2016

PROGRAMMER AVEC ZAYDOUN

Algorithmique et programmation

Ce livre est destiné aux élèves du 4eme année de l'enseignement secondaire SECTIONS SCIENTIFIQUES ainsi les élèves du 3eme 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 vu 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 structures simples et les structures de données

I. Les structures simples

Saisi des données

Je veux taper un entier dans la variable X

Analyse	Algorithme	Pascal
X = donnée	Ecrire (‘donner…’) Lire(x)	Writeln(‘donner … ?’) ; Readln(x) ;
Variable = donnée	Ecrire (‘variable…’) Lire(variable)	Writeln(‘donner … ?’) ; Readln(variable) ;

Affectation

Analyse - Algorithme	Pascal
C🡐 A+ B	C:= A+B;

Affichage

Analyse – algorithme	Pascal
Ecrire(x)	Writeln(x) ;
Ecrire(variable)	Writeln(varaiable) ;

II. Les structures de données

1) Entier ➔ en pascal : integer

- Le type entier il appartient [-37767..37767] appartient l’ensemble Z il contient au maximum 4 chiffres est sans virgule.

- Les opérateur autorisé : **+ - * div mod**

Remarque :
on n'a pas le droit d’utiliser ***/*** .

7 | 2
1 | 3

7 mod 2

7 div 2

2) réel ➔ en pascal : real

- Le type réel appartient l’ensemble R.

Les opérateur autorisé : **+ - * /**

Remarque :
Il est strictement interdit d'utiliser **div** et **mod** sur les réels.

Les fonction prédéfinies

A = analyse et P =Pascal	Exemple
Abs(reel) ➔réel + A – P Abs(entier)➔ entier +	Abs (-6,6)➔ 6,6
Sqr(entier) ➔ entier P Carré(entier) ➔ entier A Carrée (réel) ➔ réel Sqr(réel)➔ réel	Sqr(2) ➔ 2^2➔ 4
Round(réel)➔ entier P Arrondi(réel)➔ entier A	Round(5,71) ➔ 6 : (5..9) ➔ +1 Round(5,4999)➔ 5 Round(5) ➔ 5
Frac(réel) ➔ réel A- P	Frac(6,908) ➔0,908 Frac(4) ➔ 0,000
Trunc(réel) ➔ entier P Tronquer(réel) ➔ entier	Trunc(5,980) ➔ 5 Trunc(3) ➔ 3
Int(réel) ➔ réel P Ent (réel) ➔ réel A	Int(5.980)➔ 5,000 Int(3) ➔ 3,000
Sqrt(réel) ➔ réel P racine(entier) ➔réel A	Sqrt(2) ➔ 1,41 Sqrt(4) ➔ 2,000

3) booleen ➔ en pascal : boolean

La variable va prendre soit **true (vrai)** soit **false (faux)**

les opérateurs autorisés : **ET (and) OU (or) OUex (xor) NON (not)**

A	ET (and)	B	Résultat
T	Et	T	T
T	Et	F	F
F	Et	T	F
F	Et	F	F

Remarque :
ET : elle renvoie comme résultat vrai ssi tous sont vrai, sinon elle renvoie comme résultat faux

A	OU (or)	B	Résultat
T	OU	T	T
T	OU	F	T
F	OU	T	T
F	OU	F	F

Remarque :
OU : elle renvoie comme résultat vrai ssi j'en ai au moins une vrai, sinon elle renvoie comme résultat faux.

A	OUex (xor)	B	Résultat

T	OUex	T	F
T	OUex	F	T
F	OUex	T	T
F	OUex	F	F

Remarque :
- OUex : elle renvoie comme résultat vrai ssi les deux variables sont différents (vrai et faux), sinon elle renvoie comme résul6tat faux.

- Non(true) ➔ false
  Not(false) ➔ true

4) Le type caractère ➔ en pascal : char

On appelle une variable de type caractère ssi la variable contient 1 seul caractère.
Exemple de caractère :

* lettre : "A ".. "Z " ou "a" .. "z"
* chiffre : "0" .. "9"
* ponctuation : "," " : " "."
* " " (espace)
* caractère spéciaux : "@" "$"

Les fonctions prédéfinies sur les caractère :

Ord ("A") ➔ 65 Ord ("a") ➔ 97 Ord('0') ➔ 48	Ord elle renvoie le code ascui d'un caractère donnée Ord (caractère) ➔ entier
Chr (65) ➔ "A" Chr (98))➔ "b"	Chr (entier) ➔ caractère qui lui correspond
Succ("B") ➔ "C" Succ ("f") ➔ "g"	Succ (caractère)➔ le caractère suivant
Pred ("b") ➔ "a"	Pred(caractère) ➔ renvoie le caractère precedent
Upcase("c") ➔ "C" P Majus ("B") ➔ "B" A	Upcase (caractère) ➔ elle renvoie la majuscule

5) Le type chaine de caractère ➔ en pascal : string

On appelle variable de type chaine de caractère ssi elle contient 2 ou plusieurs caractère.

Exemple :

- Ch ➔ "zied" donc ch ➔ chaine
- Ch ➔ "b" donc ch ➔ caractère

Remarque :

Ch ➔ "informatique"

Ch[2] Ch[5]

Exemple :

Ch ➔ "zied"
Ch[3] ➔"a"

Ecrire (ch) ➔ "ziad"

Remarque :
Le principe de val c'est de convertir tous les chiffres qui forme la chaine ou non (loi de tous ou rien)

Les fonctions et les procédures prédéfinies sur les chaînes

Les fonctions	
X🡐 long(ch) X :=length(ch) ;	Exemple: Ch=''amira'' Donc dans x on trouve 5
P🡐 pos(c, ch) P :=pos(c, ch) ;	Exemple : C🡐 ''ir'' Ch🡐''amira'' Donc dans p on trouve 3
Ch1 🡐 Sous_chaîne (ch, p, n) Ch1 := copy(ch, p, n);	Exemple: Ch=''amira'', p=2, n=3 Donc dans ch1 on trouve ''mir''
Ch🡐ch1+ch2+ch3 Ch :=ch1+ch2+ch3;	Exemple: Ch1=''med'', ch2= '' '', ch3=''ali'' Donc dans ch on trouve ''med ali''
Les procédures	
Efface(ch, p, n) Delete(ch, p, n) ;	Ch=''farouk'', p=2, n =4 Donc dans ch on trouve ''fk''
Insert(ch1, ch2, p) ;	Ch1=''ha'' Ch2=''momed'' , p =3 Donc dans ch2 on trvouve ''mohamed''
Convch(n, ch) Str(n, ch) ;	n=2014 donc dans ch on trouve ''2014''
Valeur (ch, n, e) Val(ch, n, e) ;	Valeur(ch, n, e) ''102'' 102 0 ''10.2'' 10.2 0 ''10.2.3'' 0 5 ''1A2DBG6'' 0 2

Série N°1

Exercice n°1 :
Ecrire une analyse, un algorithme et un programme pascal, qui saisi le rayon d'un cercle de type entier et affiche la surface.

Bouillon :

- saisir le rayon
- calculer la surface
- afficher la surface.

Analyse :

Résultat = écrire(s)
S← R*R*Pi
R = donnée
Fin ex

TDO

Nom	Type
S R Pi	Réel Entier Constante = 3.14

Algorithme :

0) début ex
1) lire(R)
2) s← R*R*Pi
3) écrire(S)
4) fin ex

Exercice n°2 :

Ecrire une analyse, un algorithme et un programme pascal qui saisi la largeur et la longueur d'un rectangle et afficher le périmètre.

Bouillon :

- saisir longueur ➔ long
- saisir largeur ➔ larg
- calculer périmètre ➔ P
- afficher périmètre

Analyse :

Résultat = écrire(P)
P ← (larg+long)*2
Larg= donnée
Long= donnée
Fin

TDO

Nom	Type
Largn , long, p	Entier

Algorithme :

0) début ex
1) lire(larg)
2) lire(long)
3) p← (larg+long)*2
4) écrire(p)
5) fin

Pascal :

```
program med;
uses wincrt;
var
long, larg, p:integer;

begin
  writeln('donner longueur');
  readln(long);
  writeln('donner largeur');
  readln(larg);

  p:=(larg+long)*2;

  writeln(p);
end.
```

Exercice n°3 :

Saisir un temps T en seconde et le convertir et afficher en heure, minute et seconde.
Exemple :
3666 secondes on affiche 1 heure 1 minute et 6 secondes.

Brouillon :

- saisir le temps en seconde ➔ T
- extraire l'heure ➔ h
- extraire minute ➔ m
- extraire seconde ➔ s
- afficher (h, m, s)

Analyse :

résultat = écrire(h, m, s)
s🡐 (T mod 3600) mod 60
m 🡐 (T mod 3600) div 60
h 🡐 T div 3600
T = donnée
Fin

TDO

Nom	Type
T, h, m, s	Entier

Algorithme :

0) début ex
1) lire(T)
2) h 🡐 T div 3600
3) m 🡐 (T mod 3600) div 60
4) s🡐 (T mod 3600) mod 60
5) écrire(h, m, s)
6) fin ex

Pascal :

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

```
var
h, m, s, t:integer;

begin
   writeln('donner le temps en secondes');
   readln(T);

   h:= t div 3600;
   m:=(t mod 3600) div 60;
   s:=( t mod 3600) mod 60;

   writeln(h,' heures ', m,' minutes ',s,' secondes ');
end.
```

Exercice n°4 :

Saisir un entier de quatre chiffres et afficher la somme de ces chiffres.

Exemple :
2004 ➔ 2 + 0 + 0 + 4 on affiche 6

Brouillon :

- saisir n
- convertir n en chaîne ➔ ch
- rendre chaque caractère une valeur ➔ a , b, c , d
- somme des valeurs obetenus ➔ s
- afficher s

Analyse :

Résultat = écrire(s)
S🡐 a + b + c +d
Valeur(ch[4], d, e)
Valeur(ch[3], c, e)
Valeur(ch [2], b , e)
Valeur(ch[1], a, e)
Convch(n, ch)
n = donnée
fin

TDO

Nom	Type
A, b, c, d, e, n Ch	Entier Chaîne

Algorithme :

0) Début ex
1) Lire(n)
2) Convch(n, ch)
3) Valeur(ch[1], a, e)
4) Valeur(ch [2], b , e)
5) Valeur(ch[3], c, e)
6) Valeur(ch[4], d, e)
7) S🡐 a + b + c +d

8) Ecrire(s)
9) Fin

Pascal :

```
program ex;
uses wincrt;
var
s,a, b, c, d, e, n:integer;
ch:string;

begin
   writeln('donner un entier de quatre chiffre');
   readln(n);

   str(n, ch);

   val(ch[1], a, e);
   val(ch[2], b, e);
   val(ch[3], c, e);
   val(ch[4], d, e);

   s:=a+b+c+d;

   writeln('la somme est ', s);
end.
```

Exercice n°5 :

Saisir une date sous la forme « jj/mm/aaaa » et calculer le nombre bonheur. Le nombre bonheur et calculer de la méthode suivante :

14/02/1994

14 – 02 + (1 + 9 + 9 + 4) = 35 donc le nombre bonheur est 35.
jj-mm + a + a + a + a

Analyse :

Résultat = écrire(B)
B← j-m + (a+b+c+d)
Valeur(cha[4], d, e)
Valeur(cha[3], c, e)
Valeur(cha[2], b, e)
Valeur(cha[1], a, e)

