Sommaire

Résumé de cours

Les opérateurs arithmétiques et leur priorités

Soit l'opération suivante: res = A + B

res : résultat
A et B : opérandes
+ : opérateur

Désignation de l'opération	Priorité des opérateurs	Opérateur		Type des opérandes
		En algorithmique	*En Pascal*	
Parenthèses	1	(…)	(…)	Tout type
Multiplication	2	x	*	Entier ou réel
Division réelle		/	/	Réel
Division entière		DIV	DIV	Entier
Reste de la division entière		MOD	MOD	Entier
Addition	3	+	+	Entier ou réel
Soustraction		-	-	Entier ou réel
Egale	4	=	=	Tout type ordonné
Différent		≠	<>	Tout type ordonné
Inférieur		<	<	Tout type ordonné
Supérieur		>	>	Tout type ordonné
Inférieur ou égale		≤	<=	Tout type ordonné
Supérieur ou égale		≥	>=	Tout type ordonné
L'appartenance	5	DANS	IN	Type scalaire

<u>Remarque</u>: les opérateurs de même niveau de priorité seront évalués de gauche vers la droite (vous pouvez utiliser les parenthèses pour modifier l'ordre d'évaluation des expressions).

Les fonctions arithmétiques standards

Syntaxe en algorithme	*Syntaxe en Pascal*	*Rôle de la fonction*	*Type de x*	*Type de résultat*	*Exemples*
Abs (x)	**ABS (x)**	Retourne la valeur absolue de x.	***Entier ou réel***	***Même type que x***	R:=ABS(-6);⇨R=6 R:=ABS(-7.5); ⇨R= 7.5
Arctan (x)	**ARCTAN (x)**	Retourne la valeur en radians de l'arc tangente de x.	***Réel***	***Réel***	R:=ARCTAN(1); ⇨R= PI/4 R:=ARCTAN(0.48); ⇨R=PI/6
Arrondi (x)	**ROUND (x)**	Retourne l'entier le plus proche de x.	***Réel***	***Entier***	R:=ROUND(7.4);⇨R=7 R:=ROUND(7.5);⇨R=8 R:=ROUND(7.9);⇨R=8
Carré (x)	**SQR (x)**	Retourne le carré de x.	***Entier ou réel***	***Même type que x***	R:=SQR(3);⇨R=9 R:=SQR(3.5); ⇨R=12.25
Cos (x)	**COS (x)**	Retourne le cosinus de x (x en radians).	***Réel***	***Réel***	R:=COS(PI/2);⇨R= 0 R:=COS(PI);⇨R= -1
Exp (x)	**EXP (x)**	Retourne l'exponentielle de x.	***Réel***	***Réel***	R:=EXP(1);⇨R= 2.72 R:=EXP(-3.5);⇨R= 0.03
Ln (x)	**LN (x)**	Retourne le logarithme népérien de x si x est positif sinon il provoque une erreur.	***Réel***	***Réel***	R:=LN (1);⇨R= 0 R:=LN(3.5);⇨R=1.25
RacineCarré (x)	**SQRT(x)**	Retourne la racine carré de x si x est positif sinon il provoque une erreur.	***Réel***	***Réel***	R:=SQRT(4); ⇨R= 2 R:=SQRT(20.45); ⇨R= 4.52
Sin (x)	**SIN (x)**	Retourne le sinus de x (x en radians)	***Réel***	***Réel***	R:=SIN(PI/2);⇨R= 1 R:=SIN(PI); ⇨R= 0
Tronc (x)	**TRUNC (x)**	Retourne un entier, en ignorant la partie décimale de x.	***Réel***	***Entier***	R:=TRUNC(-1.5); ⇨R=-1 R:=TRUNC(9.5); ⇨R= 9

Les fonctions standard sur les caractères

Syntaxe en algorithme	Syntaxe en Pascal	Rôle de la fonction	Type de paramètre	Type de résultat	Exemples
CHR (N)	**CHR (N)**	Retourne le caractère dont le code ASCII est N.	***Entier***	***Caractère***	R := CHR (65) ; ⇨R sera égal à 'A' R := CHR (97) ; ⇨R sera égal à 'a'
ORD (C)	**ORD (C)**	Retourne le code ASCII du caractère C.	***Caractère***	***Entier***	R := ORD ('D') ; ⇨R sera égal à 68. R := ORD ('0') ; ⇨R sera égal à 48.
PRED (C)	**PRED (C)**	Retourne le prédécesseur de C (c'est à dire qui précède C).	***Scalaire***	***Même type de C***	N :=PRED (4) ; ⇨N sera égal à 3. R :=PRED ('D') ; ⇨R sera égal à 'C'
SUCC (C)	**SUCC (C)**	Retourne le successeur de C (c'est à dire qui suit C).	***Scalaire***	***Même type que C***	N := SUCC (3) ; ⇨ N sera égal à 4. R:= SUCC ('C'); ⇨R sera égal à 'D'
MAJUS (C)	**UPCASE (C)**	Convertir le caractère C en majuscule s'il est possible.	***Caractère***	***Caractère***	R:= UPCASE ('e'); ⇨R sera égale à E'. R:= UPCASE ('F'); ⇨R sera égale à 'F'

Les fonctions standard sur les chaînes de caractères

Syntaxe en algo	Syntaxe en Pascal	Rôle de la fonction	Exemples
Long (ch)	**LENGTH (ch)**	Retourne **un entier** représentant la longueur de la chaîne **ch.**	L:=LENGTH('Algorithme'); ⇨L= 10 L:=LENGTH(' Pascal'); ⇨L= 7
Concat (ch1, ch2, …, chn)	**CONCAT(ch1, ch2, …,chn)**	Retourne **une chaîne** qui est la somme de plusieurs chaînes dans l'ordre.	CH:=CONCAT('micro-','ordinateur'); ⇨CH= 'micro-ordinateur' CH:=CONCAT('Turbo',' ','Pascal'); ⇨CH= 'Turbo Pascal'
Sous_chaîne (ch,p,n)	**COPY (ch,p,n)**	Retourne une **sous-chaîne** de longueur **N** à partir de la position **p** dans **ch.**	CH:=COPY('Baccalauréat',1,3) ; ⇨CH= 'Bac' CH:=COPY('micro-ordinateur',7,10); ⇨CH= 'ordinateur'
Pos (ch1,ch2)	**POS (ch1,ch2)**	Retourne un **entier** représentant la position de la première occurrence de la chaîne **ch1.** Si **ch1** n'est pas dans **ch2,** elle retourne 0.	P:=POS('m','programmation');⇨P= 7 P:=POS('r','programmation');⇨P= 2 P:=POS('R','programmation');⇨P= 0

Les procédures standard sur les chaînes de caractères

Syntaxe en algo	Syntaxe en Pascal	Rôle de la fonction	Exemples
Efface (ch,p,n)	**DELETE (ch,p,n)**	Enlève **n** caractères de la chaîne **ch** à partir de la position **p**.	CH :='programmation' ; delete('programmation', 8,6); ⇨CH= 'program'
Insérer (ch1,ch2,p)	**INSERT (ch1,ch2,p)**	Insère la chaîne **ch1** dans la chaîne **ch2** à partir de la position **p**. Le caractère n°p et les suivants seront décalés vers la droite.	CH1 := '-' ; CH2 := 'Hautparleurs' INSERT (CH1,CH2,5) ; ⇨CH2= 'Haut-parleurs'
Convch (n,ch)	**STR (n,ch)**	Convertit une valeur numérique en une chaîne de caractères et l'affecte à la variable **ch**.	STR (2002, CH) ; ⇨CH= '2002' STR (15.54, CH) ; ⇨CH= '1.5540000000E+01'
Valeur (ch,n, pe)	**VAL (ch,n, pe)**	Convertit une chaîne **ch** en une valeur numérique et l'affecte à la variable **n**. Le paramètre **pe** est une variable entière qui contiendra la position de l'erreur.	VAL ('2003',n,pe) ; ⇨ n= 2003 et pe=0 VAL('06/08/1970', n,pe) ; ⇨ n= 0 et pe= 3 (le caractère / n'est pas un chiffre).

Les structures simples

Affectation		
Analyse	**Algorithmique**	**Pascal**
Variable ← expression **Exemple** A ← 8 A ← 3*5/2 A ← B A ← 3 < 5		**Variable := expression ;** **Exemple** A := 8 A := 3*5/2 A := B A := 3 < 5
Opération d'entrée		
Analyse	**Algorithmique**	**Pascal**
Variable=donnée("Message") **Exemple** a = Donnée ("Donner la valeur de a")	**Ecrire("Message")** **Lire(Variable)** **Exemple** Ecrire("Donner la valeur de a") Lire(a)	**Write('Message') ;** **Read(Variable) ; ou** **Readln(Variable) ;** **Exemple** Write('Donner la valeur de a'); Readln(a) ;

Opération de sortie		
	Analyse / Algorithmique	Pascal
Affichage d'un texte (message)	**Ecrire("Message")** **Exemple** Ecrire("Donner la valeur de a")	**Write('Message');** **ou Writeln('Message') ;** **Exemple** Writeln('Donner la valeur de a') ;
Affichage de contenu d'une variable	**Ecrire(variable)** **Exemple** Ecrire(variable)	**write(variable); ou** **Writeln(variable) ;** **Exemple** writeln(variable) ;
Affichage d'un texte et d'une variable	**Ecrire("Message",variable)** **Exemple** Ecrire("Le produit est", P)	**write('Message',variable); ou** **Writeln('Message', variable);** **Exemple** writeln('Le produit est', P) ;

Les structures conditionnelles

La forme simple réduite		
Analyse / Algorithmique		Pascal
Si condition **alors** Traitement **FinSi**		**If** condition **then** Traitement ;

La forme alternative		
Analyse / Algorithmique		Pascal

Si condition **alors**
Instruction 1.1
Instruction 1.2
…….
Instruction 1.n
Sinon
Instruction 2.1
Instruction 2.2
…….
Instruction 2.n
FinSi
		If condition **then**
		Begin
		Instruction 1.1 ;
		Instruction 1.2 ;
		…….
		Instruction 1.n ;
		End
		Else
		Begin
		Instruction 2.1 ;
		Instruction 2.2 ;
		…….
		Instruction 2.n ;
		End ;

La forme généralisée		
Analyse	**Algorithmique**	**Pascal**
[init] **Si** condition 1 **alors** traitement 1 **Sinon** **Si** condition 2 **Alors** traitement 2 **Sinon** **Si condition 3 Alors** traitement 3 …….. …….. **Sinon** **Si** condition n-1 **Alors** traitement n-1 **Sinon** traitement n **FinSi**		**Init;** **IF** condition1 **THEN** traitement1 **ELSE IF** condition2 **THEN** traitement2 **ELSE IF** condition3 **THEN** traitement3 …….. …….. **ELSE IF** condition n-1 **THEN** traitement n-1 **ELSE** traitement n **;**

La structure de contrôle conditionnelle à choix		
Analyse	**Algorithmique**	**Pascal**
[init] **Selon** sélecteur **Faire** ***Valeur_1*** : traitement 1 ***Valeur_2*** : traitement 2 ***Valeur_3*** : traitement 3 …… ***Valeur_n*** : traitement n **Sinon** traitement n+1 **Fin selon**		**Init;** **Case** sélecteur **OF** ***Valeur_1*** : traitement 1 ; ***Valeur_2*** : traitement 2 ; ***Valeur_3*** : traitement 3 ; …… ***Valeur_n*** : traitement n ; **else** traitement n+1 ; **End ;**

Les structures itératives: les boucles

<table>
<tr><th colspan="2"></th><th>Définition itérative complète</th><th colspan="2">Définition itérative à condition d'arrêt</th></tr>
<tr><td rowspan="2">Forme Générale</td><td>Analyse</td><td>Résultat=[init]
Pour compteur de Vi à Vf faire
Traitement
Fin Pour</td><td>Formulation 1
Résultat=[init]
Répéter
Traitement
Jusqu'à (Condition arrêt)</td><td>Formulation 2
Résultat = [Init]
Tant que(Non(Cond_arrêt) faire
Traitement
Fin tant que</td></tr>
<tr><td>Pascal</td><td>…..;
…. ;
For compteur : = Vi To Vf Do
Begin
….. ;
….. ;
End ;</td><td>….. ;
…… ;
Repeat
…… ;
……. ;
Until(condition arrêt);</td><td>….. ;
…… ;
While (Not(cond_arrêt)) Do
Begin
…… ;
……. ;
End ;</td></tr>
<tr><td colspan="2">Types</td><td>Compteur de type scalaire discret
(Entier / caractère)</td><td></td><td></td></tr>
<tr><td colspan="2">Nombre d'itération</td><td>Connu d'avance
Pour les entiers :
Vf – Vi + 1
Pour les caractères :
Ord (Vf) – Ord (Vi) + 1</td><td>Inconnu
Au moins une fois</td><td>Inconnue
Si la condition d'arrêt est vraie, la boucle ne s'exécute jamais.</td></tr>
<tr><td colspan="2">Initialisation & incrémentation</td><td>Le compteur est initialisé à Vi
L'incrémentation automatique se fait de 1</td><td colspan="2">Les variables qui figurent dans l'expression de la condition doivent être initialisées par l'utilisateur. Elles doivent être modifiées à l'intérieur de la boucle, sinon le boucle ne s'arrête pas. (Boucle infini).</td></tr>
</table>

Vocabulaire et syntaxe d'une fonction

Au niveau de la définition (la création et la rédaction) d'une fonction:

- **En analyse**

```
DEF FN nom_de_fonction(pf1,pf2,…:typeI;pf'1,pf'2,…:typeII ;…):type-résultat
Résultat = nom_de_fonction
nom_de_fonction ← R
…..
Fin nom_de_fonction
```

Remarque :

pf1,pf2,… : **Paramètres formels**

type-résultat: **Résultat à chercher dans le bloc du module** doit être de type simple (entier, réel, caractère, booléen, chaine de caractère). Une fonction ne peut pas retourner un tableau ou deux entiers par exemple.

R: étant le résultat de la fonction qui doit être de même type ou de type compatible que **la fonction.**

- **En algorithme**

```
0) DEF FN nom_de_fonction(pf1,pf2,… : typeI; pf'1,pf'2,… :typeII ;…):type-résultat
1)   … <traitement>
     …..
n-1) nom_de_fonction ← R
n) FIN nom_de_fonction
```

- **En Pascal**

```
FUNCTION nom_de_fonction (pf1,pf2,…:typeI;pf'1,pf'2,…: typeII;):type-résultat;
Const = ……..; {constantes locales}
Type = ……..; {types locaux}
Var   …….; {variables locales}
Begin
……. <traitement>
……..
nom_de_fonction := R;
End;
```

Au niveau de l'appel :

Une fonction peut être appelée à partir:

- du programme principal
- d'un autre sous-programme (module)

Une fonction peut être appelée de quatre manières

- Dans une instruction d'affectation:

  M←FN nom_de_fonction(pe1,pe2,…,pe'1,pe'2,…)

- Dans une instruction de sortie (écriture) :

  Ecrire(FN nom_de_fonction(pe1,pe2,…,pe'1,pe'2,…))

- Dans une instruction conditionnelle :

  Si (FN nom_de_fonction(pe1,pe2,…,pe'1,pe'2,…)=Y) alors ..

- Dans une expression arithmétique :

  Y ← 2+FN nom_de_fonction(pe1,pe2,…,pe'1,pe'2,…)

RETENONS: Les paramètres formels doivent s'accorder du point de vue **nombre**, **ordre** et **types** compatibles avec les paramètres effectifs.

Vocabulaire et syntaxe d'une procédure

Au niveau de la définition (la création et la rédaction) d'une Procédure :

✍ **En analyse**

```
DEF PROC nom_de_procédure(pf1,pf2,… :typeI ; pf1,pf2,… :typeII ;…)
Résultat = ……
      <traitement>
      …..
Fin nom_de_procédure
```

Remarque : pf1,pf2,… : **Paramètres formels**

✍ **En algorithme**

```
0) DEF PROC nom_de_procédure(pf1,pf2,…:typeI ; pf1,pf2,…: typeII;
   …)
1) <traitement>
   ……..
n) FIN nom_de_procédure
```

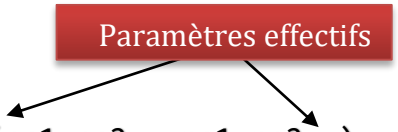
✍ **En Pascal**

```
PROCEDURE nom_de_procédure (pf1,pf2,… :typeI ;
pf'1,pf'2,… :typeII ;…);
Const = ……..; {constantes locales}
Type = ……..; {types locaux}
Var  …….; {variables locales}
Begin
        …<traitement>
End;
```

Au niveau de l'appel :

Une procédure peut être appelée à partir:

- ❖ du programme principal
- ❖ d'un autre sous-programme (à condition qu'il soit déclaré à l'intérieur de ce sous-programme ou avant)




```
PROC nom_de_procédure (pe1,pe2,…,pe1,pe2,…)
```

RETENONS: Les paramètres formels doivent s'accorder du point de vue **nombre**, **ordre** et **types** compatibles avec les paramètres effectifs.

Tri par sélection:

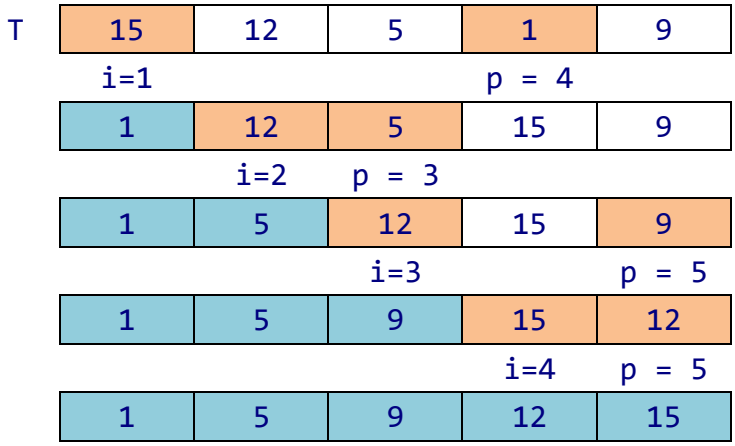
Principe

On recherche le plus petit (ou le plus grand) élément et on l'extrait, on recherche ensuite le plus petit des éléments qui restent et on l'extrait et on recommence ainsi jusqu'à ce que l'on ait extrait tous les éléments.

Méthode

1. Sélectionner le premier élément du tableau
2. Parcourir le reste du tableau à la recherche de l'élément ayant la plus petite valeur.
3. Si le cas est présent permuter les deux éléments
4. Incrémenter le premier indice
5. Refaire les étapes 2, 3, et 4 jusqu'à la fin du tableau.

Exemple: Trier dans l'ordre croissant le tableau suivant:



Algorithme:

```
0) DEF PROC Tri_selection (var T : tab ; n : entier)
1) pour i de 1 à n-1 faire
   [posmin ←i ] pour j de i+1 à n faire
                        si T[j]< T[posmin] alors
                              posmin ←j
                        fin si
                  fin pour
            si posmin <> i alors
                  aux← T[i]
                  T[i]← T[posmin]
                  T[posmin]←aux
             Fin si
        Fin pour
    2) Fin trier
```

Tri à bulle

Principe

A chaque étape on considère un couple d'éléments; s'ils ne sont pas dans le bon ordre l'un par rapport à l'autre, on les permute; on répète ce procédé en changeant de couple et ce, jusqu'à ce que plus aucun échange ne soit nécessaire.

Méthode

On effectue des passages successif sur le tableau on examinent les éléments du tableau par paire V[i] et V[i+1] et on échange les valeurs si V[i]>V[i+1], chaque passage s'arrête au niveau du dernier échange du passage précédent ainsi le 1ér passage est allé jusqu'à V[N]. On arrête les passages dés que l'un d'entre eux n'entraîne aucun échange.

Exemple

Trier dans l'ordre croissant le tableau suivant:

T				
15	3	5	1	10
i=1	i+1=2			
3	15	5	1	10
	i=2	i+1=3		
3	5	15	1	10
		i=3	i+1=4	
3	5	1	15	10
			i=4	i+1=5
3	5	1	10	15
i=1	i+1=2			
3	5	1	10	15
	i=2	i+1=3		
3	1	5	10	15
		i=3	i+1=4	
3	1	5	10	15
			i=4	i+1=5
3	1	5	10	15
i=1	i+1=2			
1	3	5	10	15
	i=2	i+1=3		
1	3	5	10	15
	i=2	i+1=3		
1	3	5	10	15
i=1	i+1=2			

Algorithme:

```
0) DEF PROC Tri_bulle (var T : tab ; n : entier)
1) Répéter
        [Echange←faux] pour i de 1 à n-1 faire
                Si (T[i]>T[i+1]) alors
                    aux← T[i]
                    T[i]← T[i+1]
                    T[i+1]←aux
                    Echange← vrai
                Fin Si
            Fin pour
        n←n-1
   Jusqu'à (n=1) ou non(Echange)
   2) Fin trier
```

Tri par insertion

Principe

On ne considère qu'un seul élément à la fois et on essaye de l'insérer en position correcte par rapport à un sous-ensemble d'éléments précédemment triés: le tri se terminera lorsque tous les éléments auront été insérés.

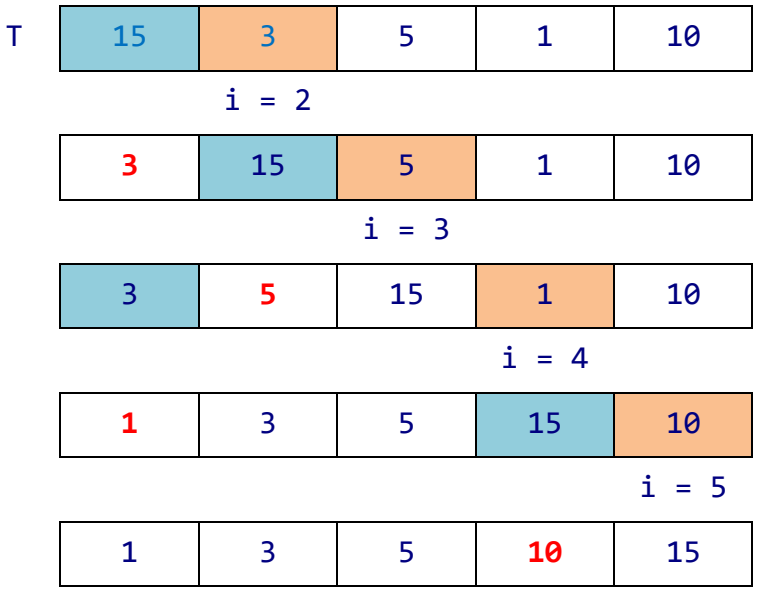
Méthode

1. On commence par le deuxième élément
2. Tant que l'élément précèdent supérieur à l'élément sélectionner on le décale à droite
3. On insert l'élément sélectionner en position correcte
4. on sélectionne l'élément suivant
5. On répète les étapes 2, 3 et 4 jusqu'au dernier élément du tableau.

Algorithme:

```
0) DEF PROC Tri_insertion(N : Entier ; Var T : Tab)
1) pour i de 2 à n faire
        aux ← T[i]
        j← i
        Tant que (T[j-1]>aux) faire
            T[j]←T[j-1]
            j←j-1
        Fin Tant que
        T[j]←aux
2) Fin Tri_insertion
```

Exemple: Trier dans l'ordre croissant le tableau suivant:



3 < 15 alors décalage de 15 vers la droite et insertion de 3 dans sa place

5 > 3 mais 5 < 15 alors décalage de 15 vers la droite et insertion de 5 dans sa place

1 < 3 alors décalage de 3, 5 et 15 vers la droite et insertion de 1 dans la place de 3

10 < 15 alors décalage de 15 vers la droite et insertion de 10 dans sa place

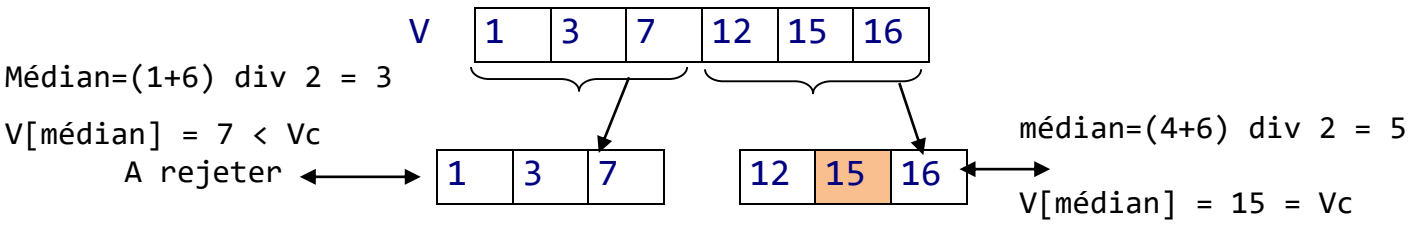
Recherche dichotomique

Principe

Cette méthode de recherche est applicable uniquement sur une liste préalablement triée. Le principe de cette méthode consiste à réduire à chaque fois l'espace de recherche sur la moitié de la liste. Le choix de l'une des deux parties dans laquelle se trouve la valeur cherchée est le résultat de l'évaluation d'un test. A chaque évaluation, on coupera l'espace de recherche en deux parties égales (à un élément près) de part et d'autre de l'élément médiane.

Exemple

Valeur chercher: Vc= 15



Algorithme:

```
0) DEF FN Recherche (A: TAB ; n, Vc : entier) : chaîne
1) [g ←1,d← n] répéter
                          m← (g+d) div2
                          Si y<A[m] alors
                           d ←m-1
                           Sinon
                                g← m+1
                   Fin Si
     jusqu'à (Vc=A[m]) ou (g>d)
2) si (y=A[m] ) alors
          Recherche ←  "existe"
Sinon
          Recherche← "non existe"
Fin si
3) Fin recherche
```

Structure Générale d'un Programme Pascal

```
PROGRAM Nom_programme ;
Uses     wincrt ;
Const    {déclaration des constantes} ;
Type     {déclaration des types de données} ;
Var      {déclaration des variables globales} ;

     PROCEDURE Nom_Procédure (paramètres formels) ;
          {Déclarations locales}
     Begin
               Instructions de la procédure ;
     End ; {fin de la procédure}

     FUNCTION Nom_Fonction (paramètres formels):type ;
          {Déclarations locales}
     Begin
               Instructions de la fonction ;
               Nom_Fonction := Résultat ;
     End ; {fin de la fonction}

BEGIN    {début du Programme Principal}

{Bloc principal du programme avec appel des
procédures et des  fonctions}

END.{ fin du P.P }
```

Remarques:

1. l'ordre de l'écriture des sous programmes n'est pas important, à l'exception ou un module utilise un autre module; dans ce cas il faut que le programme appelé soit écrit avant le programme appelant
2. un module ne s'exécute jamais s'il n'a pas été appelé par le programme principal

Exercices

Questions demandées pour tous les exercices :

1. **Analyser ce problème en décomposant en modules.**
2. **Analyser chacun de ces modules.**
3. **En déduire l'algorithme de résolution relatif à chacun de ces modules ainsi que celui du programme principal.**
4. **Traduire les algorithmes obtenus en Pascal**

1

Un entier naturel est dit parfait s'il est égal à la somme de tous ses diviseurs autres que lui-même.

Exemple :

6 est dit un nombre parfait car il vérifie 6 = 1 + 2 + 3 où 1,2 et 3 sont les diviseurs de 6 autres que 6.

On se propose de chercher tous les entiers parfaits compris entre deux valeurs données m et n telles que 2<m<n.

2

Soit un tableau T1 de N1 éléments (1≤N1<100). Les éléments de T1 sont des entiers naturels de trois chiffres.

On se propose de remplir et afficher un tableau T2 de la façon suivante :

T2[i] est égal à la somme des carrés des chiffres de T1[i].

Exemple :

Si ***T1[i] = 254*** *alors*

T2[i] = 2^2 + 5^2 +4^2 = 45

3

Un entier naturel est dit palindrome s'il est lu de la même façon de gauche à droite et de droite à gauche.

Exemple : *121 est un entier impair et palindrome.*

On cherche à afficher les N premiers entiers naturels positifs impairs et palindromes (N étant un entier naturel tel que 5 ≤ N ≤ 20).

4

On se propose d'élaborer un programme qui permet de calculer la somme factorielle des chiffres de tous les entiers de l'intervalle [10..50], en les sauvegardant dans un vecteur T.

Exemple: *Si l'entier égale à 14 alors la valeur 25 sera stoker dans T (car 1! + 4! = 25).*

Enfin afficher i fois le contenu de la case n° i (pour chaque case du tableau), en commençant par le dernier élément du vecteur T.

5

Elaborer un programme qui permet de saisir un vecteur T par n chaînes de caractères (4< n <20), cherche et affiche la longueur de la chaîne la plus longue puis affiche toutes les chaînes ayant cette longueur.

6

24 est un entier divisible par son chiffre des dizaines (2).

Elaborer un programme permettant de trouver et sauvegarder dans un vecteur V tous les entiers à deux chiffres vérifiant cette propriété, puis afficher les éléments de ce dernier : les éléments d'indice pair suivi par ceux d'indice impair.

7

Ecrire un programme Pascal intitulé **PROD_SCALAIRE** qui permet de calculer et d'afficher le produit scalaire de deux tableaux A et B de n entiers positifs (n étant un entier compris entre 5 et 50).

N.B : *Le produit scalaire de deux tableaux A et B est donné par la formule suivante :*

$$PS = \sum_{i=1}^{n} A[i] * B[i]$$

8

On appelle moyenne olympique d'un ensemble de nombres la moyenne arithmétique de tous les nombres de cet ensemble sauf le plus petit et le plus grand.

Ecrire un programme Pascal permettant de saisir un tableau de N réels (5≤ N ≤20) distincts et d'afficher leur moyenne olympique.

9

Ecrire un programme permet de lire un code d'ADN sous forme d'un tableau D de n caractères (5≤n≤ 30). Puis déterminer et afficher le code d'ARN (sous forme d'un tableau R) correspondant. Sachant que le code d'ADN utilise les lettres **A, T, C** et **G** et le code ARN correspondant est obtenu par correspondance de base :

A⇨U ‖ T⇨A ‖ C⇨G ‖ G⇨C

Exemple : (pour n = 5)

	1	2	3	4	5
D (tableau ADN)	T	A	C	G	T

	1	2	3	4	5
R (tableau ARN)	A	U	G	C	A

10

Ecrire un programme qui permet de saisir une chaîne de caractères CH_1 puis d'en extraire les deux nombres formés par les chiffres figurant dans la chaîne CH_1 (extraction à partir de la droite puis extraction à partir de la gauche).

Exemple : *Si CH1= "A45B3C2"*
Le programme Pascal affichera 4532 et 2354

11

Ecrire un programme Pascal qui permet d'insérer un entier X dans un tableau **trié** T en conservant le tableau trié.

Sachant que T et de N entiers (5 ≤ N < 30)

NB: *La solution doit comporter au moins une fonction et une procédure.*

12

Ecrire un programme pascal qui permet de saisir une chaîne non vide de longueur impaire et de l'afficher sous la forme d'un sablier

Exemples:

Ch ="**SABLIER**"

```
S  A  B  L  I  E  R
   A  B  L  I  E
      B  L  I
         L
      B  L  I
   A  B  L  I  E
S  A  B  L  I  E  R
```

Ch="**ECRAN**"

```
E  C  R  A  N
   C  R  A
      R
   C  R  A
E  C  R  A  N
```

13

Ecrire un programme qui saisit une phrase et l'affiche renversée. La phrase commence, obligatoirement, par une lettre et ses mots sont séparés par un seul espace et ne se termine pas par un espace.

Exemple :

*Votre phrase:"**RESOLUTION DE PROBLEMES**"*

*Résultat:"**PROBLEMES DE RESOLUTION**"*

14

Soit un tableau T de 20 entiers positifs. Ecrire un programme qui permet d'afficher les éléments de T compris entre deux positions P1 et P2, leur moyenne arithmétique, la valeur maximale et la valeur minimale contenues dans cet intervalle. On donne $1 \leq P1 < P2 \leq 20$.

15

Soit un tableau T1 contenant n lettres majuscules (de A à Z).

n étant un entier compris entre 5 et 20.

On désire trier en ordre croissant les éléments de T1 et les ranger dans un tableau T2 en utilisant le principe suivant :

1. Chercher la lettre qui a le plus petit code ASCII dans T1
2.
a) Ranger cette lettre dans T2
b) Remplacer cette lettre par "*" dans T1
3. Répéter n fois l'étape 1. et 2.

Ecrire un programme Pascal qui permet de :

☑ Saisir les éléments de T1.
☑ Trier les éléments de T1 et les ranger dans T2.
☑ Afficher les éléments de T2.

16

Ecrire un programme qui permet de trier par ordre décroissant les éléments d'un tableau A de n entiers positifs dans un nouveau tableau B de même dimension.

N étant un entier vérifiant **5 < n < 25.**

On utilisera la démarche suivante:

1. chercher le maximum de A
2. placer ce maximum dans B
3. remplacer le maximum par -1 dans A
4. refaire l'étape 1, 2 et 3 jusqu'à ce que le tableau A soit entièrement composé de -1.

N.B : *Prévoir l'affichage des éléments du tableau B*

17

Deux entiers naturels strictement positifs m et n sont dits nombre amis si et seulement si :

- la somme des diviseurs de m sauf lui-même est égale à n
- et la somme des diviseurs de n sauf lui-même est égale à m.

Exemple :

220 et 284 sont deux nombres amis.

En effet :

D284 = *{1, 2, 4, 71, 142, 284}*

D220 = *{1, 2, 4, 5, 10, 11, 20, 22, 44, 55, 110, 220}*

D284 et **D220** sont respectivement les ensembles de tous les diviseurs de **284** et de **220**.

284 = 1 + 2 + 4 + 5 + 10 + 11 + 20 + 22 + 44 + 55 + 110

220 = 1 + 2 + 4 + 71 + 142

Écrire un programme qui permet de déterminer puis d'afficher si deux entiers naturels donnés m et n sont amis ou non.

18

Ecrire un programme qui réalise le traitement suivant :

- choisir un entier n de l'intervalle [100, 500] et un entier m de l'intervalle [10,99]
- afficher tous les entiers de l'intervalle [1, m] en remplaçant par le caractère "*" tous les diviseurs de n ainsi que tous les entiers comportant dans leurs écritures le chiffre des unités de n.

