[j]) alors t[n] ← T1[i]
i ← i + 1

sinon

t[n] ← T2[j]

j ← j + 1

fin si

Jusqu'à (i > n1) et (j > n2)

Si i > n1 alors

Pour c de j à n2 faire

N ← N + 1

T[N] ← T2[c]

Fin pour

Sinon

Pour c de j à n1 faire

N ← N + 1

T[N] ← T1[c]

Fin pour

Fin si

TDOL

Nom	Type
I, j, c	Entier

Application

Trier un tableau contenant n élèves sachant qu'un élève est caractérisé par un nom et une moyenne, trier le tableau suivant le 3^{ème} caractère de son nom

Def proc tri_fusion(t1, t2 :tab ; n1, n2 :entier ; var t :tab ; var n :entier)

Résultat = t, n

I ← 1

J ← 1

Répéter

n ← n + 1

si t1[i].nom[3] < t2[j].nom[3] alors t[n] ← T1[i]

i ← i+1

sinon

t[n] ← T2[j]

j ← j+1

fin si

Jusqu'à (i > n1) et (j > n2)

Si i > n1 alors

Pour c de j à n2 faire

N ← N+1

T[N] ← T2[c]

Fin pour

Sinon

Pour c de j à n1 faire

N ← N+1

T[N] ← T1[c]

Fin pour

Fin si

TDOL

Nom	Type
I, j, c	Entier

3) Tri fusion : sur les enregistrement

Analyse de la procédure tri fusion

Def proc tri_fusion(t1, t2 :tab ; n1, n2 :entier ; var t :tab ; var n :entier)

Résultat = t, n

I ← 1

J ← 1

Répéter

n ← n + 1

si t1[i].champs(t2[j].champs alors T[n] ← T1[i]

i ← i+1

sinon

t[n] ← T2[j]

j ← j+1

fin si

Jusqu'à (i > n1) et (j > n2)

Si i > n1 alors

Pour c de j à n2 faire

```

        N ← N+1
        T[N] ← T2[c]
    Fin pour
Sinon
    Pour c de j à n1 faire
        N ← N+1
        T[N] ← T1[c]
    Fin pour
Fin si

```

TDOL

Nom	Type
l, j, c	Entier

II. Tri shell

1) Tri shell sur les types simple (entier, réel, caractère, chaîne)

```

Def proc tri_shell( var t : tab ; n :entier)
Résultat = t

```

```

P ← 0
Tant que p < n faire
    P ← (3*P)+1
Fin tant que

```

Calcul de pas

```

Tant que p < > 0 faire
    P ← P div 3
    Pour i de p+1 à n faire
        Aux ← t[i]
        J ← i - p
        Tant que (aux < t[j]) et (j > 0) faire
            T[j+p] ← t[j]
            J ← j - p
        Fin tant que

```

```

        T[j+p] ← aux
    Fin pour
Fin tant que
Fin proc tri_shell

```

TDOL

Nom	Type
Aux	De même type que les case du tableau
l, j	Entier
Perm	Booleén

2) Tri shell suivant la longueur de la chaîne

```

Def proc tri_shell( var t : tab ; n :entier)
Résultat = t
P ← 0
Tant que p < n faire
    P ← (3*P)+1

```

Fin tant que

Tant que $p \leq 0$ faire

$P \leftarrow P \div 3$

Pour i de p+1 à n faire

$Aux \leftarrow t[i]$

$J \leftarrow i - p$

Tant que ($long(aux) < long(t[j])$) et ($j > 0$) faire

$T[j+p] \leftarrow t[j]$

$J \leftarrow j - p$

Fin tant que

$T[j+p] \leftarrow aux$

Fin pour

Fin tant que

Fin proc tri shell

Fin proc tri shell

TDOL

Nom	Type
Aux	Chaîne
I	Entier
Perm	Booleén

3) Tri shell sur les enregistrements

Def proc tri_shell(var t : tab ; n : entier)

Résultat = t

$P \leftarrow 0$

Tant que $p < n$ faire

$P \leftarrow (3 * P) + 1$

Fin tant que

Tant que $p \leq 0$ faire

$P \leftarrow P \div 3$

Pour i de p+1 à n faire

$Aux \leftarrow t[i]$

$J \leftarrow i - p$

Tant que ($aux.champs < t[j].champs$) et ($j > 0$) faire

$T[j+p] \leftarrow t[j]$

$J \leftarrow j - p$

Fin tant que

$T[j+p] \leftarrow aux$

Fin pour

Fin tant que

Fin proc tri shell

TDOL

Nom	Type
Aux	Enregistrement
I	Entier
Perm	Booleén

4) Tri shell sur les fichiers

Enchaînement :