Valeur(chm, m, e)
Valeur(chj, j, e)
Cha← sous_chaîne(ch, 7, 4)
Chm←sous_chaîne (ch, 4, 2)
Chj←sous_chaîne(ch, 1, 2)

Ch = donnée
Fin ex

Algorithme :

0) Début ex
1) Lire(ch)
2) Chj←sous_chaîne(ch, 1, 2)
3) Chm←sous_chaîne(ch, 4, 2)
4) Cha←sous_chaîne(ch, 7, 4)
5) Valeur(chj, j, e)
6) Valeur(chm, m, e)
7) Valeur(cha[1], a, e)
8) Valeur(cha[2], b, e)
9) Valeur(cha[3], c, e)
10) Valeur(cha[4], d, e)
11) B← j-m+(a+b+c+d)
12) Ecrire(B)
13) Fin ex

Pascal :

```
program ex;
uses wincrt;
var
a, b, c, d, e, j, m:integer;
cha, chj, chm, ch:string;

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

   chj:=copy(ch, 1, 2);
   chm:=copy(ch, 4, 2);
   cha:=copy(ch, 7, 4);

   val(cha[1], a, e);
   val(cha[2], b, e);
   val(cha[3], c, e);
   val(cha[4], d, e);
   val(chm, m, e);
   val(chj, j, e);

   b:=j-m+(a+b+c+d);

   writeln(b);
end.
```

Chapitre 2 : Les structures algorithmiques de contrôle

I. Les structures de contrôle conditionnelles

6) Forme complète (2 propositions)

algorithme:

si condition **alors** Traitement 1
sinon Traitement 2
fin si

pascal:

```
if condition then
                begin
                ..... ;
                …… ;
                  end
else
        begin
          ..... ;
          ..... ;
          ..... ;
        end;
```

remarque :
- avant **else** on ne met jamais point virgule " ;"
- si j'en ai plusieurs instructions donc je dois les limités entre **begin** et **end**

7) Forme imbriqué ou généralisé (>= 3 propositions)

Algorithme :

si condition 1 **alors** Traitement 1
sinon **si** condition 2 **alors** Traitement 2
sinon **si** condition 3 **alors** Traitement 3
sinon **si** condition 4 **alors** Traitement 4
sinon
Traitement n
finsi

Remarque:
si j'ai n proposition donc je dois commencer par si et je termine par sinon et j'utilise (n-2) sinon si.

pascal:

```
if condition 1 then .....
else if condition 2 then
        begin
          .....;
          .....;
          .....;
        end
else if  condition 3 then .....
else if condition 4 then
         begin
          .....;
          .....;
          .....;
        end
else
    begin
     .....;
     .....;
    End ;
```

8) Forme à choix multiple selon ……. Faire :

Remarque:
On utilise la forme à choix multiple ssi :
- j'en ai plusieurs propositions >=3 cas
- je teste sur une seule variable
- la variable à tester doit être de ***type scalaire : (entier ou caractère ou booléen)***

algorithme :

selon variable **faire**

x1**:** .....
x2,x3**:** .....
x4..x7**:** .....
sinon
.....
fin selon

pascal:

case variable **of**
x1: ..... ;
x2,x3: ..... ;
x4..x7: ..... ;
else
..... ;
end;

Série N°2

Exercice n°1 :
Ecrire une analyse et un programme pascal permettant de saisir un entier et de vérifier si il est divisible par 5 ou non.

Brouillon :

- saisir un entier ➔ n
- vérifier n divisible par 5 ou non
- afficher ➔ msg

Analyse :

Résultat = écrire(msg)

Si n mod 5 =0 alors msg ← ‘’divsible par 5’’
Sinon msg ← ‘’nn div par 5’’
Fin si

N = donnée

TDO (tableau de déclaration des objets)

Nom	Type	Rôle
N Msg	Entier Chaîne	Donnée Résultat

Algorithme :

0) début ex
1) écrire(‘’donner un entier n’) lire(n)
2) Si n mod 5 =0 alors msg ← ‘’divisible par 5’’
Sinon msg ← ‘’ non divisible par 5’’
Fin si
3) Ecrire(msg)
4) Fin ex

Pascal :

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

begin
  writeln('donner n ');
  readln(n);

  if n mod 5=0 then msg:='div 5'
  else msg:='nn div 5';

  writeln(msg);
end.
```

Exercice n°2 :

Ecrire une analyse et un programme pascal permettant de saisir une moyenne supposé compris entre 0 et 20, puis afficher la mention qui correspond à la moyenne. (refusé, passable, assez bien, bien , tres bien)

R<10
10<= P <12
12<= AB <14
14<= B <16
TB >=16

Brouillon :

- saisir moyenne ➔ m
- vérifier la moyenne
- afficher la mention

Analyse :

Résultat = écrire(msg)

Si m<10 alors msg🡐 ‘’R’’
Sinon si (m>=10) et (m<12) alors msg🡐 ‘’P’’
Sinon si (m>=12) et (m<14) alors msg🡐 ‘’AB ‘’
Sinon si (m>=14) et (m <16) alors msg🡐 ‘’B’’
Sinon msg🡐 ‘’TB’’
Fin si

M = donnée
Fin

TDO

Nom	Type	Rôle
M	Réel	Donnée
Msg	Chaîne	Résultat

Algorithme (normal)

0) début ex
1) lire(m)

2) Si m<10 alors msg← ‘’R’’
Sinon si m>=10 et m<12 alors msg← ‘’P’’
Sinon si (m>=12) et (m<14) alors msg← ‘’AB ‘’
Sinon si m>=14 et m <16 alors msg← ‘’B’’
Sinon msg← ‘’TB’’
Fin si

2) écrire(msg)
3) fin ex

Pascal :

```
program ex;
uses wincrt;
var
m:real;
msg:string;

begin
 writeln('donner une moyenne ');
 readln(m);

 if m<10 then msg:='refusé'
 else if (m>=10) and (m<12) then msg:='passable'
 else if (m>=12) and (m<14) then msg:='Assez bien'
 else if (m>=14) and (m<16) then msg:='bien'
 else msg:='tres bien ';

 writeln(msg);
end.
```

Exercice n°3 :

Ecrire une analyse et un programme pascal permettant de saisir un caractère et afficher sa nature (voyelle, consonne, chiffre, autre)

Brouillon :

- saisir un caractère ➔ C
- verifer
- afficher (msg)

Analyse :

Résultat = écrire(msg)

Selon majus (c) faire
‘’A’’,’’E’’,’’Y’’,’’U’’,’’I’’,’’O’’ : msg← ‘’voyelle’’
‘’A’’..’’Z’’ : msg← ‘’consonne’’
‘’0’’..’’9’’ : msg← ‘’chiffres’’
Sinon msg ← ‘’autre’’
Fin selon

C = donnée

Pascal :

```
program ex;
uses wincrt;

var
c:char;
msg:string;
begin
   writeln('donner un caractère');
   readln(c);

   case upcase(c) of
   'A','E','Y','U','I','O': msg:='voyelle';
   'A'..'Z': msg:='consonne';
   '0'..'9': msg:='chiffres'
   else msg:='autre'
   end;

   writeln(msg);
end.
```

Exercice n°4 :

Ecrivez un programme qui affiche « Bonjour, madame » ou « Bonjour, mademoiselle » ou « Bonjour, monsieur » selon le sexe saisi (''F'' ou ''H'') et l'état marital saisi de la personne (''O'' : oui il est marié ou ''N'' : non il n'est pas marié).

Brouillon :

- saisir s
- saisir em
- verifier
- afficher (msg)

Analyse :

Résultat = écrire(msg)
Si majus (s) =''F'' et majus(em)= ''O'' alors msg← ''bjr madame''
Sinon Si majus (s) =''F'' et majus(em)= ''N'' alors msg← ''bjr mademoiselle''
Sinon Si majus (s) =''H'' alors msg← ''bjr monsieur''
Sinon msg←''erreur''
Fin si

Em = donnée
S = donnée

Fin ex

TDO

Nom	Type	Rôle
S	Caractère	Donnée
Em	Caractère	Donnée

Msg	Chaîne	Résultat

Exercice n°5 :

Ecrire un programme permettant de résoudre dans R, une équation de second degré sous la forme de : $ax^2+bx+c=0$, sachant que a, b et c sont saisi par l'utilisateur

Résultat = [ ]

Si (a=0) et (b=0) et (c=0) alors écrire(''R'')
Sinon si (a=0) et (b=0) et (c<>0) alors écrire(''ensmble vide'')
Sinon si (b<>0) alors écrire(-c/b)

Sinon
D← carré(b) – (4*a*c)
Si D<0 alors écrire(''enmble vide'')
Sinon si D=0 alors écrire(-b/(2*a))
Sinon écrire (-b-racine(d)/(2*a) , -b+racine(d)/(2*a))
Fin si
Fin si

C= donnée
B = donnée
A = donnée

TDO

Nom	Type
A, b, c, d	Réel

Exercice n°6 :

Ecrire une analyse et un programme pascal permettant de saisir un entier de 3 chiffres et de vérifier s'il est cubique ou non.
Exemple :
153 est dit cubique parce que $153 = 1^3+5^3+3^3$

Analyse :

Résultat = [ ]
Si (a*a*a)+(b*b*b)+(c*c*c) = n alors écrire(''cubique'')
Sinon écrire(''non cubique'')
Fin si

Valeur(ch[3], c, e)
Valeur(ch[2], b, e)
Valeur(ch[1], a, e)
Convch(n, ch)

N = donnée
Fin

TDO

Nom	Type
N, a, b, c, e Ch	Entier Chaîne

Exercice n°7 :

Ecrire un programme permettant de saisir le numéro du mois et affiche la saison qui lui correspond exemple : 1 = hiver.

Brouillon :

- saisir le n° du mois ➔ m
- vérifier la saison ➔ verifier m
- afficher la saison ➔ msg

Analyse :

Résultat = écrire(msg)

Si m dans [1, 2, 12] alors msg ← ‘’hiver’’
Sinon si m dans [3..5] alors msg← ‘’printemps’’
Sinon si m dans [6..8] alors msg ← ‘’été ‘’
Sinon si m dans [9..11] alors msg← ‘’automne’’
Sinon msg ← ‘’erreur’’
Fin si

M = donnée
Fin ex

TDO

Nom	Type
M Msg	Entier Chaîne

Analyse :

Résultat = écrire(msg)

Selon m faire
1, 2, 12 : msg← ’’hiver’’
9..11 : msg← ‘’automne’’
3..5 : msg ← ‘’peintemps’’
6..8 : msg ← ‘’été’’
Sinon
  Msg← ‘’erreur’’
Fin selon

M = donnée
Fin ex

TDO

Nom	Type
M Msg	Entier Chaîne

Algorithme :

0) début ex
1) ecrire (‘’donner le num du mois’’) lire(m)
2) selon m faire
   1, 2, 12 : ecrire(‘’hiver’’)
   9..11 : ecire(‘’automne’’)
   3..5 : ecrire(‘’printemps’’)
   6..8 : ecrire(‘’été’’)
   Sinon
   Ecrire(‘’erreur’’)
   Fin selon
3) fin ex

TDO

Nom	Type
M	entier

Pascal :

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

begin
writeln('donner num du mois');
readln(m);

case m of
1,2,12: writeln('hiver');
3..5: writeln('printemps ');
6..8: writeln('été');
9..11: writeln('autonme ')
else
 writeln('erreur');
end;

end.
```


II. Les structures itératives

1) Forme complète : Pour : nombre de répétition est connu en avance

Analyse - Algorithme	Pascal

Pour i **de** Vi **à** Vf **faire**
…….
…….
Fin pour
	For i :=Vi to Vf do
	Begin
	……
	……
	End;

Remarque:

- On utilise la boucle **pour** ssi le nombre de répétition est connu en avance (je connais l'indice de départ ainsi l'indice final)
- TO : Vi <Vf
- Downto : Vi>Vf
- I: compteur qui doit être de type scalaire (**entier** ou **caractère** ou **booléen**)

Conclusion :
On utilise la boucle **pour** dans les cas suivants :
- je connais l'indice de départ et l'indice d'arrêt
- nombre de répétition connu

2) Forme répétitive à condition d'arrêt

a) Forme répétitive à condition d'arrêt : répéter ……..jusqu'à

Algorithme	Pascal
Répéter …… ……. …….. **Jusqu'à** (condition d'arrêt)	**repeat** …… ; ……. ; …….. ; **until** (condition d'arrêt) ;

Remarque :
- on utilise répéter ssi le nombre de répétitions n'est pas connu en avance (indice de départ est connu et indice final est inconnu).
- Saisi avec une condition
- L'objet existe juste je vais faire une condition ➔ si ……. Sinon ……..