Exemple d'exécution :

Si n = 100 et m = 20 alors la liste suivante sera affichée :

** * 3 * * 6 7 8 9 * 11 12 13 14 15 16 17 18 19 **

19

On se propose d'écrire un programme permettant de déterminer et d'afficher la lettre alphabétique la plus utilisée dans un texte donné. Le texte étant saisi comme une chaîne de caractères contenant n de caractères (5 ≤ n ≤20).

Dans le cas d'ex-aequo, afficher toutes les lettres ayant la plus grande fréquence.

20

On veut écrire un programme permettant de coder un message selon le procédé suivant : permuter chaque caractère d'indice pair avec le caractère qui le précède.

Exemple :

Le codage de la chaîne de caractère: "Baccalauréat" donne "aBcclauaérta"

21

Deux joueurs lancent en même temps un dé dont les faces sont numérotées de 1 à 6. Le joueur qui obtiendra la plus grande valeur aura un point. Le jeu s'arrête quand l'un des joueurs arrive le premier à un score de 10 points.

Ecrire un programme Pascal simulant ce jeu et afficher le numéro du joueur gagnant.

On pourra utiliser la fonction prédéfinie **RANDOM (n)** qui retourne un entier de l'intervalle **[0, n – 1]**.

22

On veut écrire un programme permettant de lire un mot intitulé **CHM** composé au moins de 5 caractères et d'afficher les chaînes de caractères suivantes :

- La chaîne formée par le premier et le dernier caractère de **CHM**
- La chaîne formée par les deux premiers et les deux derniers caractères de **CHM**
- Etc.

Exemple *: Si la chaîne* ***CHM*** *contient* ***"TURBO"*** *alors le programme affichera :*

TO
TUBO
TURRBO
TURBURBO
TURBOTURBO

23

Ecrire un programme qui saisit une chaîne de caractères de longueur minimal 3 et l'affiche sous la forme d'un triangle comme indiqué ci-dessous.

Exemple :

Si la chaîne saisie est "INTERNET", on aura :

I
IN
INT
INTE
INTER
INTERN
INTERNE
INTERNET

24

On se propose de réaliser le traitement suivant sur une chaîne **CH :**

Construire une chaîne ***RES*** *à partir de la chaîne* ***CH*** *dans laquelle on rangera toutes les consonnes de CH qui sont en majuscule, suivies de toutes les voyelles de CH qui sont en majuscule, suivies de toutes les consonnes de CH qui sont en minuscule et finalement toutes les voyelles de CH qui sont en minuscule en conservant à chaque fois le même ordre d'apparition des lettres de la chaîne CH.*

Ecrire un programme qui permet de saisir une chaîne CH non vide composée de lettres alphabétiques et dont la taille ne dépasse pas 50 et de construire puis d'afficher la chaîne RES.

Exemple:

Si CH = ***"aFAbzKOikvMx"*** *Alors le programme affichera la chaîne* ***"FKMAObzkvxai"***

25

Ecrire un programme qui permet de Remplir un tableau T de n (3≤n≤20) éléments de types caractères, inverser le contenu du tableau puis afficher le résultat.

26

Ecrire un programme qui permet de saisir un entier positif N, composé de trois chiffres, de déterminer et d'afficher tous les nombres qui peuvent être formés par les chiffres de N, ainsi que le plus petit et le plus grand de ces nombres.

Exemple :

Pour N = 427, le programme affichera :

Les nombres formés par les chiffres de ***427*** *sont : 427, 472, 724, 742, 247, 274*
Le plus petit nombre est 247
Le plus grand nombre est 742

27

On se propose d'écrire un programme permettant de vérifier si un entier donné est **subline** ou non. Un entier est dit sublime si la somme de ses diviseurs y compris lui-même et le nombre de ses diviseurs sont deux nombres parfaits.

Un entier est dit parfait s'il est égal à la somme de ses diviseurs sauf lui-même

Exemple: N=12

- ❖ la somme des diviseurs SD de 12 est 1+2+3+4+6+12=28
- ❖ le nombre de diviseurs NBD de 12 est 6
- ❖ on a
  - ☑ 28 est parfait car 28=1+2+4+7+14
  - ☑ 6 est parfait car 6=1+2+3

=>donc 12 est sublime

On propose les algorithmes suivants:

```
0) Début sublime
1) Ecrire ("N=")
     Lire (N)
2) Proc diviseurs(N, SD, NBD)
3) Proc affiche (N, SD, NBD)
4) Fin sublime
```

```
0) Def fn parfait (m:entier):booléen
1) Somd<-1
     Pour i de 2 à (m div 2) faire
         Si (m mod i)=0 alors
             Somd←somd+i
          Fin si
     Fin pour
2) parfait←(somd=m)
3) Fin parfait
```

Questions:

1. écrire la procédure "**diviseurs**" permettant de calculer la somme S des diviseurs d'un entier p y compris lui-même et le nombre NB de ces diviseurs.

2. écrire la procédure "**affiche**" permettant de vérifier si un nombre p est sublime ou non. Cette procédure utilisera la fonction "**parfait**" dont l'algorithme est mentionné ci-dessus.

28

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

$$\begin{cases} P_1 = 2 \\ P_i = P_{i-2} \times \frac{i-1}{i} \times \frac{i+1}{i} & (i > 1 \text{ et } i \text{ impair}) \end{cases}$$

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

29

Ecrire une analyse et un algorithme intitulé "**Annuaire**" qui permet de remplir 2 tableaux Tnom et Ttel en parallèle en respectant que à chaque nom de Tnom correspond le numéro du téléphone de même indice

Exemple:

Tnom	WALID	KAIS	MOHAMMED	RAMZI	ABDELAZIZ
Ttel	92765324	27895312	75650222	90456032	29034598
i	1	2	3	4	5

Questions:

1. écrire un module "**Remplir_Contact**" qui permet de remplir le tableau **Tnom** en respectant qu'un nom est composé seulement de lettres majuscules et le tableau **Ttel** en respectant qu'un numéro de téléphone est composé exactement de 8 chiffres et commence obligatoirement par 7(tel fixe) ou 9(GSM Tuntel) ou 2(GSM Tunisiana) ou 5(GSM Orange)

2. écrire un module "**Tri_nom**" qui permet de trier les deux tableaux en parallèle suivant l'ordre croissant des noms

3. écrire un module "**Tri_tel**" qui permet de trier les deux tableaux en parallèle suivant l'ordre croissant des numéros

4. écrire le programme principal qui permet d'appeler les modules ci-dessus et saisir un nom pour déterminer son numéro de téléphone s'il existe ou saisir un numéro de téléphone pour connaître le nom de son propriétaire en utilisant la recherche dichotomique.

NB : 5≤Nombre de contact≤50

30

Ecrire un programme qui permet de déterminer et d'afficher tous les diviseurs suivis de tous les multiples d'un entier p donné, dans une partie d'un tableau T de n entiers donnés. Cette partie est délimitée par deux indices **ind_inf** et **ind_sup**. Avec (0<**ind_inf**<**ind_sup** ≤ n ≤ 15)

Exemple :

25	32	**43**	**4**	**32**	**72**	**80**	**15**	**24**	**2**	**48**	56	10	14
1	2	3	4	5	6	7	8	9	10	11	12	13	14

ind_inf **ind_sup**

Pour **n** = 14, **p** = 8, **ind_inf** = 3 et **ind_sup** = 11, le programme affichera :

Les diviseurs de 8 sont : 4 2

Les multiples de 8 sont : 32 72 80 24 48

31

Soit le tableau T suivant :

10	**7**	**9**	**7**	**10**	**6**	**7**	**4**	**8**	**8**

Pour chaque élément de T on ne garde que sa première occurrence et on remplace les autres par 0.

10	**7**	**9**	**0**	**0**	**6**	**0**	**4**	**8**	**0**

Pour regrouper les éléments restant au début du tableau T.

10	**7**	**9**	**6**	**4**	**8**	**0**	**0**	**0**	**0**

Ecrire un programme qui fait le traitement ci-dessus pour un tableau T de n (2≤n≤20) entiers positifs non nuls et détermine et affiche le nombre d'éléments différents de T.

32

Soit T, un tableau de n chaines de caractères représentants des prénoms de n personnes, avec N ∈ [10..100[. Ecrire un programme permettant de réaliser le traitement suivant:

1. Remplir le tableau T par N chaînes de caractères non vides,
2. Convertir tous les chaines en majuscules.
3. Trier dans l'ordre croissant le tableau T,
4. Supprimer tous les prénoms qui commencent par une lettre se trouvant dans un intervalle de lettres [L1..L2], choisi par l'utilisateur,
5. Vérifier, si un prénom saisit par l'utilisateur existe dans le tableau ou non.

Exemple :

- Soit T avec le contenu initial suivant :

T=	Hanen	Rami	Wafa	Amir	Warda	Mohamed	Ali	Mokhtar	Siwar

- Suite à l'opération de conversion le tableau sera comme suit :

T=	HANEN	RAMI	WAFA	AMIR	WARDA	MOHAMED	ALI	MOKTHAR	SIWAR

- Suite à l'opération de tri le tableau T sera comme suit :

T=	ALI	AMIR	HANEN	MOKTHAR	MOHAMED	RAMI	SIWAR	WAFA	WARDA

- Si l'intervalle des lettres choisi par l'utilisateur est ["M.."T"], tous les prénoms dont la 1ère lettre se trouve dans cet intervalle seront supprimés et le tableau T sera comme suit :

T=	ALI	AMIR	HANEN	WAFA	WARDA				

- Si l'utilisateur saisit le prénom « AMIR », le programme affiche le message « existe ».

33

Soit T un tableau de N éléments (2<N<200) de type caractère. On désire écrire un programme permettant de vérifier l'existence dans le tableau T d'un certain nombre de mots saisis dans un tableau Tm de P éléments (2<P<20).

Exemple :

Tm

BAC	Canne	Sujet

T

L	B	S	u	j	e	t	a	B	A	C	a	n	n	e	d

Remarques :

1. Les caractères de la chaîne recherchée doivent être adjacents dans le tableau T et non dispersés.
2. on remarque que les mots BAC, Canne et Sujet figurent dans le tableau T.

34

L'algorithme suivant est celui d'une fonction permettant de calculer la somme d'une partie d'éléments d'un tableau T de n entiers, délimité par les indices p1 et p2.

```
0) Def FN somme (T : TAB ; p1,p2 : entier) : entier ;
1) [s ← 0] Pour i de p1 à p2 faire
     S←s + T[i]
Fin Pour
2) somme← s
3) Fin somme
```

En exploitant la fonction dont l'algorithme est ci-dessus, Ecrire un programme qui permet de:

- Remplir automatiquement un tableau V de N entiers strictement positifs et inférieurs à 100 (5 ≤ n ≤ 20).
- Afficher l'indice (ind) de l'élément du tableau dont l'écart entre la somme (s1) des éléments qui le précédent et celle des éléments qui le succèdent (s2) soit minimal
- Afficher les sommes s1 et s2 correspondantes

Exemple :

Pour le tableau V suivant :

11	3	9	24	30	7	4	14	16	21	13	16
1	2	3	4	5	6	**7**	8	9	10	11	12

35

Soit T un tableau de N caractères alphabétique (2<N≤20)

Ecrire un programme Pascal permettant de crypter les données figurant dans le tableau T comme suit :

1. Convertir chaque caractère en sa représentation en code ASCII.
2. Permuter les chiffres des unités avec ceux des dizaines.
3. Insérer le caractère correspondant à ce nouveau code ASCII dans un tableau R. Puis afficher le tableau R obtenu.

Exemple :

Si N=3 et T= | "B" | "A" | "C" |

Alors R= | "B" | "8" | "L" |

Ord ("B") = 66, si on permute le chiffre de unité avec celui de dizaine on obtient le code ASCII du caractère "B"

Ord ("A") = 65, si on permute, on obtient (56) le code ASCII du caractère "8"

Ord ("C") = 67, si on permute, on obtient (76) le code ASCII du caractère "L"

36

On se propose de dessiner un rectangle à l'aide d'un rectangle donné. Le rectangle est formé de **L** lignes et **C** colonnes et il peut être plein ou vide, selon le choix de l'utilisateur.

Ecrire un programme permettant de :

- ☑ Saisir les dimensions **L** et **C** du rectangle, sachant que **L** et **C** sont deux entiers différents appartenant à l'intervalle [2..10]. **L** étant le nombre de lignes et **C** le nombre de colonnes.
- ☑ Saisir le caractère de dessin parmi la liste (x,+,$,*)
- ☑ Saisir le choix du dessin qui peut être soit la lettre **P** (pour plein) ou la lettre **V** (pour vide).
- ☑ Dessiner le rectangle selon les données fournies précédemment.

Exemples :

Si L=4, C=7, le caractère de dessin choisi est « x » et le choix du dessin et P alors le résultat affiché sera le suivant :

```
xxxxxxx
xxxxxxx
xxxxxxx
xxxxxxx
```

Si L=6, C=5, le caractère de dessin choisi est « $ » et le choix du dessin et V alors le résultat affiché sera le suivant :

```
$$$$$
$   $
$   $
$   $
$   $
$$$$$
```

37

Soit un tableau T de N lettres minuscules (6<=N<=100), et soient D et N deux entiers qui répondent aux conditions suivantes :

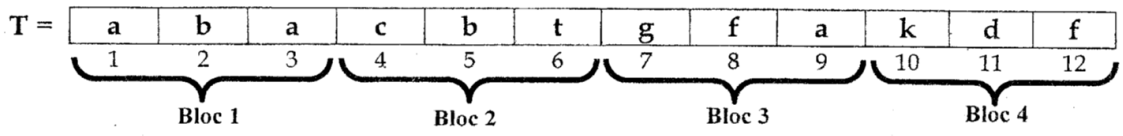
1. D est un entier diviseur de N strictement supérieur à 1.
2. M est un entier tel que N = M*D.

On se propose de trier les D éléments des M blocs disjoints qui constituent le tableau T. Ecrire un programme permettant de :

- ☑ lire les deux entiers N et D qui répondent aux conditions 1. et 2.
- ☑ remplir le tableau T par N lettres minuscules,
- ☑ trier dans l'ordre croissant, les éléments de chaque bloc du tableau T,
- ☑ afficher le tableau T après le tri.

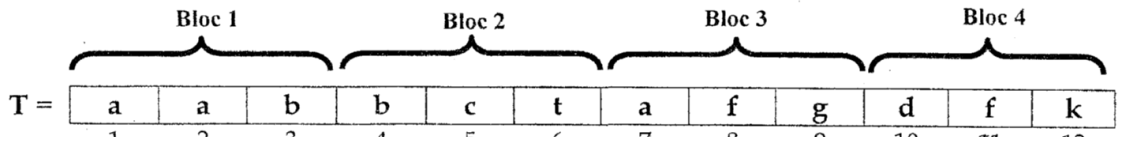
Exemple :

Si **N**=12 et **D**=3 (donc **M=4**) et si les éléments du tableau **T** sont les suivants :



Ce tableau contient 4 blocs de 3 lettres chacun.

Après le tri des éléments de chacun des blocs, le tableau T sera égal à :



38

Soit T un tableau de N chaînes de caractères non vides et dont la taille maximale est 5 caractères. On se propose d'écrire un programme Pascal permettant de réaliser le traitement suivant :

1. remplir le tableau T par N chaînes (2<=N<=30),
2. éliminer de chaque élément du tableau tous les caractères non alphabétiques,
3. convertir toutes les chaines non vides obtenues en majuscule,
4. afficher toutes les chaines non vides palindromes

N.B: *une chaîne est dite palindrome si elle se lit de la même façon de gauche à droite et de droite à gauche. Exemples : AlLA, RADAR, AA, Z*

Exemple :

Si N=5 et les éléments de T sont :

T=	A54a	15aZ	Ra8d9ar	2009	h?

☑ ***Le tableau après l'étape 2 contiendra les chaînes suivantes***

T=	Aa	aZ	Radar		h

☑ ***Le tableau après l'étape 3 contiendra les chaînes suivantes***

T=	AA	AZ	RADAR		H

☑ Le programme affichera : **AA RADAR H**

39

On veut dessiner à l'aide du caractère "*", un triangle de hauteur H (5<= H<=10), comme indiqué dans l'exemple ci-dessous.

Exemple

Si H=6, le programme affichera le triangle suivant :

```
* * * * * * * * * * *
  * * * * * * * * * *
    * * * * * * * *
      * * * * * *
        * * * *
          *
```

H= 6 (6 lignes)

Ecrire un programme qui permet de saisir H, puis d'afficher le triangle.

40

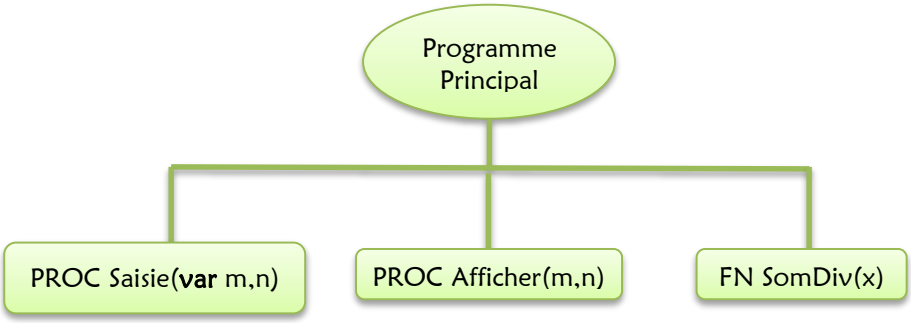
Un CODEC est un logiciel compresseur décompresseur de fichiers. En effet, les suites de bits composant un fichier comportent des similitudes comme 10000111. Plutôt que de stocker la totalité de cet octet, on gagne de la place en écrivant 14031 (qui se lit un quatre zéros trois un). Cet octet (huit bits) retrouva ensuite son format original à la décompression.

Il s'agit alors de saisir une chaîne de huit chiffres formée uniquement 0 et 1 pour désigner un octet puis la compresser suivant le principe de compression du CODEC et enfin l'afficher.

Exemple : Si octet = "10010111" Alors l'octet compressé est : "1201031"

Solution

✍ **Décomposition modulaire du problème**



✍ **Analyse de programme principal :**

Nom de programme : Parfait
Résultat : PROC Afficher(n,m)
PROC Saisie(n,m)
Fin Parfait

T.D.O Globaux

Objet	T/N	Rôle
N	Entier	Stocker la valeur de N
M	Entier	Stocker la valeur de M
Saisie	Procédure	Saisir les valeurs de N et M
Afficher	Procédure	Afficher tous les nombres parfaits compris entre n et m

✍ **Algorithme de programme principal:**

```
0) Début Parfait
1) PROC Saisie(n,m)
2) PROC Afficher(n,m)
3) Fin Parfait
```

✍ **Analyse de la procédure Saisie**

```
DEF PROC SAISIE(VAR X,Y :ENTIER)
Résultat : x,y
(x,y)=[]
Répéter
 x=donnée("Donner la valeur de N :  ")
 y=donnée("Donner la valeur de M :  ")
jusqu'à (X>Y)ET(Y>2)
Fin Saisie
```

✍ **Algorithme de la procédure Saisie**

```
0) DEF PROC SAISIE(VAR X,Y :ENTIER)
1) Répéter
   écrire("Donner la valeur de N :  ")
   lire(x)
   écrire("Donner la valeur de M :  ")
   lire(y)
   jusqu'à (X>Y)ET(Y>2)
2) Fin Saisie
```

✍ **Analyse de la procédure Afficher**

```
DEF PROC AFFICHER(X,Y :ENTIER)
Résultat : Trait
Trait=[]
Pour i de x à y faire
  Si(FN SomDiv(i)=i) alors
    Ecrire(i)
  Fin Si
Fin Pour
i : compteur
Fin Afficher
```

✍ **Algorithme de la procédure Afficher**

```
0) DEF PROC AFFICHER(X,Y :ENTIER)
1) Pour i de x à y faire
   Si(FN SomDiv(i)=i) alors
     Ecrire(i)
   Fin Si
   Fin Pour
2) Fin Afficher
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur
SomDiv	Fonction	Calculer la somme des diviseurs d'un nombre

✍ **Analyse de la Fonction SomDiv**

```
DEF FN SomDiv(X:ENTIER) : ENTIER
Résultat : SomDiv←S
Trait=[]
S=[s←0]Pour i de 1 à x div 2 faire
  Si(x mod i = 0) alors
    S←S+i
  Fin Si
Fin Pour
i : compteur
Fin SomDiv
```

✍ **Algorithme de la Fonction SomDiv**

```
0) DEF FN SomDiv(X:ENTIER) : ENTIER
1) [s←0]
2) Pour i de 1 à x div 2 faire
     Si(x mod i = 0) alors
       S←S+i
     Fin Si
   Fin Pour
3) SomDiv←S
4) Fin SomDiv
```

T.D.O Locaux

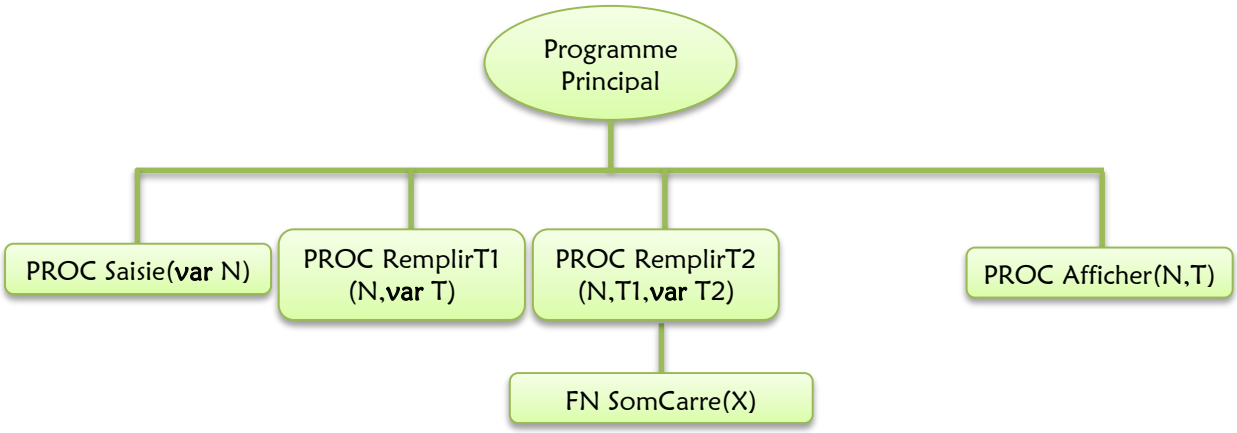
Objet	T/N	Rôle
i	Entier	Compteur
S	Entier	Calculer la somme des diviseurs de x

✍ **Traduction Pascal**

```
PROGRAM Parfait ;
USES wincrt ;
VAR
   n,m :integer ;
PROCEDURE Saisie(VAR x,y :integer) ;
BEGIN
 REPEAT
      Write('Donner la valeur de N: ');
      Readln(x) ;
      Write('Donner la valeur de M: ');
      Readln(y) ;
 UNTIL(x>y) and (y>2) ;
END;
FUNCTION SomDiv(x :integer) :integer ;
VAR
      i,S :integer ;
BEGIN
      S :=0 ;
      FOR i :=1 to x div 2 do
      If(x mod i = 0) then
            S :=S+i ;
      SomDiv := S ;
END ;
```

```
PROCEDURE Afficher(x,y :integer) ;
VAR
      i :integer ;
BEGIN
      FOR i :=x to y do
            If(SomDiv(i) = i) then
                  Writeln(i) ;
END;
BEGIN
      Saisie(n,m) ;
      Afficher(n,m) ;
END.
```

Décomposition modulaire du problème



Analyse de programme principal :

Nom de programme : Somme_Carre
Résultat : PROC Afficher(n,T2)
PROC RemplirT2(n,T1,T2)
PROC RemplirT1(n,T1)
PROC Saisie(n)
Fin Somme_Carre

Algorithme de programme principal:

0) *Début* Somme_Carre
1) PROC Saisie(n)
2) PROC RemplirT1(n,T1)
3) PROC RemplirT2(n,T1,T2)
4) PROC Afficher(n,T2)
5) *Fin* Somme_Carre

T.D.N.T

Type
TAB= Tableau de taille 100 et de type entier

T.D.O Globaux

Objet	T/N	Rôle
N	Entier	saisir la valeur de N
T1	TAB	Remplir le tableau par N entier
T2	TAB	Remplir le tableau par la somme des chiffres des éléments du tableau T1
Afficher	Procédure	Afficher un tableau de taille N
RemplirT1	Procédure	Remplir un tableau de taille de taille N
RemplirT2	Procédure	Remplir le tableau T2 par la somme des chiffres des éléments du tableau T1

Analyse de la procédure Saisie

DEF PROC SAISIE(VAR X :ENTIER)
Résultat : x
x=[]
Répéter
x=**donnée**("Donner la valeur de N : ")
jusqu'à (X≥1)ET(X≤100)
Fin Saisie

Algorithme de la procédure Saisie

0) DEF PROC SAISIE(VAR X :ENTIER)
1) Répéter
écrire("Donner la valeur de N : ")
lire(x)
jusqu'à (X≥1)ET(X≤100)
2) Fin Saisie

✍ **Analyse de la procédure Afficher**

```
DEF PROC AFFICHER(N :ENTIER,T :TAB)
Résultat : Trait
Trait=[]
Pour i de 1 à N faire
    Ecrire("T2[",i, "]=",T[i])
Fin Pour
i : compteur
Fin Afficher
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur

✍ **Algorithme de la procédure Afficher**

```
0) DEF PROC AFFICHER(N :ENTIER,T :TAB)
1) Pour i de 1 à N faire
       Ecrire("T2[",i, "]=",T[i])
2) Fin Pour
3) Fin Afficher
```

✍ **Analyse de la procédure RemplirT1**

```
DEF PROC REMPLIRT1(N :ENTIER,VAR T :TAB)
Résultat : Trait
Trait=[]
Pour i de 1 à N faire
  Répéter
  T[i]=donnée("T1[",i, "]=")
  jusqu'à (T[i]≤999)ET(T[i]≥100)
Fin Pour
i : compteur
Fin RemplirT1
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur

✍ **Algorithme de la procédure RemplirT1**

```
0) DEF PROC REMPLIRT1(N :ENTIER,VAR T:TAB)
1) Pour i de 1 à N faire
       Répéter
            écrire("T1[",i, "]=")
            lire(T[i])
       jusqu'à (T[i]≤999)ET(T[i]≥100)
   Fin Pour
2) Fin RemplirT1
```

✍ **Analyse de la procédure RemplirT2**

```
DEF PROC REMPLIRT2(N :ENTIER, T1 :TAB ,VAR T2 :TAB)
Résultat : T2
T2=[]
Pour i de 1 à N faire
  T2[i]← FN SomCarre(T1[i])
Fin Pour
i : compteur
Fin RemplirT2
```

✍ **Algorithme de la procédure RemplirT2**

```
0) DEF PROC REMPLIRT2(N:ENTIER, T1 :TAB,
   VAR T2 :TAB)
1) Pour i de 1 à N faire
       T2[i]← FN SomCarre(T1[i])
   Fin Pour
2) Fin RemplirT2
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur
SomCarre	Fonction	Calculer la somme des chiffres d'un entier

✍ Analyse de la Fonction SomCarre

```
DEF FN SomCarre(X:ENTIER) : ENTIER
    Résultat : SomCarre ←S
    S←carré(c)+ carré(d)+ carré(u)
    C←x div 100
    D←(x mod 100 div) 10
    U←x mod 10
    Fin SomCarre
```

T.D.O Locaux

Objet	T/N	Rôle
C,D,U	Entier	Déterminer les chiffres de X
S	Entier	Calculer la somme des chiffres de x

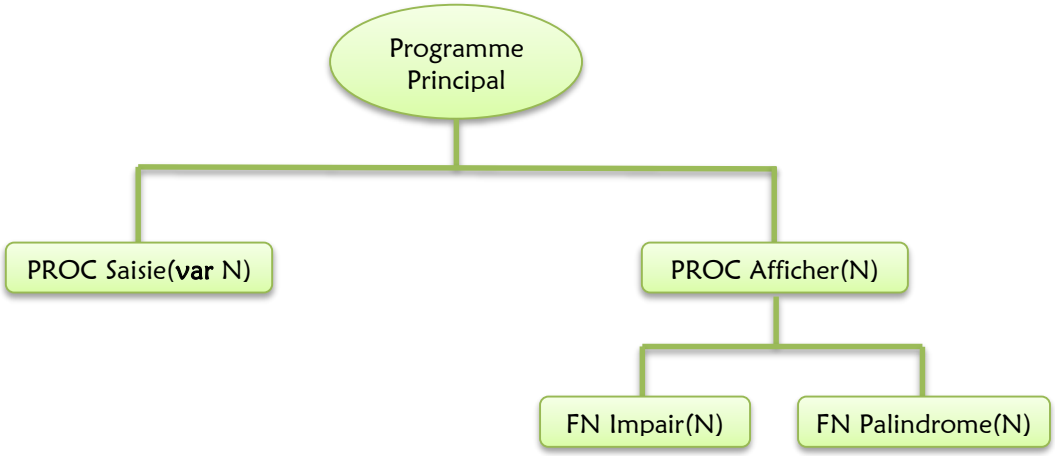
✍ Algorithme de la Fonction SomCarre

```
0) DEF FN SomCarre(X:ENTIER) : ENTIER
1) C←x div 100
2) D←(x mod 100 div) 10
3) U←x mod 10
4) S←carré(c)+ carré(d)+ carré(u)
5) SomCarre ←S
6) Fin SomCarre
```

✍ Traduction Pascal

```
PROGRAM Somme_Carre;
USES wincrt ;
TYPE
    TAB=Array[1..100] of integer ;
VAR
   n:integer ;
   T1,T2 :TAB ;
PROCEDURE Saisie(VAR x :integer) ;
BEGIN
 repeat
    Write('Donner la valeur de N: ');
    Readln(x) ;
 until(X>=1)and(X<=100);
END;
PROCEDURE RemplirT1(n:integer ;VAR
T :TAB) ;
VAR
     i :integer ;
BEGIN
     FOR i :=1 to n do
     REPEAT
       Write('T2[',i, ']= ') ;
       Readln(T[i]) ;
     UNTIL (T[i]≥100) and (T[i]≤999);
END;
FUNCTION SomCarre(x :integer):integer;
VAR
     S,c,d,u :integer ;
BEGIN
   C :=x div 100 ;
   D :=(x mod 100) div 10 ;
   U :=x mod 10 ;
   S := sqr(c)+ sqr(d)+ sqr(u) ;
   SomCarre:= S ;
END ;
PROCEDURE RemplirT2(n:integer ;T1 :TAB ;
VAR T2 :TAB) ;
VAR
     i :integer ;
BEGIN
     FOR i :=1 to n do
      T2[i]:= SomCarre(T1[i]) ;
END;

PROCEDURE Afficher(n:integer ;T :TAB) ;
VAR
     i :integer ;
BEGIN
     FOR i :=1 to n do
       Writeln('T2[',i, ']= ',T[i]) ;
END;
BEGIN
     Saisie(n) ;
     RemplirT1(n,T1) ;
     RemplirT2(n,T1,T2) ;
     Afficher(n,T2) ;
END.
```

✍ **Décomposition modulaire du problème**



✍ **Analyse de programme principal :**

Nom de programme : Entier_palindrome
Résultat : PROC Afficher(n)
PROC Saisie(n)
Fin Entier_palindrome

T.D.O Globaux

Objet	T/N	Rôle
N	Entier	Stocker la valeur de N
Saisie	Procédure	Saisir les valeurs de N
Afficher	Procédure	Afficher les n premiers nombres impairs et palindromes.

✍ **Algorithme de programme principal:**

```
0) Début Entier_palindrome
1) PROC Saisie(n)
2) PROC Afficher(n)
3) Fin Entier_palindrome
```

✍ **Analyse de la procédure Saisie**

```
DEF PROC SAISIE(VAR X :ENTIER)
Résultat : x
x=[]
Répéter
 x=donnée("Donner la valeur de N :  ")
jusqu'à (X≥5)ET(X≤20)
Fin Saisie
```

✍ **Algorithme de la procédure Saisie**

```
0) DEF PROC SAISIE(VAR X :ENTIER)
1) Répéter
   écrire("Donner la valeur de N :
   ")
   lire(x)
   jusqu'à (X≥5)ET(X≤20)
2) Fin Saisie
```

✍ **Analyse de la procédure Afficher**

```
DEF PROC AFFICHER(N:ENTIER)
Résultat : Trait
Trait=[][i←1,x←0]Répéter
  Si(FN Impair(i))et(FN Palindrome(i))
    alors
         Ecrire(i)
         x←x+1
  Fin Si
  i←i+1
jusqu'à (x=N)
Fin Afficher
```

✍ **Algorithme de la procédure Afficher**

```
0) DEF PROC AFFICHER(N:ENTIER)
1) [i←1,x←0]Répéter
  Si(FN Impair(i))et(FN Palindrome(i))
  alors
     Ecrire(i)
     x←x+1
  Fin Si
  i←i+1
   jusqu'à (x=N)
2) Fin Afficher
```

T.D.O Locaux

Objet	T/N	Rôle
i,x	Entier	Compteur
Impair	Fonction	Vérifier la parité d'un entier
palindrome	Fonction	Vérifier si un entier est palindrome ou non ?