- transfert du fichier au tableau
- tri tableau
- transfert tableau au fichier

Analyse de la procédure transfer f t

Def proc transfer_f_t(var f :fich ; var t :tab ; varv n :entier)

Résultat = t, n

N ← 0

Ouvrir(f)

Tant que non (fin_fichier(f))

 n ← n+1

 lire(f, t[n])

fin tant que

fin proc transfer_f_t

Analyse de la procédure tri sur un tableau

Def proc tri_shell(var t : tab ; n :entier)

Résultat = t

Pour i de 2 à n faire

 Aux ← t[i]

 J ← i-1

 Tant que (aux.champs < t[j].champs) et (j > 0) faire

 T[j+1] ← t[j]

 J ← j-1

 Fin tant que

 T[j+1] ← aux P ← 0

Tant que p < n faire

 P ← (3*P)+1

Fin tant que

Tant que p <= 0 faire

 P ← P div 3

Pour i de p+1 à n faire

 Aux ← t[i]

 J ← i-p

 Tant que (aux.champs < t[j].champs) et (j > 0) faire

 T[j+p] ← t[j]

 J ← j-p

 Fin tant que

 T[j+p] ← aux

Fin pour

Fin tant que

Fin proc tri shell

TDOL

Nom	Type
Aux	De même type que les case du tableau
i	Entier
Perm	Booleén

Analyse de la procédure transfer t f

```

Def proc transfer_t_f (t :tab ; n :entier ; var f :fich)
Résultat =f
Recreer(f)
Pour i de 1 à n faire
  Ecrire (f, t[i])
Fin pour
Fin proc transfer_t_f

```

TDOL

Nom	Type
i	Entier

Pascal :

```

program ex;
uses wincrt;

```

```

type
fich= file of integer;
tab = array[1..20] of integer;
var
f:fich;

```

```

{***** les modules *****}

```

```

procedure tri_shell(var t:tab ; n:integer);
var
i, j, p:integer;
aux:integer;
begin

```

```

  p:=0;
  while p<n do
  begin
    p:=3*p+1;
  end;

```

```

  while(p<>0) do
  begin
    p:=p div 3;

```

```

  for i:=p+1 to n do
  begin
    aux:=t[i];
    j:=i-p;

```

```

    while (aux<t[j]) and (j>0)do

```

```

begin
    t[j+p]:=t[j];
    j:=j-p;
end;

    t[j+p]:=aux;
end;
end;
end;

```

```

procedure saisi(var f:fich);
var
    i, x, n:integer;
begin
    writeln('donner n ');
    readln(n);

    reset(f);
    for i:=1 to n do
        begin
            writeln('donner x ');
            readln(x);
            write(f, x);
        end;
    end;

```

```

    close(f);
end;

procedure affich(var f:fich);
var
    x:integer;
begin
    reset(f);
    while(not(eof(f))) do
        begin
            read(f, x);
            writeln(x);
        end;
    end;
    close(f);
end;

```

```

procedure transfer_f_t(var f:fich; var t:tab ; var n:integer);
begin
    reset(f);
    n:=0;
    while(not(eof(f))) do
        begin
            n:=n+1;
            read(f, t[n]);
        end;
    end;

    close(f);
end;

```

```

procedure transfer_t_f(t:tab ; n:integer; var f:fich);
var
  i:integer;
begin
  rewrite(f);
  for i:=1 to n do
  begin
    write(f,t[i]);
  end;

  close(f);
end;

```

```

procedure tri(var f:fich);
var
  n:integer;
  t:tab;
begin
  transfer_f_t(f, t, n);
  tri_shell(t, n);
  transfer_t_f(t, n, f);
end;

```

```

{***** PP *****)

```

```

begin
  assign(f, 'd:\bac 2015\tri.dat');
  rewrite(f);

  saisi(f);
  tri(f);
  affich(f);
end.

```

Pascal avec les enregistrements :

```

program ex;
uses wincrt;

type
  pers= record
    nom:string[20];
    age:integer;
  end;

  tab = array[1..20] of pers;

var
  t:tab;
  n:integer;

```

```

{***** les modules *****)

```

```

procedure tri_inser(var t:tab ; n:integer);

```

```

var
  i, j, p:integer;
  aux:pers;
begin

  p:=0;
  while p<n do
  begin
    p:=(3*p)+1;
  end;

  while p<>0 do
  begin
    p:=p div 3;
    for i:=p+1 to n do
    begin
      aux:=t[i];
      j:=i-p;
      while (aux.age<t[j].age) and (j>0) do
      begin
        t[j+p]:=t[j];
        j:=j-p;
      end;

      t[j+p]:=aux;
    end;
  end;
end;

```

```

procedure saisi( var n:integer; var t:tab );
var
  i:integer;
begin
  repeat
    writeln('donner n ');
    readln(n);
  until n>0;

  for i:=1 to n do
  begin
    writeln('donner t[',i,']');
    with t[i] do
    begin
      readln(nom);
      readln(age);
    end;
  end;
end;
procedure affich(t:tab ; n:integer);
var
  i:integer;
begin
  for i:=1 to n do

```

```

begin
    write(t[i].nom, ' ', t[i].age, ' ');
end;

writeln;
end;

{***** PP *****)
begin
    saisi(n, t);
    affich(t, n);
    tri_inser(t, n);
    affich(t, n);
end.