Remarque :
- répéter : on exécute puis on teste donc répéter elle est exécuter au moins une seule fois

b) Forme répétive à condition d'arrêt : Tant que ……..faire

Algorithme	Pascal

Tant que (condition) **faire**
……..
…..
…….
Fin tq
	while (condition) **do**
	begin
	……..
	…..
	End ;

Remarque :
- on utilise tant que ssi le nombre de répétitions n'est pas connu en avance.
- je dois tester puis exécuter ➔ tant que
- Tant que : on teste puis on exécute donc tant que peut ne pas être exécuter.

Série N°3

Exercice n°1 :

Saisir un tableau T par n caractère avec n compris entre 4 et 20, afficher le nombre de lettre dans le tableau.

Explication :
Saisir un tableau T par n caractère avec n compris entre 4 et 20 ➔ le nom du tableau est T, le nombre de case du tableau est n compris entre 4 et 20, et le tableau contient à l'intérieur des caractère.

Brouillon :

- Saisir la taille du tableau ➔ n compris entre 4 et 20
- saisir tableau T
- calculer le nombre de lettre ➔ s
- afficher s

Analyse :

Résultat = écrire(s)
S←0
Pour i de 1 à n faire
  Si majus(t[i]) dans [''A''..''Z''] alors s←s+1
  Fin si
Fin pour

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

répéter
N= donnée
Jusqu'à n dans [4..20]

TDNT

Type
Tab = tableau [1..20] de caractère

TDO

Nom	Type
T n, i, s	Tab Entier

Exercice n°2 :
Saisir un tableau par n entier avec n compris entre 2 et 10, afficher la somme du tableau et le nombre des valeurs impaires dans le tableau.

Brouillon :

- saisir taille ➔ n compris entre 2 et 10
- saisir un tableau
- calcul somme ➔ s
- calcul des nombre impaires ➔ p
- afficher s et p

Analyse :

Résultat = écrire(s, p)

P←0
Pour i de 1 à n faire
  Si t[i] mod 2=0 alors p←p+1
Fin pour

S←0
Pour i de 1 à n faire
  S← s+t[i]
Fin pour

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

Répéter
  N=donnée
Jusqu'à n dans [2..10]

TDNT

Type
Tab = tableau [1..20] de caractère

TDO

Nom	Type
T	Tab
n, i, s	Entier

Exercice n°3 :
Saisir un tableau par n réel avec n compris entre 10 et 30, afficher la moyenne du tableau.

Brouillon :

- saisir la taille n compris entre 10 et 30
- saisir le tableau T
- calculer la moyenne ➔ moy
- afficher moy

Analyse :

résultat = écrire(moy)
s←0
pour i de 1 à n faire
  s← s+t[i]
fin pour

moy← s/n

```
pour i de 1 à n faire
  t[i]= donnée
fin pour

répéter
  n = donnée
jusqu'à n dans [10..30]
```

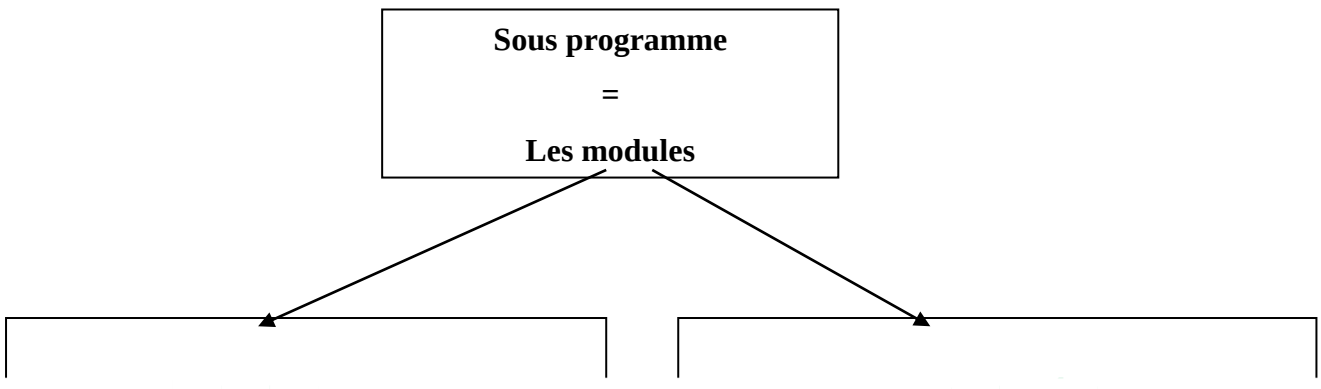
TDNT

Type
Tab = tableau [1..30] de réel

TDO

Nom	Type
N	Entier
S, moy	Réel
T	Tab

Chapitre 3 : Les sous programmes



La fonction renvoie **un seul résultat** de	La procédure renvoie **un ou plusieurs**

Activité :
Saisir un tableau par n chaîne de caractère non vides, avec n compris entre 5 et 30, afficher la somme total de l'ordre de chaque voyelle dans l'alphabet français.

Exemple :
N=5
T

Ala	Amine	Emna	sandra	Ska
1 + 1	+ 1 + 9 + 5	+5 +1	+ 1+1	+ 1 = 26

Brouillon :

- saisir la taille ➔ proc saisi
- remplir le tableau ➔ proc rempli
- somme ➔ fn somme
- afficher somme ➔ proc affich

1) Analyse principale

- Dans l'analyse principal on ne trouve plus ni conditionnelle, ni répétitive on trouve seulement des simples appel des modules (fonction ou procédure).
- Lors de l'appel d'une procédure : proc nom (????) : juste apple direct sans reçoit « ← » .
- Lors de l'appel d'une fonction : ? ← fn nom (????) : apple avec reçoit « ← » .
- Proc nom (**variable(s) donnée(s)** ; **variable(s) résultat(s)**) .
- Fn nom (**variable(s) donnée(s)**) : **type de résultat** .

Résultat =
Proc affich (s)
S← fn somme (T , n)
Proc rempli (n, **T**)
Proc saisi (**n**)
Fin ex

Le paramètre entre parenthèse lors de l'appel s'appelle paramètre effectif

TDNT (le TDNT il est réalisé qu'une seule fois sous l'analyse principale et si nous avons un tableau ou une matrice dans notre exercice)

Type
Tab = tableau [1..30] de chaîne

TDOG (tableau de déclaration des objets globaux ➔ il contient les variables de l'analyse principale ainsi les modules appelées dans l'analyse principale)

Nom	Type
S, n	Entier
T	Tab
Affich , rempli, saisi	Procédure
Somme	Fonction

2) Analyse des modules

Analyse des modules ➔ création des modules (la définition des modules).
Lors de la définition des modules nous commençons toujours avec les modules les plus faciles.

Analyse de la procédure rempli

```
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
```

Si le paramètre appartient à l'entête va subir un changement avant et après l'exécution de la procédure on doit ajouter le terme **var**

Le résultat dans la procédure est toujours le paramètre qui va subir un changement (var) sinon on écrit [ ]

TDOL (tableau de déclaration des objets locaux ➔ il contient les variables utilisé dans un module et qui n'existe pas dans l'entête du module concerné)

Nom	Type
I	Entier

Analyse de la procédure saisi

```
Def proc saisi (var n :entier)
Résultat = n
Répéter
  N = donnée
Jusqu'à n>=4 et n<=30
Fin proc saisi
```

Analyse de la procédure affich

```
Def proc affich (s :entier)
Résultat = [ ]
   Écrire(s)
Fin proc affich
```

Analyse de la fonction somme

```
Def fn somme (t :tab ; n :entier ) : entier
Résultat = somme ← s
S←0
Pour i de 1 à n faire
   Pour j de 1 à long(t[i]) faire
     Si majus (t[i][j]) dans [''A'',''E'',''Y'',''U'',''I'',''O''] alors
                                        S←s +(ord(majus(t[i][j])-64)
    Fin si
   Fin pour
Fin pour
Fin fn somme
```

TDOL

Nom	Type
I, j , s	Entier

Série N°3

Exercice n°1 :

Saisir un tableau par n entier positifs de quatre chiffres avec n compris entre 2 et 30, transférer dans un tableau V, la somme de chaque éléments du tableau, afficher les éléments du tableau T qui sont divisible par les éléments de V.

T

2004	2008	2010	4010	3520	6985

V

6	10	3	5	10	28

Les éléments à afficher : 2004, 2010, 4010, 3520

Brouillon :

- saisi taille ➔ proc
- remplir tableau ➔ proc
- transfer vers V ➔ proc
- afficher ➔ proc

Analyse principale :

Résultat = [ ]
  Proc affich (T, V, n)
  Proc tarnsfer (t, n, v)
  Proc rempli (n, T)
  Proc taille (n)

TDNT

Type
Tab = tableau [1..30] d'entier

TDOG

Nom	Type
T, v	Tab
N	Entier
Affich, taille, transfer, rempli	Procédure

Analyse de la procédure

Def proc taille (var n :entier)
Résultat = n
Répéter
  N = donnée
Jusqu'à n>=4 et n<=30
Fin proc taille

Analyse de la procédure rempli

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]>=1000 et t[i]<=9999
Fin pour
Fin proc rempli

TDOL

Nom	Type
I	Entier

Analyse de la procédure affich

Def proc affich (t, v :tab ; n :entier)
Résultat = [ ]
Pour i de 1 à n faire
  Si t[i] mod v[i] =0 alors écrire(t[i])
   Fin si
Fin pour
Fin proc affich

TDOL

Nom	Type
I	Entier

Analyse de la procédure transfer

```
Def proc transfer (t :tab ; n :entier ; var v :tab)
Résultat = v
Pour i de 1 à n faire
  Convch(t[i], ch)
   Pour j de 1 à long(ch) faire
      Valeur(ch[j], x, e)
      s←s+x
  fin pour
    v[i]←s
fin pour
fin proc transfer
```

TDOL

Nom	Type
I, x, e, s	Entier
Ch	Chaîne

Exercice n°2 :

Saisir une chaîne composée uniquement par des chiffres et 7 caractères au minimum, puis affiché si la chaîne saisie est divisible par 11 ou non en appliquant le principe suivant :

- faire la somme des indices paire de la chaîne (S1) et la somme des indices impairs de la chaîne (S2)
- si la valeur absolue de S1-S2 égale à 11 ou égale à 0, on affiche que la chaîne est divisible par 11 sinon le cas contraire.