✍ **Analyse de la Fonction Impair**

```
DEF FN Impair(X:ENTIER) : BOOLEEN
Résultat : Impair ← x mod 2 = 0
Fin Impair
```

Comparer le reste de la division de x par 2 avec 0 : le résultat obtenu sera vrai ou faux

✍ **Algorithme de la Fonction Impair**

```
0) DEF FN Impair(X:ENTIER) : BOOLEEN
1) Impair ← x mod 2 ≠ 0
2) Fin Impair
```

✍ **Analyse de la Fonction Palindrome**

```
DEF FN Palindrome(X:ENTIER) : BOOLEEN
Résultat : Palindrome ← ch = ch1
Ch1=[ch1←""]
Pour i de long(ch) à 1 (pas=-1) faire
    ch1←ch1+ch[i]
Fin pour
Convch(x,ch)
i:compteur
Fin Palindrome
```

✍ **Algorithme de la Fonction Palindrome**

```
0) DEF FN Palindrome(X:ENTIER):BOOLEEN
1) Convch(x,ch)
2) [ch1←""]
3) Pour i de long(ch) à 1 (pas=-1)faire
       ch1←ch1+ch[i]
   Fin pour
4) Palindrome ← ch = ch1
5) Fin Palindrome
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur
Ch, Ch1	Chaine de caractère	Chaines auxiliaires

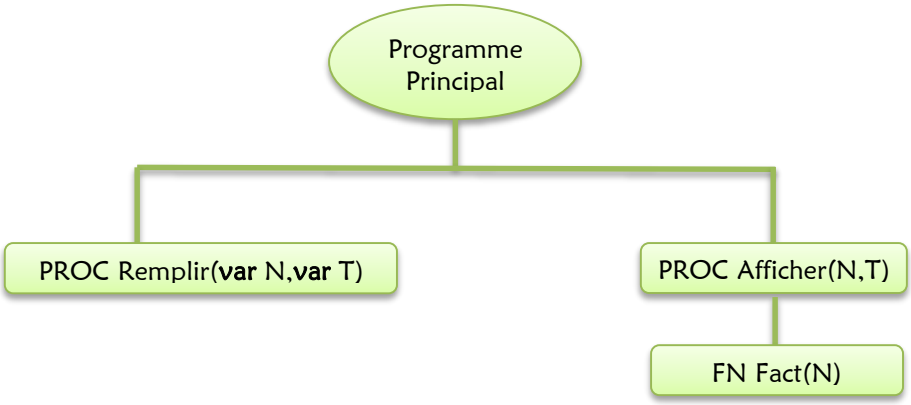
✍ **Traduction Pascal**

```
PROGRAM Entier_palindrome;
USES wincrt ;
VAR
   n :integer ;
PROCEDURE Saisie(VAR x :integer) ;
BEGIN
 repeat
    Write('Donner la valeur de N: ');
    Readln(x) ;
 until(x>=5)and(x<=20);
END;
PROCEDURE Afficher(n :integer) ;
VAR
     x,i :integer ;
BEGIN
     Str(x,ch) ;ch1 :='' ;
     For i :=length(ch) downto 1 do
        Ch1 := Ch1+ch[i] ;
     Palindrome:= ch=ch1 ;
END ;
BEGIN
X:=0 ; i:=1 ;
Repeat
If(Impair(i)) and (Palindrome(i)) then
begin
     Writeln(i) ; x:=x+1 ;
end ;
i :=i+1 ;
Until(x=n) ;
```

```
FUNCTION Impair(x :integer) :boolean ;
BEGIN
      Impair := x mod 2 <> 0 ;
END ;
FUNCTION  Palindrome(x:integer):
boolean;
VAR
     Ch,ch1 :string ;
     i :integer ;
END ;

BEGIN
      Saisie(n) ;
      Afficher(n) ;
END.
```

Exercice 4

✍ **Décomposition modulaire du problème**



✍ **Analyse de programme principal :**

Nom de programme : Som_Fact
Résultat : PROC Afficher(n,T)
PROC Remplir(n,T)
Fin Som_Fact

✍ **Algorithme de programme principal:**

0) *Début* Som_Fact
1) PROC Remplir (n,T)
2) PROC Afficher(n,T)
3) *Fin* Som_Fact

T.D.N.T

Type
TAB= Tableau de taille 40 et de type entier

T.D.O Globaux

Objet	T/N	Rôle
N	Entier	Stocker la taille de tableau
T	TAB	Remplir le tableau par N entier
Afficher	Procédure	Afficher un tableau de taille N
Remplir	Procédure	Remplir le tableau par la somme des factorielles des chiffres d'un entier de 10 à 50.

✍ Analyse de la procédure Afficher

```
DEF PROC AFFICHER(N :ENTIER,T :TAB)
Résultat : Trait
Trait=[]
Pour i de 1 à N faire
   Pour j de 1 à i faire
     Ecrire(T[i])
   Fin Pour
Fin Pour
i,j : compteur
Fin Afficher
```

✍ Algorithme de la procédure Afficher

```
0) DEF PROC AFFICHER(N :ENTIER,T :TAB)
1) Pour i de 1 à N faire
     Pour j de 1 à i faire
       Ecrire(T[i])
     Fin Pour
   Fin Pour
2) Fin Afficher
```

T.D.O Locaux

Objet	T/N	Rôle
i,j	Entier	Compteur

✍ Analyse de la procédure Remplir

```
DEF PROC REMPLIR(VAR N :ENTIER, VAR T :TAB)
Résultat : T
T =[]
Pour i de 10 à 50 faire
  T[i-9]← FN Fact(i div 10)+
          FN Fact(i mod 10)
Fin Pour
N ←50 -10+1
i : compteur
Fin Remplir
```

✍ Algorithme de la procédure Remplir

```
0) DEF PROC REMPLIR(VAR N:ENTIER,VAR
   T:TAB)
1) N←50-10+1
2) Pour i de 10 à 50 faire
     T[i-9]← FN Fact(i div 10)+
             FN Fact(i mod 10)
   Fin Pour
3) Fin Remplir
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur
Fact	Fonction	Calculer la factorielle d'un entier

✍ Analyse de la Fonction Fact

```
DEF FN Fact(X:ENTIER) : ENTIER
Résultat : Fact ← f
F=[f←1]Pour i de 2 à x faire
          F←f*i
Fin pour
i:compteur
Fin Fact
```

✍ Algorithme de la Fonction Fact

```
0) DEF FN Fact(X:ENTIER) : ENTIER
1) [f←1]
2) Pour i de 2 à x faire
     f←f*i
   Fin pour
3) Fact ← f
4) Fin Fact
```

T.D.O Locaux

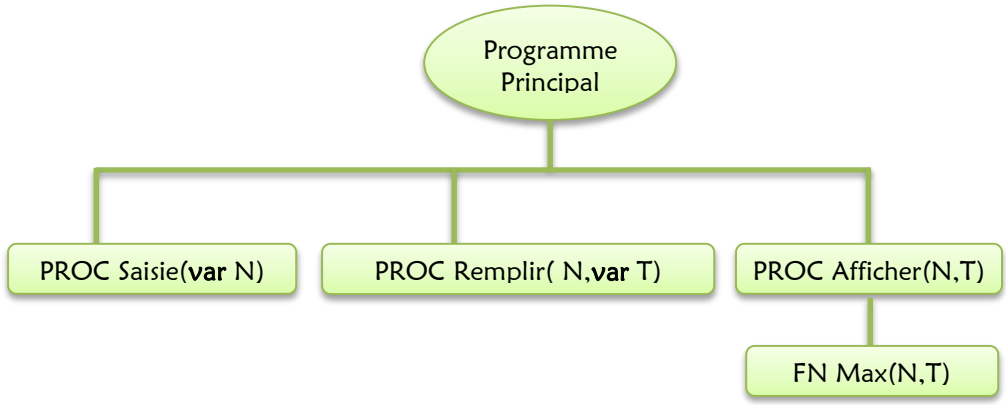
Objet	T/N	Rôle
i	Entier	Compteur
f	Entier long	Calculer la factorielle de x

Traduction Pascal

```
PROGRAM Som_Fact;
USES wincrt ;
TYPE
    TAB=Array[1..50] of integer ;
VAR
   n:integer ;
   T :TAB ;
PROCEDURE Remplir(VAR n:integer ;VAR
T :TAB) ;
VAR
     i :integer ;
FUNCTION Fact(x :integer):integer;
VAR
     f,i :integer ;
BEGIN
   f :=1 ;
   for i :=2 to x do
     f :=f*i ;
   Fact :=f ;
END;
BEGIN
  N :=50-10+1 ;
FOR i :=10 to 50 do
T[i-9]:=Fact(i div 10)+Fact(i mod 10);
END;
PROCEDURE Afficher(n:integer ;T :TAB) ;
VAR
     i :integer ;
BEGIN
     FOR i :=1 to n do
       Writeln('T[',i, ']= ',T[i]) ;
END;
BEGIN
     Remplir(n,T) ;
     Afficher(n,T) ;
END.
```

Exercice 5

Décomposition modulaire du problème



Analyse de programme principal :

Nom de programme : Chaine_longue
Résultat : PROC Afficher(n,T)
PROC Remplir(n,T)
PROC Saisie(n)
Fin Chaine_longue

Algorithme de programme principal:

0) *Début* Chaine_longue
1) PROC Saisie(n)
2) PROC Remplir(n,T)
3) PROC Afficher(n,T)
4) *Fin* Chaine_longue

T.D.N.T

Type
TAB= Tableau de taille 20 et de type chaine de caractère

T.D.O Globaux

Objet	T/N	Rôle
N	Entier	stocker la taille du tableau
T	TAB	Remplir le tableau par N chaines de caractère
Afficher	Procédure	Afficher les chaines ayant la longueur maximale
Remplir	Procédure	Remplir le tableau par n chaines de caractère.
Saisie	Procédure	Saisir la taille du tableau

✍ **Analyse de la procédure Saisie**

```
DEF PROC SAISIE(VAR X :ENTIER)
Résultat : x
x=[]Répéter
 x=donnée("Donner la valeur de N :  ")
jusqu'à (x>4)ET(x<20)
Fin Saisie
```

✍ **Algorithme de la procédure Saisie**

```
0) DEF PROC SAISIE(VAR X :ENTIER)
1) Répéter
     écrire("Donner la valeur de N: ")
     lire(x)
     jusqu'à (x>4)ET(x<20)
2) Fin Saisie
```

✍ **Analyse de la procédure Remplir**

```
DEF PROC REMPLIR(N :ENTIER,VAR T :TAB)
Résultat : T
T =[]Pour i de 1 à N faire
   T[i]=donnée("T[",i, "]=")
Fin Pour
i : compteur
Fin Remplir
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur

✍ **Algorithme de la procédure Remplir**

```
0) DEF PROC REMPLIR(N :ENTIER,VAR T:TAB)
1) Pour i de 1 à N faire
       écrire("T[",i, "]=")
       lire(T[i])
   Fin Pour
2) Fin Remplir
```

✍ **Analyse de la procédure Afficher**

```
DEF PROC AFFICHER(N :ENTIER,T :TAB)
Résultat : Trait
Trait=[]Pour i de 1 à N faire
          si long(T[i])=m alors
               Ecrire(T[i])
          Fin Si
     Fin Pour
Ecrire("la longueur de la chaîne la
     plus longue : ",m)
m←FN Max(n,T)
i : compteur
Fin Afficher
```

✍ **Algorithme de la procédure Afficher**

```
0) DEF PROC AFFICHER(N :ENTIER,T :TAB)
1) m←FN Max(n,T)
2) Ecrire("la longueur de la chaîne la
       plus longue : ",m)
3) Pour i de 1 à N faire
       si long(T[i])=m alors
            Ecrire(T[i])
       Fin Si
     Fin Pour
4) Fin Afficher
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur
m	Entier	Stocker la longueur maximale des chaines
Max	Fonction	Déterminer la longueur maximale des chaines

✍ **Analyse de la Fonction Max**

```
DEF FN Max(X:ENTIER, T :TAB) : ENTIER
Résultat : Max ← m
m=[m←long(T[1])]
Pour i de 1 à x faire
     si(long(T[i])>m) alors
          m←long(T[i])
     Fin Si
Fin pour
i:compteur
Fin Max
```

✍ **Algorithme de la Fonction Max**

```
0) DEF FN Max(X:ENTIER, T:TAB) : ENTIER
1) [m←long(T[1])]
2) Pour i de 1 à x faire
       si(long(T[i])>m) alors
            m←long(T[i])
       Fin Si
   Fin pour
3) Max ← m
4) Fin Max
```

T.D.O Locaux

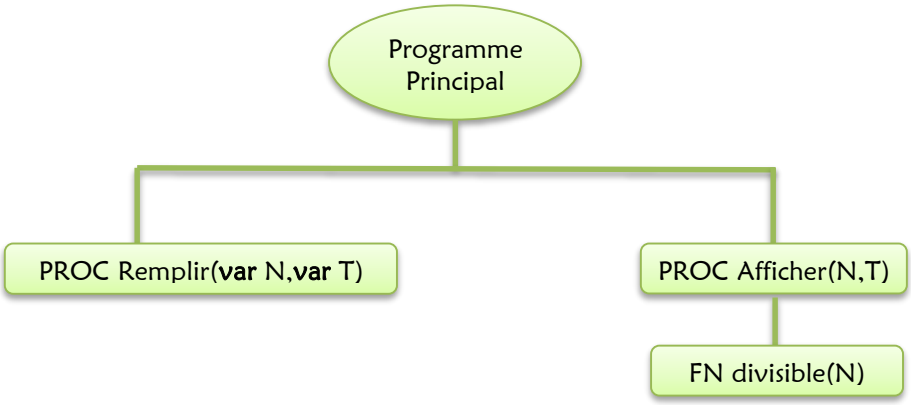
Objet	T/N	Rôle
i	Entier	Compteur
m	Entier	Déterminer la longueur maximale des chaines

✍ **Traduction Pascal**

```
PROGRAM Chaine_longue;
USES wincrt ;
TYPE
    TAB=Array[1..20] of string[30] ;
VAR
   n:integer ;
   T :TAB ;
PROCEDURE Saisie(VAR x :integer) ;
BEGIN
 repeat
    Write('Donner la valeur de N: ');
    Readln(x) ;
 until(x>4)and(x<20);
END;
PROCEDURE Remplir(n:integer;VAR T:TAB);
VAR   i :integer ;
BEGIN
     FOR i :=1 to n do
     begin
       Write('T[',i, ']= ') ;
       Readln(T[i]) ;
     end;
END;
```

```
PROCEDURE Afficher(n:integer ;T :TAB) ;
VAR
     m,i :integer ;
FUNCTION Max(x:integer;T:TAB):integer;
VAR
     m,i :integer ;
BEGIN
   m :=length(T[1]) ;
     FOR i :=1 to n do
     if(length (T[i])>m) then
          m :=length(T[i]) ;
   Max :=m ;
END;
BEGIN
  m :=Max(n,T) ;
  writeln(' la longueur de la chaîne la
     plus longue : ',m) ;
  FOR i :=1 to n do
   if length (T[i])=m then
    Writeln(T[i]) ;
END;
BEGIN
Saisie(n);Remplir(n,T);Afficher(n,T);
END.
```

✍ **Décomposition modulaire du problème**



✍ **Analyse de programme principal :**

Nom de programme : Div_Dizaine
Résultat : PROC Afficher(n,T)
PROC Remplir(n,T)
Fin Div_Dizaine

✍ **Algorithme de programme principal:**

0) *Début* Div_Dizaine
1) PROC Remplir (n,T)
2) PROC Afficher(n,T)
3) *Fin* Div_Dizaine

T.D.N.T

Type
TAB= Tableau de taille 90 et de type entier

T.D.O Globaux

Objet	T/N	Rôle
N	Entier	Stocker la taille de tableau
T	TAB	Remplir le tableau par N entier
Afficher	Procédure	Afficher un tableau de taille N
Remplir	Procédure	Remplir les entiers de deux chiffres divisibles par leurs chiffres de dizaine

✍ **Analyse de la procédure Afficher**

```
DEF PROC AFFICHER(N :ENTIER,T :TAB)
Résultat : Trait
Trait=[]Pour i de 1 à N faire
  Si i mod 2 =0 alors
    Ecrire(T[i])
  Fin Si
Fin Pour
Pour i de 1 à N faire
  Si i mod 2 ≠ 0 alors
    Ecrire(T[i])
  Fin Si
Fin Pour
i,j : compteur
Fin Afficher
```

✍ **Algorithme de la procédure Afficher**

```
0) DEF PROC AFFICHER(N :ENTIER,T :TAB)
1) Pour i de 1 à N faire
  Si i mod 2 =0 alors
     Ecrire(T[i])
  Fin Si
Fin Pour
2) Pour i de 1 à N faire
  Si i mod 2 ≠ 0 alors
     Ecrire(T[i])
  Fin Si
   Fin Pour
3) Fin Afficher
```

T.D.O Locaux

Objet	T/N	Rôle
i,j	Entier	Compteur

✍ **Analyse de la procédure Remplir**

```
DEF PROC REMPLIR(VAR N :ENTIER, VAR T :TAB)
Résultat : T
T=[n←0]
Pour i de 10 à 99 faire
  Si FN divisible(i) alors
    n←n+1
    T[n]← i
  Fin Si
Fin Pour
i : compteur
Fin Remplir
```

✍ **Algorithme de la procédure Remplir**

```
0) DEF PROC REMPLIR(VAR N:ENTIER,VAR T:TAB)
1) [N←0]Pour i de 10 à 99 faire
  Si FN divisible (i) alors
     n←n+1
     T[n]← i
  Fin Si
  Fin Pour
2) Fin Remplir
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur
divisible	Fonction	Tester si un entier est divisible par son chiffre de dizaine

✍ **Analyse de la Fonction divisible**

```
DEF FN Divisible(X:ENTIER) : BOOLEEN
Résultat : divisible ← (x mod (x div 10))= 0
Fin divisible
```

✍ **Algorithme de la Fonction divisible**

```
0) DEF FN Divisible(X:ENTIER) : BOOLEEN
1) divisible ← (x mod (x div 10))= 0
2) Fin divisible
```

✍ **Traduction Pascal**

```
PROGRAM Div_Dizaine;
USES wincrt ;
TYPE
    TAB=Array[1..90] of integer ;
VAR
   n:integer ;
   T :TAB ;
PROCEDURE Afficher(n:integer ;T :TAB);
VAR  i :integer ;
BEGIN
     FOR i :=1 to n do
       if(i mod 2 =0) then
          Write(T[i]) ;
     FOR i :=1 to n do
       if(i mod 2 ≠0) then
          Write(T[i]) ;
```

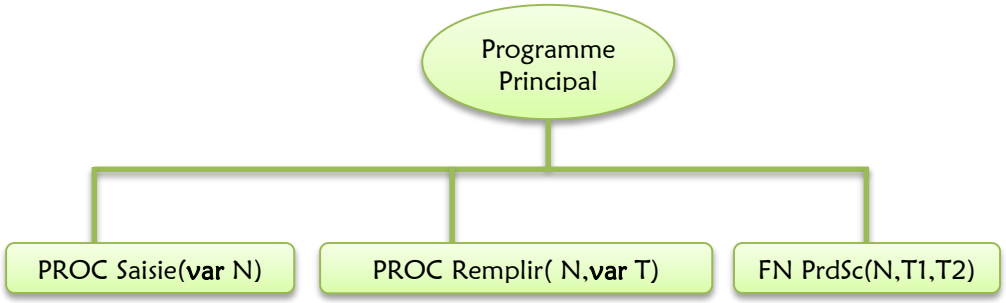
```
PROCEDURE Remplir(VAR n:integer;VAR
T:TAB);
VAR  i :integer ;
FUNCTION Divisible(x:integer):Boolean;
BEGIN
   Divisible:=(x mod (x div 10))= 0;
END;
BEGIN
N :=0 ;
FOR i :=10 to 99 do
  if(divisible(i)) then
  begin
    n :=n+1 ;
    T[n] :=i ;
   end ;
END;
```

```
END;
```

```
BEGIN
      Remplir(n,T) ;
      Afficher(n,T) ;
END.
```

Exercice 7

✍ **Décomposition modulaire du problème**



✍ **Analyse de programme principal :**

Nom de programme : Produit_scalire
Résultat: Ecrire("Le produit scalaire est : ", FN PrdSc(n,A,B))
PROC Remplir(n,A)
PROC Remplir(n,B)
PROC Saisie(n)
Fin Produit_scalire

✍ **Algorithme de programme principal:**

```
0) Début Produit_scalire
1) PROC Saisie(n)
2) PROC Remplir(n,A)
3) PROC Remplir(n,B)
4) Ecrire("Le produit
   scalaire est : ", FN
   PrdSc(n,A,B))
5) Fin Produit_scalire
```

T.D.N.T

Type
TAB= Tableau de taille 50 et de type entier

T.D.O Globaux

Objet	T/N	Rôle
N	Entier	stocker la taille du tableau
A,B	TAB	Remplir les tableaux par N entiers positifs
Remplir	Procédure	Remplir un tableau par N entiers positifs.
Saisie	Procédure	Saisir la taille du tableau
PrdSc	Fonction	Calculer le produit scalaire de deux tableaux

✍ **Analyse de la procédure Saisie**

```
DEF PROC SAISIE(VAR X :ENTIER)
Résultat : x
x=[]Répéter
 x=donnée("Donner la valeur de N :  ")
jusqu'à (x≥5)ET(x≤50)
Fin Saisie
```

✍ **Algorithme de la procédure Saisie**

```
0) DEF PROC SAISIE(VAR X :ENTIER)
1) Répéter
     écrire("Donner la valeur de N: ")
     lire(x)
     jusqu'à (x≥5)ET(x≤50)
2) Fin Saisie
```

✍ **Analyse de la procédure Remplir**

```
DEF PROC REMPLIR(N :ENTIER,VAR T :TAB)
Résultat : T
T =[]Pour i de 1 à N faire
Répéter
          T[i]=donnée("T[",i, "]=")
Jusqu'à (T[i]≥0)
Fin Pour
i : compteur
Fin Remplir
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur

✍ **Algorithme de la procédure Remplir**

```
0) DEF PROC REMPLIR(N :ENTIER,VAR T:TAB)
1) Pour i de 1 à N faire
     Répéter
       écrire("T[",i, "]=")
       lire(T[i])
     Jusqu'à (T[i]≥0)
   Fin Pour
2) Fin Remplir
```

✍ **Analyse de la Fonction PrdSc**

```
DEF FN PrdSc(X:ENTIER, T1,T2 :TAB): ENTIER LONG
Résultat : PrdSc ← s
S=[s←0]
Pour i de 1 à x faire
     S←s+T1[i]*T2[i]
Fin pour
i:compteur
Fin PrdSc
```

✍ **Algorithme de la Fonction PrdSc**

```
0) DEF FN PrdSc (X:ENTIER,T1,T2:TAB): ENTIER
       LONG
1) [s←0]
2) Pour i de 1 à x faire
       S←s+T1[i]*T2[i]
   Fin pour
3) PrdSc ← s
4) Fin PrdSc
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur
s	Entier long	Calculer le produit scalaire de deux vecteurs

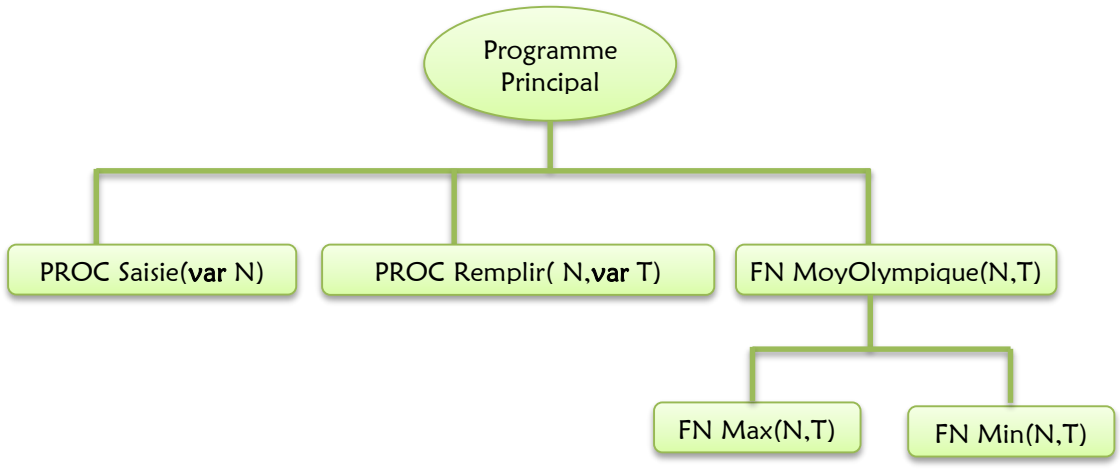
✍ **Traduction Pascal**

```
PROGRAM Produit_scalire;
USES wincrt ;
TYPE TAB=Array[1..20] of integer ;
VAR
   n:integer ;
   A,B :TAB ;
PROCEDURE Saisie(VAR x :integer) ;
BEGIN
 repeat
    Write('Donner la valeur de N: ');
    Readln(x) ;
 until(x>=5)and(x<=50);
END;
PROCEDURE Remplir(n:integer;VAR T:TAB);
VAR   i :integer ;
BEGIN
  FOR i :=1 to n do
  repeat
       Write('valeur ',i,'=') ;
       Readln(T[i]) ;
  Until(T[i]≥0) ;
END;
```

```
FUNCTION PrdSc(x:integer;T1,T2:TAB):
longint;
VAR  s : longint ;
     i :integer ;
BEGIN
   S :=0 ;
   FOR i :=1 to n do
     S :=s+ T1[i]*T2[i] ;
   PrdSc:=s ;
END;
BEGIN
Saisie(n) ;
Remplir(n,A);
Remplir(n,B);
Write(' Le produit scalaire est : ',
 PrdSc(n,A,B)) ;
END.
```

Exercice 8

✍ **Décomposition modulaire du problème**



✍ **Analyse de programme principal :**

Nom de programme : Moyenne_Olympique
Résultat: Ecrire("La moyenne olympique est : ",
FN MoyOlympique(n,T))
PROC Remplir(n,T)
PROC Saisie(n)
Fin Moyenne_Olympique

✍ **Algorithme de programme principal:**

0) *Début* Moyenne_Olympique
1) PROC Saisie(n)
2) PROC Remplir(n,T)
3) Ecrire("La moyenne olympique est : ", FN MoyOlympique(n,T))
4) *Fin* Moyenne_Olympique

T.D.N.T

Type
TAB= Tableau de taille 20 et de type réel

T.D.O Globaux

Objet	T/N	Rôle
N	Entier	stocker la taille du tableau
T	TAB	Remplir les tableaux par N réels
Remplir	Procédure	Remplir un tableau par N réels
Saisie	Procédure	Saisir la taille du tableau
MoyOlympique	Fonction	Calculer la moyenne olympique

✍ **Analyse de la procédure Saisie**

```
DEF PROC SAISIE(VAR X :ENTIER)
Résultat : x
x=[]Répéter
 x=donnée("Donner la valeur de N :  ")
jusqu'à (x≥5)ET(x≤20)
Fin Saisie
```

✍ **Algorithme de la procédure Saisie**

```
0) DEF PROC SAISIE(VAR X :ENTIER)
1) Répéter
     écrire("Donner la valeur de N: ")
     lire(x)
     jusqu'à (x≥5)ET(x≤20)
2) Fin Saisie
```

✍ **Analyse de la procédure Remplir**

```
DEF PROC REMPLIR(N :ENTIER,VAR T :TAB)
Résultat : T
T=[]Pour i de 1 à N faire
          T[i]=donnée("T[",i, "]=")
Fin Pour
i : compteur
Fin Remplir
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur

✍ **Algorithme de la procédure Remplir**

```
0) DEF PROC REMPLIR(N :ENTIER,VAR T:TAB)
1) Pour i de 1 à N faire
      écrire("T[",i, "]=")
      lire(T[i])
   Fin Pour
2) Fin Remplir
```

✍ **Analyse de la Fonction MoyOlympique**

```
DEF FN Moyolympique(N:ENTIER, T:TAB): REEL
Résultat : MoyOlympique ← m
m←(s-FN min(n,T)- FN max(n,T))/(n-2)
S=[s←0]
Pour i de 1 à n faire
     S←s+T[i]
Fin pour
i:compteur
Fin MoyOlympique
```

✍ **Algorithme de la Fonction MoyOlympique**

```
0) DEF FN Moyolympique(N:ENTIER, T:TAB):
REEL
1) [s←0]
2) Pour i de 1 à n faire
      S←s+T[i]
   Fin pour
3) m←(s-FN min(n,T)-FN max(n,T))/(n-2)
4) MoyOlympique ← m
5) Fin MoyOlympique
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur
s	Réel	Calculer la somme des éléments d'un vecteur
m	Réel	Calculer la moyenne olympique

✍ **Analyse de la Fonction Max**

```
DEF FN Max(N:ENTIER, T :TAB) : REEL
Résultat : Max ← m
m=[m←T[1]]
Pour i de 1 à n faire
    si(T[i]>m) alors
        m←T[i]
    Fin Si
Fin pour
i:compteur
Fin Max
```

✍ **Algorithme de la Fonction Max**

```
0) DEF FN Max(N:ENTIER, T:TAB) : REEL
1) [m←T[1]]
2) Pour i de 1 à n faire
       si(T[i]>m) alors
           m←T[i]
       Fin Si
   Fin pour
3) Max ← m
4) Fin Max
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur
m	Réel	Déterminer la valeur maximale d'une liste des valeurs

✍ **Analyse de la Fonction Min**

```
DEF FN Min(N:ENTIER, T :TAB) : REEL
Résultat : Min ← m
m=[m←T[1]]
Pour i de 1 à n faire
    si(T[i]<m) alors
        m←T[i]
    Fin Si
Fin pour
i:compteur
Fin Min
```

✍ **Algorithme de la Fonction Min**

```
0) DEF FN Min(N:ENTIER, T:TAB) : REEL
1) [m←T[1]]
2) Pour i de 1 à n faire
       si(T[i]<m) alors
           m←T[i]
       Fin Si
   Fin pour
3) Min ← m
4) Fin Min
```

T.D.O Locaux

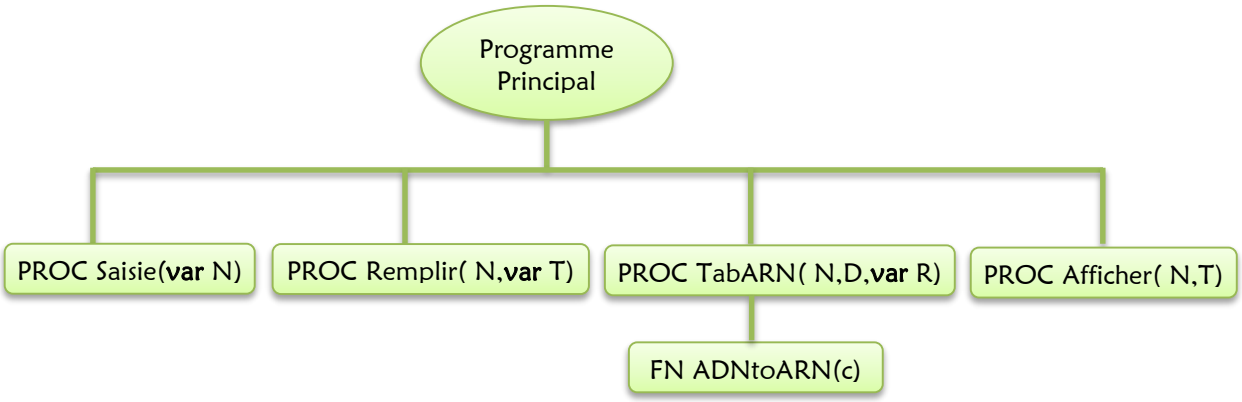
Objet	T/N	Rôle
i	Entier	Compteur
m	Réel	Déterminer la valeur minimale d'une liste des valeurs

Traduction Pascal

```
PROGRAM Moyenne_Olympique;
USES wincrt ;
TYPE TAB=Array[1..20] of real ;
VAR
   n:integer ;
   T :TAB ;
FUNCTION MoyOlympique (n:integer;T:TAB)
:real;
VAR  i :integer ;
     S,m:real ;
FUNCTION Max(n:integer;T:TAB):real;
VAR
      i :integer ;
      m :real ;
BEGIN
   m := T[1] ;
   FOR i :=1 to n do
      if(T[i]>m) then
         m := T[i] ;
   Max :=m ;
END;
FUNCTION Min(n:integer;T:TAB):real;
VAR
      i :integer ;
      m :real ;
BEGIN
   m := T[1] ;
   FOR i :=1 to n do
      if(T[i]<m) then
         m := T[i] ;
   Min :=m ;
END;
BEGIN
   S :=0 ;
   FOR i :=1 to n do
      S :=s+ T[i];
   m :=(s-Min(n,T)-Max(n,T))/(n-2) ;
   MoyOlympique:=m ;
END;
```

```
PROCEDURE Saisie(VAR x :integer) ;
BEGIN
 repeat
    Write('Donner la valeur de N: ');
    Readln(x) ;
 until(x>=5)and(x<=20);
END;
PROCEDURE Remplir(n:integer;VAR
T:TAB);
VAR   i :integer ;
BEGIN
  FOR i :=1 to n do
   begin
      Write('T[',i, ']= ') ;
      Readln(T[i]) ;
   end ;
END;

BEGIN
Saisie(n) ;
Remplir(n,T);
Write('La moyenne olympique est : ',
MoyOlympique(n,T):2:3);
END.
```

✍ Décomposition modulaire du problème



✍ Analyse de programme principal :

Nom de programme : ADN_ARN
Résultat: PROC Afficher(N,R)
PROC TabARN(n,D,R)
PROC Remplir(n,D)
PROC Saisie(n)
Fin ADN_ARN

✍ Algorithme de programme principal:

0) *Début* ADN_ARN
1) PROC Saisie(n)
2) PROC Remplir(n,D)
3) PROC TabARN(n,D,R)
4) PROC Afficher(N,R)
5) *Fin* ADN_ARN

T.D.N.T

Type
TAB= Tableau de taille 30 et de type caractère

T.D.O Globaux

Objet	T/N	Rôle
N	Entier	stocker la taille du tableau
D	TAB	Remplir le tableau par le code ADN
R	TAB	Remplir le tableau par le code ARN du tableau D
Remplir	Procédure	Remplir un tableau par un code ADN.
Saisie	Procédure	Saisir la taille du tableau
TabARN	Fonction	Convertir un code ADN en code ARN

✍ Analyse de la procédure Saisie

DEF PROC SAISIE(VAR X :ENTIER)
Résultat : x
x=[]*Répéter*
x=**donnée**("Donner la valeur de N : ")
jusqu'à (x≥5)ET(x≤30)
Fin Saisie

✍ Algorithme de la procédure Saisie

0) DEF PROC SAISIE(VAR X :ENTIER)
1) Répéter
écrire("Donner la valeur de N: ")
lire(x)
jusqu'à (x≥5)ET(x≤30)
2) Fin Saisie

✍ **Analyse de la procédure Remplir**

```
DEF PROC REMPLIR(N :ENTIER,VAR T :TAB)
Résultat : T
T=[]Pour i de 1 à N faire
Répéter
     T[i]=donnée
Jusqu'à
Majus(T[i])dans["A","T","C","G"]
Fin Pour
i : compteur
Fin Remplir
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur

✍ **Algorithme de la procédure Remplir**

```
0) DEF PROC REMPLIR(N :ENTIER,VAR T:TAB)
1) Pour i de 1 à N faire
   Répéter
      lire(T[i])
   Jusqu'à
        Majus(T[i]) dans["A","T","C","G"]
  Fin Pour
2) Fin Remplir
```

✍ **Analyse de la procédure Afficher**