```

Chapitre 5 : Les algorithmes récurrents

I. calcul sur les suites récurrentes :

1) 1ère ordre (Calculer les n premier terme de la suite avec n donnée) :

Programme Principale nous avons l'initialisation :
 $U \leftarrow \text{valeur}$

```

Def fn suite_1(u : type ; n : entier) : type
Résultat = suite_1
Up ← U
Si n= indice alors suite ← U
sinon
    U ← formule
    Suite ← Suite(u, n-1)
Fin si

```

2) 2ème ordre (Calculer les n premier terme de la suite avec n donnée) :

Programme Principale nous avons l'initialisation :

```

Up ← valeur1
U ← valeur 2

```

```

Def fn suite_1(U,Up,n :entier) : type
Résultat = suite_1
Upp ← Up
Up ← U
Si n= indice alors suite ← U
sinon
    U ← formule
    Suite ← Suite(u, up, n-1)
Fin si

```

3) 1ère ordre (Calculer les termes de la suite jusqu'à la différence entre deux termes consécutif est $< 10^{-4}$) :

Programme Principale nous avons l'initialisation :

```

U ← valeur

```

```

Def fn suite_1(U : type) : type
Résultat = suite_1
Up ← U
U ← formule
Si abs(U-Up) ≤  $10^{-4}$  alors suite ← U
sinon
    Suite ← Suite(u)
Fin si

```

4) 2ème ordre (Calculer les termes de la suite jusqu'à la différence entre deux termes consécutif est $< 10^{-4}$) :

Programme Principale nous avons l'initialisation :

```

Up ← valeur
U ← valeur

```

```

Def fn suite_1(U,Up : type) : type
Résultat = suite_1
Upp ← Up
Up ← u
U ← formule
Si abs(1/1-1/n) ≤  $10^{-4}$  alors suite ← 1/1

```

sinon

Suite ← Suite(u, up)

Fin si

Série N°6

Exercice n°1 :

Ecrivez un programme permettant de calculer les n premiers termes de la suite de Héron définie par :

$$U_0 = x$$

$$U_1 = x + 1$$

$$U_n = (U_{n-1} / 2) + (x / 2 * U_{n-2})$$

Avec x un réel positif donnée.

Analyse principale (suite de 2ème ordre):

Résultat = proc suite (x, n)

Proc saisi (x, n)

Fin ex

TDO

Nom	Type
X	Réel
N	Entier
Suite, saisi	Procédure

Analyse de la procédure suite

```
Def proc suite(x :réel ; n :entier)
Résultat = []
[Up ← x, U ← x+1]
pour i de 2 à n faire
    Upp ← Up
    Up ← U
    U ← (Up/2) + (x/2Upp)
    Ecrire(U)
Fin pour
Fin proc suite
```

TDOL

Nom	Type
U, Up, Upp	Réel
I	Entier

Analyse de la procédure suite (récursive)

```
Def proc suite(U, Up :réel ; x :réel ; n :entier)
Résultat = []
Si n= 1 alors écrire(U)
Sinon
    Upp ← Up
    Up ← U
    U ← (Up/2) + (x/2Upp)
    Ecrire(U)
    Pro suite(U, Up , x, n-1)
```

Analyse de la procédure saisi

```
Def proc saisi (var x :réel ; var n :entier)
Résultat = x, n
X= répéter
    X= donnée
Jusqu'à x>0
N= répéter
    N = donnée
Jusqu'à n>0
Fin proc saisi
```

Exercice n°2 :

Soient deux suites U et V définies par :
 $U_1 = 1$ et $U_2 = 2$

$$U_n = U_{n-1} + U_{n-2} \text{ pour } n \geq 3$$

$$V_n = U_n / U_{n-1} \text{ pour } n \geq 2$$

La suite V_n tend vers une limite appelé nombre d'or.

On suppose que le n ème terme de la suite V soit V_n . Dont un nombre approché du nombre d'or avec une précision ϵ selon des que $|V_n - V_{n-1}| < \epsilon$ selon

$\epsilon = 10^{-4}$.