Analyse principale :

```
Résultat = [ ]
  Proc affich(ch)
  Ch= proc saisi(ch)
Fin ex
```

TDOG

Nom	Type
Ch	Chaîne

Analyse de la procédure saisi

```
Def proc saisi (var ch :chaîne)
Résultat = ch
Répéter
  ch= donnée
  s←0
```

```
   pour i de 1 à long(ch) faire
    si ch[i] dans [‘’0’’..’’9’’] alors s←s+1
    fin si
   fin pour
jusqu’à (long(ch)>=7) et (s= long(ch))
fin proc saisi
```

TDOL

Nom	Type
I, s	Entier

Analyse de la procédure affich

```
Def proc affich (ch :chaîne)
Résultat [ ]
Si fn verif(ch) = vrai alors écrire(‘’divisible par 11’’)
  Sinon écrire(‘’non divisible par 11’’)
Fin si
Fin proc affich
```

TDOL

Nom	Type
Verif	Fonction

Analyse de la procédure verif

```
Def fn verif(ch :chaîne) : booleén
Résultat = verif
S1←0
S2← 0
Pour  I de 1 à long(ch) faire
   Valeur(ch[i], x, e)
   Si i mod 2=0 alors s1←s1+x
    Sinon s2←s2+x
   Fin si
Fin pour

   Si abs(s1+s2) dans [0, 11]  alors verif← vrai
     Sinon verif←faux
   Fin si
Fin fn verif
```

TDOL

Nom	Type
S1, s2, i, x, e	Entier

Pascal :

```
program ex;
uses wincrt;
var
  ch:string;

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

```
procedure saisi(var ch:string);
var
x:longint;
e:integer;
begin
 repeat
       writeln('donner une chaîne');
       readln(ch);

       val(ch, x, e);
 until (length(ch)>=7) and (e=0);
end;

function verif(ch:string):boolean;
var
 s1, s2, x, e, i:integer;
begin
    s1:=0;
    s2:=0;
    for i:=1 to length(ch) do
    begin
        val(ch[i],x, e);
          if i mod 2=0 then s2:=s2+x
        else s1:=s1+x;
    end;

    if abs(s1-s2) in [0,11] then verif:=true
    else verif:=false;
end;

procedure affich (ch:string);
begin
    if verif(ch)=true then writeln('divisibla par 11')
    else writeln('non divisible par 11');
end;

{************** PP **************}
begin
    saisi(ch);
    affich(ch);
end.
```

Exercice n°3 :

Saisir un tableau T par n entier positif de trois chiffres avec n compris entre 4 et 30, afficher si le tableau est palindrome ou non.

T

124	522	633	633	522	124

T est palindrome

T

124	514	633	633	522	124

T n'est pas palindrome

Brouillon :

- saisi taille ➔ proc
- saisi tableau ➔ proc
- Affichage ➔ proc

Analyse principale :

Résultat = [ ]
Proc affich (t, n)
T = Proc tableau(n, t)
N = Proc taille (n)

TDNT

Type
Tab = tableau [1..30] d'entier

TDOG

Nom	Type
N	Entier
T	Tab
Affich, tableau, taille	Procédure

Analyse de la procédure taille

```
Def proc taille (var n :entier)
Résultat = n
Répéter
   N = donnée
Jusqu'à n>=4 et n <=30
Fin proc taille
```

Analyse de la procédure tableau

```
Def proc tableau (n :entier ; var t :tab)
Résultat = t
Pour i de 1 à n faire
  Répéter
   T[i] = donnée
   Jusqu' t[i]>=100 et t[i]<=999
Fin pour
```

TDOL

Nom	Type
I	Entier

Analyse de la procédure affich

```
Def proc affich (t :tab ; n :entier)
Résultat = [ ]
I←0
J←n+1
Répéter
   I←i+1
```

J←j-1
Jusqu'à t[i]<> t[j] ou i>=j

Si i>= j alors écrire(''palindrome '')
Sinon écrire(''non palindrome'')
Fin si
Fin proc affich

TDOL

Nom	Type
I, j	Entier

Pascal :

```
program ex;
uses wincrt;
type
tab = array[1..30] of integer;
var  { les variables du pp }
n:integer;
t:tab;

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

procedure affich(t:tab ; n:integer);
var
  i, j:integer;
begin
    i:=0;
    j:=n+1;
    repeat
      i:=i+1;
      j:=j-1;
    until (i>=j) or (t[i]<>t[j]);

    if i>=j then writeln('palindrome ')
    else writeln('non palindrome');
end;

procedure rempli(n:integer; var t:tab );
var
 i:integer;
begin
    for i:=1 to n do
    begin
          repeat
            writeln('donner t[',i,']');
            readln(t[i]);
       until (t[i]>=100) and (t[i]<=999);
    end;
end;
```

```
procedure taille (var n:integer);
begin
   repeat
         writeln('donner n ');
         readln(n);
   until (n>=4) and (n<=30);
end;

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

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

Exercice n°4 :

Saisir une chaîne non vide contenant que des lettres alphabétiques afficher le nombre de touche à appuyer sur un téléphone portable pour saisir cette chaîne.

1	2 A b c	3 D e f
4 G h I	5 J k L	6 M N O
7 P Q R S	8 T U V	9 W X Y Z

Exemple :
Ahmed = 1+2+1+2+1 = 7

Brouillon :

- saisir la chaîne ➔ proc saisi
- calcul ➔ fn calcul
- affiche ➔ proc affich

Analyse principale :

```
Résultat = [ ]
  Proc affich (s)
  S← fn calcul (ch)
  Ch = Proc saisi (ch)
Fin
```

TDOG

Nom	Type
Ch	Chaîne
S	Entier
Affich, saisi	Procedure
Calcul	Fonction

Analyse de la procedure affich

```
Def proc affich (s:entier)
Résultat = [ ]
Écrire(s)
```

Fin proc affich

Analyse de la procédure saisi

Def proc saisi (var ch :chaîne)
Résultat = ch
Répéter
  Ch = donnée
  S←0
   Pour i de 1 à long(ch) faire
      Si majus(ch[i]) dans [''A''..''Z''] alors s←s+1
       Fin si
     Fin pour
Jusqu'à (ch<>'') et (s=long(ch))

TDOL

Nom	Type
I , s	Entier

Analyse de la fonction calcul

Def fn calcul (ch :chaîne) : entier
Résultat = calcul ←s
S←0
Pour i de 1 à long(ch) faire
  Si majus (ch[i]) dans[''A'',''D'',''J'',''G'',''M'',''P'',''T'',''W''] alors s←s+1
  Sinon Si majus (ch[i]) dans[''B'',''E'',''H'',''K'',''N'',''R'',''U'',''X''] alors s←s+2
   Sinon si majus(ch[i]) dans [''S'',''Z''] alors s←s+4
    Sinon s←s+3
  Fin si
Fin pour
Fin fn calcul

TDOL

Nom	Type
S, i	Entier

Pascal :

```
program ex;
uses wincrt;
var  { les variables du pp }
ch:string;
nb:integer;

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

procedure affich(nb:integer);
begin
   writeln('le nombre de touche est ', nb);
end;

procedure saisi(var ch:string);
var
  s,i:integer;
```

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

         s:=0;
         for i:=1 to length(ch) do
         begin
            if upcase(ch[i]) in ['A'..'Z'] then s:=s+1;
       end;
   until (ch<>'') and (length(ch)=s);
end;

function calcul(ch:string):integer;
var
  i, nb:integer;
begin
    nb:=0;
    for i:=1 to length(ch) do
    begin
         case upcase(ch[i]) of
         'A','D','G','J','M','P','T','W': nb:=nb+1;
         'B', 'E', 'H','K', 'N','Q','U','X': nb:=nb+2;
         'S','Z': nb:=nb+4
         else nb:=nb+3;
         end;
    end;

    calcul:=nb;
end;

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

begin
   saisi(ch);
   nb:=calcul(ch);
   affich(nb);
end.
```

Chapitre 4 : Les algorithmes de tri & de recherche

I. Tri à bulles

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

```
Def proc tri_bull( var t : tab ; n :entier)
Résultat = t
Répéter
    Perm🡐 faux
    Pour i de 1 à (n-1) faire
      Si t[i]>t[i+1] alors
                          Perm🡐vrai
                          Aux🡐t[i]
                          T[i]🡐t[i+1]
                           T[i+1]🡐aux
      Fin si
  Fi pour
Jusqu'à Perm= faux
Fin proc tri bull
```

TDOL

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

2) Tri bull suivant la longueur de la chaîne

```
Def proc tri_bull( var t : tab ; n :entier)
Résultat = t
Répéter
    Perm🡐 faux
    Pour i de 1 à (n-1) faire
      Si long(t[i])>long(t[i+1]) alors
                          Perm🡐vrai
                          Aux🡐t[i]
                          T[i]🡐t[i+1]
                           T[i+1]🡐aux
      Fin si
  Fi pour
Jusqu'à Perm= faux
Fin proc tri bull
```

TDOL

Nom	Type
Aux	Chaîne
I	Entier
Perm	Booleén

Pascal :

```
program ex;
uses wincrt;
type
tab = array[1..20] of string;
var
t:tab;
n:integer;

{********************  les mdoules ********************}

procedure tri_bull(var t:tab ; n:integer);
var
 i:integer;
 aux:string;
 perm:boolean;
begin
    repeat
      perm:=false;
      for i:=1 to n-1 do
      begin
          if t[i]>t[i+1] then
                              begin
                                  perm:=true;
                                  aux:=t[i];
                                  t[i]:=t[i+1];
                                  t[i+1]:=aux;
                        end;
      end;

    until perm=false;
end;

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

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

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

II. Tri insertion

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

```
Def proc tri_insertion( var t : tab ; n :entier)
Résultat = t
Pour i de 2 à n faire
   Aux←t[i]
    J←i-1
    Tant que (aux<t[j]) et (j>0) faire
        T[j+1]←t[j]
        J←j-1
    Fin tant que

    T[j+1]←aux
Fin pour
Fin proc tri insertion
```

TDOL

Nom	Type
Aux I, j Perm	De même type que les case du tableau Entier Booléen

2) Tri insertion suivant la longueur de la chaîne

```
Def proc tri_insertion( var t : tab ; n :entier)
Résultat = t
Pour i de 2 à n faire
   Aux←t[i]
    J←i-1
    Tant que (long(aux)<long(t[j])) et (j>0) faire
        T[j+1]←t[j]
        J←j-1
    Fin tant que

    T[j+1]←aux

Fin proc tri insertion
```

TDOL

Nom	Type
Aux	Chaîne
I	Entier
Perm	Booleén

Pascal :

```
program ex;
uses wincrt;
type
tab = array[1..20] of integer;
var
t:tab;
n:integer;

{********************  les mdoules ********************}

procedure tri_inser(var t:tab ; n:integer);
var
  i, j:integer;
  aux:integer;
begin
    for i:=2 to n do
    begin
          aux:=t[i];
          j:=i-1;

          while (aux<t[j]) and (j>0) do
          begin
              t[j+1]:=t[j];
           j:=j-1;
          end;

          t[j+1]:=aux;
    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,']');
        readln(t[i]);
    end;
end;
```

```
procedure affich(t:tab ; n:integer);
var
i:integer;
begin
   for i:=1 to n do
   begin
      write(t[i],' ');
   end;

   writeln;
end;

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

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

III. Tri selection

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

Analyse de la procédure tri_selection