```
DEF PROC AFFICHER(N :ENTIER,T :TAB)
Résultat : Trait
Trait=[]Pour i de 1 à N faire
          Ecrire(T[i])
        Fin Pour
        Ecrire("Le code ARN est :")
i : compteur
Fin Afficher
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur

✍ **Algorithme de la procédure Afficher**

```
0) DEF PROC AFFICHER(N :ENTIER,T :TAB)
1) Ecrire("Le code ARN est :")
2) Pour i de 1 à N faire
      Ecrire(T[i])
   Fin Pour
3) Fin Afficher
```

✍ **Analyse de la procédure TabARN**

```
DEF PROC TabARN(N :ENTIER,D:TAB,VAR R:TAB)
Résultat : R
R=[]Pour i de 1 à N faire
          R[i]←FN ADNtoARN(D[i])
        Fin Pour
i : compteur
Fin TabARN
```

✍ **Algorithme de la procédure TabARN**

```
0) DEF PROC TabARN(N:ENTIER,D:TAB,VAR
   R:TAB)
1) Pour i de 1 à N faire
      R[i]←FN ADNtoARN(D[i])
  Fin Pour
2) Fin TabARN
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur
ADNtoARN	Fonction	Convertir un caractère de code ADN en caractère du code ARN

Analyse de la Fonction ADNtoARN

```
DEF FN ADNtoARN(C : CARACTERE): CARACTERE
Résultat : ADNtoARN ← c
Selon majus(c) faire
     "A" : c ← "U"
     "T" : c ← "A"
     "C" : c ← "G"
     "G" : c ← "C"
Fin selon
Fin ADNtoARN
```

Algorithme de la Fonction ADNtoARN

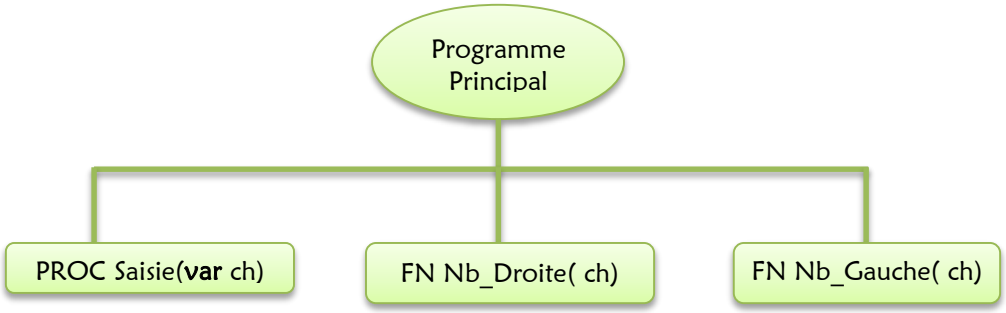
```
0) DEF FN ADNtoARN(C : CARACTERE): CARACTERE
1) Selon majus(c) faire
     "A" : c ← "U"
     "T" : c ← "A"
     "C" : c ← "G"
     "G" : c ← "C"
   Fin selon
2) ADNtoARN ← c
3) Fin ADNtoARN
```

Traduction Pascal

```
PROGRAM ADN_ARN;
USES wincrt ;
TYPE TAB=Array[1..30] of char ;
VAR
   n:integer ;
   A,B :TAB ;
PROCEDURE Saisie(VAR x :integer) ;
BEGIN
 repeat
    Write('Donner la valeur de N: ');
    Readln(x) ;
 until(x>=5)and(x<=30);
END;
PROCEDURE Remplir(n:integer;VAR T:TAB);
VAR   i :integer ;
BEGIN
Write('Le code ADN: ') ;
FOR i :=1 to n do
repeat
    Read(T[i]) ;
Until UPCASE(T[i])in ['A','T','C','G'];
END;
PROCEDURE Afficher(n:integer ;T :TAB);
VAR  i :integer ;
BEGIN
     Write('Le code ARN est : ') ;
     FOR i :=1 to n do
         Write(T[i]) ;
END;
```

```
PROCEDURE TabARN(n:integer;D :TAB;
VAR R:TAB);
VAR   i :integer ;
      FUNCTION ARNtoADN(c :char):char;
      BEGIN
         Case UPCASE(c) of
            'A' : c := 'U' ;
            'T' : c := 'A' ;
            'C' : c := 'G' ;
            'G' : c := 'C' ;
         End ;
         ARNtoADN:=c ;
      END;
BEGIN
FOR i :=1 to n do
   R[i] := ARNtoADN(D[i]) ;
END;

BEGIN
      Saisie(n) ;
      Remplir(n,D);
      TabARN(n,D,R);
      Afficher(n,R);
END.
```

✍ **Décomposition modulaire du problème**



✍ **Analyse de programme principal :**

```
Nom de programme : Extraire_Nombre
Résultat: Ecrire("Nombre droite : ",
FN Nb_Droite(ch)," Nombre gauche : ",
FN Nb_Gauche(ch))
PROC Saisie(ch)
Fin Extraire_Nombre
```

✍ **Algorithme de programme principal:**

```
0) Début Extraire_Nombre
1) PROC Saisie(ch)
2) Ecrire("Nombre droite : ",
FN Nb_Droite(ch)," Nombre
gauche : ",FN Nb_Gauche(ch))
3) Fin Extraire_Nombre
```

T.D.O Globaux

Objet	T/N	Rôle
ch	Chaine de caractère	stocker une chaine de caractère
Saisie	Procédure	Saisir une chaine de caractère
Nb_Droite	Fonction	Extraire un nombre formé par les chiffres de ch de droite à gauche
Nb_Gauche	Fonction	Extraire un nombre formé par les chiffres de ch de gauche à droite

✍ **Analyse de la procédure Saisie**

```
DEF PROC SAISIE(VAR CH:chaine de caractères)
Résultat : ch
ch=[]Répéter
 ch=donnée("Donner une chaine :  ")
jusqu'à long(ch)≠0
Fin Saisie
```

✍ **Algorithme de la procédure Saisie**

```
0) DEF PROC SAISIE(VAR ch:chaine
   de caractères)
1) Répéter
   écrire("Donner une chaine  ")
   lire(ch)
   jusqu'à long(ch)≠0
2) Fin Saisie
```

✍ **Analyse de la procédure Nb_Droite**

```
DEF PROC NB_Droite(CH:chaine de caractères) :ENTIER
Résultat : Nb_Droite←nb
Si ch1="" alors   nb←0
Sinon  valeur(ch1,nb,e)
```

```
Fin si
Pour i de long(ch) à 1 (pas=-1)faire
 Si(ch[i])dans['0'..'9'] alors
    Ch1←ch1+ch[i]
  Fin si
Fin Pour
i : compteur
Fin Nb_Droite
```

✍ **Algorithme de la procédure Nb_Droite**

```
0) DEF PROC NB_Droite(CH:chaine de
   caractères) :ENTIER
1) Pour i de long(ch) à 1 (pas=-1)faire
    Si(ch[i])dans['0'..'9'] alors
      Ch1←ch1+ch[i]
     Fin si
   Fin Pour
2) Si ch1="" alors   nb←0
   Sinon  valeur(ch1,nb,e)
   Fin si
3) Nb_Droite←nb
4) Fin Nb_Droite
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur
e	Entier	Position d'erreur
nb	Entier	Convertir la chaine ch1 en nombre
Ch1	chaine	Chaine auxiliaire

✍ **Analyse de la procédure Nb_Gauche**

DEF PROC NB_Gauche(CH:chaine de caractères) :ENTIER

```
Résultat : Nb_Gauche ←nb
Si ch1="" alors   nb←0
Sinon  valeur(ch1,nb,e)
Fin si
Pour i de 1 à long(ch) faire
 Si(ch[i])dans['0'..'9'] alors
    Ch1←ch1+ch[i]
  Fin si
Fin Pour
i : compteur
Fin Nb_Gauche
```

✍ **Algorithme de la procédure Nb_ Gauche**

```
0) DEF PROC NB_Gauche(CH:chaine de
   caractères) :ENTIER
1) Pour i de 1 à long(ch) faire
    Si(ch[i])dans['0'..'9'] alors
      Ch1←ch1+ch[i]
     Fin si
   Fin Pour
2) Si ch1="" alors   nb←0
   Sinon  valeur(ch1,nb,e)
   Fin si
3) Nb_Gauche←nb
4) Fin Nb_Gauche
```

✍ **T.D.O Locaux**

Objet	T/N	Rôle
i	Entier	Compteur
e	Entier	Position d'erreur
nb	Entier	Convertir la chaine ch1 en nombre
Ch1	chaine	Chaine auxiliaire

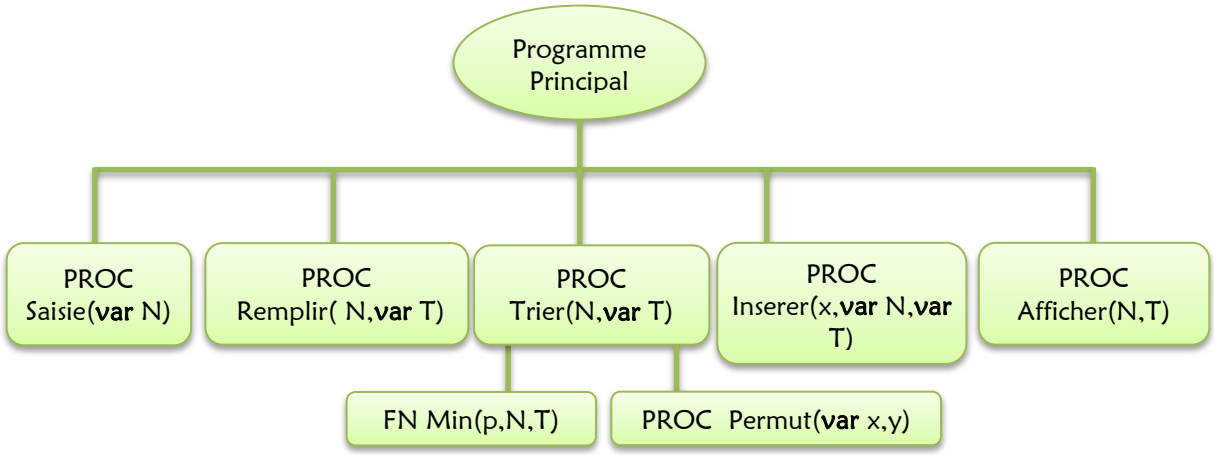
✍ **Traduction Pascal**

```
PROGRAM Extraire_Nombre;
USES wincrt ;
VAR Ch :string ;
PROCEDURE Saisie(VAR ch :string) ;
BEGIN
 repeat
    Write('Donner une chaine: ');
    Readln(ch) ;
 until length(ch)<>0 ;
END;
FUNCTION Nb_Droite(ch :string):longint;
VAR  nb :longint
     i,e :integer ;
     Ch1  :string ;
BEGIN
     FOR i :=length(ch) downto 1 do
      If ch[i] in ['0','9'] then
        Ch1 := Ch1+ Ch[i] ;
val(ch1,nb,e) ;  Nb_Droite:=nb ;
END;
```

```
FUNCTION Nb_Gauche(ch:string): longint;
VAR  nb : longint ;
     i,e :integer ;
     Ch1  :string ;
BEGIN
     FOR i :=1 to length(ch) do
      If ch[i] in ['0','9'] then
        Ch1 := Ch1+ Ch[i] ;
val(ch1,nb,e) ; Nb_Gauche:=nb ;
END;
BEGIN
Saisie(ch) ;
Write('Nombre droite : ',
Nb_Droite(ch),' Nombre gauche : ',
 Nb_gauche(ch)) ;
END.
```

Exercice 11

✍ **Décomposition modulaire du problème**



✍ **Analyse de programme principal :**

Nom de programme : Inserer_Nombre
Résultat : PROC Afficher(n,T)
PROC Inserer(x,n,T)
PROC Trier(n,T)
PROC Remplir(n,T)
PROC Saisie(n)
X=donnée("Donner l'entier à insérer")
Fin Inserer_Nombre

✍ **Algorithme de programme principal:**

0) *Début* Inserer_Nombre
1) PROC Saisie(n)
2) PROC Remplir(n,T)
3) PROC Trier(n,T)
4) Ecrire("Donner l'entier à insérer")
5) Lire(x)
6) PROC Inserer(x,n,T)
7) PROC Afficher(n,T)
8) *Fin* Inserer_Nombre

T.D.N.T

Type
TAB= Tableau de taille 30 et de type entier

T.D.O Globaux

Objet	T/N	Rôle
N	Entier	stocker la taille du tableau
T	TAB	Remplir le tableau par N entiers
x	Entier	Saisir un entier à insérer dans le tableau T
Afficher	Procédure	Afficher un tableau
Remplir	Procédure	Remplir le tableau par n entiers.
Saisie	Procédure	Saisir la taille du tableau
Trier	Procédure	Trier le tableau T dans l'ordre croissant
Insérer	Procédure	Insérer l'entier x dans le Tableau T en conservant l'ordre

✍ **Analyse de la procédure Saisie**

```
DEF PROC SAISIE(VAR X :ENTIER)
Résultat : x
x=[]Répéter
 x=donnée("Donner la valeur de N :  ")
jusqu'à (x≥5)ET(x≤30)
Fin Saisie
```

✍ **Algorithme de la procédure Saisie**

```
0) DEF PROC SAISIE(VAR X :ENTIER)
1) Répéter
     écrire("Donner la valeur de N: ")
     lire(x)
     jusqu'à (x≥5)ET(x≤30)
2) Fin Saisie
```

✍ **Analyse de la procédure Remplir**

```
DEF PROC REMPLIR(N :ENTIER ;VAR T :TAB)
Résultat : T
T =[]Pour i de 1 à N faire
  T[i]=donnée("T[",i, "]=")
Fin Pour
i : compteur
Fin Remplir
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur

✍ **Algorithme de la procédure Remplir**

```
0) DEF PROC REMPLIR(N :ENTIER ;VAR T:TAB)
1) Pour i de 1 à N faire
       écrire("T[",i, "]=")
       lire(T[i])
   Fin Pour
2) Fin Remplir
```

✍ **Analyse de la procédure Afficher**

```
DEF PROC AFFICHER(N :ENTIER ;T :TAB)
Résultat : Trait
Trait=[]Pour i de 1 à N faire
          Ecrire("T[",i, "]=",T[i])
        Fin Pour
i : compteur
Fin Afficher
```

T.D.O Locaux

✍ **Algorithme de la procédure Afficher**

```
0) DEF PROC AFFICHER(N:ENTIER ;T :TAB)
1) Pour i de 1 à N faire
       Ecrire("T[",i, "]=",T[i])
   Fin Pour
2) Fin Afficher
```

Objet	T/N	Rôle
i	Entier	Compteur

✍ **Analyse de la procédure Trier**

```
DEF PROC TRIER(N :ENTIER ;VAR T :TAB)
Résultat : T
T =[]Pour i de 1 à N-1 faire
       posmin←FN Min(i,N,T)
       si posmin ≠ i alors
        PROC Permut(T[posmin],T[i])
       Fin Si
     Fin Pour
i : compteur
Fin Trier
```

✍ **Algorithme de la procédure Trier**

```
0) DEF PROC TRIER(N :ENTIER ;VAR T:TAB)
1) Pour i de 1 à N faire
       posmin←FN Min(i,N,T)
      si posmin ≠ i alors
        PROC Permut(T[posmin],T[i])
      Fin Si
   Fin Pour
2) Fin Trier
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur
Min	Fonction	Déterminer la position de minimum dans un tableau à partir d'une position pos
Permut	Procédure	Permuter deux entiers dans un tableau

✍ **Analyse de la Fonction Min**

```
DEF FN Min(pos,n:ENTIER;T:TAB):ENTIER
Résultat : Min ← posmin
posmin =[posmin←pos]
Pour i de pos+1 à n faire
     si(T[i]<T[posmin]) alors
          posmin←i
     Fin Si
Fin pour
i:compteur
Fin Min
```

✍ **Algorithme de la Fonction Min**

```
0) DEF FN Min(pos,n:ENTIER ;T:TAB):
ENTIER
1) [posmin←pos]
Pour i de pos+1 à n faire
    si(T[i]<T[posmin]) alors
       posmin←i
    Fin Si
Fin pour
2) Min ← posmin
3) Fin Min
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur
posmin	entier	Déterminer la position de minimum dans un tableau à partir d'une position pos

✍ **Analyse de la procédure Permut**

```
DEF PROC PERMUT(VAR X,Y :ENTIER)
Résultat : x,y
          aux←x
          x←y
          y←aux
Fin Permut
```

✍ **Algorithme de la procédure Permut**

```
0) DEF PROC PERMUT(VAR X,Y :ENTIER)
1) Aux←x
2) X←y
3) Y←aux
4) Fin Permut
```

T.D.O Locaux

Objet	T/N	Rôle
aux	Entier	Variable auxiliaire

✍ **Analyse de la procédure Insérer**

```
DEF PROC INSERER(X:ENTIER ;VAR N :ENTIER ;VAR T :TAB)
Résultat : N,T
T[i+1]←x
N←N+1
[i←N]Tant que (T[i]>x)et(i>1) faire
      T[i+1]←T[i]
      i←i-1
Fin Tant que
Fin Trier
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur

✍ **Algorithme de la procédure Insérer**

```
0) DEF PROC INSERER(X:ENTIER; VAR N:ENTIER;
   VAR T:TAB)
1) [i←N]Tant que (T[i]>x)et(i>1) faire
      T[i+1]←T[i]
      i←i-1
   Fin Tant que
2) T[i+1]←x
3) N←N+1
4) Fin Inserer
```

✍ **Traduction Pascal**

```
PROGRAM Inserer_Nombre;
USES wincrt ;
TYPE TAB=Array[1..20] of integer ;
VAR
   N,x:integer ;
   T :TAB ;
PROCEDURE Saisie(VAR x :integer) ;
BEGIN
 repeat
    Write('Donner la valeur de N: ');
    Readln(x) ;
 until(x>=5)and(x<=30);
END;
PROCEDURE Remplir(n:integer;VAR T:TAB);
VAR   i :integer ;
BEGIN
     for i :=1 to n do
     begin
       Write('T[',i, ']= ') ;
       Readln(T[i]) ;
     end;
END;
PROCEDURE Afficher(n:integer ;T :TAB) ;
VAR  i :integer ;
BEGIN
write('T=') ;
for i :=1 to n do
    Write(T[i],' ') ;
END;
PROCEDURE Inserer(x:integer; VAR
```

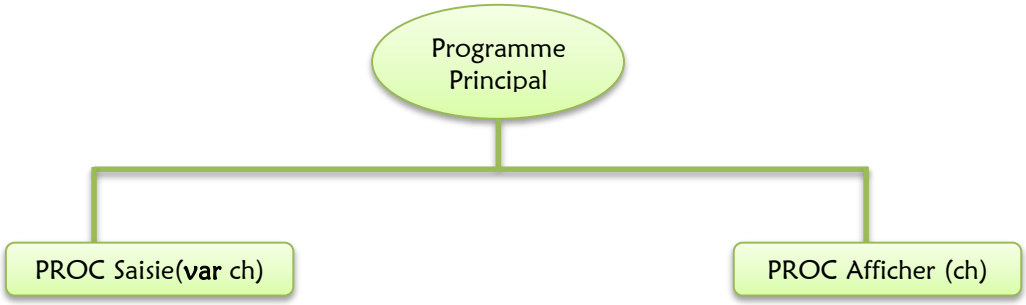
```
PROCEDURE Trier(n:integer ; VAR
T :TAB) ;
VAR posmin,i :integer ;
FUNCTION Min(pos,n:integer;T:TAB):integer;
VAR   posmin,i :integer ;
BEGIN
      posmin:=pos;
      for i := pos+1 to n do
            if(T[i]<T[posmin]) then
                  posmin:=i;
      Min := posmin;
END;
PROCEDURE Permut(VAR x,y:integer);
VAR  aux :integer ;
BEGIN
      aux :=x ;
      x :=y ;
      y :=aux ;
END;
BEGIN
      for i :=1 to n do
      begin
        Posmin := Min(i,N,T) ;
           if posmin <> i then
             Permut(T[posmin],T[i]) ;
      end;
END;



BEGIN
 Saisie(n) ;
```

```
n:integer ;VAR T:TAB);
VAR   i :integer ;
BEGIN
      i:=N ;
      while (T[i]>x)and(x>1)do
      Begin
            T[i+1] :=T[i] ;
            i :=i-1 ;
      end ;
      T[i+1] :=x ;
      N :=N+1 ;
END;
```

```
 Remplir(n,T);
 Trier(n,T) ;
 Write('Donner l'entier à insérer: ');
 Readln(x) ;
 Inserer(x,n,T) ;
 Afficher(n,T) ;
END.
```

Exercice 12

✍ **Décomposition modulaire du problème**



✍ **Analyse de programme principal :**

Nom de programme : Sablier
Résultat: PROC Afficher(ch)
PROC Saisie(ch)
Fin Sablier

✍ **Algorithme de programme principal:**

0) *Début* Sablier
1) PROC Saisie(ch)
2) PROC Afficher(ch)
3) *Fin* Sablier

T.D.O Globaux

Objet	T/N	Rôle
ch	Chaine de caractère	stocker une chaine de caractère
Saisie	Procédure	Saisir une chaine de caractère
Afficher	Procédure	Afficher une chaine sous forme d'un sablier

✍ **Analyse de la procédure Saisie**

DEF PROC SAISIE(VAR CH:chaine de caractères)
Résultat : ch
ch=[]*Répéter*
 ch=**donnée**("Donner une chaine : ")
jusqu'à long(ch) mod 2 ≠0
Fin Saisie

✍ **Algorithme de la procédure Saisie**

0) DEF PROC SAISIE(VAR ch:chaine de caractères)
1) Répéter
écrire("Donner une chaine : ")
lire(ch)
jusqu'à long(ch) mod 2 ≠ 0
2) Fin Saisie

✍ **Analyse de la procédure Afficher**

```
DEF PROC AFFICHER(CH :chaine de caractères)
Résultat : Trait
Trait=[]Pour i de 1 à long(ch) faire
     Si i<=long(ch)div 2 alors
        n← long(ch)-2*(i-1)
        P← i
     Sinon
        n← 2*i-long(ch)
        P← long(ch)-i +1
     Fin Si
     nb←p-1+n
     Ecrire(sous-chaine(ch,p,n):nb)
     Fin Pour
i : compteur
Fin Afficher
```

✍ **Algorithme de la procédure Afficher**

```
0) DEF PROC AFFICHER(CH: chaine de
   caractères)
1) Pour i de 1 à long(ch) faire
   Si i<=long(ch)div 2 alors
      n ← long(ch)-2*(i-1)
      P←i
   Sinon
      n ← 2*i-long(ch)
      P← long(ch)-i +1
   Fin Si
   nb←p-1+n
   Ecrire(sous-chaine(ch,p,n):nb)
    Fin Pour
2) Fin Afficher
```

T.D.O Locaux

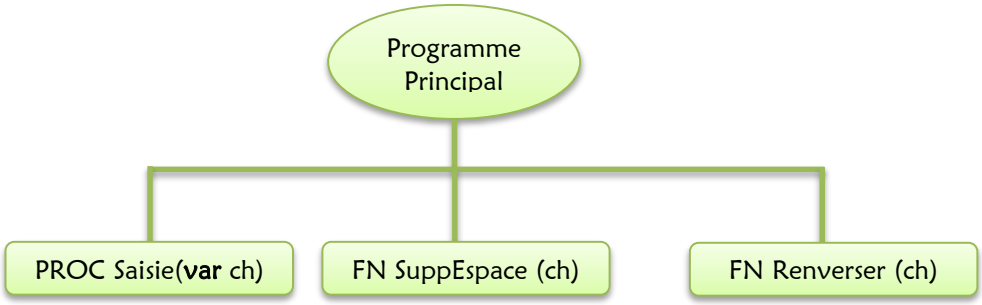
Objet	T/N	Rôle
i	Entier	Compteur
n	Entier	Calculer le nombre de caractère à afficher
p	Entier	Calculer la position dont lequel on va commencer à afficher la chaine ch
nb	Entier	Calculer le nombre position pour afficher la chaine ch

✍ **Traduction Pascal**

```
PROGRAM Sablier ;
USES wincrt ;
VAR Ch :string ;
PROCEDURE Afficher(ch :string) ;
VAR    n,p,nb,i :integer;
BEGIN
for i:=1 to length(ch) do
begin
     if(i<=length(ch) div 2) then
     begin
          n:= length(ch)-2*(i-1) ;
          p:=i;
     end
     else
     begin
          n:=2*i-length(ch) ;
          p:=length(ch)-i+1;
     end;
nb:=p-1+n;   writeln(copy(ch,p,n):nb) ;
end;
END;
```

```
PROCEDURE Saisie(VAR ch :string) ;
BEGIN
 REPEAT
    Write('Donner une chaine: ');
    Readln(ch) ;
 UNTIL length(ch) mod 2 <> 0 ;
END;
BEGIN
    Saisie(ch) ;
    Afficher(ch) ;
END.
```

✍ **Décomposition modulaire du problème**



✍ **Analyse de programme principal :**

```
Nom de programme : Renverser_chaine
Résultat: écrire("Chaine résultat: ",
                 FN Renverser(ch))
Ch←FN SuppEspace(ch)
PROC Saisie(ch)
Fin Renverser_chaine
```

✍ **Algorithme de programme principal:**

```
0) Début Renverser_chaine
1) PROC Saisie(ch)
2) Ch←FN SuppEspace(ch)
3) écrire("Chaine résultat:
   ",FN Renverser(ch))
4) Fin Renverser_chaine
```

T.D.O Globaux

Objet	T/N	Rôle
ch	Chaine de caractère	stocker une chaine de caractère
Saisie	Procédure	Saisir une chaine de caractère
SuppEspace	Fonction	Supprimer les espaces inutiles dans une chaine
Renverser	Fonction	Renversée les mots d'une phrase

✍ **Analyse de la procédure Saisie**

```
DEF PROC SAISIE(VAR CH:chaine de caractères)
Résultat : ch
ch=[]Répéter
 ch=donnée("Donner une chaine :  ")
jusqu'à Majus(ch[1]) dans ["A".."Z"]
Fin Saisie
```

✍ **Algorithme de la procédure Saisie**

```
0) DEF PROC SAISIE(VAR ch:chaine
   de caractères)
1) Répéter
   écrire("Donner une chaine : ")
   lire(ch)
   jusqu'à Majus(ch[1]) dans
   ["A".."Z"]
2) Fin Saisie
```

✍ **Analyse de la Fonction Renverser**

```
DEF FN Renverser(CH:chaine de caractères): chaine de caractères
Résultat : Renverser ← ch1
Ch1 =[p←1]
Pour i de 1 à long(ch) faire
    si(ch[i]=" ") alors
      mot←" "+sous-chaine(ch,p,i-p)
      p←i+1
    Fin Si
```

```
    si(i=long(ch)) alors
      mot←sous-chaine(ch,p,i-p+1)
    Fin Si
    Insérer(mot,ch1,1)
Fin pour
i:compteur
Fin Renverser
```

✍ **Algorithme de la Fonction Renverser**

```
0) DEF FN Renverser(ch :chaine de
     caractères): chaine de caractères
1) [p←1] Pour i de 1 à long(ch) faire
     si(ch[i]=" ") alors
       mot←" "+sous-chaine(ch,p,i-p)
       p←i+1
     Fin Si
     si(i=long(ch)) alors
       mot←sous-chaine(ch,p,i-p+1)
     Fin Si
     Insérer(mot,ch1,1)
Fin pour
2) Renverser ← ch1
3) Fin Renverser
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur
p	entier	Déterminer la position de première lettre de chaque mot
Ch1	Chaine de caractères	Contient la phrase renversée
mot	Chaine de caractères	Contient un mot de la phrase

✍ **Analyse de la Fonction SuppEspace**

DEF FN SuppEspace(CH:chaine de caractères): chaine de caractères

```
Résultat : SuppEspace ← ch
Si ch[long(ch)] = " " alors
    Efface(ch,long(ch),1)
Fin Si
Ch =[i←1]
Répéter
Si ch[i] = " " alors
    Si i=1 alors p←1
    Sinon p←i+1
    Fin si
[nb←0,j←p]
Tant que(ch[j]=" ")et(j≤long(ch))faire
    j←j+1
    nb←nb+1
Fin Tant que
Efface(ch,p,nb)
Fin Si
i←i+1
Jusqu'à i>long(ch)
Fin SuppEspace
```

✍ **Algorithme de la Fonction SuppEspace**

```
0) DEF FN SuppEspace(ch :chaine de
     caractères): chaine de caractères
1) [i←1]Répéter
Si ch[i] = " " alors
     Si i=1 alors p←1
     Sinon p←i+1
     Fin si
[nb←0,j←p]
Tant que(ch[j]=" ")et(j≤long(ch))faire
     j←j+1
     nb←nb+1
Fin Tant que
Efface(ch,p,nb)
Fin Si
i←i+1
Jusqu'à i>long(ch)
2) Si ch[long(ch)] = " " alors
     Efface(ch,long(ch),1)
Fin Si
3) SuppEspace ← ch
4) Fin SuppEspace
```

T.D.O Locaux

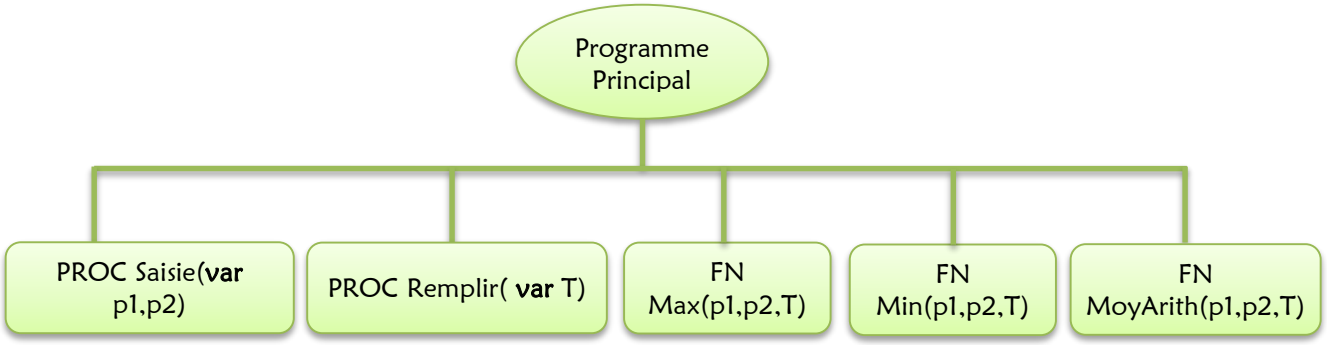
Objet	T/N	Rôle
i,j	Entier	Compteur
p	Entier	Déterminer une position de départ pour supprimer les espaces
nb	Entier	Calculer le nombre des espaces à supprimer

✍ **Traduction Pascal**

```
PROGRAM Renverser_chaine;
USES wincrt ;
VAR Ch :string ;
PROCEDURE Saisie(VAR ch :string) ;
BEGIN
 repeat
    Write('Donner une chaine: ');
    Readln(ch) ;
 until Upcase(ch[1]) in ['A'..'Z'] ;
END;
FUNCTION SuppEspace(ch:string):string ;
VAR    p,nb,i,j :integer;
BEGIN
i:=1;
Repeat
if ch[i] = ' ' then
begin
     if i=1 then p:=1
     else p:=i+1;
nb:=0;j:=p;
while(ch[j]=' ')and(j<=length(ch))do
begin
     j:=j+1;
     nb:=nb+1;
end;
delete(ch,p,nb);
end;
i:=i+1;
until i>length(ch);
if ch[length (ch)] = ' ' then
     delete(ch,length(ch),1);
SuppEspace := ch;
END;
```

```
FUNCTION Renverser(ch:string):string ;
VAR    p,i :integer ;
       Ch1,Mot :string ;
BEGIN
P:=1;ch1:= '';
for i:=1 to length(ch) do
begin
     if ch[i]= ' ' then
      begin
        mot:= ' '+copy(ch,p,i-p);
        ch1:=mot+ch1;
         p:=i+1;
     end;
     if i=length(ch) then
      begin
        mot:= copy (ch,p,i-p+1);
        ch1:=mot+ch1;
       end;
end;
Renverser := ch1;
END;
BEGIN
     Saisie(ch) ;
     Ch :=SuppEspace(ch) ;
     Write('Chaine résultat: ',
     Renverser(ch));

END.
```

Décomposition modulaire du problème



Analyse de programme principal :

```
Nom de programme : Calcul
Résultat: Ecrire("La moyenne arithmétique: ",
      FN MoyArith(p1,p2,T))
Ecrire("Le min: ",FN Min(p1,p2,T))
Ecrire("Le maximum: ",FN Max(p1,p2,T))
PROC Afficher(p1,p2,T)
PROC Remplir(T)
PROC Saisie(p1,p2)
Fin Calcul
```

T.D.N.T

Type
TAB= Tableau de taille 20 et de type entier

T.D.O Globaux

Objet	T/N	Rôle
P1,p2	Entier	Saisir deux positions dans le tableau
T	TAB	Remplir les tableaux par N réels
Remplir	Procédure	Remplir un tableau par N réels
Saisie	Procédure	Saisir les deux positions p1 et p2
Afficher	Procédure	Afficher les éléments du tableau compris entre p1 et p2
Max	Fonction	Déterminer le maximum entre p1 et p2
Min	Fonction	Déterminer le minimum entre p1 et p2
MoyArith	Fonction	Calculer la moyenne des entiers compris entre p1 et p2

Algorithme de programme principal:

```
0) Début Calcul
1) PROC Saisie(p1,p2)
2) PROC Remplir(T)
3) PROC Afficher(p1,p2,T)
4) Ecrire("Le maximum: ",FN
   Max(p1,p2,T))
5) Ecrire("Le min: ",FN
   Min(p1,p2,T))
6) Ecrire("La moyenne
   arithmétique: ", FN
   MoyArith (n,T))
7) Fin Calcul
```

Analyse de la procédure Saisie

```
DEF PROC SAISIE(VAR P1,P2 :ENTIER)
Résultat : p1,p2
x=[]Répéter
 p1=donnée("P1=:  ")
 p2=donnée("P2=:  ")
jusqu'à (p1≥1)ET(p2≤20)ET(p1≤p2)
Fin Saisie
```

Algorithme de la procédure Saisie