Correction :

```

Def proc suite ;
Résultat = écrire(V)
[U ← 2, Up ← 1, V ← U/Up ]
Répéter
    Upp ← Up
    Up ← U
    Vp ← V
    U ← Up+Upp
    V ← U/Up
Jusqu'à abs (V-Vp) ≤ 10e-4
Fin proc suite
    
```

TDOL

Nom	Type
Upp, U, Up	Entier
V, Vp	Réel

Correction (récursive) :

```

Def fn suite (Up, U, V :réel) : réel
Résultat = suite

    Upp ← Up
    Up ← U
    Vp ← V
    U ← Up+Upp
    V ← U/Up
Si abs (V-Vp) ≤ 10e-4 alors suite ← V
Sinon suite ← fn suite(Up, U, V)
Fin si
Fin fn suite
    
```

Exercice n°3 :

Soit la suite de thue morse définie par :

```

U0 = "0"
U1 = "01"
U2 = "0110"
U3 = "01101001"
U4 = "0110100110010110"
    
```

Saisir un entier $n > 0$ et saisir un caractère C qui est égale à '0' ou '1' et calculer les différentes termes de la suite de thue morse.

Pascal :

```
program ex;  
uses wincrt;
```

```
var  
u:string;  
n:integer;
```

```
{***** Les Modules *****}
```

```
procedure saisi(var n:integer; var u:string);  
begin
```

```
    repeat  
        writeln('donner n ');  
        readln(n);  
    until n>0;
```

```
    repeat  
        writeln('donner u ');  
        readln(u);  
    until (u='0') or (u='1');
```

```
end;
```

```
procedure suite(n:integer; c:string);
```

```
var  
    i, j:integer;  
    up, u:string;
```

```
begin
```

```
    U:=c;
```

```
    for i:=1 to n do
```

```
    begin
```

```
        Up:=U;
```

```
        for j:=1 to length(Up) do
```

```
        begin
```

```
            if up[j]='0' then U:=Up+'1'
```

```
            else U:=Up+'0';
```

```
        end;
```

```
        writeln('U',i,' = ',u);
```

```
    end;
```

```
end;
```

```
{***** PP *****}
```

```
begin
```

```
saisi(n, u);
```

```
suite(n, u);
```

```
end.
```

Exercice n°4 :

Ecrivez un programme permettant de calculer le terme n , sachant que le calcul s'arrête lorsque la différence entre deux termes consécutif $\leq 10^{-4}$.

$$U_0 = x$$

$$U_n = U_{n-1} - (U_{n-1}/2)$$

Avec x un réel positif donnée.

Analyse principale (suite de 2^{ème} ordre):

Résultat = écrire(fn suite(x))
Proc saisi (x, n)

Fin ex

TDO

Nom	Type
X	Réel
N	Entier
Saisi	Procédure
Suite	Fonction

Analyse de la fonction suite

Def fn suite(x :réel) :réel

Résultat = suite $\leftarrow$ U

[U $\leftarrow$ x]

répéter

Up $\leftarrow$ U

U $\leftarrow$ Up - (Up/2)

Jusqu'à $\text{abs}(U - \text{Up}) \leq 10^{-4}$

Fin fn suite

TDOL

Nom	Type
U, Up	Réel

Analyse de la procédure saisi

Def proc saisi (var x :réel)

Résultat = x, n

X= répéter

X= donnée

Jusqu'à $x > 0$

Fin proc saisi

Analyse de la fonction récursive :

Def fn suite(U, x :réel) :réel

Résultat = suite

Up $\leftarrow$ U

Si $\text{abs}(U - \text{Up}) \leq 10^{-4}$ alors suite $\leftarrow$ U

Sinon

$U \leftarrow U_p - (U_p/2)$

Suite(U, x)

Fin si

Fin fn suite

Pascal :

```
program ex;  
uses wincrt;
```

```
var  
x:real;
```

```
{***** Les Modules *****}
```

```
procedure saisi (var x:real);  
begin  
    repeat  
        writeln('donne x');  
        readln(x);  
    until x>0;  
end;
```

```
{***** PP *****}
```

```
function suite(x:real):real;  
var  
U, Up:real;  
begin  
    U:=x;
```

```
    repeat  
        Up:=U;  
        U:=Up-(Up/2);  
    until abs(U-Up)<=1e-4;
```

```
    suite:=U;  
end;
```

```
begin  
    saisi(x);  
    writeln('la suite est ',suite(x):0:6);  
end.
```

Exercice n°5 :

Ecrivez un programme permettant de calculer le terme n , sachant que le calcul s'arrête lorsque la différence entre deux termes consécutif $\leq 10^{-4}$.

$U_0 = x$

$U_1 = x/2$

$U_n = U_{n-1} - (U_{n-2}/2)$

Avec x un réel positif donnée.