```
Def proc tri_select (var t :tab ; n :entier)
Résultat = T
Pour i de 1 à (n-1) faire
  p← fn posmin(i, t, n)
  si i<>p alors
                aux←t[i]
                t[i]←t[p]
                t[p]←aux
   fin si
fin pour
fin proc tri_select
```

TDOL

Nom	Type
I, p Aux Posmin	Entier De même type que les cases du tableau Fonction

Analyse de la fonction posmin

```
Def fn posmin (deb : entier ; t :tab ; n :entier) :entier
Résultat = posmin ← pmin
Pmin ← deb
Pour i de (deb+1) à n faire
   Si t[pmin]>t[i] alors pmin ←i
   Fin si
Fin pour
Fin fn posmin
```

TDOL

Nom	Type
i , pmin	Entier

2) Tri selection : suivant la longueur de la chaîne

Analyse de la procédure tri_selection

```
Def proc tri_select (var t :tab ; n :entier)
Résultat = T
Pour i de 1 à (n-1) faire
   p🡐 fn posmin(i, t, n)
   si i<>p alors
                  aux🡐t[i]
                  t[i]🡐t[p]
                  t[p]🡐aux
    fin si
fin pour
fin proc tri_select
```

TDOL

Nom	Type
I, p	Entier
Aux	Chaîne
Posmin	Fonction

Analyse de la fonction posmin

```
Def fn posmin (dep : entier ; t :tab ; n :entier) :entier
Résultat = posmin 🡐 pmin
Pmin 🡐 dep
Pour i de (dep+1) à n faire
    Si long(t[pmin])>long(t[i]) alors pmin 🡐i
    Fin si
Fin pour
Fin fn posmin
```

TDOL

Nom	Type
i , pmin	Entier

Pascal :

```
program ex;
uses wincrt;
type
tab = array[1..20] of integer;
var
t:tab;
n:integer;

{********************  les mdoules ********************}
```

```
function posmin(deb:integer; t:tab ; n:integer):integer;
var
pmin, i:integer;
begin
   pmin:=deb;

   for i:=(deb+1) to n do
   begin
      if t[pmin]>t[i] then pmin:=i;
   end;

   posmin:=pmin;
end;

procedure tri_select(var t:tab; n:integer);
var
i, p:integer;
aux:integer;
begin
   for i:=1 to n-1 do
   begin
         p:= posmin(i, t, n);
         if p<>i then
                  begin
                        aux:=t[i];
                        t[i]:=t[p];
                        t[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,']');
       readln(t[i]);
   end;
end;

procedure affich(t:tab ; n:integer);
var
i:integer;
begin
   for i:=1 to n do
   begin
      write(t[i],' ');
```

```
    end;
end;

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

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


Chapitre 5 : Les algorithmes arithmétiques

I. PPCM

```
Def fn PPCM (a, b :entier) :entier
Résultat =

Si a>b alors
                Max←a
                Min←b
                Maxi←a
Sinon
                Max←b
                Min←a
                Maxi←b
Fin si

Tant que (max mod min<>0) faire
```

```
    Max←max+maxi
Fin tant que
Fin fn PPCM
```

II. PGCD

```
Def fn PGCD (a, b :entier) : entier
Résultat = PGCD← a
Tant que (a<>b) faire
  Si a>b alors a←a-b
    Sinon b←b-a
   Fin si
Fin tant que
Fin fn PGCD
```

III. Nombre premier

```
Def fn premier(n :entier) :booleén
Résultat =
Si (s=0) alors premier←vrai
Sinon premier←faux
Fin si

[S←0] pour i de 2 à (n-1) faire
                Si n mod i=0 alors s←s+1
                 Fin si
          Fin pour
Fin fn premier
```

IV. Puissance

```
Def fn puis(x, n :entier) :entier
Résultat = puis←p
[p←1] pour i de 1 à n faire
          P←P*x
        Fin pour
Fin fn puis
```

V. Factoriel

```
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
```

VI. La conversion entre bases de numération

1) Conversion des bases (10 vers base 2)

```
Def fb conv_10_2(n :entier ) : chaîne
Résultat = conv_10_2← ch

Ch←''
Répéter
  R← N mod 2
  N← N div 2
  Convch(r, c)
  Ch← c  + ch
Jusqu'à n=0
Fin fn conv_10_2
```

2) Conversion des bases (10 vers base 8)

```
Def fb conv_10_8(n :entier ) : chaîne
Résultat = conv_10_8← ch

Ch←''
Répéter
  R← N mod 8
  N← N div 8
  Convch(r, c)
  Ch← c  + ch
Jusqu'à n=0
Fin fn conv_10_8
```

3) Conversion des bases (10 vers base 16)

```
Def fb conv_10_16(n :entier ) : chaîne
Résultat = conv_10_16← ch

Ch←''
Répéter
  R← N mod 16
  N← N div 16
  Si r<10 alors  Convch(r, c)
     Sinon c← chr(r+55)
  Fin si

  Ch← c  + ch
Jusqu'à n=0
Fin fn conv_10_16
```

4) Conversion des bases (10 vers base qq)

```
Def fb conv_10_qq(n :entier ; b :entier  ) : chaîne
Résultat = conv_10_qq← ch

Ch←''
Répéter
```

```
   R← N mod b
   N← N div b
   Si r<10 alors  Convch(r, c)
      Sinon c← chr(r+55)
   Fin si

   Ch← c  + ch
Jusqu'à n=0
Fin fn conv_10_qq
```

5) Conversion des bases (qq vers base 10)

```
Def fb conv_qq_10(n :entier ; b :entier  ) : chaîne
Résultat = conv_qq_10 ←s
[s←0, j←0]  pour i de long(ch) à 1 faire
                    Si ch[i] dans ['A'..'F'] alors x← ord(ch[i])-55
                          Sinon valeur(ch[i], x, e)
                     Fin si

                     S← S+ (x* puis(b, j))
                      J←J+1
                  Fin pour
Fin fn conv_qq_10
```

Série N°4

Exercice n°1 :

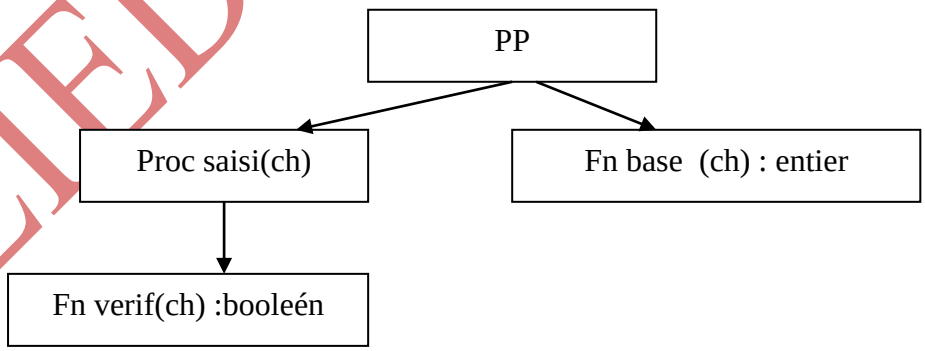
Ecrire un programme permettant de saisir une chaîne contenant que des chiffres et des lettres ('A'..'F') alphabétique. Afficher la base correspondant à cette chaîne.

Exemple :

1254 la base est 6

1542B ➔ la base est 12

Brouillon :



Analyse Principale :

```
Résultat = écrire( b )
  b ← Fn base(ch)
  ch = proc saisi(ch)
fin ex
```

TDOG

Nom	Type

b	Entier
Ch	Chaîne
Base	Fonction
Saisi	Procédure

Analyse de la procédure saisi

```
Def proc saisi(var ch :chaîne)
Résultat = ch
Repéter
   Ch = donnée
Jusqu'à verif(ch)= vrai
Fin proc saisi
```

Analyse de la fonction verif

```
Def fn verif(ch :chaîne) : booléen
Résultat = verif← test
S←0
Pour I de 1à long(ch) faire
    Si ch[i] dans ['0''..''9'', 'A''..''F''] alors s←s+1

  Fin si
Fin pour

Si s=long(ch) alors test←vrai
Sinon test←faux
Fin si
Fin fn verif
```

TDOL

Nom	Type
I	Entier
Test	Booléen
S	Entier

Analyse de la fonction base

```
Def fn base (ch :chaîne) :entier
Résultat = base ← b
Max←ch[1]
Pour i de 2 à long(ch) faire
   Si ch[i] >max alors max← ch[i]
    Fin si
Fin pour

Si max dans ['A''..''F''] alors b← ord(max)-55+1
Sinon valeur (max, b, e)
          b←b+1
Fin si
Fin fn base
```

TDOL

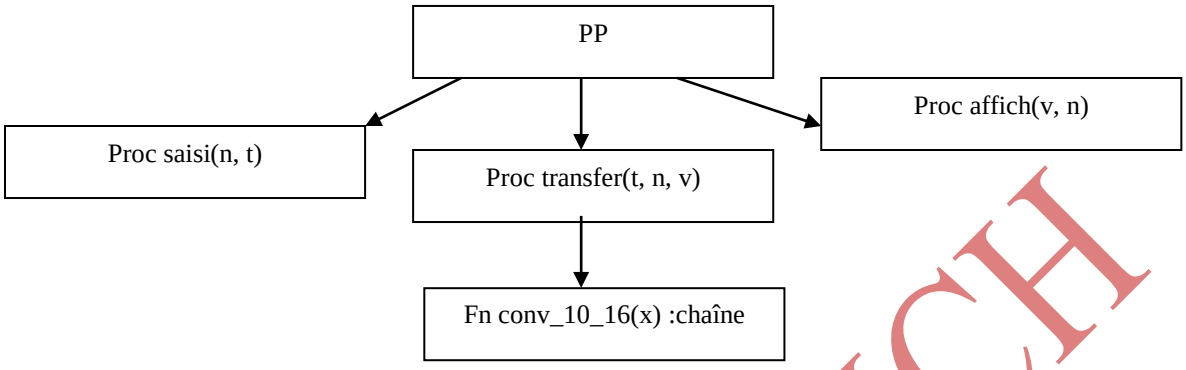
Nom	Type
B, i, e	Entier

Max	Caractère

Exercice n°2 :

Ecrire un programme pascal permettant de saisir un tableau T par n entier positifs, puis transférer les éléments du tableau T dans un tableau V en base 16.