```
0) DEF PROC SAISIE(VAR P1,P2:ENTIER)
1) Répéter
     écrire("P1= ") lire(p1)
     écrire("P2= ") lire(p2)
     jusqu'à (p1≥1)ET(p2≤20)ET(p1≤p2)
2) Fin Saisie
```

✍ **Analyse de la procédure Remplir**

```
DEF PROC REMPLIR(VAR T :TAB)
Résultat : T
T=[]Pour i de 1 à 20 faire
          Répéter
          T[i]=donnée("T[",i, "]=")
          jusqu'à T[i]≥0
Fin Pour
i : compteur
Fin Remplir
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur

✍ **Algorithme de la procédure Remplir**

```
0) DEF PROC REMPLIR(VAR T:TAB)
1) Pour i de 1 à 20 faire
   Répéter
     écrire("T[",i, "]=")
     lire(T[i])
     jusqu'à T[i]≥0
   Fin Pour
2) Fin Remplir
```

✍ **Analyse de la Fonction MoyArith**

```
DEF FN MoyArith(P1,P2:ENTIER ; T:TAB): REEL
Résultat : Moyarith ← s/(p1-p2+1)
S=[s←0]
Pour i de p1 à p2 faire
     S←s+T[i]
Fin pour
i:compteur
Fin MoyArith
```

✍ **Algorithme de la Fonction MoyArith**

```
0) DEF FN MoyArith(P1,P2:ENTIER;T:TAB):REEL
1) [s←0]
2) Pour i de p1 à p2 faire
       S←s+T[i]
   Fin pour
3) MoyArith ←s/(p1-p2+1)
4) Fin MoyArith
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur
s	Réel	Calculer la somme des éléments d'un vecteur

✍ **Analyse de la Fonction Max**

```
DEF FN Max(P1,P2:ENTIER,T:TAB):ENTIER
Résultat : Max ← m
m=[m←T[p1]]
Pour i de p1 à p2 faire
     si(T[i]>m) alors
          m←T[i]
     Fin Si
Fin pour
i:compteur
Fin Max
```

✍ **Algorithme de la Fonction Max**

```
0) DEF FN Max(P1,P2:ENTIER,T:TAB):ENTIER
1) [m←T[p1]]
2) Pour i de p1 à p2 faire
       si(T[i]>m) alors
            m←T[i]
       Fin Si
   Fin pour
3) Max ← m
4) Fin Max
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur
m	Réel	Déterminer la valeur maximale d'une liste des valeurs

✍ **Analyse de la Fonction Min**

```
DEF FN Min(P1,P2:ENTIER,T:TAB):ENTIER
Résultat : Max ← m
m=[m←T[p1]]
Pour i de p1 à p2 faire
    si(T[i]<m) alors
        m←T[i]
    Fin Si
Fin pour
i:compteur
Fin Min
```

✍ **Algorithme de la Fonction Min**

```
0) DEF FN Min(P1,P2:ENTIER,T:TAB):ENTIER
1) [m←T[1]]
2) Pour i de 1 à n faire
       si(T[i]<m) alors
           m←T[i]
       Fin Si
   Fin pour
3) Min ← m
4) Fin Min
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur
m	Réel	Déterminer la valeur minimale d'une liste des valeurs

✍ **Analyse de la procédure Afficher**

```
DEF PROC AFFICHER(P1,P2 :ENTIER,T :TAB)
Résultat : Trait
Trait=[]Pour i de p1 à p2 faire
          Ecrire("T[",i, "]=",T[i])
        Fin Pour
i : compteur
Fin Afficher
```

✍ **Algorithme de la procédure Afficher**

```
0) DEF PROC AFFICHER(P1,P2:ENTIER,T:TAB)
1) Pour i de p1 à p2 faire
       Ecrire("T[",i, "]=",T[i])
   Fin Pour
2) Fin Afficher
```

T.D.O Locaux

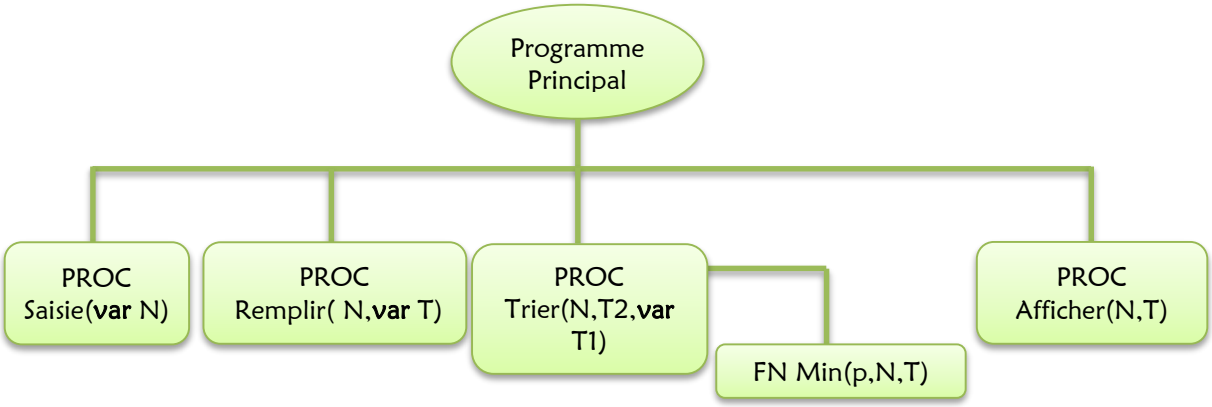
Objet	T/N	Rôle
i	Entier	Compteur

✍ **Traduction Pascal**

```
PROGRAM Calcul;
USES wincrt ;
TYPE TAB=Array[1..20] of integer ;
VAR
   n:integer ;
   T :TAB ;
FUNCTION MoyArith (p1,p2:integer;T:TAB)
:real;
VAR  i :integer ;
     S:integer ;
BEGIN
   S :=0 ;
   FOR i :=p1 to p2 do
     S :=s+ T[i];
   MoyArith:= s/(p2-p1+1) ;
END;
FUNCTION Max(p1,p2:integer;T:TAB):
integer;
VAR
      i,m :integer ;
BEGIN
   m := T[p1] ;
   FOR i :=p1+1 to p2 do
     if(T[i]>m) then
        m := T[i] ;
   Max :=m ;
END;
FUNCTION Min(p1,p2:integer;T:TAB):
integer;
VAR
      i,m :integer ;
BEGIN
   m := T[p1] ;
   FOR i :=p1 to p2 do
     if(T[i]<m) then
        m := T[i] ;
   Min :=m ;
END;
```

```
PROCEDURE Saisie(VAR p1,p2:integer) ;
BEGIN
 repeat
    Write('P1: '); Readln(p1) ;
    Write('P2: '); Readln(p2) ;
 until(p1>=5)and(p2<=20)and(p1<=p2);
END;
PROCEDURE Remplir(VAR T:TAB);
VAR   i :integer ;
BEGIN
  for i :=1 to 20 do
   repeat
      Write('T[',i, ']= ') ;
      Readln(T[i]) ;
   until T[i]>=0 ;
END;
PROCEDURE Afficher(p1,p2:integer ;
VAR T:TAB);
VAR   i :integer ;
BEGIN
  for i :=p1 to p2 do
     Write('T[',i, ']= ',T[i]) ;
END;

BEGIN
Saisie(p1,p2) ;
Remplir(T);
Afficher(p1,p2,T) ;
Write('Le maximum: ',Max(p1,p2,T));
Write('Le minimum: ',Min(p1,p2,T));
Write('La moyenne arithmétique: ',
MoyArith(n,T):2:3);
END.
```

✍ **Décomposition modulaire du problème**



✍ **Analyse de programme principal :**

Nom de programme : Tri
Résultat : PROC Afficher(n,T2)
PROC Trier(n,T1,T2)
PROC Remplir(n,T1)
PROC Saisie(n)
Fin Tri

✍ **Algorithme de programme principal:**

0) *Début* Tri
1) PROC Saisie(n)
2) PROC Remplir(n,T1)
3) PROC Trier(n,T1,T2)
4) PROC Afficher(n,T2)
5) *Fin* Tri

T.D.N.T

Type
TAB= Tableau de taille 20 et de type caractère

T.D.O Globaux

Objet	T/N	Rôle
N	Entier	stocker la taille du tableau
T1	TAB	Remplir le tableau par N lettres majuscules
T2	TAB	Contient les éléments de T1 dans l'ordre croissant
Afficher	Procédure	Afficher un tableau
Remplir	Procédure	Remplir le tableau par n lettres majuscules.
Saisie	Procédure	Saisir la taille du tableau
Trier	Procédure	Trier le tableau T dans l'ordre croissant

✍ **Analyse de la procédure Saisie**

DEF PROC SAISIE(VAR X :ENTIER)
Résultat : x
x=[]*Répéter*
x=**donnée**("Donner la valeur de N : ")
jusqu'à (x≥5)ET(x≤20)
Fin Saisie

✍ **Algorithme de la procédure Saisie**

0) DEF PROC SAISIE(VAR X :ENTIER)
1) Répéter
écrire("Donner la valeur de N: ")
lire(x)
jusqu'à (x≥5)ET(x≤30)
2) Fin Saisie

✍ **Analyse de la procédure Remplir**

```
DEF PROC REMPLIR(N :ENTIER ;VAR T :TAB)
Résultat : T
T =[]Pour i de 1 à N faire
     répéter
T[i]=donnée("T[",i, "]=")
Jusqu'à Majus(T[i])dans["A".."Z"]
Fin Pour
i : compteur
Fin Remplir
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur

✍ **Algorithme de la procédure Remplir**

```
0) DEF PROC REMPLIR(N :ENTIER ;VAR T:TAB)
1) Pour i de 1 à N faire
   répéter
      écrire("T[",i, "]=")
      lire(T[i])
   jusqu'à Majus(T[i])dans["A".."Z"]
  Fin Pour
2) Fin Remplir
```

✍ **Analyse de la procédure Afficher**

```
DEF PROC AFFICHER(N :ENTIER ;T :TAB)
Résultat : Trait
Trait=[]Pour i de 1 à N faire
          Ecrire("T[",i, "]=",T[i])
        Fin Pour
i : compteur
Fin Afficher
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur

✍ **Algorithme de la procédure Afficher**

```
0) DEF PROC AFFICHER(N :ENTIER ;T:TAB)
1) Pour i de 1 à N faire
      Ecrire("T[",i, "]=",T[i])
   Fin Pour
2) Fin Afficher
```

✍ **Analyse de la procédure Trier**

```
DEF PROC TRIER(N:ENTIER;T1:TAB;VAR T2:TAB)
Résultat : T
T2 =[]Pour i de 1 à N faire
       posmin←FN Min(N,T1)
     T2[i]←T1[posmin]
     T1[posmin]←"*"
     Fin Pour
i : compteur
Fin Trier
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur
Min	Fonction	Déterminer la position de minimum dans un tableau
posmin	Entier	Contient la position de minimum dans un tableau

✍ **Algorithme de la procédure Trier**

```
0) DEF PROC TRIER(N:ENTIER;T1:TAB;VAR
   T2:TAB)
1) Pour i de 1 à N faire
   posmin←FN Min(N,T1)
   T2[i]←T1[posmin]
   T1[posmin]←"*"
  Fin Pour
2) Fin Trier
```

Analyse de la Fonction Min

```
DEF FN Min(n:ENTIER ;T:TAB):ENTIER
Résultat : Min ← posmin
posmin =[posmin←1]
Pour i de 2 à n faire
Si T[i]≠"*" alors
    Si (T[i]<T[posmin]) ou
       (T[posmin]="*")  alors
          posmin←i
    Fin Si
Fin Si
Fin pour
i:compteur
Fin Min
```

Algorithme de la Fonction Min

```
0) DEF FN Min(n:ENTIER ;T:TAB): ENTIER
1) [posmin←1]
Pour i de 2 à n faire
Si T[i]≠"*" alors
    Si (T[i]<T[posmin]) ou
       (T[posmin]="*")  alors
          posmin←i
     Fin Si
Fin Si
Fin pour
2) Min ← posmin
3) Fin Min
```

T.D.O Locaux

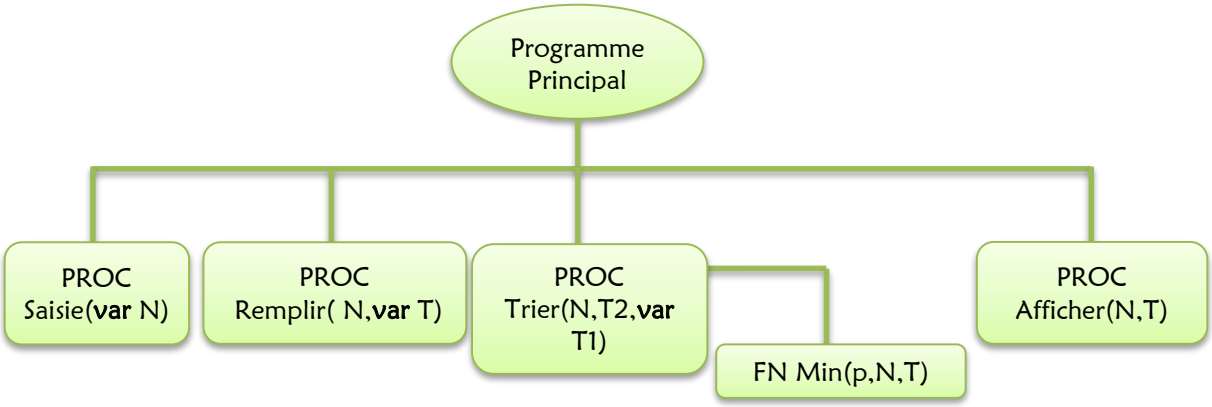
Objet	T/N	Rôle
i	Entier	Compteur
posmin	entier	Déterminer la position du petit caractère dans un tableau

Traduction Pascal

```
PROGRAM Tri;
USES wincrt ;
TYPE TAB=Array[1..20] of char ;
VAR N:integer ;
    T1,T2 :TAB ;
PROCEDURE Saisie(VAR x :integer) ;
BEGIN
 repeat
    Write('Donner la valeur de N: ');
    Readln(x) ;
 until(x>=5)and(x<=20);
END;
PROCEDURE Remplir(n:integer;VAR T:TAB);
VAR   i :integer ;
BEGIN
  for i :=1 to n do
   repeat
     Write('T[',i, ']= ') ;
     Readln(T[i]) ;
   until Upcase(T[i])in['A'..'Z'] ;
END;
PROCEDURE Afficher(n:integer ;T :TAB) ;
VAR  i :integer ;
BEGIN
for i :=1 to n do  Write(T[i],' ') ;
END;
```

```
PROCEDURE Trier(n:integer ;T1 :TAB ;
VAR T2 :TAB) ;
VAR posmin,i :integer ;
FUNCTION Min(n:integer;T:TAB):integer;
VAR   posmin,i :integer ;
BEGIN
posmin:=1;
      for i := 2 to n do
      if T[i]<>'*' then
          if (T[i]<T[posmin]) or
             (T[posmin]="*")  then
                 Posmin :=i ;
Min := posmin;
END;
BEGIN
      for i :=1 to n do
      begin
      Posmin := Min(N,T1) ;
      T2[i]:=T1[posmin];
      T1[posmin]:='*';
      end;
END;
BEGIN
 Saisie(n) ; Remplir(n,T1);
 Trier(n,T1,T2) ;
 Afficher(n,T2) ;
END.
```

✍ **Décomposition modulaire du problème**



✍ **Analyse de programme principal :**

Nom de programme : Tri
Résultat : PROC Afficher(n,B)
PROC Trier(n,A,B)
PROC Remplir(n,A)
PROC Saisie(n)
Fin Tri

✍ **Algorithme de programme principal:**

0) *Début* Tri
1) PROC Saisie(n)
2) PROC Remplir(n,A)
3) PROC Trier(n,A,B)
4) PROC Afficher(n,B)
5) *Fin* Tri

T.D.N.T

Type
TAB= Tableau de taille 25 et de type entier

T.D.O Globaux

Objet	T/N	Rôle
N	Entier	stocker la taille du tableau
A	TAB	Remplir le tableau par N entiers
B	TAB	Contient les éléments de T1 dans l'ordre croissant
Afficher	Procédure	Afficher un tableau
Remplir	Procédure	Remplir le tableau par n entiers.
Saisie	Procédure	Saisir la taille du tableau
Trier	Procédure	Trier le tableau T dans l'ordre croissant

✍ **Analyse de la procédure Saisie**

DEF PROC SAISIE(VAR X :ENTIER)
Résultat : x
x=[]*Répéter*
x=**donnée**("Donner la valeur de N : ")
jusqu'à (x>5)ET(x<25)
Fin Saisie

✍ **Algorithme de la procédure Saisie**

0) DEF PROC SAISIE(VAR X :ENTIER)
1) Répéter
écrire("Donner la valeur de N: ")
lire(x)
jusqu'à (x>5)ET(x<25)
2) Fin Saisie

✍ **Analyse de la procédure Remplir**

```
DEF PROC REMPLIR(N :ENTIER ;VAR T :TAB)
Résultat : T
T =[]Pour i de 1 à N faire
     répéter
T[i]=donnée("T[",i, "]=")
Jusqu'à T[i]≥0
Fin Pour
i : compteur
Fin Remplir
```

✍ **Algorithme de la procédure Remplir**

```
0) DEF PROC REMPLIR(N :ENTIER ;VAR T:TAB)
1) Pour i de 1 à N faire
   répéter
      écrire("T[",i, "]=")
      lire(T[i])
   jusqu'à T[i]≥0
  Fin Pour
2) Fin Remplir
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur

✍ **Analyse de la procédure Afficher**

```
DEF PROC AFFICHER(N :ENTIER ;T :TAB)
Résultat : Trait
Trait=[]Pour i de 1 à N faire
          Ecrire("T[",i, "]=",T[i])
         Fin Pour
i : compteur
Fin Afficher
```

✍ **Algorithme de la procédure Afficher**

```
0) DEF PROC AFFICHER(N :ENTIER ;T:TAB)
1) Pour i de 1 à N faire
      Ecrire("T[",i, "]=",T[i])
   Fin Pour
2) Fin Afficher
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur

✍ **Analyse de la procédure Trier**

```
DEF PROC TRIER(N:ENTIER;A:TAB;VAR B:TAB)
Résultat : T
T2 =[]Pour i de 1 à N faire
       posmax←FN Max(N,T1)
     T2[i]←T1[posmax]
     T1[posmax]←-1
     Fin Pour
i : compteur
Fin Trier
```

✍ **Algorithme de la procédure Trier**

```
0) DEF PROC TRIER(N:ENTIER;A:TAB;VAR
   B:TAB)
1) Pour i de 1 à N faire
   posmax←FN Max(N,T1)
   T2[i]←T1[posmax]
   T1[posmax]←-1
  Fin Pour
2) Fin Trier
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur
Max	Fonction	Déterminer la position de maximum dans un tableau
posmax	Entier	Contient la position de maximum dans un tableau

✍ **Analyse de la Fonction Max**

```
DEF FN Max(n:ENTIER ;T:TAB):ENTIER
Résultat : Max ← posmax
posmax =[posmax←1]
Pour i de 2 à n faire
Si T[i]≠-1 alors
    Si (T[i]>T[posmax]) ou
       (T[posmax]=-1)  alors
          posmax←i
    Fin Si
Fin Si
Fin pour
i:compteur
Fin Max
```

✍ **Algorithme de la Fonction Max**

```
0) DEF FN Max(n:ENTIER ;T:TAB): ENTIER
1) [posmax←1]
Pour i de 2 à n faire
Si T[i]≠-1 alors
    Si (T[i]>T[posmax]) ou
       (T[posmax]=-1)  alors
          posmax←i
     Fin Si
Fin Si
Fin pour
2) Max ← posmax
3) Fin Max
```

T.D.O Locaux

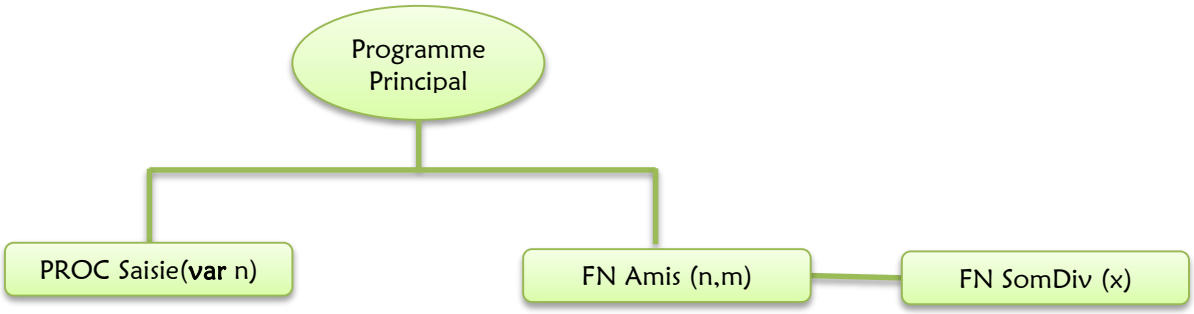
Objet	T/N	Rôle
i	Entier	Compteur
posmax	entier	Déterminer la position du grand caractère dans un tableau

✍ **Traduction Pascal**

```
PROGRAM Tri;
USES wincrt ;
TYPE TAB=Array[1..25] of integer ;
VAR N:integer ;
    T1,T2 :TAB ;
PROCEDURE Saisie(VAR x :integer) ;
BEGIN
 repeat
    Write('Donner la valeur de N: ');
    Readln(x) ;
 until(x>5)and(x<25);
END;
PROCEDURE Remplir(n:integer;VAR T:TAB);
VAR   i :integer ;
BEGIN
  for i :=1 to n do
   repeat
     Write('T[',i, ']= ') ;
     Readln(T[i]) ;
   until T[i]>=0;
END;
PROCEDURE Afficher(n:integer ;T :TAB) ;
VAR  i :integer ;
BEGIN
for i :=1 to n do  Write(T[i],' ') ;
END;
```

```
PROCEDURE Trier(n:integer ;A:TAB; VAR
B:TAB) ;
VAR posmax,i :integer ;
FUNCTION Max(n:integer;T:TAB):integer;
VAR   posmax,i :integer ;
BEGIN
posmax:=1;
      for i := 2 to n do
      if T[i]<>-1 then
          if (T[i]<T[posmax]) or
             (T[posmax]=-1)  then
                 Posmax :=i ;
Max := posmax;
END;
BEGIN
      for i :=1 to n do
      begin
      Posmax := Max(N,A) ;
      B[i]:=A[posmax];
      A[posmax]:=-1;
      end;
END;
BEGIN
 Saisie(n) ; Remplir(n,A);
 Trier(n,A,B) ;
 Afficher(n,B) ;
END.
```

✍ **Décomposition modulaire du problème**



✍ **Analyse de programme principal :**

Nom de programme : Nombre_Amis
Résultat : trait
Trait=[]**Si** FN Amis(n,m) **alors**
Ecrire(n," et ",m, " sont amis")
Sinon
Ecrire(n," et ",m," ne sont pas amis")
Fin Si
PROC Saisie(m)
PROC Saisie(n)
Fin Nombre_Amis

✍ **Algorithme de programme principal:**

0) *Début* Nombre_Amis
1) PROC Saisie(n)
2) PROC Saisie(m)
3) **Si** FN Amis(n,m) **alors**
Ecrire(n," et ",m, " sont amis")
Sinon
Ecrire(n," et ",m," ne sont pas amis")
Fin Si
4) *Fin* Nombre_Amis

T.D.O Globaux

Objet	T/N	Rôle
N	Entier	Stocker la valeur de N
M	Entier	Stocker la valeur de M
Saisie	Procédure	Saisir un entier naturel
Amis	Fonction	Vérifier si n et m sont amis ou non

✍ **Analyse de la procédure Saisie**

DEF PROC SAISIE(VAR X :ENTIER)
Résultat : x
x=[]
Répéter
x=**donnée**("Donner la valeur de N : ")
jusqu'à X≥0
Fin Saisie

✍ **Algorithme de la procédure Saisie**

0) DEF PROC SAISIE(VAR X :ENTIER)
1) Répéter
écrire("Donner un entier : ")
lire(x)
jusqu'à X≥0
2) Fin Saisie

Analyse de la fonction Amis

```
DEF FN AMIS(X,Y :ENTIER) :BOOLEEN
Résultat : amis←(s1=y)et(s2=x)
S1←FN SomDiv(x)
S2←FN SomDiv(y)
Fin Amis
```

Algorithme de la fonction Amis

```
0) DEF FN AMIS(X,Y :ENTIER) :BOOLEEN
1) S1←FN SomDiv(x)
2) S2←FN SomDiv(y)
3) Amis←(s1=y)et(s2=x)
4) Fin Amis
```

T.D.O Locaux

Objet	T/N	Rôle
S1	Entier	Contient la somme des diviseurs de x sauf lui même
S2	Entier	Contient la somme des diviseurs de y sauf lui même
SomDiv	Fonction	Calculer la somme des diviseurs d'un nombre

Analyse de la Fonction SomDiv

```
DEF FN SomDiv(X:ENTIER) : ENTIER
Résultat : SomDiv←S
Trait=[]
S=[s←0]Pour i de 1 à x div 2 faire
  Si(x mod i = 0) alors
    S←S+i
  Fin Si
Fin Pour
i : compteur
Fin SomDiv
```

Algorithme de la Fonction SomDiv

```
0) DEF FN SomDiv(X:ENTIER) : ENTIER
1) [s←0]
2) Pour i de 1 à x div 2 faire
     Si(x mod i = 0) alors
       S←S+i
     Fin Si
   Fin Pour
3) SomDiv←S
4) Fin SomDiv
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur
S	Entier	Calculer la somme des diviseurs de x

Traduction Pascal

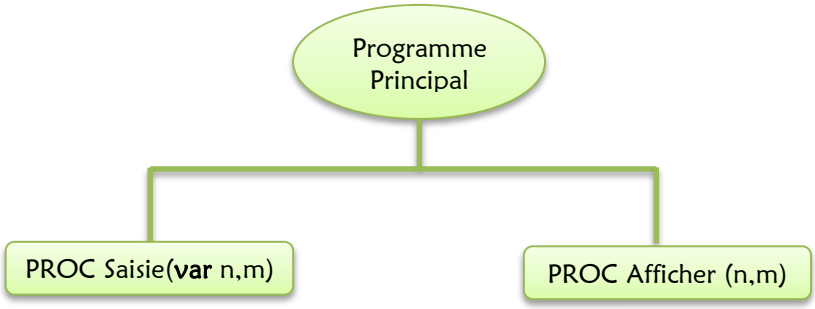
```
PROGRAM Nombre_Amis;
USES wincrt ;
VAR
   n,m :integer ;
PROCEDURE Saisie(VAR x :integer) ;
BEGIN
     repeat
          Write('Donner un entier: ');
          Readln(x) ;
     until(x>=0);
END;
FUNCTION SomDiv(x :integer) :integer ;
VAR  i,S :integer ;
BEGIN
     S :=0 ;
     for i :=1 to x div 2 do
FUNCTION Amis(x,y:integer):boolean;
VAR  S1,s2 :integer ;
BEGIN
     S1:=SomDiv(x);
     S2:=SomDiv(y);
     Amis:=(s1=y)and(s2=x);
END;
BEGIN
Saisie(n) ;
Saisie(m) ;
if Amis(n,m) then
   write(n,' et ',m, ' sont amis')
else
   write (n,' et ',m,' ne sont pas
amis');
```

```
      If(x mod i = 0) then
            S :=S+i ;
      SomDiv := S ;
END ;
```

```
END.
```

Exercice 18

✍ **Décomposition modulaire du problème**



✍ **Analyse de programme principal :**

Nom de programme : Ex18
Résultat : PROC Afficher(n,m)
PROC Saisie(n,m)
Fin Ex18

✍ **Algorithme de programme principal:**

0) *Début* Ex18
1) PROC Saisie(n,m)
2) PROC Afficher(n,m)
3) *Fin* Ex18

T.D.O Globaux

Objet	T/N	Rôle
N	Entier	Stocker la valeur de N
M	Entier	Stocker la valeur de M
Saisie	Procédure	Saisir deux entiers n et m
Afficher	Procédure	afficher tous les entiers de l'intervalle [1, m] en remplaçant par le caractère "*" tous les diviseurs de n ainsi que tous les entiers comportant dans leurs écritures le chiffre des unités de n

✍ **Analyse de la procédure Saisie**

```
DEF PROC SAISIE(VAR X,Y :ENTIER)
Résultat : x,Y
x=[]
Répéter
 x=donnée("Donner la valeur de N :  ")
 y=donnée("Donner la valeur de M :  ")
jusqu'à (X dans [100..500]) et
        (Y dans [10..99])
Fin Saisie
```

✍ **Algorithme de la procédure Saisie**

```
0) DEF PROC SAISIE(VAR X,Y :ENTIER)
1) Répéter
     écrire("Donner la valeur de N :  ")
     lire(x)
     écrire("Donner la valeur de M :  ")
     lire(y)
jusqu'à (X dans [100..500]) et
      (Y dans [10..99])
2) Fin Saisie
```

✍ **Analyse de la procédure Afficher**

```
DEF PROC AFFICHER(N,M :ENTIER)
Résultat : Trait
Trait=[]pour i de 1 à m faire
Convch(i,ch2)
Si (n mod i =0)ou(pos(ch1,ch2)≠0)alors
 Ecrire("*")
Sinon
 Ecrire(i)
Fin Si
Fin pour
Convch(m mod 10,ch1)
Fin Afficher
```

✍ **Algorithme de la procédure Afficher**

```
0) DEF PROC AFFICHER(N,M :ENTIER)
1) Convch(m mod 10,ch1)
2) pour i de 1 à m faire
   Convch(i,ch2)
   Si (n mod i=0) ou (pos(ch1,ch2)≠0)
   alors Ecrire("*")
   Sinon Ecrire(i)
   Fin Si
   Fin pour
3) Fin Afficher
```

T.D.O Locaux

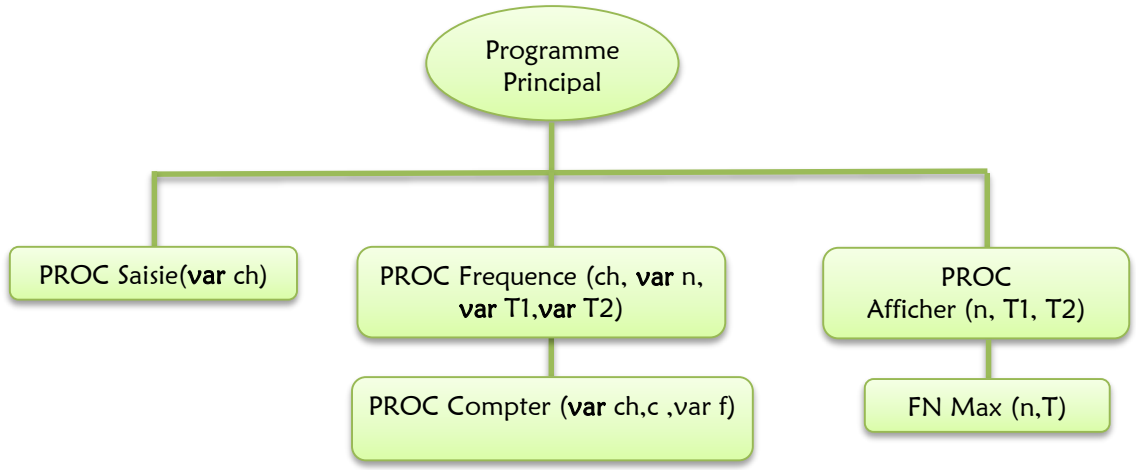
Objet	T/N	Rôle
I	Entier	compteur
Ch1,ch2	Chaine de caractères	Chaines auxiliaires

✍ **Traduction Pascal**

```
PROGRAM ex18;
USES wincrt ;
VAR
   n,m :integer ;
PROCEDURE Saisie(VAR x,y :integer) ;
BEGIN
 repeat
      Write('Donner la valeur de N: ');
      Readln(x) ;
      Write('Donner la valeur de M: ');
      Readln(y) ;
until(x>=100)and(x<=500)and(y in [10..99]);
END;
PROCEDURE Afficher(n,m :integer);
VAR  i:integer ;
     Ch1,ch2 :string ;
BEGIN
str(m mod 10,ch1);
for i :=1 to m do
begin
   str(i,ch2);
   if (n mod i=0) or (pos(ch1,ch2)<>0) then
   write('*')
   else write(i);
end;
END ;
```

```
BEGIN
      Saisie(n,m) ;
      Afficher(n,m) ;
END.
```

✍ **Décomposition modulaire du problème**



✍ **Analyse de programme principal :**

Nom de programme : Fréquence_Lettre
Résultat: PROC Afficher(n,T1,T2)
PROC Frequence (ch,n,T1,T2)
PROC Saisie(ch)
Fin Fréquence_Lettre

✍ **Algorithme de programme principal:**

0) *Début* Fréquence_Lettre
1) PROC Saisie(ch)
2) PROC Frequence (ch,n,T1,T2)
3) PROC Afficher(n,T1,T2)
4) *Fin* Fréquence_Lettre

T.D.N.T

Type
TChar= Tableau de taille 20 et de type caractère
TInt= Tableau de taille 20 et de type entier

T.D.O Globaux

Objet	T/N	Rôle
Ch	Chaine de caractère	stocker une chaine de caractère
T1	TChar	Stocker les lettres utilisées dans la phrase
T2	TInt	Stocker les fréquences des lettres utilisées dans la phrase
Saisie	Procédure	Saisir une chaine de caractère
Frequence	Procédure	Remplir deux tableaux respectivement par des lettres et leurs fréquences dans la phrase.
Afficher	Procédure	Afficher la lettre alphabétique la plus utilisée dans un texte

✍ **Analyse de la procédure Saisie**

DEF PROC SAISIE(VAR CH:chaine de caractères)
Résultat : ch
ch=[]*Répéter*
ch=**donnée**("Donner une chaine : ")
jusqu'à long(ch) dans [5..20]
Fin Saisie

✍ **Algorithme de la procédure Saisie**

0) DEF PROC SAISIE(VAR ch:chaine de caractères)
1) Répéter
écrire("Donner une chaine : ")
lire(ch)
jusqu'à long(ch) dans [5..20]
2) Fin Saisie