Analyse principale (suite de 1er ordre):

Résultat = écrire(fn suite(x))
Proc saisi (x, n)
Fin ex

TDO

Nom	Type
X	Réel
N	Entier
Saisi	Procédure
Suite	Fonction

Analyse de la fonction suite

Def fn suite(x :réel) :réel
Résultat = suite $\leftarrow$ U
[Up $\leftarrow$ x ; U $\leftarrow$ x/2]
répéter
 Upp $\leftarrow$ Up
 Up $\leftarrow$ U
 U $\leftarrow$ Up -(Upp/2)
Jusqu'à abs(U-Up) $\leq 10^{-4}$
Fin fn suite

TDOL

Nom	Type
U, Up, Upp	Réel

Analyse de la procédure saisi

Def proc saisi (var x :réel)
Résultat = x, n
X= répéter
 X= donnée
 Jusqu'à x > 0
Fin proc saisi

Exercice n°6 :

Sachant que $\pi/4 = \frac{1}{1} - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \dots$

Ecrire un programme Pascal qui permet d'afficher $\pi/4$ en utilisant la formule ci-dessus.

Le calcul s'arrête quand la différence entre deux termes consécutifs devient inférieure ou égale à 10^{-4} . La dernière somme calculée est une valeur approchée de $\pi/4$.

Correction :

Def fn suite : réel
Résultat = suite $\leftarrow$ S

$[s \leftarrow 1, \text{sig} \leftarrow -1, i \leftarrow 1]$
 Répéter
 $i \leftarrow i+2$
 $\text{Sig} \leftarrow -\text{sig}$
 $\text{Sp} \leftarrow S$
 $S \leftarrow \text{sp} + (\text{sp} + (1/i) * \text{sig})$
 Jusqu'à $\text{abs}(s - \text{sp}) < 1e-4$
 Fin fn suite

TDOL

Nom	Type
S, sp	Reel
Sig	Entier
i	Entier

Correction (recursive) :

Def fn suite(S :rée ; sig : entier ; i :entier) : réel
 Résultat = suite
 $i \leftarrow i+2$
 $\text{Sig} \leftarrow -\text{sig}$
 $\text{Sp} \leftarrow S$
 $S \leftarrow \text{sp} + (\text{sp} + (1/i) * \text{sig})$
 Si $\text{abs}(s - \text{sp}) < 1e-4$ alors suite $\leftarrow s$
 Sinon suite $\leftarrow \text{fn suite}(s, \text{sig}, i)$
 Fin si
 Fin fn suite

II. calcul matricielles :

Série N°7

Exercice n°1 :

Déduire un module permettant de déterminer le minimum dans une matrice supposée carrée.

Correction :

```
Def fn min(m :mat ; n :entier) :entier
Résultat = min ← mini
Mini ← m[1, 1]
Pour L de 1 à N faire
    Pour C de 1 à n faire
        Si m[L, C] < mini alors mini ← M[L, C]
    Fin si
Fin pour
Fin pour
Fin fn min
```

TDOL

Nom	Type
L, C	Entier

Exercice n°2 :

Déduire un module permettant d'afficher le minimum de chaque ligne .

14	15	2	3	2
7	10	12	17	7
6	-4	2	8	-4
3	4	1	2	1

Correction :

```

Def proc min_ligne(m :mat ; n :entier)
Résultat = [ ]
Pour L de 1 à N faire
  Mini ← m[L, 1]
  Pour c de 1 à n faire
    Si M[L, C]<mini alors mini ← M[L, C]
  Fin si
Fin pour
Ecrire(mini)
Fin pour
Fin proc min_ligne

```

TDOL

Nom	Type
Mini, L, C	Entier

Exercice n°3 :

Déduire un module permettant d'afficher le minimum de chaque colonne .

14	15	2	3
7	10	12	17
6	-4	2	8
3	4	1	2

3 - 4 1 2

Correction :

```

Def proc min_col(m :mat ; n :entier)
Résultat = [ ]
Pour C de 1 à N faire
  Mini ← m[1, C]
  Pour L de 1 à n faire
    Si M[L, C]<mini alors mini ← M[L, C]
  Fin si
Fin pour
Ecrire(mini)
Fin pour
Fin proc min_ligne

```

TDOL

Nom	Type
Mini, L, C	Entier

Exercice n°4 :

Déduire un module permettant de calculer la somme du 1^{er} diagonal

14	15	2	3
7	10	12	17
6	-4	2	8
3	4	1	2

Correction :

```

Def fn somme_1diag(m :mat ; n :entier) : entier

```

Résultat = somme_1diag ← s

S ← 0

Pour i de 1 à n faire

 S ← s+m[i, i]

Fin pour

Fin fn somme

TDOL

Nom	Type
S, i	Entier

Exercice n°5 :

Déduire un module permettant de calculer la somme du 2^{ème} diagonal

14	15	2	3
7	10	12	17
6	-4	2	8
3	4	1	2

Correction :