Brouillon :



Pascal :

```
program ex;
uses wincrt;
type
tab_e = array[1..10] of integer;
tab_c= array[1..10] of string;
var
t:tab_e;
n:integer;
v:tab_c;

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

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

    for i:=1 to n do
    begin
          repeat
              writeln('donner t[',i,']');
              readln(t[i]);
       until t[i]>0;
    end;
end;

procedure affich(v:tab_c; n:integer);
var
  i:integer;
begin
```

```
    for i:=1 to n do
    begin
          writeln(v[i]);
    end;
end;

function conv_10_16(n:integer):string;
var
   ch, c:string;
   r:integer;
begin
     ch:='';

     repeat

           r:= n mod 16;
           n := n div 16;
           if r<10 then str(r, c)
           else c:=chr(r+55);
           ch:=c+ch;

     until n=0;

     conv_10_16:=ch;
end;

procedure transfer(n:integer; t:tab_e; var v:tab_c);
var
 i:integer;
begin
    for i:=1 to n do
    begin
       v[i]:=conv_10_16(t[i]);
    end;
end;

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

begin
   saisi(n, t);
   transfer(n, t, v);
   affich(v, n);
end.
```

Chapitre 6 : Les algorithmes récurrents

I. Calcul sur les suites:

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

```
Def proc suite(n : entier)
Résultat = [ ]
U← valeur initialisation
Pour i de (indice +1) à n faire
    Up← U
    U← formule
    Ecrire (U)
Fin pour
Fin proc suite
```

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

```
Def proc suite(n : entier)
Résultat = [ ]
Up← valeur initialisation 1
U← valeur initialisation 2
Pour i de (indice +1) à n faire
    Upp← Up
    Up← U
    U← formule
    Ecrire (U)
Fin pour
```

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

```
Def proc suite( ?:type)
Résultat = [ ]
U← valeur
Répéter
   Up← U
   U← formule)
Jusqu'à abs(U-Up)< 10-4
Ecrire(U)
Fin proc suite
```

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

Def proc suite(n : entier)
Résultat = [ ]
Up← valeur1
U← valeur 2
Répéter
  Upp← Up
  Up← U
  U← formule)
Jusqu'à abs (U-Up)< 10^{-4}

Série N°5

Exercice 1 :

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

U_0= x
U_n+1 = (U_n / 2) + (x/2U_n)
Avec x un réel positif.

Analyse principale :

Résultat= proc suite_aff(n,a)
TT= proc saisi(a)
  Proc saisi(n)

Algorithme principal :

0) début ex
1) proc saisi_n (n)
2) proc saisi_x (x)
3) proc suite_aff (n,x)
4) fin ex

Analyse de la suite_aff

Résultat= []
TT= [U←a] pour i de 1 à n-1 faire
  U← (U / 2) + (x /2 U)
  Ecrire(U)
  Fin pour

Algorithme de la suite_aff

0) début procedure suite_aff(n,a :entier)
1) [U←a] pour i de 1 à n-1 faire
  U← (U / 2) + (x /2 U)
  Ecrire(U)
  Fin pour
2) fin proc suite_aff

Exercice n°2 :

Soit la suite $(P_i)_{i \text{ impair}}$ définie par :

$$\begin{cases} P_1 = 2 \\ P_{i+1} = P_{i-1} \cdot \dots \end{cases}$$

Ecrire un programme Pascal qui permet de calculer et d'afficher les termes de la suite P jusqu'à ce que la différence entre deux termes consécutifs devient inférieure ou égale à 10^{-4}.

Pascal :

```
program ex;
uses wincrt;
var
U:real;

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

procedure suite_aff(U:real);
var
i:integer;
Uav:real;
begin

i:=1;
repeat
i:=i+2;
Uav:=U;

U:=Uav*((i-1)/i)*((i+1)/i);
writeln(U:0:5,' l différe est ',abs(U- Uav):0:6);
until abs(U- Uav)<1e-4;
end;

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

begin
U:=2;
suite_aff(U);
end.
```

Exercice n°3 :

Ecrire un programme Pascal qui permet de calculer puis d'afficher la racine carrée d'un réel positif x donné en utilisant la suite suivante :
Il s'agit de calculer les premiers termes de cette suite jusqu'à ce que la différence entre deux termes successifs devient inférieure ou égale à 10^{-4}.
Le dernier terme calculé est une valeur approchée de $\sqrt{x}$ à 10^{-4} près.

$$\begin{cases} U_0 = (1 + x)/2 \\ U_{n+1} = (U_n + x/U_n)/2 \end{cases}$$

```
program ex;
uses wincrt;
var
U,x:real;
```

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

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

procedure suite_aff(x:real);
var
Uav:real;

begin
U:=(1+x)/2;

repeat
Uav:=U;
U:=(Uav +(x/Uav))/2;
until abs(U-Uav)<1e-4;

writeln(U:0:2);
end;

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

begin
saisi(x);
suite_aff(x);
end.
```

Série N°6

Exercice 1 :

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

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).

N.B :

La solution doit comporter au moins une fonction et une procédure.

Pascal :

```
program ex;
uses wincrt;
var
x:real;

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

 {*****  focntion puissance *****}

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

 puis:=p;
 end;

 {*****  focntion factoriel ****}

 function fact(n:integer):integer;
 var
 i,f:integer;
 begin

 f:=1;
 for i:=1 to n do
 begin
  f:=f*i;
```

```
end;
fact:=f;
end;

function calcul(x:real):real;
var
i,j:integer;
sav,s:real;

begin

s:=0;
i:=1;
j:=0;

repeat
j:=j+1;

sav:=s;
if j mod 2<>0 then s:=sav+(puis(x,i)/fact(i))
                else s:=sav-(puis(x,i)/fact(i));
i:=i+2;
until abs(s-sav)<=1e-4;

calcul:=s;
end;

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

begin
writeln('donner x');
readln(x);

writeln(calcul(x):0:3);
end.
```

Exercice n°2 :

Soit la somme Sn suivante :

Sn=1/11+3/22 + 5/33 +7/44 +….. +(2n-1)/nn

Ecrire un programme Pascal intitulé SOMME permettant de calculer et d'afficher la somme Sn pour un entier n positif donné < 10 en utilisant la formule ci-dessus.

N.B :

La solution doit comporter au moins deux modules.

Analyse :

```
Résultat= somme
TT = somme ← s
        [s←0] pour i de 1 à n faire
                S←s+((2*i) - 1)/((i*10)+i)
               Fin pour
```

Analyse principale :

```
Résultat= écrire(somme(n))
```

TT = saisi(n)

Exercice n°3 :

Soit la somme Sn suivante :
Sachant que **$6+6/2^2 + 6/3^2 +6/4^2 +\ldots..+6/n^2$ tend vers π^2 ,** écrire un programme Pascal permettant de calculer puis d'afficher une valeur approchée de **π^2** , Le calcul s'arrête quand la différence entre deux termes consécutifs devient inférieure ou égale à 10^{-4}

Algorithme somme :

0) fonction somme (i :entier) :réel
1) [s←0] répéter
   I←i+1 ;
   Sav ← s
   S← sav+ (6/carrée(i))
   Jusqu' à abs (s-sav) <= 10^{-4}
2) somme← s
3) fin fonction somme

Algorithme principal :

0) début ex
1) i←0
2) écrire(somme(i))
3) fin ex

Exercice n°4 :

Soit l'expression mathématique suivante : **$\pi/4 = 1 – 1/3 + 1/5 – 1/7 + 1/9 - \ldots$**
Écrire un programme Pascal qui utilise l'expression ci-dessus pour déterminer et afficher une valeur approchée de π à 10^{-4} prés .

N.B :

1. Le calcul s'arrête quand la différence entre deux valeurs consécutives de cette expression devient strictement inférieure à 10^{-4} .
2. La solution doit comporter au moins une fonction et une procédure.

Algorithme somme :

0) fonction somme(i :entier) :reél
1) [s←1, j←0] répéter
   I←i+2
   Sav← s
   Si j mod 2<>0 alors S← sav +(1/i)
   Sinon s←sav- (1/i)
   J←j+1
   Jusqu'abs(s-sav) <=1E-4
2) somme ←S

Exercice n°5 :

Soient deux entiers récurrentes u et v :

$V_0 = 0$ **$V_{n+1}=$** $\sqrt{\frac{1+Vn}{2}}$

Et $U_0 = 2$ $U_{n+1} = \frac{Un}{V_{n+1}}$

Calculer la suite Un et Vn.

Algorithme somme :

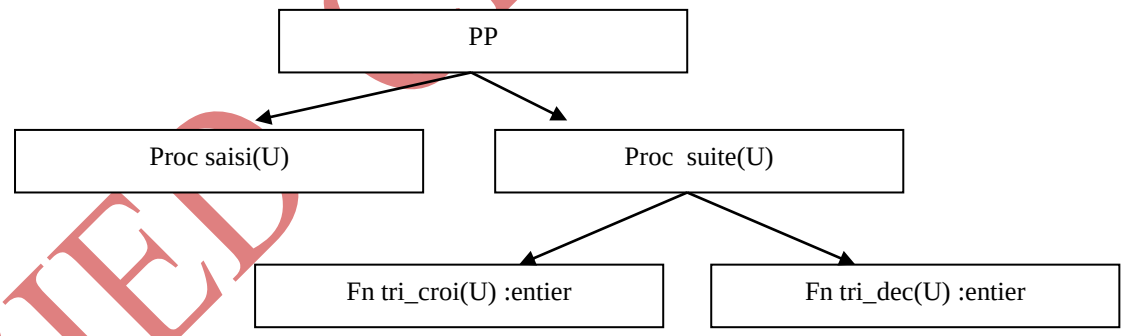
0) procédure suite(n :entier)
1) [U←2, V←0] pour i de 1 à n faire
   V←racine((1+v)/2)
   U← U/V
   Ecrire(U , V)
   Fin pour
2) fin procédure suite

Exercice n°6 (examen 20/02/2012)

Soit U_0 un nombre de quatre chiffre, on forme à l'aide de quatre chiffres de U_0 le plus grand entier naturel Max et le plus petit entier naturel Min. leurs différence (max-min) donne un nombre U_1 , on refait le travail pour U_1 et on obtient ainsi une suite. Cette suite est stationnaire c'est-à-dire qu'elle devient constante à partir d'un certain rang. Ecrire une analyse d'un programme qui calcul et affiche les termes de la suite et jusqu'à ce qu'elle devienne constante. Le premier terme U_0 est un entier donné de quatre chiffres.

Exemple :
2541

5421 - 1245 = 4176
7641 – 1467 =



Pascal :

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

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

function tri_croi(n:integer):integer;
var
  ch:string;
  i,x ,e:integer;
  aux:char;
  perm:boolean;
begin
```

```
  str(n, ch);
  repeat
     perm:=false;
     for i:=1 to length(ch)-1 do
     begin
        if ch[i]>ch[i+1] then
                        begin
                           perm:=true;
                           aux:=ch[i];
                           ch[i]:=ch[i+1];
                           ch[i+1]:=aux;
                        end;
     end;
  until perm=false;

  val(ch, x, e);
  tri_croi:=x;
end;

function tri_dec(n:integer):integer;
var
  ch:string;
  i,x ,e:integer;
  aux:char;
  perm:boolean;
begin
  str(n, ch);
  repeat
     perm:=false;
     for i:=1 to length(ch)-1 do
     begin
        if ch[i]<ch[i+1] then
                        begin
                           perm:=true;
                           aux:=ch[i];
                           ch[i]:=ch[i+1];
                           ch[i+1]:=aux;
                        end;
     end;
  until perm=false;

  val(ch, x, e);
  tri_dec:=x;
end;

procedure suite(u:integer);
var
 up:integer;
begin
    repeat
       up:=u;
       u:=tri_dec(up)-tri_croi(up);
       writeln(u);
    until (up=u);
end;
```

```
procedure saisi(var u:integer);
begin
   repeat
      writeln('donner U');
      readln(u);
   until (u>=1000) and (u<=9999);
end;

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

begin
   saisi(u);
   suite(u);
end.
```

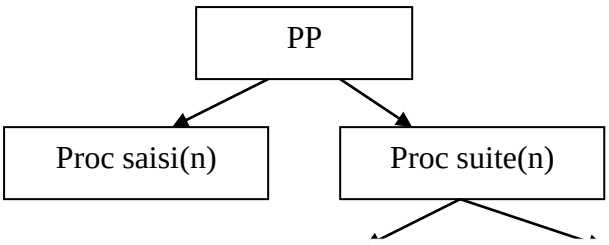

Série N°7

Exercice 1 :

Ecrivez un programme permettant de calculer et afficher les n premiers termes de la suite suivante :

$U_0 = 2$

$U_n = (U_{n-1}\ !\ ) + (U_{n-1})^n$



Fn fact(n) :entier	Fn puis(x, n) :entier

Analyse principale

Résultat = pro suite(n)
N = proc saisi(n)
Fin ex

TDOG

Nom	Type
N Suite, saisi	Entier Procédure

Analyse de la procédure suite

Def proc suite(n :entier)
Résultat = [ ]
U←2
Pour i de 1 à n faire
Up←U
U← fn fact(Up) + fn puis(Up, i)
Ecrire(U)
Fin pour
Fin proc suite

TDOL

Nom	Type
U, i, Up Fact , puis	Entier Fonction

Analyse de la fonction fact

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

TDOL

Nom	Type
I, F	Entier

Analyse de la fonction puis

Def fn puis (X, n :entier)
Résultat = puis ←P
P←1
Pour i de 1 à n faire
P← P*X
Fin pour

Fin fn puis

TDOL

Nom	Type
I, p	Entier

Analyse de la procédure saisi

```
Def proc saisi (var n :entier)
Résultat = n
Répéter
   N = donnée
Jusqu'à n>0
Fin proc saisi
```

Exercice 2 :

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 et n>=2.