✍ Analyse de la procédure Fréquence

```
DEF PROC FREQUENCE (CH:chaine de caractères;var n:entier;var T1:TChar;var
T2:TInt)
Résultat : T1,T2,n
n←i-1
(T1,T2) =[i←1,ch1←ch]
répéter
          T1[i]←Ch1[i]
          PROC Compter(ch1,ch1[i],f)
          T2[i]←f
          i←i+1
jusqu'à ch1=""

Fin Fréquence
```

✍ Algorithme de la procédure Fréquence

```
0) DEF PROC FREQUENCE (CH:chaine de
     caractères; var n:entier; var
     T1:TChar;var T2:TInt)
1) [i←1,ch1←ch]répéter
     T1[i]←Ch1[i]
     PROC Compter(ch1,ch1[i],f)
     T2[i]←f
     i←i+1
jusqu'à ch1=""
2) N ←i-1
3) Fin Fréquence
```

T.D.O Locaux

Objet	T/N	Rôle
I	Entier	Compteur
F	Entier	Contient la fréquence d'un caractère dans une chaine
Compter	Procédure	Compter la fréquence d'un caractère dans une chaine

✍ Analyse de la procédure Compter

```
DEF PROC Compter (var CH:chaine de caractères; c:caractère; var f:entier)
Résultat : ch,f
[f←0]
Tant que Pos(c,ch)≠0 faire
     f←f+1
     Efface(ch,Pos(c,ch),1)
Fin Tant que

Fin Compter
```

✍ Algorithme de la procédure Compter

```
0) DEF PROC Compter (var CH:chaine de
     caractères; c:entier;var f :entier)
1) [f←0]
Tant que Pos(c,ch)≠0 faire
     f←f+1
     Efface(ch,Pos(c,ch),1)
Fin Tant que
2) Fin Compter
```

✍ Analyse de la procédure Afficher

```
DEF PROC AFFICHER(N :ENTIER ;T1 :TCHAR ;T2 :TINT)
Résultat : Trait
Trait=[]Pour i de 1 à N faire
      Si (T2[i]=T2[posmax]) alors
Ecrire(T1[i]," f=",T2[i])
Fin si
       Fin Pour
Posmax←FN MAX(n,T2)
i : compteur
Fin Afficher
```

✍ Algorithme de la procédure Afficher

```
0) DEF PROC AFFICHER(N :ENTIER ;
   T1 :TCHAR ;T2 :TINT)
1) Posmax←FN MAX(n,T2)
2) Pour i de 1 à N faire
   Si (T2[i]=T2[posmax]) alors
   Ecrire(T1[i]," f=",T2[i])
   Fin si
    Fin Pour
3) Fin Afficher
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur
posmax	Entier	Contient la position du la fréquence la plus grande

✍ **Analyse de la Fonction Max**

```
DEF FN Max(n:ENTIER ;T:TINT):ENTIER
Résultat : Mqx ← posmax
posmax =[posmax←1]
Pour i de 2 à n faire
    Si T[i]>T[posmax] alors
        posmax←i
    Fin Si
Fin pour
i:compteur
Fin Max
```

✍ **Algorithme de la Fonction Max**

```
0) DEF FN Max(n:ENTIER;T:TINT) :ENTIER
1) [posmax←1]
Pour i de 2 à n faire
    Si T[i]>T[posmax] alors
        posmax←i
    Fin Si
Fin pour
2) Max ← posmax
3) Fin Max
```

T.D.O Locaux

Objet	T/N	Rôle
I	Entier	Compteur
posmax	entier	Déterminer la position du la fréquence la plus grande

✍ **Traduction Pascal**

```
PROGRAM Frequence_Lettre;
USES wincrt ;
TYPE Tint= array[1..20] of integer ;
     TChar=array[1..20] of char ;
VAR Ch :string ;
    T1 :TChar ;T2 :Tint ;
    N :integer ;
PROCEDURE Saisie(VAR ch :string) ;
BEGIN
 repeat
    Write('Donner une chaine: ');
    Readln(ch) ;
 until length(ch) in [5..20] ;
END;
PROCEDURE Afficher(n:integer;T1:TChar
;T2:TInt);
VAR    i,posmax :integer ;
FUNCTION Max(n:integer;T:TInt):integer;
VAR   posmax,i :integer ;
BEGIN
      posmax:=1;
      for i := 2 to n do
       if T[i]<T[posmax]then
            Posmax :=i ;
Max := posmax;
```

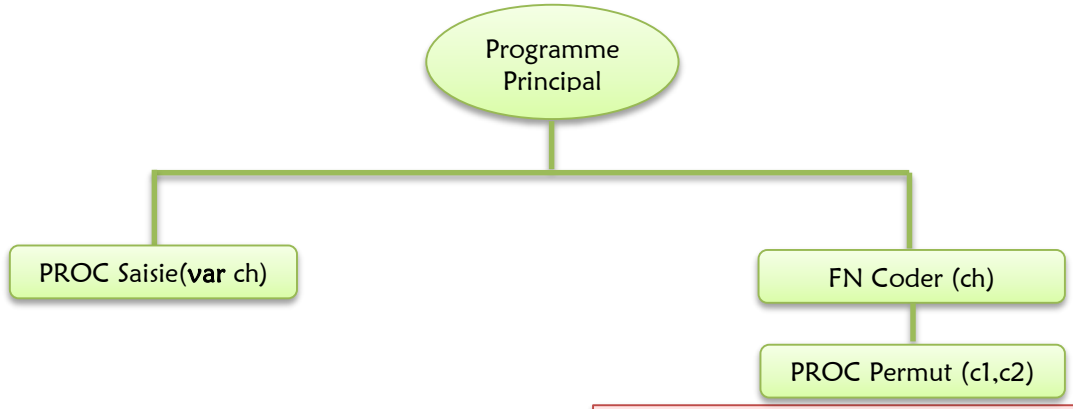
```
PROCEDURE Compter(var ch:string; c:char;
var f :integer);
BEGIN
     f:=0;
     while Pos(c,ch)<>0 do
     begin
          f:=f+1;
          delete(ch,Pos(c,ch),1);
     End;
END;
PROCEDURE Frequence(ch:string ; var
n :integer ;var T1 :TChar ;var T2 :Tint);
VAR  i,f :integer;
BEGIN
i:=1;ch1:=ch;
Repeat
     T1[i] :=Ch[i] ;
     Compter(ch1,ch[i],f) ;
     T2[i] :=f ;
     i:=i+1;
until ch1 = '';
n:=i-1;
END;
```

```
END;
BEGIN
Posmax:= MAX(n,T2);
for i:=1 to n do
   if (T2[i]=T2[posmax]) then
   writeln(T1[i],' f=',T2[i])
END ;
```

```
BEGIN
      Saisie(ch);
      Frequence (ch,n,T1,T2);
      Afficher(n,T1,T2);
END.
```

Exercice 20

✍ **Décomposition modulaire du problème**



✍ **Analyse de programme principal :**

Nom de programme : Codage
Résultat: écrire("La chaine après codage :", FN Coder(ch))
PROC Saisie(ch)
Fin Codage

✍ **Algorithme de programme principal:**

0) *Début* Codage
1) PROC Saisie(ch)
2) écrire("La chaine après codage :", FN Coder(ch))
3) *Fin* Codage

T.D.O Globaux

Objet	T/N	Rôle
ch	Chaine de caractère	stocker une chaine de caractère
Saisie	Procédure	Saisir une chaine de caractère
Coder	Fonction	permuter chaque caractère d'indice pair avec le caractère qui le précède

✍ **Analyse de la procédure Saisie**

DEF PROC SAISIE(VAR CH:chaine de caractères)
Résultat : ch
ch=[]*Répéter*
 ch=**donnée**("Donner une chaine : ")
jusqu'à long(ch) > 1
Fin Saisie

✍ **Algorithme de la procédure Saisie**

0) DEF PROC SAISIE(VAR ch:chaine de caractères)
1) Répéter
   écrire("Donner une chaine : ")
   lire(ch)
 jusqu'à long(ch) > 1
2) Fin Saisie

Analyse de la fonction Coder

```
DEF FN Coder (CH:chaine de caractères) : chaine de caractères
Résultat : Coder←ch
For i de 1 à long(ch) faire
     Si i mod 2 = 0 alors
       PROC Permut(ch1[i-1],ch1[i])
     Fin Si
Fin pour
i :Compteur
Fin Coder
```

Algorithme de la Fonction Coder

```
0) DEF FN Coder(CH: chaine de
     caractères) : chaine de caractères
1) For i de 1 à long(ch) faire
     Si i mod 2 = 0 alors
       PROC Permut(ch1[i-1],ch1[i])
     Fin Si
Fin pour
2) Coder←ch
3) Fin Coder
```

T.D.O Locaux

Objet	T/N	Rôle
I	Entier	Compteur
Permut	Procédure	permuter deux caractères d'une chaine

Analyse de la procédure Permut

```
DEF PROC PERMUT(VAR X,Y :CARACTERE)
Résultat : x,y
          aux←x
          x←y
          y←aux
Fin Permut
```

Algorithme de la procédure Permut

```
0) DEF PROC PERMUT(VAR X,Y : CARACTERE)
1) Aux←x
2) X←y
3) Y←aux
4) Fin Permut
```

T.D.O Locaux

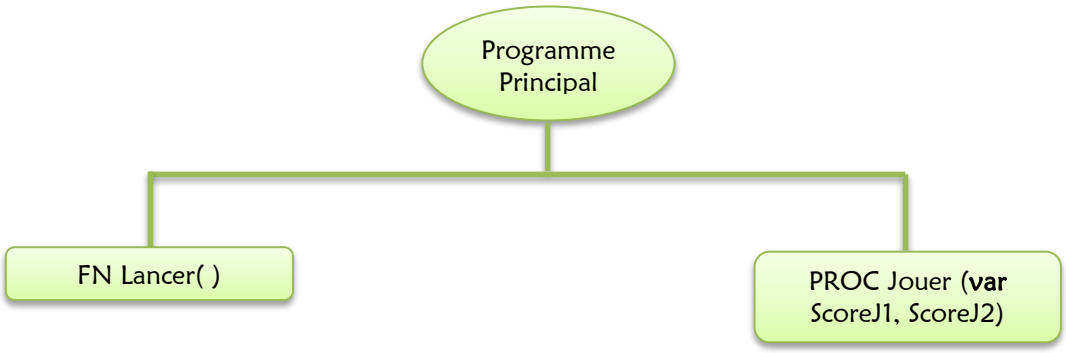
Objet	T/N	Rôle
aux	Entier	Variable auxiliaire

Traduction Pascal

```
PROGRAM Codage;
USES wincrt ;
VAR Ch :string ;
Function Coder(ch :string) :string;
VAR    i:integer ;
      PROCEDURE Permut(VAR x,y:char);
      VAR  aux :char ;
      BEGIN
            aux :=x ;
            x :=y ;
            y :=aux ;
      END;
BEGIN
for i:=1 to length(ch) do
   if i mod 2 = 0 then
      Permut(ch1[i-1],ch1[i]) ;
Coder:=ch;
END ;
```

```
PROCEDURE Saisie(VAR ch :string) ;
BEGIN
 repeat
    Write('Donner une chaine: ');
    Readln(ch) ;
 until length(ch) > 1 ;
END;
BEGIN
     Saisie(ch);
     writeln('La chaine après codage :',
     Coder(ch))
END.
```

✍ **Décomposition modulaire du problème**



✍ **Analyse de programme principal :**

```
Nom de programme : Jeu_Dé
Résultat: Trait
Trait=[] Si ScoreJ1=10 alors
écrire("Le joueur 1 gagne la partie")
Sinon
écrire("Le joueur 2 gagne la partie")
Fin Si
Ecrire("Joueur1=",ScoreJ1," Joueur2=",
ScoreJ2)
PROC Jouer(ScoreJ1, ScoreJ2)
Fin Jeu_Dé
```

✍ **Algorithme de programme principal:**

```
0) Début Jeu_Dé
1) PROC Jouer(ScoreJ1, ScoreJ2)
2) Ecrire("Joueur1=",ScoreJ1,"
   Joueur2=",ScoreJ2)
3) Si ScoreJ1=10 alors
écrire("Le joueur 1 gagne la
partie")
 Sinon
écrire("Le joueur 2 gagne la
partie")
Fin Si
4) Fin Jeu_Dé
```

T.D.O Globaux

Objet	T/N	Rôle
ScoreJ1	Entier	Stocker le score de joueur 1
ScoreJ2	Entier	Stocker le score de joueur 2
Jouer	Procédure	Contrôler le déroulement de jeu de dé

✍ **Analyse de la procédure Jouer**

```
DEF PROC JOUER(VAR scj1,SCJ2 :ENTIER)
Résultat : ScJ1,ScJ2
[ScJ1←0, ScJ2←0]
Répéter
 N1 ←FN Lancer
 N2 ←FN Lancer
Ecrire("Joueur 1 obtient:",N1,
 " Joueur 2 obtient:",N2)
Si N1>N2 alors ScJ1←ScJ1+1
Sinon ScJ2←ScJ2+1
Fin Si
jusqu'à (ScJ1 = 10) ou (ScJ2 = 10)
Fin Jouer
```

✍ **Algorithme de la procédure Saisie**

```
0) DEF PROC JOUER(VAR Scj1,SCJ2 :ENTIER)
1) [ScJ1←0, ScJ2←0]
Répéter
      N1 ←FN Lancer
      N2 ←FN Lancer
     Ecrire("Joueur 1 obtient:",N1,
      " Joueur 2 obtient:",N2)
     Si N1>N2 alors ScJ1←ScJ1+1 Fin Si
     Si N1<N2 alors ScJ2←ScJ2+1 Fin Si
jusqu'à (ScJ1 = 10) ou (ScJ2 = 10)
2) Fin Jouer
```

T.D.O Locaux

Objet	T/N	Rôle
N1	Entier	Stocker la valeur de la face obtenue par le joueur 1
N2	Entier	Stocker la valeur de la face obtenue par le joueur 2
Lancer	Fonction	Générer un nombre aléatoire entre 1 et 6

✍ **Analyse de la fonction Lancer**

```
DEF FN Lancer: entier
Résultat : Lancer ← Random(6)+1

Fin Lancer
```

✍ **Algorithme de la Fonction Lancer**

```
0) DEF FN Lancer: entier
1) Lancer←Random(6)+1
2) Fin Lancer
```

✍ **Traduction Pascal**

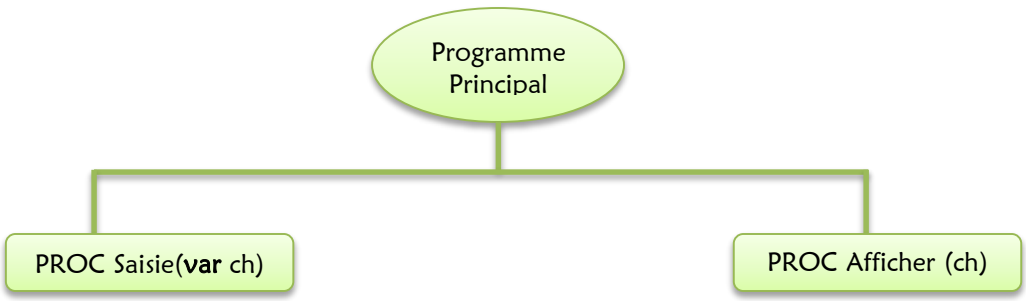
```
PROGRAM Jeu_De;
USES wincrt ;
VAR ScoreJ1,ScoreJ2 :integer ;
PROCEDURE Jouer(ScJ1, ScJ2:integer);
VAR     N1,N2:integer ;
      FUNCTION Lancer :integer ;
      BEGIN
          Lancer :=Random(6)+1 ;
      END;
BEGIN
ScJ1:=0; ScJ2:=0;
Randomize;
repeat
    N1 :=Lancer ;
    N2 :=Lancer ;
writeln('Joueur 1 obtient:',N1,
 ' Joueur 2 obtient:',N2) ;
if N1>N2 then ScJ1 :=ScJ1+1
else ScJ2 :=ScJ2+1 ;
until (ScJ1 = 10) or (ScJ2 = 10);
END ;
```

```
BEGIN
Jouer(ScoreJ1, ScoreJ2);
writeln('Joueur1=',ScoreJ1, '
Joueur2=',ScoreJ2);
if ScoreJ1=10 alors
writeln ('Le joueur 1 gagne la partie')
else
writeln ('Le joueur 2 gagne la partie');
END.
```

Random(6) permet de générer un nombre aléatoire compris entre 0 et 5

Exercice 22

✍ **Décomposition modulaire du problème**



✍ **Analyse de programme principal :**

Nom de programme : chaine
Résultat: PROC Afficher(chm)
PROC Saisie(chm)
Fin chaine

✍ **Algorithme de programme principal:**

0) *Début* chaine
1) PROC Saisie(chm)
2) PROC Afficher(chm)
3) *Fin* chaine

T.D.O Globaux

Objet	T/N	Rôle
chm	Chaine de caractère	stocker une chaine de caractère
Saisie	Procédure	Saisir une chaine de caractère
Afficher	Procédure	Afficher une chaine

✍ **Analyse de la procédure Saisie**

DEF PROC SAISIE(VAR CH:chaine de caractères)
Résultat : ch
ch=[]*Répéter*
ch=**donnée**("Donner une chaine : ")
jusqu'à long(ch) ≥ 5
Fin Saisie

✍ **Algorithme de la procédure Saisie**

0) DEF PROC SAISIE(VAR ch:chaine de caractères)
1) Répéter
écrire("Donner une chaine : ")
lire(ch)
jusqu'à long(ch) ≥ 5
2) Fin Saisie

✍ **Analyse de la procédure Afficher**

DEF PROC AFFICHER(CH :chaine de caractères)
Résultat : Trait
Trait=[]**Pour i de 1 à long(ch) faire**
Ecrire(sous-chaine(ch,1,i)+
sous-chaine(ch,long(ch)-i+1,i))
Fin Pour
i : compteur
Fin Afficher

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur

✍ **Algorithme de la procédure Afficher**

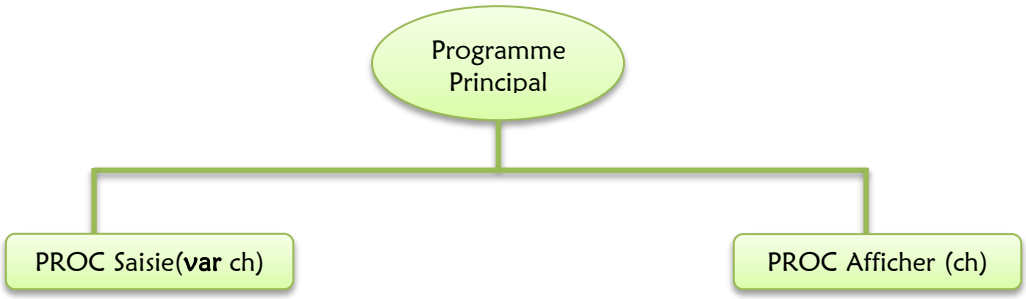
0) DEF PROC AFFICHER(CH: chaine de caractères)
1) Pour i de 1 à long(ch) faire
Ecrire(sous-chaine(ch,1,i)+
sous-chaine(ch,long(ch)-i+1,i))
Fin Pour
2) Fin Afficher

✍ **Traduction Pascal**

```
PROGRAM chaine ;
USES wincrt ;
VAR Chm :string ;
PROCEDURE Afficher(ch :string) ;
VAR    i :integer;
BEGIN
for i:=1 to length(ch) do
writeln(copy(ch,1,i)+copy(ch, length(ch)-
i+1, i) ) ;
END;
```

```
PROCEDURE Saisie(VAR ch :string) ;
BEGIN
 REPEAT
    Write('Donner une chaine: ');
    Readln(ch) ;
 UNTIL length(ch)>=5 ;
END;
BEGIN
    Saisie(chm) ;
    Afficher(chm) ;
END.
```

✍ **Décomposition modulaire du problème**



✍ **Analyse de programme principal :**

Nom de programme : chaine
Résultat: PROC Afficher(ch)
PROC Saisie(ch)
Fin chaine

✍ **Algorithme de programme principal:**

0) *Début* chaine
1) PROC Saisie(ch)
2) PROC Afficher(ch)
3) *Fin* chaine

T.D.O Globaux

Objet	T/N	Rôle
ch	Chaine de caractère	stocker une chaine de caractère
Saisie	Procédure	Saisir une chaine de caractère
Afficher	Procédure	Afficher une chaine sous forme d'un triangle

✍ **Analyse de la procédure Saisie**

DEF PROC SAISIE(VAR CH:chaine de caractères)
Résultat : ch
ch=[]*Répéter*
ch=**donnée**("Donner une chaine : ")
jusqu'à long(ch) ≥ 3
Fin Saisie

✍ **Algorithme de la procédure Saisie**

0) DEF PROC SAISIE(VAR ch:chaine de caractères)
1) Répéter
écrire("Donner une chaine : ")
lire(ch)
jusqu'à long(ch) ≥ 3
2) Fin Saisie

✍ **Analyse de la procédure Afficher**

DEF PROC AFFICHER(CH :chaine de caractères)
Résultat : Trait
Trait=[]**Pour i de 1 à long(ch) faire**
Ecrire(sous-chaine(ch,1,i))
Fin Pour
i : compteur
Fin Afficher

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur

✍ **Algorithme de la procédure Afficher**

0) DEF PROC AFFICHER(CH: chaine de caractères)
1) Pour i de 1 à long(ch) faire
Ecrire(sous-chaine(ch,1,i))
Fin Pour
2) Fin Afficher

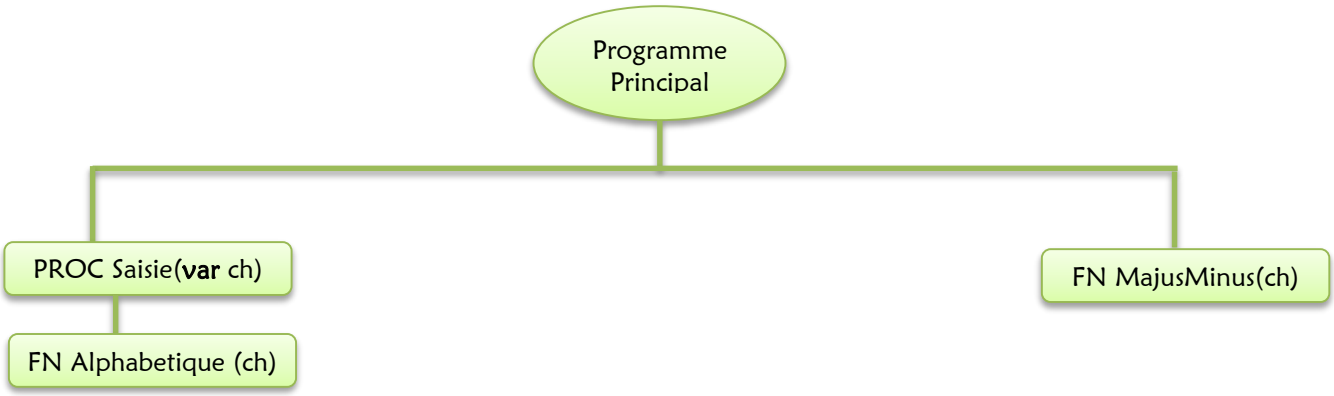
Traduction Pascal

```
PROGRAM chaine ;
USES wincrt ;
VAR Ch :string ;
PROCEDURE Afficher(ch :string) ;
VAR    i :integer;
BEGIN
for i:=1 to length(ch) do
   writeln(copy(ch,1,i)) ;
END;
```

```
PROCEDURE Saisie(VAR ch :string) ;
BEGIN
 REPEAT
    Write('Donner une chaine: ');
    Readln(ch) ;
 UNTIL length(ch)>=3 ;
END;
BEGIN
    Saisie(ch) ;
    Afficher(ch) ;
END.
```

Exercice 24

Décomposition modulaire du problème



Analyse de programme principal :

Nom de programme : chaine
Résultat:
écrire("Chaine résultat : ",Res)
Res← Fn MajusMinus(ch)
PROC Saisie(ch)
Fin chaine

Algorithme de programme principal:
0) *Début* chaine
1) PROC Saisie(ch)
2) **Res**← Fn MajusMinus(ch)
3) écrire("Chaine résultat : ",Res)
4) *Fin* chaine

T.D.O Globaux

Objet	T/N	Rôle
Ch	Chaine de caractère	stocker une chaine de caractère
Saisie	Procédure	Saisir une chaine de caractère
MajusMinus	Fonction	Réorganiser les lettres d'une chaine donnée

✍ **Analyse de la procédure Saisie**

```
DEF PROC SAISIE(VAR CH:chaine de caractères)
Résultat : ch
ch=[]Répéter
 ch=donnée("Donner une chaine :  ")
jusqu'à (long(ch) dans [1..50]) et
        (FN Alphabetique(ch))
Fin Saisie
```

✍ **Algorithme de la procédure Saisie**

```
0) DEF PROC SAISIE(VAR ch:chaine
   de caractères)
1) Répéter
   écrire("Donner une chaine : ")
   lire(ch)
  jusqu'à (long(ch) dans [1..50])
et (FN Alphabetique(ch))
2) Fin Saisie
```

T.D.O Locaux

Objet	T/N	Rôle
Alphabetique	Fonction	Tester si une chaine est composée seulement par des lettres alphabétiques ou non

✍ **Analyse de la fonction Alphabétique**

```
DEF FN ALPHABETIQUE(CH :chaine de caractères) :BOOLEEN
Résultat : Alphabétique←test
Test=[test←vrai,i←1]
Répéter
  Si Non(Majus(Ch[i])dans["A".."Z"])
    Alors
         Test← faux
  Fin Si
i←i+1
jusqu'à (test=faux) ou (i>long(ch))
Fin Alphabétique
```

T.D.O Locaux

Objet	T/N	Rôle
I	Entier	Compteur
test	booléen	Tester si la chaine est composée seulement par des lettres alphabétiques

✍ **Algorithme de la fonction Alphabétique**

```
0) DEF FN ALPHABETIQUE(CH :chaine de
   caractères) :BOOLEEN
1) Répéter
  Si Non(Majus(Ch[i])dans["A".."Z"])
     Alors
          Test← faux
  Fin Si
   i←i+1
   jusqu'à (test=faux) ou
   (i>long(ch))
2) Alphabétique←test
3) Fin Alphabétique
```

✍ **Analyse de la fonction MajusMinus**

```
DEF FN MAJUSMINUS(CH :chaine de caractères) : chaine de caractères
Résultat : MajusMinus← CMajus+ VMajus+ CMinus+ VMinus
[Vmajus←"", Vminus←"", Cmajus←"", Cminus←""]
pour i de 1 à long(ch) faire
Selon ch[i] faire
"A","E","I","O","U","Y" : VMajus← VMajus+ ch[i]
"a","e","i","o","u","y" : VMinus← VMinus+ ch[i]
"A".."Z" : CMajus← CMajus+ ch[i]
"a".."z" : CMinus← CMinus+ ch[i]
```

Fin Selon
Fin Pour
i :compteur
Fin MajusMinus

Algorithme de la fonction MajusMinus

0) DEF FN MAJUSMINUS (CH :chaine de caractères) : chaine
1) [Vmajus←"", Vminus←"", Cmajus←"", Cminus←""]
pour i de 1 à long(ch) faire
Selon ch[i] faire
"A","E","I","O","U","Y" : VMajus← VMajus+ ch[i]
"a","e","i","o","u","y" : VMinus← VMinus+ ch[i]
"A".."Z" : CMajus← CMajus+ ch[i]
"a".."z" : CMinus← CMinus+ ch[i]
Fin Selon
Fin Pour
2) **MajusMinus** ← CMajus+ VMajus+ CMinus+ VMinus
3) **Fin MajusMinus**

T.D.O Locaux

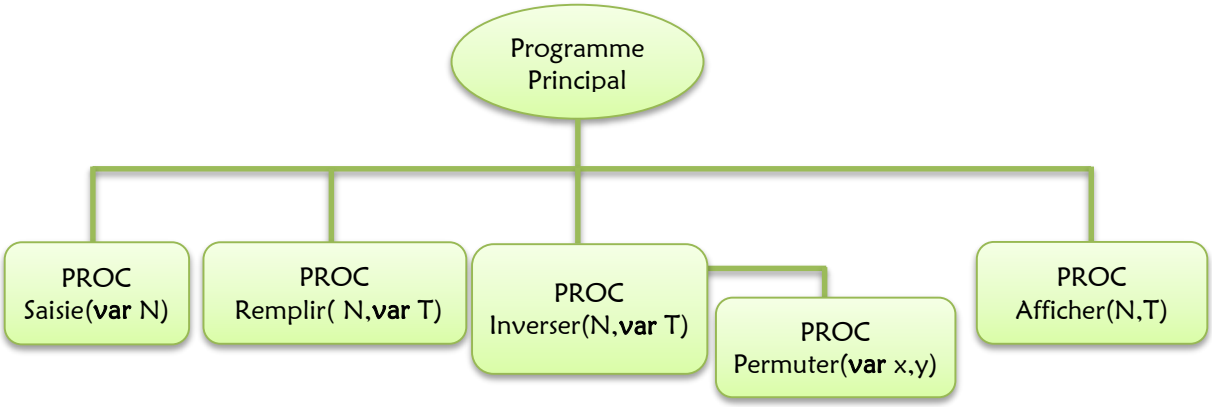
Objet	T/N	Rôle
i	Entier	Compteur
VMajus	chaine	Contient les lettres voyelles Majuscules
VMinus	chaine	Contient les lettres voyelles Minuscules
CMajus	chaine	Contient les lettres consonnes Majuscules
CMinus	chaine	Contient les lettres consonnes Minuscules

Traduction Pascal

```
PROGRAM chaine ;
USES wincrt ;
VAR Ch,res :string ;
PROCEDURE Saisie(VAR ch :string) ;
FUNCTION Alphabetique(ch :string):Boolean;
VAR
   i :integer ;    test :boolean ;
BEGIN
    Test :=true ; i :=1 ;
Repeat
     If NOT(Upcase(Ch[i])in['A'..'Z'])
then
         Test :=false ;
    i :=i+1 ;
Until (test=false)OR(i>length(ch)) ;
 Alphabetique :=test ;
END ;
BEGIN
 REPEAT
    Write('Donner une chaine: ');
    Readln(ch) ;
 UNTIL (length(ch)in [1..50]) AND
       (Alphabetique(ch)) ;
END;
```

```
FUNCTION MajusMinus(ch:string) :String;
VAR i :integer ;
Vmajus, Vminus, Cmajus, Cminus:string ;
BEGIN
   Vmajus:=''; Vminus:=''; Cmajus:='';
   Cminus:='';
   For i :=1 to length(ch) do
   Case ch[i] of
   'A', 'E', 'I', 'O', 'U', 'Y' :
   VMajus:= VMajus+ ch[i];
   'a', 'e', 'i', 'o', 'u', 'y' :
   VMinus:= VMinus+ ch[i];
   'A'.. 'Z' : CMajus:= CMajus+ ch[i];
   'a'.. 'z' : CMinus:= CMinus+ ch[i];
   END;
MajusMinus := CMajus+ VMajus+ CMinus+
VMinus;
END ;
BEGIN
    Saisie(ch) ;
    Res :=  MajusMinus(ch);
    write('Chaine résultat : ',Res)

END.
```

✍ Décomposition modulaire du problème



✍ Analyse de programme principal :

Nom de programme : Inverser_Tableau
Résultat : PROC Afficher(n,T)
PROC Inverser(n,T)
PROC Remplir(n,T)
PROC Saisie(n)
Fin Inverser_Tableau

✍ **Algorithme de programme principal:**

0) *Début* Inverser_Tableau
1) PROC Saisie(n)
2) PROC Remplir(n,T)
3) PROC Inverser(n,T)
4) PROC Afficher(n,T)
5) *Fin* Inverser_Tableau

T.D.N.T

Type
TAB= Tableau de taille 20 et de type caractère

T.D.O Globaux

Objet	T/N	Rôle
N	Entier	stocker la taille du tableau
T	TAB	Remplir le tableau par N caractères
Afficher	Procédure	Afficher un tableau
Remplir	Procédure	Remplir le tableau par n caractères.
Saisie	Procédure	Saisir la taille du tableau
Inverser	Procédure	Inverser les éléments d'un tableau T

✍ Analyse de la procédure Saisie

```
DEF PROC SAISIE(VAR N:entier)
Résultat : n
n=[]Répéter
 n=donnée("Donner la taille de tableau: ")
jusqu'à n dans [3..20]
Fin Saisie
```

✍ **Algorithme de la procédure Saisie**