Def fn somme_2diag(m :mat ; n :entier) : entier

Résultat = somme_2diag ← s

S ← 0

Pour i de 1 à n faire

 S ← s+m[i, n- i+1]

Fin pour

Fin fn somme

TDOL

Nom	Type
S, i	Entier

Exercice n°6 :

Ecrire un module permettant de vérifier si la matrice est symétrique par rapport à la deuxième diagonale ou non (la matricé est supposé carrée)

14	12	10	1	2
18	19	5	1	1
20	21	10	5	10
22	20	21	19	12
30	22	20	18	14

Correction :

Def fn sym_2(m:mat ; n:entier):booleén

Résultat = sym_2 ← test

Test ← vrai

```

Pour L de 1 à N faire
  Pour C de 1 à n faire
    Si  $M[L, C] \neq M[N-C+1, N-L+1]$  alors test ← faux
  Fin si
Fin pour
Fin pour
Fin fn sym_2

```

TDOL

Nom	Type
L, C	Entier
Test	Booléen

Exercice n°7 :

Ecrire un module permettant de vérifier si la matrice est symétrique par rapport à la première diagonale ou non (on suppose que la matrice est carrée)

14	12	10	10	22
12	19	5	1	1
10	5	10	5	10
10	1	5	19	18
22	22	10	18	14

Correction :

```

Def fn sym_1diag(m : mat ; n : entier) : booléen
  Résultat = sym_1diag ← test
  Test ← vrai
  Pour L de 1 à N faire
    Pour C de 1 à N faire
      Si  $M[L, C] \neq M[C, L]$  alors test ← faux
    Fin si
  Fin pour
Fin fn sym_1diag

```

TDOL

Nom	Type
Test	Booléen
L, C	Entier

Exercice n°8 :

Ecrire des modules permettant de trier par ordre croissant chaque ligne d'une matrice.

1^{er} module : permet de trier une seule ligne (NL) d'une matrice

2^{ème} module : je fais appel au 1^{er} module pour l'appliquer aux différents lignes de la matrice.

1^{er} module :

```
Def proc tri_ligne(NL :entier ; var m :mat ; n :entier)
Résultat = m
Répéter
Perm ← faux
Pour i de 1 à n faire
    Si M[NL, i]>M[NL, i+1] alors
        Perm ← vrai
        Aux ← M[NL, i]
        M[NL, i] ← M[NL, i+1]
        M[NL, i+1] ← aux
    Fin si
Fin pour
Jusqu'à perm=faux
```

2^{ème} module :

```
Def proc tri_mat ( var m :mat ; n :entier)
Résultat = m
Pour L de 1 à n faire
    Proc tri_ligne(L, m, n)
Fin pour
Fin proc tri _mat
```

Exercice n°9 :

Ecrire des modules permettant de trier par ordre croissant chaque colonne d'une matrice.

1^{er} module : permet de trier une seul colonne (NC) d'une matrice

2^{ème} module : je fais appel au 1^{er} module pour l'appliquer aux différents colonne de la matrice.

1^{er} module :

```
Def proc tri_col(NC :entier ; var m :mat ; n :entier)
Résultat = m
Répéter
Perm ← faux
Pour i de 1 à n faire
    Si M[i, NC]>M[i+1, NC] alors
        Perm ← vrai
        Aux ← M[i, NC]
        M[i, NC] ← M[i+1, NC]
        M[i+1, NC] ← aux
    Fin si
Fin pour
Jusqu'à perm=faux
```

2^{ème} module :

```
Def proc tri_mat ( var m :mat ; n :entier)
Résultat = m
Pour C de 1 à n faire
  Proc tri_ligne(C, m, n)
Fin pour
Fin proc tri_mat
```

ZIED CHAOUCH

Chapitre 6 : Les algorithmes approximation

I. Calcul des aires

1) Principe :

Si nous voulons calculer l'aire d'une fonction dans un intervalle $[a,b]$, on peut utiliser l'une des deux méthodes suivantes : (méthode de rectangle ou méthode de trapèze).

$$F(x) = 1/(x^2+1)$$

2) Méthode de rectangle :

0) def Fn F(x :réel) :réel

1) $F \leftarrow 1/(\text{carré}(x) + 1)$

2) Fin Fn F

Algorithme de la fonction aire rectangle :