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 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>=2
Fin proc saisi
```

Exercice n°3 :

Soit la suite suivante :

$U_1 = 1$
$U_2 = 1$
$U_n = 4* U_{n-1} + 2 *U_{n-2}$ (pour n>2)

Pour un entier N donnée, calculez et afficher les n premier termes de la suite.

Analyse principale :

```
Résultat = proc suite (n)
           Proc saisi (n)
 Fin ex
```

TDO

Nom	Type
N	Entier
Suite, saisi	Procédure

Analyse de la procédure suite

```
Def proc suite(n :entier)
Résultat = []
[Up←1, U←1]
 pour i de 3 à n faire
          Upp←Up
          Up←U
          U←4*Up +2*Upp
          Ecrire(U)
       Fin pour
Fin proc suite
```

TDOL

Nom	Type
U, Up, Upp	entier
I	Entier

Analyse de la procédure saisi

```
Def proc saisi (var n :entier)
Résultat = x, n
N= répéter
```

N = donnée
Jusqu'à n>2
Fin proc saisi

Exercice n°4 :

Soient deux suites U et V définies par :

$U_1 = 1$ et $U_2=2$

$U_n= U_{n-1} + U_{n-2}$ pour n>=3

$V_n= U_n / U_{n-1}$ pour n>=2

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

On suppose que le n ème terme de la suite V soit Vn. Don un nombre approché du nombre d'or avec une précision epselon des que $| V_n – V_{n-1}| <$ epselon(10-4)

Analyse de la procédure suite

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)< 10^e-4
Fin proc suite

TDOL

Nom	Type
Upp, U, Up	Entier
V, Vp	Réel

Exercice 5 :

Soit la suite définie par :

$U_0=N/2$

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

- Ou n est un entier naturel donnée.
- En admettant que sa convergence, vérifiez que sa limite est racine de N.
- Calculer ses termes successives, tant que la différence entre deux termes consécutifs dépasse 10^{-6}, c'st à dire $| U_{n+1} - U_n| >10^{-6}$.
- Afficher le nombre d'itération effectué

Pascal :

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

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

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

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

procedure suite(n:integer);
var
   i:integer;
   up, u:real;
begin
   u:=n/2;
   repeat
        up:=u;
        u:=(up + n/up)/2
   until abs(u-up)<=1e-6;

   writeln(u:0:3);
end;

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

begin
    saisi(n);
    suite(n);
end.
```

II. Triangle de pascal :

```
Def proc triagle_pas(var m :mat ; n :entier)
Résultat = m
M[1,1]🡐1
M[2,1]🡐1
M[2,2]🡐1
Pour L de 3 à N faire
  Pour C de 1 à L faire
    Si (C=1) ou (c=L) alors M[L, C]🡐1
     Sinon M[L, C] 🡐 M[L-1, C-1]+ M[L-1, C]
     Fin si
  Fin pour
Fin pour
Fin proc triangle_pas
```

TDOL

Nom	Type
L, C	Entier

Pascal :

```
program ex;
uses wincrt;
type
mat = array[1..10, 1..10] of integer;
var
m:mat;
n:integer;
```

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

procedure saisi(var n: integer);
begin
    repeat
          writeln('donner n ');
          readln(n);
    until n>=3;
end;

procedure triangle_pas(n:integer; var m:mat);
var
 L, C:integer;
begin
    m[1,1]:=1;
    m[2,1]:=1;
    m[2,2]:=1;

    for L:=3 to n do
    begin
          for c:=1 to L do
          begin
              if (c=1) or (c=L) then m[L, C]:=1
           else M[L, c]:=m[L-1, c-1]+M[L-1,c];
        end;
    end;
end;

procedure affich(m:mat; n:integer);
var
   l, c:integer;
begin
    for l:=1 to n do
    begin
          for c:=1 to L do
          begin
            write(m[l,c]:5);
          end;
       writeln;
    end;
end;

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

begin
    saisi(n);
    triangle_pas(n, m);
    affich(m, n);
end.
```

Série N°8

Exercice 1 :

Saisir une matrice de façon aléatoire qui contient des entiers compris entre 10 et 50, sachant que la matrice est carrée de N ligne et colonne, avec n compris entre 3 et 10.

travail demandé :

- afficher la matrice.
- Calculer la somme de 1ere diagonale.
- Calculer la somme 2ème diagonale.
- Trier chaque ligne de la matrice par ordre croissant.
- La somme de chaque colonne.
- La somme total de la matrice.

Pascal :

```
program ex;
uses wincrt;

type
mat = array[1..10, 1..10] of integer;
var
m:mat;
n:integer;

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

procedure saisi(var n:integer; var m:mat);
var
i,j :integer;
begin
randomize;
```

```
    repeat
         writeln('donner n ');
         readln(n);
    until (n>=3) and (n<=10);

    for i:=1 to n do
    begin
        for j:=1 to n do
        begin
            m[i,j]:=random(50-10+1)+10;
        end;
    end;
end;

procedure tri_tab(nl:integer; n:integer;var m:mat);
var
 i:integer;
 aux:integer;
 perm:boolean;
begin
    repeat
        perm:=false;
        for i:=1 to n-1 do
        begin
            if m[nl,i]>m[nl, i+1] then
                        begin
                          perm:=true;
                          aux:=m[nl,i];
                          m[nl, i]:=m[nl,i+1];
                          m[nl, i+1]:=aux;
                        end;
        end;
    until perm=false;
end;

procedure tri(var m:mat ; n:integer);
var
 i:integer;
begin
    for i:=1 to n do
    begin
        tri_tab(i, n, m);
    end;
end;

procedure affich (m:mat; n:integer);
var
 i, j:integer;
begin
    for i:=1 to n  do
    begin
        for j:=1 to n do
        begin
            write(m[i, j]:5);
```

```
        end;
        writeln;
    end;
end;

function somme1(m:mat; n:integer):integer;
var
 s, i:integer;
begin
    s:=0;
    for i:=1 to n do
    begin
        s:=s+m[i, i];
    end;

    somme1:=s;
end;

function somme2(m:mat; n:integer):integer;
var
 s, i:integer;
begin
    s:=0;
    for i:=1 to n do
    begin
        s:=s+m[i, n-i+1];
    end;

    somme2:=s;
end;

function somme(m:mat; n:integer):integer;
var
 s, i, j:integer;
begin
    s:=0;
    for i:=1 to n do
    begin
      for j:=1 to n do
      begin
        s:=s+m[i, j];
      end;
    end;

    somme:=s;
end;

procedure somme_col(m:mat; n:integer);
var
  s, i, j:integer;
begin
    for j:=1 to n do
    begin
        s:=0;
        for i:=1 to n do
```

```
        begin
            s:=s+m[i,j];
        end;
        writeln(s);
    end;
end;
```

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

begin
    saisi(n, m);
    affich(m, n);
    writeln('la somme de 1er diago est ',somme1(m, n));
    writeln('la somme de 2eme diago est ', somme2(m, n));
    tri(m, n);
    writeln('************');
    affich(m, n);
    somme_col(m, n);
    writeln('somme total est ', somme(m, n));
end.
```

Exercice n°2 :

Saisir une matrice carré par N lettre majuscule, avec c nompris entre 4 et 30
concaténer chaque ligne de la matrice et le mettre dans un tableau T.
Afficher à partir du tableau T que les chaîne palindrôme.

Exemple :

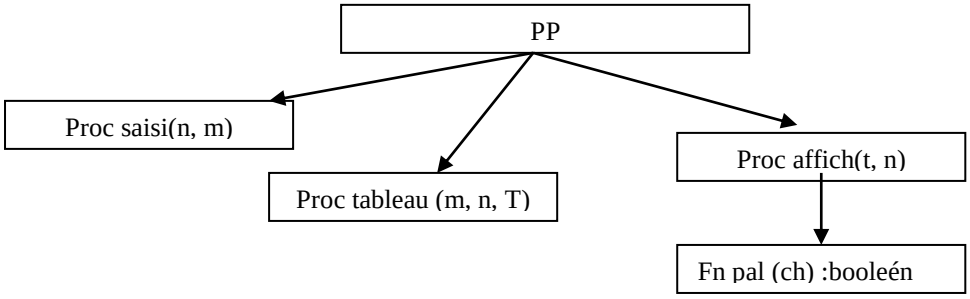
m

A	L	I	P
R	D	R	D
A	D	C	A
C	B	B	C

T

ALIP	RDRD	ADCA	CBBC

Les chaînes palindrome sont CBBC



Pascal :

```
program ex;
uses wincrt;

type
mat=array[1..10,1..10] of char ;
tab = array[1..10] of string;

var
 m:mat;
 n:integer;
 t:tab;

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

 procedure saisi(var n:integer; var m:mat);
 var
  i, j:integer;
 begin
   repeat
      writeln('donner n ');
      readln(n);
   until (n>=4) and (n <=30);

   for i:=1 to n do
   begin
      for j:=1 to n do
      begin
         writeln('donner m[',i,',',j,']');
         readln(m[i,j]);
      end;
   end;
 end;

 procedure tableau(m:mat; n:integer; var t:tab);
 var
   i, j:integer;
 begin
    for i:=1 to n do
    begin
       t[i]:='';
       for j:=1 to n do
       begin
          t[i]:=t[i]+m[i,j];
       end;
    end;
 end;

 function pal(ch:string):boolean;
 var
  i, j:integer;
 begin
    i:=0;
    j:=length(ch)+1;
```

```
    repeat
         i:=i+1;
         j:=j-1;
    until (ch[i]<>ch[j]) or (i>=j);

    if i>=j then pal:=true
    else pal:=false;
end;

procedure affich(t:tab; n:integer);
var
  i:integer;
begin
    for i:=1 to n do
    begin
        if pal(t[i])=true then writeln(t[i]);
    end;
end;

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

begin
 saisi(n, m);
 tableau(m, n, t);
 affich(t, n);
end.
```

Série N°9

Exercice 1 :

Saisir une matrice carrée de taille n avec n compris entre 5 et 20, sachant que la matrice M contient que des entiers positifs.

- Afficher la matrice
- Afficher la somme de la matrice.