```
0) DEF PROC SAISIE(VAR n :entier)
1) Répéter
     écrire("Donner la taille du
     tableau: ")
     lire(n)
   jusqu'à n dans [3..20]
2) Fin Saisie
```

✍ Analyse de la procédure Remplir

DEF PROC REMPLIR(N :ENTIER ;*VAR* T :TAB)

Résultat : T
T=[]**Pour i de 1 à N faire**
T[i]=**donnée**("T[",i, "]=")
Fin Pour
i : compteur
Fin Remplir

T.D.O Locaux

Objet	T/N	Rôle
I	Entier	Compteur

✍ Algorithme de la procédure Remplir

0) DEF PROC REMPLIR(N :ENTIER ;*VAR* T: TAB)
1) Pour i de 1 à N faire
écrire("T[",i, "]=")
lire(T[i])
Fin Pour
2) Fin Remplir

✍ Analyse de la procédure Afficher

DEF PROC AFFICHER(N :ENTIER ;T :TAB)

Résultat : Trait
Trait=[]**Pour i de 1 à N faire**
Ecrire("T[",i, "]=",T[i])
Fin Pour
i : compteur
Fin Afficher

T.D.O Locaux

Objet	T/N	Rôle
I	Entier	Compteur

✍ Algorithme de la procédure Afficher

0) DEF PROC AFFICHER(N :ENTIER ;T:TAB)
1) Pour i de 1 à N faire
Ecrire("T[",i, "]=",T[i])
Fin Pour
2) Fin Afficher

✍ Analyse de la procédure Inverser

DEF PROC INVERSER(N:ENTIER; *VAR* T:TAB)

Résultat : T
Pour i de 1 à N div 2 faire
PROC Permuter(T[i],T[n-i+1])
Fin Pour
i : compteur
Fin Inverser

T.D.O Locaux

✍ Algorithme de la procédure Inverser

0) DEF PROC INVERSER(N:ENTIER; *VAR* T:TAB)
1) Pour i de 1 à N div 2 faire
PROC Permuter(T[i],T[n-i+1])
Fin Pour
2) Fin Inverser

Objet	T/N	Rôle
I	Entier	Compteur
Permuter	Procédure	Permuter deux caractères

✍ Analyse de la procédure Permuter

DEF PROC Permuter(*VAR* x,y :caractère)

Résultat : x,y
Aux←x
X←y
Y←aux
Fin Permuter

T.D.O Locaux

✍ Algorithme de la procédure Permuter

0) DEF PROC Permuter(*VAR* x,y :caractère)
1) Aux←x
2) X←y
3) Y←aux
4) Fin Permuter

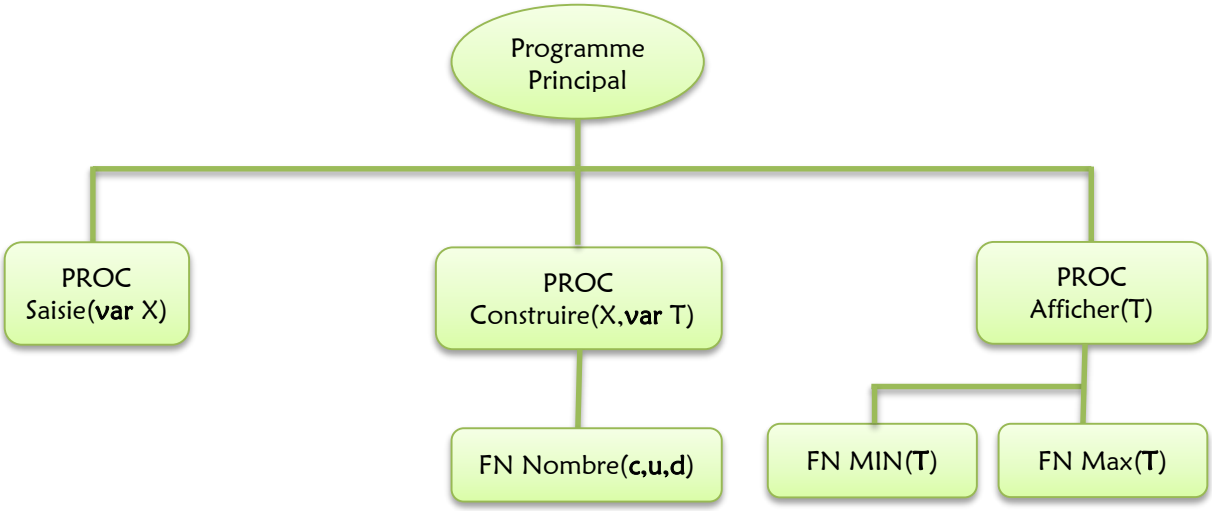
Objet	T/N	Rôle
Aux	caractère	Variable auxiliaire

✍ **Traduction Pascal**

```
PROGRAM Inverser_tableau;
USES wincrt ;
TYPE TAB= array[1..20] of char ;
VAR n :integer ;
    T :TAB ;
PROCEDURE Saisie(VAR n :integer) ;
BEGIN
 Repeat
Write('Donner la taille du tableau:');
Readln(n) ;
 until n in [3..20] ;
END;
PROCEDURE Remplir(n:integer;VAR T:TAB);
VAR   i :integer ;
BEGIN
  for i :=1 to n do
  begin
     Write('T[',i, ']= ') ;
     Readln(T[i] ) ;
  end ;
END;
PROCEDURE Afficher(n:integer;T:TAB) ;
VAR  i :integer ;
BEGIN
for i :=1 to n do
 Write(T[i],' ') ;
END;
```

```
PROCEDURE Inverser(n:integer;var T:TAB);
VAR
    i:integer ;
  PROCEDURE Permuter(var x,y :char);
  VAR
      aux :char ;
  BEGIN
      Aux :=x ;
      x :=y ;
      y :=aux ;
  END ;
BEGIN
 for i :=1 to n div 2 do
      Permuter(T[i], T[n-i+1]) ;
END;
BEGIN
      Saisie(n);
      Remplir(n,T);
      Inverser(n,T);
      Afficher(n,T) ;
END.
```

✍ **Décomposition modulaire du problème**



✍ **Analyse de programme principal :**

Nom de programme : ConstruireNombres
Résultat : PROC Afficher(T)
PROC Construire(X,T)
PROC Saisie(X)
Fin ConstruireNombres

✍ **Algorithme de programme principal:**

```
0) Début ConstruireNombres
1) PROC Saisie(x)
2) PROC Construire(x,T)
3) PROC Afficher(T)
4) Fin ConstruireNombres
```

T.D.N.T

Type
TAB= Tableau de taille 6 et de type entiers

T.D.O Globaux

Objet	T/N	Rôle
X	Entier	stocker un entier de 3 chiffres
T	TAB	Remplir le tableau par 6
Afficher	Procédure	Afficher un tableau
Construire	Procédure	Remplir le tableau par 6 entiers de 3 chiffres qui compose le nombre X
Saisie	Procédure	Saisir le nombre X

✍ **Analyse de la procédure Saisie**

DEF PROC SAISIE(VAR X:entier)
Résultat : x
x=[]*Répéter*
x=**donnée**("Donner un entier: ")
jusqu'à (x ≥ 100) et (x≤999)
Fin Saisie

✍ **Algorithme de la procédure Saisie**

```
0) DEF PROC SAISIE(VAR x :entier)
1) Répéter
     écrire("Donner entier: ")
     lire(x)
   jusqu'à (x ≥ 100) et (x≤999)
2) Fin Saisie
```

✍ **Analyse de la procédure Construire**

```
DEF PROC CONSTRUIRE(X :ENTIER ;VAR T :TAB)
Résultat : T
T[1]← FN Nombre(c,u,d)
T[2]← FN Nombre(c,d,u)
T[3]← FN Nombre(d,u,c)
T[4]← FN Nombre(d,c,u)
T[5]← FN Nombre(u,c,d)
T[6]← FN Nombre(u,d,c)
C←X div 100
D←(X mod 100) div 10
U←X mod 10
Fin Construire
```

✍ **Algorithme de la procédure Construire**

```
0) DEF PROC CONSTRUIRE(X :ENTIER;VAR T:TAB)
1) C←X div 100
2) D←(X mod 100) div 10
3) U←X mod 10
4) T[1]← FN Nombre(c,u,d)
5) T[2]← FN Nombre(c,d,u)
6) T[3]← FN Nombre(d,u,c)
7) T[4]← FN Nombre(d,c,u)
8) T[5]← FN Nombre(u,c,d)
9) T[6]← FN Nombre(u,d,c)
10) Fin Construire
```

T.D.O Locaux

Objet	T/N	Rôle
c,u,d	Entier	Déterminer les chiffres d'un nombre
Nombre	Fonction	Construire un nombre de 3 chiffres

✍ **Analyse de la Fonction Nombre**

```
DEF FN Nombre(C,D,U:ENTIER) : ENTIER
Résultat : Nombre ← c*100+d*10+u
Fin Nombre
```

✍ **Algorithme de la Fonction Nombre**

```
0) DEF FN Nombre(C,D,U:ENTIER):ENTIER
1) Nombre ← c*100+d*10+u
2) Fin Nombre
```

✍ **Analyse de la procédure Afficher**

```
DEF PROC AFFICHER(T :TAB)
Résultat : écrire("Max=", FN Max(T))
           écrire("Min=", FN Min(T))
     Pour i de 1 à 6 faire
          Ecrire("T[",i, "]=",T[i])
     Fin Pour
i : compteur
Fin Afficher
```

✍ **Algorithme de la procédure Afficher**

```
0) DEF PROC AFFICHER(T:TAB)
1) Pour i de 1 à 6 faire
       Ecrire("T[",i, "]=",T[i])
   Fin Pour
2) écrire("Max=", FN Max(T))
3) écrire("Min=", FN Min(T))
4) Fin Afficher
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur
Min	Fonction	Déterminer le minimum dans un tableau
Max	Fonction	Déterminer le maximum dans un tableau

Analyse de la Fonction Max

```
DEF FN Max(T :TAB) : ENTIER
Résultat : Max ← m
m=[m←T[1]]
Pour i de 1 à 6 faire
     si(T[i]>m) alors
          m←T[i]
     Fin Si
Fin pour
i:compteur
Fin Max
```

T.D.O Locaux

Objet	T/N	Rôle
I	Entier	Compteur
M	Entier	Déterminer la valeur maximale d'une liste des valeurs

Algorithme de la Fonction Max

```
0) DEF FN Max(T:TAB) : ENTIER
1) [m←T[1]]
2) Pour i de 1 à 6 faire
       si(T[i]>m) alors
            m←T[i]
       Fin Si
   Fin pour
3) Max ← m
4) Fin Max
```

Analyse de la Fonction Min

```
DEF FN Min(T :TAB) : ENTIER
Résultat : Min ← m
m=[m←T[1]]
Pour i de 1 à n faire
     si(T[i]<m) alors
          m←T[i]
     Fin Si
Fin pour
i:compteur
Fin Min
```

T.D.O Locaux

Objet	T/N	Rôle
I	Entier	Compteur
M	Entier	Déterminer la valeur minimale d'une liste des valeurs

Algorithme de la Fonction Min

```
0) DEF FN Min(T:TAB) : ENTIER
1) [m←T[1]]
2) Pour i de 1 à n faire
       si(T[i]<m) alors
            m←T[i]
       Fin Si
   Fin pour
3) Min ← m
4) Fin Min
```

Traduction Pascal

```
PROGRAM ConstruireNombres;
USES wincrt ;
TYPE TAB= array[1..6] of integer ;
VAR X :integer ;
    T :TAB ;
PROCEDURE Saisie(VAR X :integer) ;
BEGIN
 Repeat
Write('Donner un entier:');
Readln(x) ;
 until (x>=100) and (x<=999) ;
END;
```

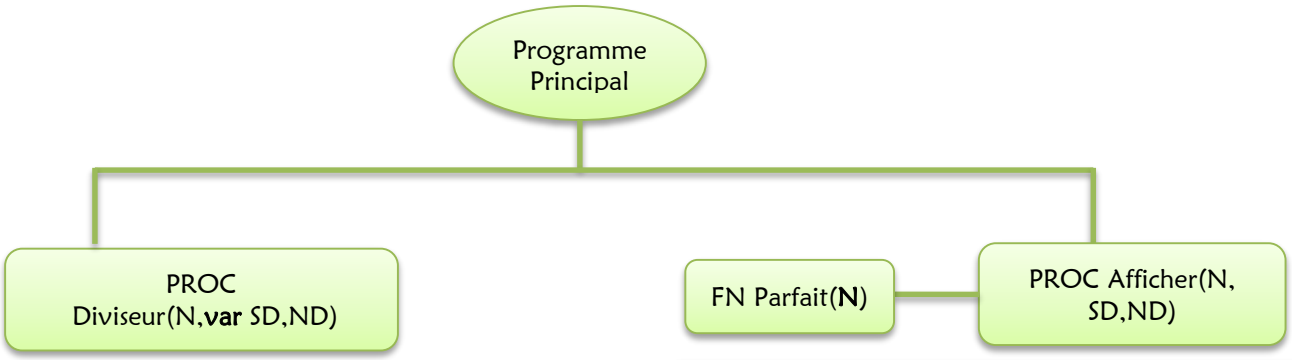
```
PROCEDURE Afficher(T:TAB) ;
VAR  i :integer ;
FUNCTION Max(T:TAB): integer;
VAR
      i,m :integer ;
BEGIN
   m := T[1] ;
   FOR i :=1 to 6 do
      if(T[i]>m) then
         m := T[i] ;
   Max :=m ;
END;
FUNCTION Min(T:TAB): integer;
VAR
```

```
PROCEDURE Construire(X:integer;VAR T:TAB);
VAR   i,c,u,d :integer ;
FUNCTION Nombre(c,u,d:integer): integer;
BEGIN
   Nombre:= c*100+d*10+u ;
END;
BEGIN
      C:=X div 100;
      D:= (X mod 100) div 10;
      U:=X mod 10;
      T[1]:= Nombre(c,u,d);
      T[2]:= Nombre(c,d,u);
      T[3]:= Nombre(d,u,c);
      T[4]:= Nombre(d,c,u);
      T[5]:= Nombre(u,c,d);
      T[6]:= Nombre(u,d,c);
END;
```

```
      i,m :integer ;
BEGIN
   m := T[1] ;
   FOR i :=1 to 6 do
      if(T[i]<m) then
         m := T[i] ;
   Min :=m ;
END;
BEGIN
   for i :=1 to 6 do
       Write(T[i],' ') ;
Writeln ;
Writeln ('Max=', Max(T));
   Writeln ('Min=', Min(T));
END;
BEGIN
      Saisie(x);
      Construire(x,T);
      Afficher(T) ;
END.
```

Exercice 27

✍ **Décomposition modulaire du problème**



✍ **Analyse de programme principal :**

Nom de programme : Subline
Résultat : PROC Afficher(n,SD,NB)
PROC Diviseurs(n,SD,NB)
N=donnée("Donner un entier :")
Fin Subline

✍ **Algorithme de programme principal:**

0) *Début* Subline
1) *Ecrire("Donner un entier :")*
2) *Lire(N)*
3) PROC Diviseurs(N,SD,NB)
4) PROC Afficher(N,SD,NB)
5) *Fin* Subline

T.D.O Globaux

Objet	T/N	Rôle
N	Entier	Stocker la valeur de N
SD	Entier	Calculer la somme des diviseurs de N
NB	Entier	Calculer le nombre des diviseurs de N
Diviseurs	Procédure	Calculer la somme des diviseurs et le nombre des diviseurs de N
Afficher	Procédure	Afficher si le nombre N est Subline ou non

✍ **Analyse de la procédure Diviseurs**

```
DEF PROC DIVISEURS(N :ENTIER ;VAR SD,NB :ENTIER)
Résultat : SD,NB
(SD,NB)=[SD←0, NB←0]
Pour i de 1 à N faire
  Si N mod i =0 alors
    SD←SD+i
    NB←NB+1
  Fin Si
Fin Pour
i : compteur
Fin Diviseurs
```

✍ **Algorithme de la procédure Diviseurs**

```
0) DEF PROC DIVISEURS (N:ENTIER;VAR SD,NB :ENTIER)
1) [SD←0, NB←0]
Pour i de 1 à N faire
  Si N mod i =0 alors
    SD←SD+i
    NB←NB+1
  Fin Si
Fin Pour
2) Fin Diviseurs
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur

✍ **Analyse de la procédure Afficher**

```
DEF PROC AFFICHER(N,SD,NB :ENTIER)
Résultat : Trait
Trait=[] Si (FN Parfait(SD)) et
           (FN Parfait(NB)) alors
         Ecrire(N," est subline")
         Sinon
         Ecrire(N," n'est subline")
         Fin Si
Fin Afficher
```

✍ **Algorithme de la procédure Afficher**

```
0) DEF PROC AFFICHER(N,SD,NB :ENTIER)
1) Si (FN Parfait(SD)) et
      (FN Parfait(NB)) alors
           Ecrire(N," est subline")
   Sinon
           Ecrire(N," n'est subline")
   Fin Si
2) Fin Afficher
```

T.D.O Locaux

Objet	T/N	Rôle
Parfait	Fonction	Tester si un entier est parfait ou non

Analyse de la Fonction Parfait

```
DEF FN Parfait(N:ENTIER) : BOOLEEN
Résultat : Parfait←S=N
Trait=[]
S=[s←1]Pour i de 2 à N div 2 faire
  Si(N mod i = 0) alors
    S←S+i
  Fin Si
Fin Pour
i : compteur
Fin Parfait
```

Algorithme de la Fonction Parfait

```
0) DEF FN Parfait(N:ENTIER) : BOOLEEN
1) [s←1]
2) Pour i de 2 à N div 2 faire
     Si(N mod i = 0) alors
       S←S+i
     Fin Si
   Fin Pour
3) Parfait←S=N
4) Fin Parfait
```

T.D.O Locaux

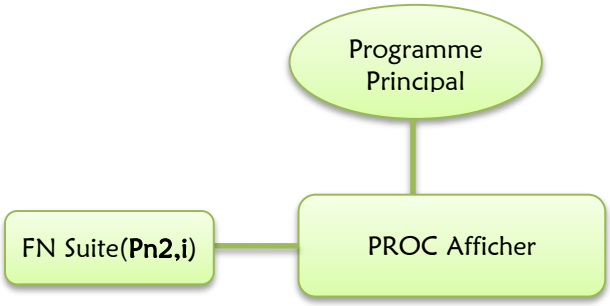
Objet	T/N	Rôle
i	Entier	Compteur
S	Entier	Calculer la somme des diviseurs de x

Traduction Pascal

```
PROGRAM Subline ;
USES wincrt ;
VAR
   n,SD, NB :integer ;
PROCEDURE Diviseurs(X :integer; VAR
SD,NB :integer) ;
VAR i :integer ;
BEGIN
   SD :=0 ; NB :=0 ;
For i:=1 to N do
  if N mod i =0 then
    begin
    SD:=SD+i;
    NB:=NB+1;
   End;
END;
```

```
PROCEDURE Afficher(N ,SD,NB :integer) ;
FUNCTION Parfait(N:integer):Boolean;
VAR   i,S :integer ;
BEGIN
S :=1; FOR i :=2 to N div 2 do
         If(N mod i = 0) then
            S :=S+i ;
Parfait := S=N ;
END ;
BEGIN
If(Parfait(SD)) and (Parfait(NB)) then
     Writeln(N,' est subline')
Else
    Writeln(N,' n''est pas subline');
END;
BEGIN
     Write('Donner un entier :') ;
     Readln(N) ;
     Diviseurs(N,SD,NB) ;
     Afficher(N,SD,NB) ;
END.
```

✍ **Décomposition modulaire du problème**



✍ **Analyse de programme principal :**

Nom de programme : SuitePn
Résultat : PROC Afficher
Fin SuitePn

✍ **Algorithme de programme principal:**

0) *Début* SuitePn
1) PROC Afficher
2) *Fin* SuitePn

T.D.O Globaux

Objet	T/N	Rôle
Afficher	Procédure	Afficher les termes successifs d'une suite

✍ **Analyse de la procédure Afficher**

DEF PROC AFFICHER
Résultat :
[Pn2←2,i←3]
Répéter
  Pn← Fn Suite(Pn2,i)
  Ecrire("P",i,"=",Pn)
  i←i+2
Jusqu'à abs(Pn-Pn2) < 1/carré(100)
Fin Afficher

✍ **Algorithme de la procédure Diviseurs**

0) DEF PROC AFFICHER
1) **[Pn2←2,i←3]**
Répéter
  Pn← Fn Suite(Pn2,i)
  Ecrire("P",i,"=",Pn)
  i←i+2
Jusqu'à abs(Pn-Pn2) < 1/carré(100)
2) Fin AFFICHER

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur
Pn	Réel	Calculer le terme d'indice n
Pn2	Réel	Calculer le terme d'indice n-2
Suite	Fonction	Calculer le terme i d'une suite

✍ **Analyse de la Fonction Suite**

DEF FN Suite(P:REEL;I:ENTIER):REEL
Résultat : Suite←P*(i-1)/i*(i+1)/i

Fin Suite

✍ **Algorithme de la Fonction Suite**

0) DEF FN Suite(P:REEL;I:ENTIER):REEL
1) Suite← P*(i-1)/i*(i+1)/i
2) Fin Suite

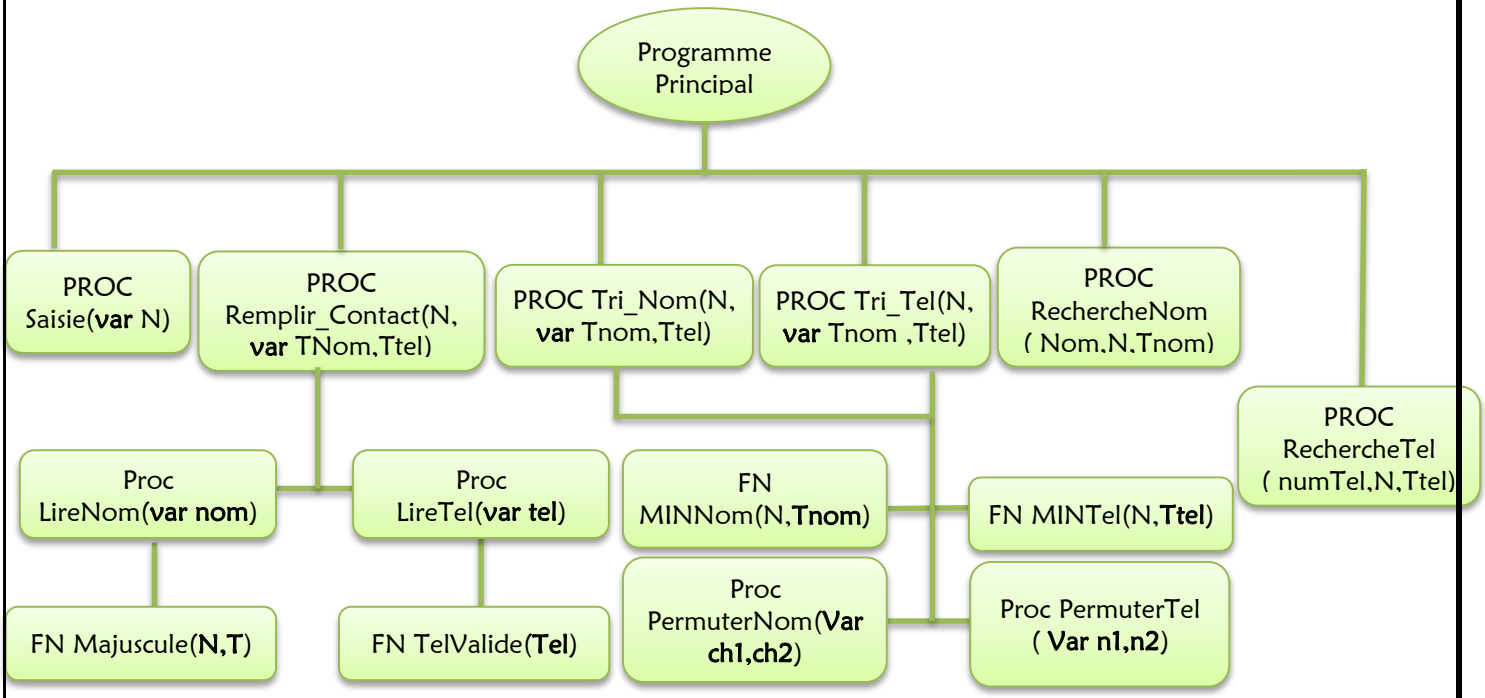
Traduction Pascal

```
PROGRAM SuitePn ;
USES wincrt ;
PROCEDURE Afficher ;
VAR i :integer ;
    Pn,Pn2 :real ;
     FUNCTION Suite(p :real ;i:integer):real;
     BEGIN
          Suite := P*(i-1)/i*(i+1)/i ;
     END ;
BEGIN
     Pn2:=2; i:=3;
     Repeat
       Pn:= Suite(Pn2,i);
      writeln('P',i, '=',Pn) ;
       i:=i+2;
     Until abs(Pn-Pn2) < 1/sqr(100);
END;
```

```
BEGIN
     Afficher ;
END.
```

Exercice 29

Décomposition modulaire du problème



✍ Analyse de programme principal :

Nom de programme : Annuaire
Résultat:PROC RechercheTel(numTel,N,Ttel)
PROC LireTel(numTel)
PROC Tri_Tel(N,Tnom, Ttel)
PROC RechercheNom(Nom,N,Tnom)
PROC LireNom(Nom)
PROC Tri_Nom(N,Tnom, Ttel)
PROC Remplir_Contact(N,Tnom, Ttel)
PROC Saisie(N)
Fin Annuaire

✍ Algorithme de programme principal:

```
0) Début Annuaire
1) PROC Saisie(N)
2) PROC Remplir_Contact(N,Tnom,
   Ttel)
3) PROC Tri_Nom(N, Tnom, Ttel)
4) PROC LireNom(Nom)
5) PROC RechercheNom(Nom,N,Tnom)
6) PROC Tri_Tel(N, Tnom, Ttel)
7) PROC LireTel(numTel)
8) PROC
   RechercheTel(numTel,N,Ttel)
9) Fin Annuaire
```

T.D.N.T

Type
T_Nom = chaine de caractère de taille 20
T_Tel = chaine de caractère de taille 8
TABNom= Tableau de taille 50 et de type *T_Nom*
TABTel= Tableau de taille 50 et de type *T_Tel*

T.D.O Globaux

Objet	T/N	Rôle
N	Entier	stocker le nombre des contacts
Tnom	TABNom	Remplir le tableau par N noms
Ttel	TABTel	Remplir le tableau par N Numéros de téléphone
numTel	T_Tel	Stocker un numéro de téléphone
Nom	T_Nom	Stocker un nom d'une personne
Saisie	Procédure	Saisir le nombre des contacts
Remplir_Contact	Procédure	Remplir les deux tableaux Tnom et Ttel
Tri_Tel	Procédure	Trier le tableau Ttel
Tri_Nom	Procédure	Trier le tableau Tnom
LireNom	Procédure	saisir un nom de personne
LireTel	Procédure	saisir un numéro de téléphone
RechercheNom	Procédure	Vérifier l'existence d'un nom dans l'annuaire
RechercheTel	Procédure	Vérifier l'existence d'un numéro de téléphone dans l'annuaire

✍ Analyse de la procédure Saisie

```
DEF PROC SAISIE(VAR N:entier)
Résultat : N
N=[]Répéter
 N=donnée("Donner le nombre de contact:")
jusqu'à (N ≥ 5) et (N≤50)
Fin Saisie
```

✍ Algorithme de la procédure Saisie

```
0) DEF PROC SAISIE(VAR N :entier)
1) Répéter
     écrire("Donner le nombre de
     contact: ")
     lire(N)
   jusqu'à (N ≥ 5) et (N≤50)
2) Fin Saisie
```

✍ Analyse de la procédure LireNom

```
DEF PROC LIRENOM(VAR NOM:T_Nom)
Résultat : nom
Nom=[]Répéter
 Nom=donnée("Nom: ")
jusqu'à FN Majuscule(Nom)
Fin LireNom
```

✍ **Algorithme de la procédure LireNom**

```
0) DEF PROC LIRENOM(VAR NOM:T_Nom)
1) Répéter
     écrire("Nom: ")
     lire(Nom)
   jusqu'à FN Majuscule(Nom)
2) Fin LireNom
```

T.D.O Locaux

Objet	T/N	Rôle
Majuscule	Fonction	Tester si la chaine est composée seulement par des lettres majuscules ou non

✍ Analyse de la fonction Majuscule

```
DEF FN MAJUSCULE(CH:T_Nom):BOOLEEN
Résultat : Majuscule←test
Test=[test←vrai,i←1]
Répéter
  Si Non(Ch[i] dans["A".."Z"]) Alors
          Test← faux
  Fin Si
i←i+1
jusqu'à (test=faux) ou (i>long(ch))
Fin Majuscule
```

✍ **Algorithme de la fonction Majuscule**

```
0) DEF FN MAJUSCULE(CH:T_Nom):
     BOOLEEN
1) Répéter
   Si Non(Ch[i]dans["A".."Z"]) Alors
           Test← faux
   Fin Si
    i←i+1
    jusqu'à (test=faux) ou
    (i>long(ch))
2) Majuscule←test
3) Fin Majuscule
```

T.D.O Locaux

Objet	T/N	Rôle
I	Entier	Compteur
test	booléen	Tester si la chaine est composée seulement par des lettres majuscules

✍ Analyse de la procédure LireTel

```
DEF PROC LIRETEL(VAR TEL:T_Tel)
Résultat : tel
tel =[]Répéter
 tel =donnée("N° Tel: ")
jusqu'à FN TelValide(tel)
Fin LireTel
```

✍ **Algorithme de la procédure LireTel**

```
0) DEF PROC LIRETEL(VAR TEL:T_tel)
1) Répéter
     écrire("N° Tel ")
     lire(tel)
   jusqu'à FN TelValide (tel)
2) Fin LireTel
```

T.D.O Locaux

Objet	T/N	Rôle
TelValide	Fonction	Tester si un n°Tel donné est valide ou non

Analyse de la fonction TelValide

```
DEF FN TELVALIDE(TEL :T_TEL):BOOLEEN
Résultat : TelValide←test
Si long(ch)≠8 alors
    test←faux
Fin Si
Test=[test←vrai,i←1]
Répéter
Si Non(tel[i] dans["0".."9"]) ou
    Non(tel[1] dans["2","4","5","7","9"])
  Alors
        Test← faux
  Fin Si
i←i+1
jusqu'à (test=faux) ou (i>long(tel))
Fin TelValide
```

T.D.O Locaux

Objet	T/N	Rôle
I	entier	compteur
test	booléen	Tester si un numéro de Tel est valide ou non

Algorithme de la fonction TelValide

```
4) DEF FN TELVALIDE(TEL :ENTIER):
   BOOLEEN
5) [test←vrai,i←1]
Répéter
Si Non(tel[i] dans["0".."9"]) ou
Non(tel[1] dans
["2","4","5","7","9"])
  Alors
        Test← faux
  Fin Si
i←i+1
jusqu'à (test=faux) ou (i>long(tel))
6) Si long(ch)≠8 alors
    test←faux
Fin Si
7) TelValide←test
8) Fin TelValide
```

Analyse de la procédure Remplir_Contact

```
DEF PROC REMPLIR_CONTACT(N :ENTIER ; VAR Tnom:TABNom;VAR Ttel:TABTtel)
Résultat : Tnom,Ttel
(Tnom,Ttel) =[]
Pour i de 1 à N faire
  PROC LireNom(Tnom[i])
  PROC LireTel(Ttel[i])
Fin Pour
Fin Remplir_Contact
```

T.D.O Locaux

Objet	T/N	Rôle
I	Entier	Compteur

Algorithme de la procédure Remplir_Contact

```
0) DEF PROC REMPLIR_CONTACT(N :ENTIER ; VAR
   Tnom:TABNom; VAR Ttel:TABTtel)
1) Pour i de 1 à N faire
  PROC LireNom(Tnom[i])
  PROC LireTel(Ttel[i])
Fin Pour
2) Fin Remplir_Contact
```

Analyse de la procédure Tri_Tel

```
DEF PROC TRI_TEL(N :ENTIER ; VAR Tnom:TABNom;VAR Ttel:TABTtel)
Résultat : Tnom,Ttel
Pour i de 1 à N-1 faire
posmin←FN MinTel(i,N,Ttel)
si posmin ≠ i alors
 PROC PermutTel(Ttel[posmin],Ttel[i])
 PROC PermutNom(Tnom[posmin],Tnom[i])
Fin Si
Fin Pour
i : compteur
Fin Tri_Tel
```

Algorithme de la procédure Tri_Tel

```
0) DEF PROC TRI_TEL(N :ENTIER ; VAR
   Tnom:TABNom;VAR Ttel:TABTtel)
1) Pour i de 1 à N-1 faire
   posmin←FN MinTel(i,N,Ttel)
   si posmin ≠ i alors
    PROC PermutTel(Ttel[posmin],Ttel[i])
    PROC PermutNom(Tnom[posmin],Tnom[i])
   Fin Si
Fin Pour
2) Fin Tri_Tel
```

T.D.O Locaux

Objet	T/N	Rôle
I	Entier	Compteur
MinTel	Fonction	Déterminer la position de plus petit numéro de téléphone dans **Ttel** à partir d'une position **pos**

✍ **Analyse de la Fonction MinTel**

```
DEF FN Mintel(pos,n:ENTIER; Ttel:TABTtel):ENTIER
Résultat : MinTel ← posmin
posmin =[posmin←pos]
Pour i de pos+1 à n faire
    si(Ttel[i]<Ttel[posmin]) alors
        posmin←i
    Fin Si
Fin pour
i:compteur
Fin MinTel
```

✍ **Algorithme de la Fonction MinTel**

```
0) DEF FN Mintel(pos,n:ENTIER;Ttel:TABTtel)
      : ENTIER
1) [posmin←pos]
Pour i de pos+1 à n faire
   si(Ttel[i]<Ttel[posmin]) alors
     posmin←i
   Fin Si
Fin pour
2) MinTel ← posmin
3) Fin MinTel
```

T.D.O Locaux

Objet	T/N	Rôle
I	Entier	Compteur
Posmin	entier	Déterminer la position de plus petit numéro de téléphone dans **Ttel** à partir d'une position **pos**

✍ **Analyse de la procédure PermutTel**

```
DEF PROC PERMUTTEL(VAR X,Y :T_TEL)
Résultat : x,y
          aux←x
          x←y
          y←aux
Fin PermutTel
```

✍ **Algorithme de la procédure PermutTel**

```
0) DEF PROC PERMUTTEL(VAR X,Y : T_TEL)
1) Aux←x
2) X←y
3) Y←aux
4) Fin PermutTel
```

T.D.O Locaux

Objet	T/N	Rôle
aux	T_Tel	Variable auxiliaire

✍ **Analyse de la procédure Tri_Nom**

```
DEF PROC TRI_NOM(N :ENTIER ; VAR Tnom:TABNom;VAR Ttel:TABTtel)
Résultat : Tnom,Ttel
Pour i de 1 à N-1 faire
posmin←FN MinNom(i,N,TNom)
si posmin ≠ i alors
 PROC PermutTel(Ttel[posmin],Ttel[i])
 PROC PermutNom(Tnom[posmin],Tnom[i])
Fin Si
Fin Pour
```

```
i : compteur
Fin Tri_Nom
```

✍ **Algorithme de la procédure Tri_Nom**

```
0) DEF PROC TRI_NOM(N:ENTIER;VAR Tnom:TABNom;VAR Ttel:TABTtel)
1) Pour i de 1 à N-1 faire
     posmin←FN MinNom (i,N,TNom)
     si posmin ≠ i alors
       PROC PermutTel(Ttel[posmin],Ttel[i])
       PROC PermutNom(Tnom[posmin],Tnom[i])
     Fin Si
Fin Pour
2) Fin Tri_Nom
```

T.D.O Locaux

Objet	T/N	Rôle
I	Entier	Compteur
MinNom	Fonction	Déterminer la position de plus petit nom dans **TNom** à partir d'une position **pos**

✍ **Analyse de la Fonction MinNom**

```
DEF FN MinNom(pos,n:ENTIER; Ttel:TABTtel):ENTIER
Résultat : MinNom ← posmin
posmin =[posmin←pos]
Pour i de pos+1 à n faire
     si(TNom[i]<TNom[posmin]) alors
          posmin←i
     Fin Si
Fin pour
i:compteur
Fin MinNom
```

✍ **Algorithme de la Fonction MinNom**