0) def Fn aire_rectangle (a, b : réel ; n : entier) : réel

1) $\text{pas} \leftarrow (b-a)/n$

$x \leftarrow a$

$s \leftarrow 0$

pour i de 1 à n faire

$S \leftarrow s + (\text{pas} * \text{fn } f(x))$

$x \leftarrow x + \text{pas}$

Fin pour

2) Aire_rectangle $\leftarrow s$

3) fin Fn rectangle

En pascal:

```
program ex;
uses winCRT;
var
a,b:real;
n:integer;
```

```
{***** focntion F *****}
```

```
function f(x:real):real;
begin
f:=sqr(x)-1;
end;
```

```
{***** fonction rectangle *****}
```

```
function rectangle (a,b:real; n:integer):real;  
var  
pas,s,x:real;  
i:integer;  
begin
```

```
pas:=(b-a)/n;
```

```
s:=0;  
x:=a;
```

```
for i:=1 to n do  
begin  
s:=s+(pas*f(x));  
x:=pas+x;  
end;
```

```
rectangle:=s;  
end;
```

```
{***** PP *****}
```

```
begin  
writeln('donner le nombre de partion ');  
readln(n);
```

```
writeln('donner les bornes a et b');  
readln(a,b);
```

```
writeln('aire= ',rectangle(a,b,n):0:3);  
end.
```

3) Méthode de trapèze :

- 0) def Fn F(x :réel) :réel
- 1) $F \leftarrow \text{carré}(x)-1$
- 2) Fin Fn F

Algorithme de la fonction tarpéze :

- 0) def Fn trapèze (a, b : réel ; n : entier) : réel
- 1) $\text{pas} \leftarrow (b-a)/n$
- 2) $[x \leftarrow a, s \leftarrow 0]$ pour i de 1 à n faire
 - $S \leftarrow s + (((\text{fn } f(x) + \text{fn } f(x+\text{pas})) * \text{pas}) / 2$
 - $X \leftarrow x + \text{pas}$Fin pour
- 3) trapèze $\leftarrow s$
- 4) fin Fn trapèze

En pascal :

```
program ex;  
uses wincrt;  
var  
a,b:real;  
n:integer;
```

```
{***** focntion F *****}
```

```
function f(x:real):real;  
begin  
f:=sqr(x)-1;  
end;
```

```
{***** fonction trapeze *****}
```

```
function trapeze (a,b:real; n:integer):real;  
var  
pas,s,x:real;  
i:integer;  
begin
```

```
pas:=(b-a)/n;
```

```
s:=0;  
x:=a;
```

```
for i:=1 to n do  
begin  
s:=s+(((f(x)+f(x+pas))*pas)/2);  
x:=pas+x;  
end;
```

```
trapeze:=s;  
end;
```

```
{***** PP *****}
```

```
begin  
writeln('donner le nombre de partion ');  
readln(n);
```

```
writeln('donner les bornes a et b');  
readln(a,b);
```

```
writeln('aire= ',trapeze(a,b,n):0:3);  
end.
```

II. Point fixe

Généralement on aura la formule de la fonction $f(x)$, puis il nous demande de calculer le point fixe :

S'il nous donne $f(x) = ? \rightarrow f \leftarrow ?$
On va chercher le point fixe qui est X .

La méthode à apprendre :

- fonction f
- fonction pt_fixe (toujours la même)

Exemple :

$F(x) = 1 - \sin(x)$

Analyse de la fonction f:

```
def Fn F ( x :réel) :réel
résultat = F ← 1-sin(x)
fin Fn F
```

Algorithme de la fonction f :

- 0) def Fn F (x :réel) :réel
- 1) $F \leftarrow 1 - \sin(x)$
- 2) Fin Fn F

Analyse de point fixe A apprendre :

```
def Fn pt_fixe :réel
Résultat= pt_fixe ← X
[x ← 0]
  Répéter
    Xp ← x
    X ← fn f(xp)
  Jusqu'à abs(X-Xp) <= 10-4
Fin Fn pt_fixe
```

TDOL

Nom	Type
Xp	Réel
X	Réel
F	Fonction

Algorithme de point fixe :

```
0) def Fn pt_fixe(eps :réel) :réel
1) [x ← 0]
  Répéter
    Xav ← x
    X ← f(Xav)
  Jusqu'à abs(X-Xav) <= eps
```

- 3) pt_fixe ← X
- 4) Fin Fn pt_fixe

Remarque :

-Généralement $\epsilon = 10^{-6}$

Pascal : avec calcul de nombre d'itération :

```

program ex;
uses wincrt;

const
eps=1e-6;

{***** fonction F *****}

function f(x:real):real;
begin
  f:=1-sin(x);
end;

{***** procédure point fixe *****}

procedure pt_fix(eps:real);
var
  x,Xav:real;
  i:integer;
begin
  x:=0;

  i:=0;
  repeat
    i:=i+1;
    Xav:=X;
    X:=f(Xav);
  until abs(x-xav)<=eps;

  writeln(' point fixe est ',x:0:3, ' nombre de itération est ',i);
end;

{***** PP *****}
begin
  pt_fix(eps);
end.