Brouillon :

- saisi taille de la matrice n et saisir la matrice M ➔ proc
- afficher la martice ➔ proc
- la somme matrice ➔ fn

Analyse programme principale

Résultat = écrire (fn somme(m, n))
Porc affich(m, n)
Proc saisi (m, n)
Fin ex

TDNT

TYPE
MAT = tableau [1..10, 1..10] d'entier

TDOG

Nom	Type
M	Mat
N	Entier
somme	Fn
Saisi, affich	Proc

Analyse de la procédure saisi

```
Def proc saisi (var m :mat ; var n :entier) ;
Résultat = m, n
M=
Pour i de 1 à n faire
   Pour j de 1 à n faire
     M[i, j]=donnée
    Fin pour
Fin pour

Répéter
  N= donner
Jusqu'à n dans [ ?.. ?]
Fin proc saisi
```

TDOL

Nom	Type
I, j	Entier

Analyse de la fonction somme

```
Def fn somme  ( m :mat ; n :entier) :entier;
Résultat = somme←s
[s←0]
Pour i de 1 à n faire
   Pour j de 1 à n faire
     S←s+M[i, j]
    Fin pour
Fin pour
Fin proc saisi
```

TDOL

Nom	Type
I, j, s	Entier

Analyse de la procedure afich

```
Def proc affich  ( m :mat ; n :entier)
Résultat = [ ]

Pour i de 1 à n faire
   Pour j de 1 à n faire
     Écrire(M[i, j])
    Fin pour
Fin pour
Fin proc saisi
```

TDOL

Nom	Type
I, j, s	Entier

Pascal :

```
program ex;
uses wincrt;

type
mat = array[1..20, 1..20] of integer;

var
n:integer;
m:mat;

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

procedure saisi(var n:integer; var m:mat);
var
  i, j:integer;
begin
   repeat
```

```
        writeln('donner n ');
        readln(n);
    until (n>=3) and (n<=20);

    for i:=1 to n do
    begin
        for j:=1 to n do
        begin
            repeat
                writeln('donner m[',i,',',j,']');
                readln(m[i,j]);
            until m[i, j]>0;
        end;
    end;
end;

procedure affich(m:mat; n:integer);
var
 i, j:integer;
begin
    for i:=1 to n do
    begin
        for j:=1 to n do
         begin
          write(m[i,j]:5);
         end;
         writeln;
    end;
end;

function somme(m:mat; n:integer):integer;
var
 i, j, s:integer;
begin
    s:=0;
    for i:=1 to n do
    begin
        for j:=1 to n do
         begin
          s:=s+m[i,j];
         end;
    end;

    somme:=s;
end;



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

begin
    saisi(n, m);
    affich(m, n);
   writeln('la somme est ', somme(m, n));
```

end.

Exercice n°2 :

Saisir une matrice carrée de taille n avec n compris entre 5 et 20, sachant que la matrice M contient que des entiers positifs.

- La somme de la 1ère diagonale. ——————
- La somme de la 2ème diagonale. ----------

2	5	7	1	9
14	5	8	0	1
5	7	9	1	11
10	52	69	14	13
7	10	44	3	7

Analyse programme principale

Résultat = écrire (fn somme1(m, n), fn somme2(m, n))
Proc saisi (m, n)
Fin ex

TDNT

TYPE
MAT = tableau [1..10, 1..10] d'entier

TDOG

Nom	Type
M N Somme1, somme2,	Mat Entier Fn

Analyse de la procédure saisi

Def proc saisi (var m :mat ; var n :entier) ;
Résultat = m, n
M=
Pour L de 1 à n faire
  Pour C de 1 à n faire
    M[L, C]=donnée

Fin pour
Fin pour

Répéter
N= donner
Jusqu'à n dans [?.. ?]
Fin proc saisi

TDOL

Nom	Type
L, C	Entier

Analyse de la fonction somme1

Def fn somme1 (m :mat ; n :entier) :entier;
Résultat = somme1←s
[s←0]
Pour L de 1 à n faire
S←s+M[L, L]
Fin pour
Fin proc somme1

TDOL

Nom	Type
L, s	Entier

Analyse de la fonction somme2

Def fn somme2 (m :mat ; n :entier) :entier;
Résultat = somme2←s
[s←0]
Pour L de 1 à n faire
S←s+M[L, n-L+1]
Fin pour
Fin proc somme2

TDOL

Nom	Type
L, s	Entier

Exercice n°3

Saisir une matrice carrée de taille n avec n compris entre 5 et 20, sachant que la matrice M contient que des entiers positifs, afficher la somme de chaque ligne.

Analyse programme principale

Résultat = proc somme_lig(m, n)
Proc saisi (m, n)
Fin ex

TDNT

TYPE
MAT = tableau [1..10, 1..10] d'entier

TDOG

Nom	Type
M	Mat
N	Entier
Somme	Fn

Analyse de la procédure saisi

```
Def proc saisi (var m :mat ; var n :entier) ;
Résultat = m, n
M=
Pour L de 1 à n faire
   Pour C de 1 à n faire
     M[L, C]=donnée
    Fin pour
Fin pour

Répéter
  N= donner
Jusqu'à n dans [ ?.. ?]
Fin proc saisi
```

TDOL

Nom	Type
L, C	Entier

Analyse de la procedure_somme_lig

```
Def proc somme_ligne(m : mat ; n :entier)
Résultat
Pour i de 1 à N faire
  S←0
  Pour j de 1 à N faire
     S←S+m[i, j]
  Fin pour
  Ecrire(s)
Fin pour
```

TDOL

Nom	Type
I, j, s	Entier

Pascal :

```
program ex;
uses wincrt;
type
mat = array[1..20, 1..20] of integer;
var
s1, s2, n:integer;
m:mat;

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

procedure saisi(var n:integer; var m:mat);
```

```
var
  i, j:integer;
begin
    repeat
         writeln('donner n ');
         readln(n);
    until (n>=3) and (n<=20);


    for i:=1 to n do
    begin
       for j:=1 to n do
       begin
          repeat
              writeln('donner m[',i,',',j,']');
              readln(m[i,j]);
          until m[i, j]>0;
       end;
    end;
end;

          procedure somme_lig(m:mat ; n:integer);
          var
           i,j,s :integer;
          begin
            for i:=1 to n do
            begin
               s:=0;
               for j:=1 to n do
               begin
                  s:=s+m[i,j];
               end;
               writeln('la somme de la ligne ', i,' est ', s);
            end;
          end;



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

    begin
      saisi(n, m);
      somme_lig(m, n);
    end.
```

Exercice n°4 :

Ecrire un programme permettant de saisir une matrice de L ligne et C colonne, avec L compris entre 2 et 5 et C est compris entre 3 et 7.
Afficher la valeur maximale de la matrice.
Afficher la valeur maximale de chaque ligne
Afficher la valeur minimale de chaque colonne

5	2	11	3

4	8	21	3
7	16	1	0

La valeur maximale de la matrice est 21
La valeur maximale pour chaque ligne :

- ligne 1 = 11
- ligne 2 = 21
- ligne 3 = 16

Analyse principale :

Résultat = écrire(fn max(M, L, C))
Proc affich_max (M, L, C)
Proc saisi (M, L, C)
Fin ex

TDNT

Type
Mat = tableau [1..5, 1..7] d'entier

TDO

Nom	Type
M	Mat
L, C	Entier
saisi, affich_max	Procédure
max	Fonction

Analyse de la procédure saisi

Def proc saisi(var L, c :entier ; var m :mat)
Résultat = l, c, m
Répéter
L= donnée
Jusqu'à L dans [2..5]

Répéter
C= donnée
Jusqu'à C dans [3..7]

Pour i de 1 à L faire
Pour j de 1 à C faire
M[i, j]=donnéd
Fin pour
Fin pour
Fin proc saisi

TDOL

Nom	Type
I, j	Entier

Analyse de la procédure affich_max

```
def proc affich_max(m :mat ; L, C :entier)
résultat = []
[] = Pour i de 1 à L faire
        Max←m[i,1]
        Pour j de 2 à C faire
          Si max<m[i, j] alors max← m[i, j]
           Fin si
        Fin pour
          Ecrire(max)
      Fin pour
Fin proc affich_max
```

TDOL

Nom	Type
i, j, max	Entier

Analyse de la fonction max

```
Def fn max (M :mat ; L, C :entier) :entier
Résultat = max← maxi
[maxi←m[1,1]]
Pour i de 1 à L faire
  Pour j de 1 à C faire
    Si maxi<m[i, j] alors maxi← m[i, j]
     Fin si
  Fin pour
Fin pour
Fin fn max
```

TDOL

Nom	Type
I, j	Entier

Analyse de la procédure affich_min

```
def proc affich_min(m :mat ; L, C :entier)
résultat = []
[] = Pour j de 1 à C faire
        Max←m[1,j]
        Pour j de 2 à L faire
          Si min > m[i, j] alors min ← m[i, j]
           Fin si
        Fin pour
          Ecrire(min)
      Fin pour
Fin proc affich_min
```

TDOL

Nom	Type
i, j, min	Entier

Exercice n°5 :

Ecrire un programme permettant de saisir une matrice carré de n entier, avec n compris entre

4 et 10, afficher la valeur maximale dans sa ligne et le minimum dans sa colonne.
Exemple :

5	25	15	9
20	30	4	5
1	30	6	38
-1	36	4	14

Le programme affiche 25, parce que c'est le maximum dans sa ligne et le minimum dans sa
colonne.

Pascal :

```
program ex;
uses wincrt;
type
mat = array[1..10, 1..10] of integer;
var
m:mat;
n:integer;

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

procedure saisi(var m:mat; var n:integer);
var
i,j:integer;
begin
   repeat
          writeln('donner la taille ');
        readln(n);
   until n>0;

   for i:=1 to n do
   begin
        for j:=1 to n do
        begin
          write('donner m[',i,',',j,']= ');
         readln(m[i,j]);
      end;
   end;
end;

function  max_ligne(nl:integer; m:mat; n:integer):integer;
var
j, max, ind_max:integer;
begin
    max:=m[nl, 1];
    ind_max:=1;
    for j:=2 to n do
```

```
    begin
        if m[nl,j]>max then
                        begin
                        max:=m[nl, j];
                        ind_max:=j;
                    end;
    end;

    max_ligne:=ind_max;
end;

function  min_col(nc:integer; m:mat; n:integer):integer;
var
i, min, ind_min :integer;
begin
    min:=m[1, nc];
    ind_min:=1;
    for i:=2 to n do
    begin
        if m[i,nc]<min then
                        begin
                         min:=m[i, nc];
                         ind_min:=i;
                    end;

    end;

    min_col:=ind_min;
end;
procedure affich (m:mat; n:integer);
var
i, c, nc, L:integer;
begin

   for i:=1 to n do
   begin
         nc:=max_ligne(i, m, n);
         L:=min_col(nc, m, n);
         if i=L then writeln(m[i,nc]);
   end;
end;

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

begin
   saisi(m, n);
   affich (m, n);
end.
```

Exercice n°6 :

Ecrire un programme permettant de saisir une matrice carrée (N*N) et vérifier si la matrice est symétrique par rapport au deuxième diagonale ou non

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

Analyse de la procédure symétrique

```
Def fn sym(m :mat ; n :entier) :booleén
Résultat = sym←test
Test←vrai
Pour i de 1 à n faire
   Pour j de 1 à n faire
      Si m[i,j]<>m[n-j+1, n-i+1] alors test←faux
       Fin si
  Fin pour
Fin pour
Fin fn sym
```

TDOL

Nom	Type
I,j	Entier