```

III. valeur approché

Exercice n°1

Sachant que $\sin(x) = \frac{x}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \dots$

pour x très proche de zéro.

Ecrire un programme Pascal qui permet d'afficher $\sin(x)$ en utilisant la formule ci-dessus.

Le calcul s'arrête quand la différence entre deux termes consécutifs devient inférieure ou égale à 10^{-4} . La dernière somme calculée est une valeur approchée de $\sin(x)$.

Analyse principale :

Résultat = écrire(calcul(x))

X = proc saisi(x)

Fin PP

TDOG

Nom	Type
Calcul	Fn
Saisi	Proc
X	Réel

Analyse de la fonction calcul

Def fn calcul (x :réel) :réel

Résultat = calcul $\leftarrow$ s

S $\leftarrow$ x

Sig $\leftarrow$ 1

i $\leftarrow$ 1

Répéter

 I $\leftarrow$ i+2

 Sp $\leftarrow$ S

 Sig $\leftarrow$ -sig

 S $\leftarrow$ Sp + (puis(x, i)/fact(i))*sig

Jusqu'à $\text{abs}(sp - s) \leq 10^{-4}$

TDOL

Nom	Type
Sig, i	Entier
Sp, s	Réel
Puis, fact	Fonction

Analyse de la fonction puis

Def fn pusi (x :réel ; n :entier) :réel

Résultat = puis $\leftarrow$ p

P $\leftarrow$ 1

Pour i de 1 à n faire

 P $\leftarrow$ p*x

Fin pour

Fin fn puis

TDOL

Nom	Type
I	Entier
P	Réel

Analyse de la fonction fact

```

Def fn fact (n :entier) : entier long
Résultat = fact ← F
F ← 1
Pour i de 1 à n faire
    F ← F*i
Fin pour
Fin fn puis

```

TDOL

Nom	Type
I	Entier
P	Réel

Pascal :

```

program ex;
uses wincrt;
var
x:real;

{***** fcontion factoriel *****}

function fact(n:integer):integer;
var
f,i:integer;
begin
    f:=1;
    for i:=1 to n do
        begin
            f:=f*i;
        end;

    fact:=f;
end;
{***** focntion puissance *****}

function puis(x:real; n:integer):real;
var
p:real;
i:integer;
begin
    p:=1;

    for i:=1 to n do
        begin
            p:=p*x;
        end;
    puis:=p;
end;

{***** fonction approche *****}

function calcul(x:real):real;

```

```

var
  sp, s:real;
  i,sig:integer;

begin
  s:=0;
  i:=1;
  sig:=1;

  repeat
    Sp:=S;
    S:=Sp+(sig*(puis(x,i)/fact(i)));
    i:=i+2;
    sig:=-sig;
  until abs(S-Sp)<1e-4;

  calcul:=s;
end;

```

```
{***** PP *****}
```

```

begin
  repeat
    writeln('donner x ');
    readln(x);
  until (x>0) and (x<1);
  writeln(calcul(x):0:3);
end.

```

Série N°8

Exercice n°1 (exercice n°5 page 207) :

Chercher le point fixe la fonction $f(x) = \text{racine}(1+x)$

Traduisez et testez la solution obtenu, enregistrer le travail en pascal sous le nom pf_fixe2

Analyse principale :

Résultat = écrire(pt_fixe)

Fin ex

TDOG

Nom

Tvne

Pts_fixe	Fonction
----------	----------

Analyse de la fonction pt_fixe

```

Def fn pt_fixe : réel
Résultat = pt_fixe ← x
[x ← 0]
Répéter
  Xp ← x
  X ← f(x)
Jusqu'à abs(x-xp) <= eps
Fin fn pt_fixe

```

Nom	Type
X, xp	Réel
F	Fonction

Analyse de la fonction f

```

Def fn f(x :réel) :réel
Résultat = f ← racine(1+x)
Fin fn f

```

Exercice n°2 (exercice n°6 page 207) :

Calculer l'aire de :

$$\int_1^5 (x + 3 \sin(x)) dx$$

En utilisant la méthode de trapèze.

Pascal :

```

program ex;
uses wincrt;
var
a,b:real;
n:integer;

{***** Les Modules *****}

function f(x:real):real;
begin
  f:=x+3*sin(x);
end;

function trapeze(a, b:real; n:integer):real;
var
s,x,pas:real;
i:integer;

begin
  s:=0;
  x:=a;
  pas:=(b-a)/n;

```

```
for i:=1 to n do
begin
  s:=s+((f(x)*f(x+pas)*pas)/2);
  x:=x+pas;
end;

trapeze:=s;
end;

{***** PP *****}

begin
  writeln('donner le nombre de partition ');
  readln(n);
  writeln(trapeze(1, 5, n):0:4);
end.
```