```
0) DEF FN MinNom(pos,n:ENTIER;Ttel:TABTtel)
       : ENTIER
1) [posmin←pos]
Pour i de pos+1 à n faire
   si(TNom[i]<TNom[posmin]) alors
     posmin←i
   Fin Si
Fin pour
2) MinNom ← posmin
3) Fin MinNom
```

T.D.O Locaux

Objet	T/N	Rôle
I	Entier	Compteur
Posmin	entier	Déterminer la position de plus petit nom dans **TNom** à partir d'une position **pos**

✍ **Analyse de la procédure PermutNom**

```
DEF PROC PERMUTNOM(VAR NOM1,NOM2 :T_NOM)
Résultat : nom1,nom2
          aux←nom1
          nom1← nom2
          nom2←aux
Fin PermutNom
```

✍ **Algorithme de la procédure PermutNom**

```
0) DEF PROC PERMUTNOM(VAR NOM1,NOM2 : T_NOM)
1) aux←nom1
2) nom1← nom2
3) nom2←aux
4) Fin PermutNom
```

T.D.O Locaux

Objet	T/N	Rôle
aux	T_Nom	Variable auxiliaire

Analyse de la procédure RechercheNom

```
DEF PROC RECHERCHE_NOM(NOM :T_NOM ;N :ENTIER ; Tnom:TABNom)
Résultat : Trait
Trait=[]si (Nom=Tnom[m] ) alors
   ecrire("Le nom existe dans l'annuaire")
Sinon
  ecrire("Le nom n'existe pas dans l'annuaire")
Fin si
[g ←1,d← N] répéter
          m← (g+d) div2
          Si Nom<Tnom[m] alors
                  d ←m-1
           Sinon
                  g← m+1
          Fin Si
jusqu'à (Nom=Tnom[m]) ou (g>d)
Fin RechercheNom
```

T.D.O Locaux

Objet	T/N	Rôle
g	Entier	Partie gauche
d	Entier	Partie droite
m	Entier	Milieu de tableau

Algorithme de la procédure RechercheNom

```
0) DEF PROC RECHERCHE_NOM(NOM:T_NOM;N:ENTIER;
   Tnom:TABNom)
1) [g ←1,d← N] répéter
      m← (g+d) div2
      Si Nom<Tnom[m] alors
              d ←m-1
       Sinon
              g← m+1
      Fin Si
   jusqu'à (Nom=Tnom[m]) ou (g>d)
2) si (Nom=Tnom[m] ) alors
      ecrire("Le nom existe dans l'annuaire")
   Sinon
  ecrire("Le nom n'existe pas dans
l'annuaire")
   Fin si
3) Fin RechercheNom
```

Analyse de la procédure RechercheTel

```
DEF PROC RECHERCHE_TEL(NumTel :T_Tel ;N :ENTIER ; Ttel:TABTtel)
Résultat :Trait
Trait=[]si (numTel=Ttel[m] ) alors
   ecrire("Le numéro de téléphone existe dans l'annuaire")
Sinon
  ecrire("Le numéro de téléphone n'existe pas dans l'annuaire")
Fin si
[g ←1,d← N] répéter
     m← (g+d) div2
     Si NumTel<Ttel[m] alors

             d ←m-1
      Sinon
             g← m+1
     Fin Si
jusqu'à (NumTel=Ttel[m])ou(g>d)
Fin RechercheTel
```

T.D.O Locaux

Objet	T/N	Rôle
g	Entier	Partie gauche
d	Entier	Partie droite
m	Entier	Milieu de tableau

✍ **Algorithme de la procédure RechercheTel**

```
0) DEF PROC RECHERCHE_NOM(NumTel :T_Tel ;N:ENTIER; Ttel:TABTtel)
1) [g ←1,d← N] répéter
      m← (g+d) div2
      Si NumTel<Ttel[m] alors
              d ←m-1
       Sinon
              g← m+1
      Fin Si
   jusqu'à (NumTel=Ttel [m]) ou (g>d)
2) si (NumTel=Ttel[m] ) alors
      ecrire("Le numéro de téléphone existe dans l'annuaire")
   Sinon
  ecrire("Le numéro de téléphone n'existe pas dans l'annuaire")
   Fin si
3) Fin RechercheNom
```

✍ **Traduction Pascal**

```
PROGRAM Annuaire;
USES wincrt ;
TYPE
T_Nom= string[20] ;
T_Tel= string[8] ;
TABTel=Array[1..50] of T_Tel;
TABNom=Array[1..50] of T_Nom;
VAR
   N:integer ;
   numTel :T_Tel; Nom :T_Nom ;
   Tnom :TABNom; Ttel:TABTel ;
PROCEDURE Saisie(VAR N :integer);
BEGIN
 repeat
Write('Donner le nombre de contact:
');
Readln(N) ;
 until(N>=5)and(N<=50);
END;
PROCEDURE LireNom(VAR Nom :T_Nom);
FUNCTION Majuscule(ch :T_Nom):Boolean;
VAR   i:integer; test:boolean ;
BEGIN
   Test :=true ; i :=1 ;
   Repeat
     If Not(Ch[i]in['A'.. 'Z']) then
           Test:= false;
      i:=i+1;
   until (test=false)or(i>length(ch)) ;
   Majuscule :=test ;
END;
BEGIN
```

```
PROCEDURE Tri_Tel(n:integer ; VAR
Tnom:TABNom ; VAR Ttel:TABTel) ;
VAR posmin,i :integer ;
FUNCTION MinTel(pos,n:integer;T:TABTel):integer;
VAR   posmin,i :integer ;
BEGIN
      posmin:=pos;
      for i := pos+1 to n do
            if(T[i]<T[posmin]) then
                  posmin:=i;
      MinTel := posmin;
END;
BEGIN
for i :=1 to n do
begin
Posmin := MinTel(i,N,Ttel) ;
  if posmin <> i then
  begin
      PermutTel(Ttel[posmin],Ttel[i]) ;
      PermutNom(Tnom[posmin],Tnom[i]) ;
  end;
end;
END;
PROCEDURE Tri_Nom(n:integer ; VAR
Tnom:TABNom ; VAR Ttel:TABTel) ;
VAR posmin,i :integer ;
FUNCTION MinNom(pos,n:integer;T:TABNom):integer;
VAR   posmin,i :integer ;
BEGIN
      posmin:=pos;
      for i := pos+1 to n do
            if(T[i]<T[posmin]) then
                  posmin:=i;
```

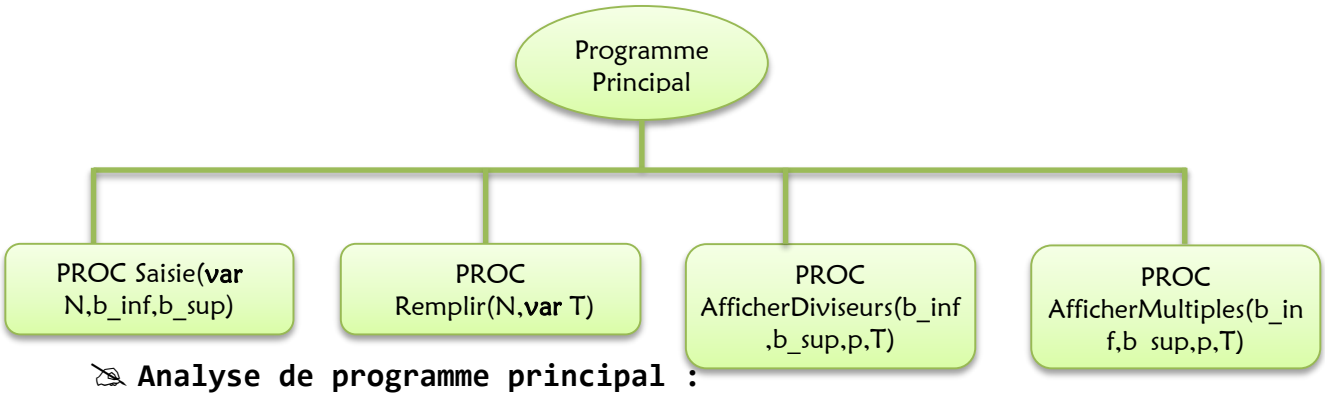
```
       repeat
            Write('Nom: ');
            Readln(Nom) ;
       Until Majuscule(Nom);
END;
PROCEDURE LireTel(VAR numtel :T_Tel);
FUNCTION TelValide(tel:T_tel):Boolean;
VAR   i :integer; test:boolean ;
BEGIN
Test:=true;i:=1;
Repeat
if NOT(tel[i] in ['0'.. '9']) OR
NOT(tel[1] in ['2','4','5','7','9'])
then
     Test:= false;
i:=i+1;
until (test=false) or (i>length(tel));
if length(tel)<>8 then
    test:=false;
TelValide :=test ;
END;
BEGIN
       repeat
            Write('N°Tel: ');
            Readln(numtel) ;
       Until TelValide(numtel);
END;
PROCEDURE Remplir_Contact(n:integer;
VAR Tnom:TABNom ; VAR Ttel:TABTel);
VAR   i :integer ;
BEGIN
      for i :=1 to n do
      begin
       writeln('==> Contact n°',i) ;
        LireNom(Tnom[i]);
        LireTel(Ttel[i]);
      end;
END;
PROCEDURE PermutTel(VAR x,y:T_Tel);
VAR  aux : T_Tel ;
BEGIN
      aux :=x ;
      x :=y ;
      y :=aux ;
END;
PROCEDURE PermutNom(VAR nom1,nom2:
T_Nom);
VAR  aux : T_Nom;
BEGIN
      aux :=nom1 ;
      MinNom := posmin;
END;
BEGIN
for i :=1 to n do
begin
Posmin := MinNom(i,N,TNom) ;
  if posmin <> i then
  begin
     PermutTel(Ttel[posmin],Ttel[i]) ;
     PermutNom(Tnom[posmin],Tnom[i]) ;
  end;
end;
END;
PROCEDURE RechercheNom(nom:T_Tel; VAR
n:integer ;VAR Tnom:TABNom);
VAR   g,d,m :integer ;
BEGIN
g :=1;d:= N; Repeat
        m:= (g+d) div 2;
        if Nom<Tnom[m] then
               d := m-1
        else
               g:= m+1;
until (Nom=Tnom[m]) or (g>d);
if (Nom=Tnom[m] ) then
 writeln('Le nom existe dans l''annuaire')
else
writeln('Le nom n''existe pas dans
l''annuaire');
END;
PROCEDURE RechercheTel(numTel:T_Nom; VAR
n:integer ;VAR Ttel:TABTel);
VAR   g,d,m :integer ;
BEGIN
g :=1;d:= N; Repeat
        m:= (g+d) div 2;
        if numTel<Ttel[m] then
               d := m-1
        else
               g:= m+1;
until (numTel=Ttel[m]) or (g>d);
if (numTel = Ttel [m] ) then
 writeln('Le numéro de téléphone existe dans
l''annuaire')
else
writeln('Le numéro de téléphone n''existe
pas dans l''annuaire');
END;
```

```
      nom1 := nom2 ;
      nom2 :=aux ;
END;
```

```
BEGIN
      Saisie(N);
      Remplir_Contact(N,Tnom, Ttel);
      Writeln('== Recherche d'un nom ==');
      Tri_Nom(N, Tnom, Ttel);
      LireNom(Nom);
      RechercheNom(Nom,N,Tnom);
      Writeln('== Recherche d'un N°Tel==');
      Tri_Tel(N, Tnom, Ttel);
      LireTel(numTel);
      RechercheTel(numTel,N,Ttel);
END.
```

Exercice 30

✍ **Décomposition modulaire du problème**



✍ **Analyse de programme principal :**

Nom de programme : DiviseurMultiple
Résultat: PROC AfficherMultiples(b_inf,b_sup,p,T)
PROC AfficherDiviseurs(b_inf,b_sup,p,T)
P=donnée("Donner un entier :")
PROC Remplir(N,T)
PROC Saisie(N, b_inf,b_sup)
Fin DiviseurMultiple

T.D.N.T

Type
TAB= Tableau de taille 15 et de type entier

✍ **Algorithme de programme principal:**

0) *Début* DiviseurMultiple
1) PROC Saisie(N, b_inf,b_sup)
2) PROC Remplir(N,T)
3) Ecrire("Donner un entier")
4) Lire(p)
5) PROC AfficherDiviseurs (b_inf,b_sup,p,T)
6) PROC AfficherMultiples (b_inf,b_sup,p,T)
7) *Fin* DiviseurMultiple

T.D.O Globaux

Objet	T/N	Rôle
N	Entier	Stocker la taille de tableau
b_inf,b_sup	Entier	Stocker deux positions dans le tableau
T	TAB	Remplir un tableau par N entiers
P	Entier	Saisir la valeur de p
Remplir	Procédure	Remplir un tableau par N entiers
Saisie	Procédure	Saisir les valeurs de N, b_inf et b_sup
AfficherDiviseurs	Procédure	Afficher les diviseurs de p dans le tableau compris entre b_inf et b_sup
AfficherMultiples	Procédure	Afficher les multiples de p dans le tableau compris entre b_inf et b_sup

✍ **Analyse de la procédure Saisie**

```
DEF PROC SAISIE(VAR N,B_INF,B_SUP:ENTIER)
Résultat : N,b_inf,b_sup
(N,b_inf,b_sup)=[]Répéter
 N=donnée("Donner la taille de tableau:")
 B_inf=donnée("B_inf=  ")
 B_Sup=donnée("B_Sup=  ")
jusqu'à (b_inf≥1)ET(b_sup≤N) ET
        (b_inf≤b_sup)ET(N≤15)
Fin Saisie
```

✍ **Algorithme de la procédure Saisie**

```
0) DEF PROC SAISIE(VAR N, B_INF,
   B_SUP :ENTIER)
1) Répéter
   écrire("Donner la taille de
   tableau : ") lire(N)
   écrire("B_inf = ") lire(B_inf)
   écrire("B_Sup = ") lire(B_Sup)
jusqu'à (b_inf≥1)ET(b_sup≤N) ET
    (b_inf≤b_sup)ET(N≤15)
2) Fin Saisie
```

✍ **Analyse de la procédure Remplir**

```
DEF PROC REMPLIR(N :ENTIER ; VAR T :TAB)
Résultat : T
T=[]Pour i de 1 à N faire
          Répéter
          T[i]=donnée("T[",i, "]=")
          jusqu'à T[i]≥0
Fin Pour
i : compteur
Fin Remplir
```

✍ **Algorithme de la procédure Remplir**

```
0) DEF PROC REMPLIR(N :ENTIER ; VAR T:TAB)
1) Pour i de 1 à N faire
   Répéter
      écrire("T[",i, "]=")
      lire(T[i])
      jusqu'à T[i]≥0
  Fin Pour
2) Fin Remplir
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur

✍ Analyse de la procédure AfficherDiviseurs

```
DEF PROC AFFICHERDIVISEURS(B_INF,B_SUP,P :ENTIER ;T :TAB)
Résultat : Trait
Trait=[]
Pour i de b_inf à b_sup faire
Si p mod T[i] = 0 alors
    Ecrire(T[i] ," ")
Fin Si
Fin Pour
i : compteur
Fin AfficherDiviseurs
```

✍ Algorithme de la procédure AfficherDiviseurs

```
0) DEF PROC AFFICHERDIVISEURS(B_INF,B_SUP,
   P :ENTIER ; T:TAB)
1) Pour i de b_inf à b_sup faire
   Si p mod T[i] = 0 alors
       Ecrire(T[i] ," ")
   Fin Si
Fin Pour
2) Fin AfficherDiviseurs
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur

✍ Analyse de la procédure AfficherMultiples

```
DEF PROC AFFICHERMULTIPLES(B_INF,B_SUP,P :ENTIER ;T :TAB)
Résultat : Trait
Trait=[]
Pour i de b_inf à b_sup faire
Si T[i] mod p = 0 alors
    Ecrire(T[i]," ")
Fin Si
Fin Pour
i : compteur
Fin AfficherMultiples
```

✍ Algorithme de la procédure AfficherMultiples

```
0) DEF PROC AFFICHERMULTIPLES(B_INF,B_SUP,
   P :ENTIER ; T:TAB)
1) Pour i de b_inf à b_sup faire
Si T[i] mod p = 0 alors
    Ecrire(T[i] ," ")
Fin Si
   Fin Pour
2) Fin AfficherMultiples
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur

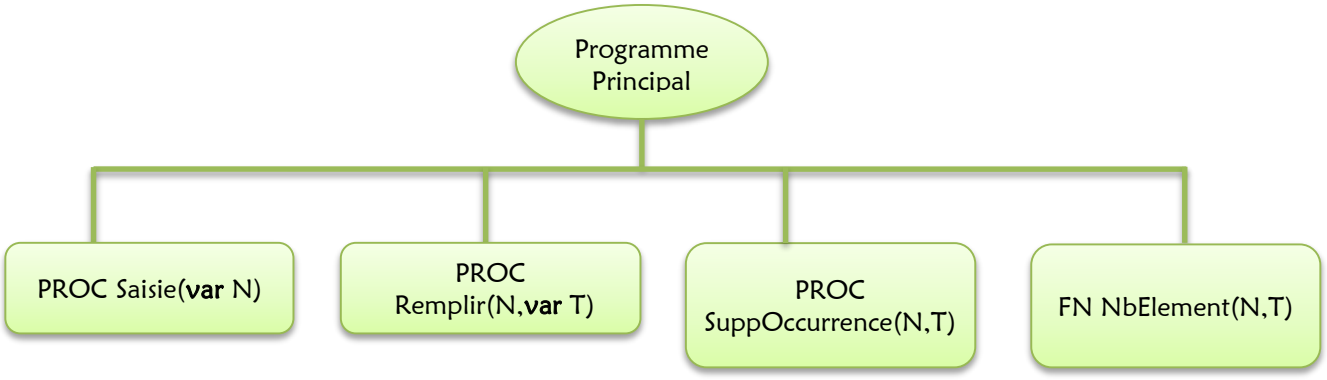
✍ Traduction Pascal

```
PROGRAM DiviseurMultiple;
USES wincrt ;
TYPE TAB=Array[1..15] of integer ;
VAR
   N,p,b_inf,b_sup:integer ;
   T :TAB ;
PROCEDURE Saisie(VAR N,b_inf,b_sup
:integer) ;
BEGIN
 repeat
    Write('Donner la taille de
tableau :'); Readln(N) ;
    Write('b_inf= '); Readln(b_inf) ;
    Write('b_sup= '); Readln(b_sup) ;
 Until (b_inf>=1) and (b_sup<=N) and
(b_inf <= b_sup) and (n<=15);
END;
PROCEDURE AfficherMultiples
(b_inf,b_sup,p :integer ; VAR T:TAB);
VAR   i :integer ;
BEGIN
 for i := b_inf to b_sup do
    if T[i]  mod p = 0 then
      Write(T[i], ' ') ;
END;
```

```
PROCEDURE Remplir(N :integer ;VAR
T:TAB);
VAR   i :integer ;
BEGIN
  for i :=1 to N do
   repeat
      Write('T[',i, ']= ') ;
      Readln(T[i]) ;
   until T[i]>=0 ;
END;
PROCEDURE AfficherDiviseurs
(b_inf,b_sup,p :integer ; VAR T:TAB);
VAR   i :integer ;
BEGIN
 for i :=b_inf to b_sup do
    if p mod T[i] = 0 then
      Write(T[i], ' ') ;
END;
BEGIN
Saisie(N,b_inf,b_sup) ;
Remplir(N,T);
Write('Donner la valeur de P :') ;
Readln(P) ;
AfficherDiviseurs(b_inf,b_sup,p,T) ;
AfficherMultiples(b_inf,b_sup,p,T) ;
END.
```

Exercice 31

✍ **Décomposition modulaire du problème**



✍ **Analyse de programme principal :**

```
Nom de programme : ElementDistinct
Résultat: écrire("Le tableau contient ",
FN NbElement(N,T), "
 éléments distinct")
PROC SuppOccurrence(N,T)
PROC Remplir(N,T)
PROC Saisie(N)
Fin ElementDistinct
```

✍ **Algorithme de programme principal:**

```
0) Début ElementDistinct
1) PROC Saisie(N)
2) PROC Remplir(N,T)
3) PROC SuppOccurrence(N,T)
4) écrire("Le tableau contient ",FN
   NbElement(N,T)," éléments
   distinct")
5) Fin ElementDistinct
```

T.D.N.T

Type
TAB= Tableau de taille 20 et de type entier

T.D.O Globaux

Objet	T/N	Rôle
N	Entier	Stocker la taille de tableau
T	TAB	Remplir un tableau par N entiers
P	Entier	Saisir la valeur de p
Remplir	Procédure	Remplir un tableau par N entiers
Saisie	Procédure	Saisir les valeurs de N, b_inf et b_sup
NbElement	Fonction	Calculer le nombre d'éléments distinct de T
SuppOccurrence	Procédure	Remplacer les éléments qui se répètent par 0

✍ **Analyse de la procédure Saisie**

```
DEF PROC SAISIE(VAR N:ENTIER)
Résultat : N
N=[]Répéter
 N=donnée("Donner la taille de tableau:")
 jusqu'à (2≤N) ET (N≤20)
Fin Saisie
```

✍ **Algorithme de la procédure Saisie**

```
0) DEF PROC SAISIE(VAR N:ENTIER)
1) Répéter
     écrire("Donner la taille de
     tableau : ") lire(N)
   jusqu'à (2≤N) ET (N≤20)
2) Fin Saisie
```

✍ **Analyse de la procédure Remplir**

```
DEF PROC REMPLIR(N :ENTIER ;VAR T :TAB)
Résultat : T
T=[]Pour i de 1 à N faire
          Répéter
          T[i]=donnée("T[",i, "]=")
          jusqu'à T[i]>0
Fin Pour
i : compteur
Fin Remplir
```

✍ **Algorithme de la procédure Remplir**

```
0) DEF PROC REMPLIR(N :ENTIER ;VAR T:TAB)
1) Pour i de 1 à N faire
     Répéter
       écrire("T[",i, "]=")
       lire(T[i])
       jusqu'à T[i]>0
   Fin Pour
2) Fin Remplir
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur

✍ **Analyse de la procédure SuppOccurrence**

```
DEF PROC SUPPOCCURRENCE(N :ENTIER ; VAR T :TAB)
Résultat : T
Pour i de 1 à N-1 faire
Si T[i] > 0 alors
     Pour j de i+1 à N faire
       Si T[i]=T[j] alors
         T[j]←0
       Fin Si
     Fin Pour
Fin Si
Fin Pour
i,j : compteur
Fin SuppOccurrence
```

✍ **Algorithme de la procédure SuppOccurrence**

```
0) DEF PROC SUPPOCCURRENCE (N:ENTIER;VAR T:TAB)
1) Pour i de 1 à N-1 faire
   Si T[i] > 0 alors
   Pour j de i+1 à N faire
     Si T[i]=T[j] alors
       T[j]←0
     Fin Si
   Fin Pour
   Fin Si
Fin Pour
2) Fin SuppOccurrence
```

T.D.O Locaux

Objet	T/N	Rôle
i,j	Entier	Compteur

✍ **Analyse de la fonction NbElement**

```
DEF FN NBELEMENT(N :ENTIER ;T :TAB) :ENTIER
Résultat : NbElement←Nb
Nb=[Nb←0]
Pour i de 1 à N faire
Si T[i] > 0 alors
     Nb←Nb+1
Fin Si
Fin Pour
i : compteur
Fin NbElement
```

✍ **Algorithme de la procédure NbElement**

```
0) DEF FN NBELEMENT (N:ENTIER; T:TAB) :ENTIER
1) [Nb←0]Pour i de 1 à N faire
   Si T[i] > 0 alors
   Nb←Nb+1
   Fin Si
Fin Pour
2) NbElement←Nb
3) Fin NbElement
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur
NB	Entier	Calculer le nombre d'éléments distinct de T

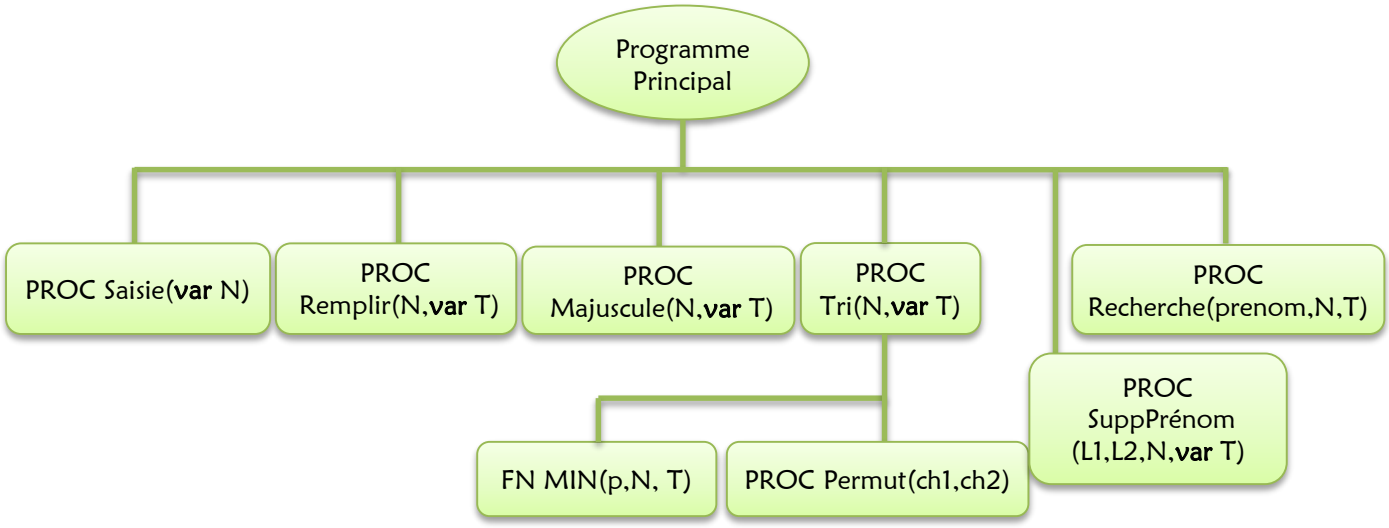
✍ **Traduction Pascal**

```
PROGRAM ElementDistinct;
USES wincrt ;
TYPE TAB=Array[1..20] of integer ;
VAR
   N:integer ;
   T :TAB ;
PROCEDURE Saisie(VAR N:integer) ;
BEGIN
 repeat
    Write('Donner la taille de
tableau :'); Readln(N) ;
Until (2<=N) and (n<=20);
END;
PROCEDURE SuppOccurrence (N:integer ;
VAR T:TAB);
VAR   i,j :integer ;
BEGIN
 for i := 1 to N-1 do
   if T[i] > 0 then
     for j :=i+1 to N do
       if T[i]=T[j] then
         T[j]:=0;
END;
```

```
PROCEDURE Remplir(N:integer; VAR T:TAB);
VAR   i :integer ;
BEGIN
  for i :=1 to N do
   repeat
      Write('T[',i, ']= ') ;
      Readln(T[i]) ;
   until T[i]>0 ;
END;
FUNCTION NbElement(N:integer;VAR
T:TAB) :integer;
VAR   i,NB :integer ;
BEGIN
 Nb :=0 ;
for i :=1 to N do
    if T[i] > 0 then
      Nb:=Nb+1;
NbElement :=Nb ;
END;
BEGIN
Saisie(N) ;
Remplir(N,T);
SuppOccurrence(N,T);
writeln('Le tableau contient ',
NbElement(N,T),' éléments distinct')
END.
```

Exercice 32

✍ **Décomposition modulaire du problème**



✍ **Analyse de programme principal:**

Nom de programme : Exercice32
Résultat:PROC Recherche (prenom,N,T)
prenom=donnée("Donner le prénom cherché:"
PROC SuppPrénom(L1,L2,N,T)
L1=donnée("Donner un L1 :")
L2=donnée("Donner un L2 :")
PROC Tri(N,T)
PROC Majuscule(N,T)
PROC Remplir (N,T)
PROC Saisie(N)
Fin Exercice32

✍ **Algorithme de programme principal:**

0) *Début* Exercice32
1) PROC Saisie(N)
2) PROC Remplir(N,T)
3) PROC Majuscule (N, T)
4) PROC Tri(N,T)
5) Ecrire("Donner un L1 :")
6) Lire(L1)
7) Ecrire("Donner un L2 :")
8) Lire(L2)
9) PROC SuppPrénom(L1,L2,N,T)
10) Ecrire("Donner le prénom cherché:")
11) Lire(prenom)
12) PROC Recherche(prenom,N,Ttel)
13) *Fin* Exercice32

T.D.N.T

Type
TAB= Tableau de taille 100 et de type chaine de caractères

T.D.O Globaux

Objet	T/N	Rôle
N	Entier	stocker le nombre des contacts
T	TAB	Remplir le tableau par N noms
L1, L2	caractère	Saisir deux caractères
Prénom	chaine	Stocker un prénom d'une personne
Saisie	Procédure	Saisir la taille de tableau
Remplir	Procédure	Remplir le tableau T par N prénoms
Tri	Procédure	Trier le tableau T
Majuscule	Procédure	Convertir tous les prénoms en majuscule
SuppPrénom	Procédure	Supprimer tous les prénoms dont le caractère n°1 est compris entre L1 et L2
Recherche	Procédure	Vérifier l'existence d'un prénom dans le tableau

✍ **Analyse de la procédure Saisie**

DEF PROC SAISIE(VAR N:entier)
Résultat : N
N=[]*Répéter*
N=**donnée**("Donner la taille de tableau:")
jusqu'à (N ≥ 10) et (N<100)
Fin Saisie

✍ **Algorithme de la procédure Saisie**

0) DEF PROC SAISIE(VAR N :entier)
1) Répéter
écrire("Donner le nombre de contact: ")
lire(N)
jusqu'à (N ≥ 10) et (N<100)
2) Fin Saisie

✍ Analyse de la procédure Remplir

```
DEF PROC REMPLIR (N:ENTIER; VAR T:TAB)
Résultat : T
T =[]Pour i de 1 à N faire
     Répéter
       T[i]=donnée("T[",i, "]=")
     jusqu'à long(T[i])≠0
Fin Pour
Fin Remplir
```

T.D.O Locaux

Objet	T/N	Rôle
I	Entier	Compteur

✍ Algorithme de la procédure Remplir

```
0) DEF PROC REMPLIR (N:ENTIER; VAR T:TAB)
1) Pour i de 1 à N faire
   Répéter
     écrire("T[",i, "]=")
     lire(T[i])
jusqu'à long(T[i])≠0
Fin Pour
2) Fin Remplir
```

✍ Analyse de la procédure Majuscule

```
DEF PROC MAJUSCULE(N:ENTIER; VAR T:TAB)
Résultat : T
Pour i de 1 à N faire
Ch←T[i]
     Pour j de 1 à long(ch) faire
        ch[j]←Majus(ch[j])
     Fin Pour
T[i] ←Ch
Fin Pour
Fin Majuscule
```

T.D.O Locaux

Objet	T/N	Rôle
i,j	Entier	Compteur
ch	chaine	Chaine auxiliaire

✍ Algorithme de la procédure Majuscule

```
0) DEF PROC MAJUSCULE(N:ENTIER;VAR T:TAB)
1) Pour i de 1 à N faire
   Ch←T[i]
   Pour j de 1 à long(ch) faire
   ch[j]←Majus(ch[j])
     Fin Pour
     T[i] ←Ch
Fin Pour
2) Fin Majuscule
```

✍ Analyse de la procédure Tri

```
DEF PROC TRI(N:ENTIER; VAR T:TAB)
Résultat : T
Pour i de 1 à N-1 faire
posmin←FN Min (i,N,T)
si posmin ≠ i alors
 PROC Permut(T[posmin],T[i])
Fin Si
Fin Pour
i : compteur
Fin Tri
```

✍ Algorithme de la procédure Tri

```
0) DEF PROC TRI (N :ENTIER ; VAR T:TAB)
1) Pour i de 1 à N-1 faire
   posmin←FN Min(i,N,T)
   si posmin ≠ i alors
    PROC Permut(T[posmin],T[i])
   Fin Si
Fin Pour
2) Fin Tri
```

T.D.O Locaux

Objet	T/N	Rôle
I	Entier	Compteur
Min	Fonction	Déterminer la position de plus petit prénom dans **T** à partir d'une position **pos**

✍ **Analyse de la Fonction Min**

```
DEF FN Min (pos,n:ENTIER; T:TAB):ENTIER
Résultat : Min ← posmin
posmin =[posmin←pos]
Pour i de pos+1 à n faire
     si(T[i]<T [posmin]) alors
          posmin←i
     Fin Si
Fin pour
i:compteur
Fin Min
```

✍ **Algorithme de la Fonction Min**

```
0) DEF FN Min(pos,n:ENTIER;T:TAB):ENTIER
1) [posmin←pos]
Pour i de pos+1 à n faire
   si(T[i]<T[posmin]) alors
     posmin←i
   Fin Si
Fin pour
2) Min← posmin
3) Fin Min
```

T.D.O Locaux

Objet	T/N	Rôle
I	Entier	Compteur
posmin	entier	Déterminer la position de plus petit prénom dans le tableau T à partir d'une position **pos**

✍ **Analyse de la procédure Permut**

```
DEF PROC PERMUT (VAR CH1,CH2:CHAINE)
Résultat : ch1,ch2
           aux← ch1
           ch1← ch2
           ch2←aux
Fin Permut
```

✍ **Algorithme de la procédure Permut**

```
0) DEF PROC PERMUTTEL(VAR CH1,CH2:CHAINE)
1) aux← ch1
2) ch1← ch2
3) ch2←aux
4) Fin Permut
```

T.D.O Locaux

Objet	T/N	Rôle
aux	chaine	Variable auxiliaire

✍ **Analyse de la procédure SuppPrenom**

```
DEF PROC SUPPPRENOM(L1,L2 :CARACTERE ;N :ENTIER ; VAR T :TAB)
Résultat : T
Pour i de 1 à N faire
Ch←T[i]
Si ch[1] dans[L1..L2] alors
  T[i]←""
Fin Si
Fin Pour
i: compteur
Fin SuppPrenom
```

✍ **Algorithme de la procédure SuppPrenom**

```
0) DEF PROC SUPPPRENOM (L1,L2:CARACTERE;
   N:ENTIER;VAR T:TAB)
1) Pour i de 1 à N faire
   Ch←T[i]
   Si ch[1] dans[L1..L2] alors
     T[i]←""
   Fin Si
Fin Pour
2) Fin SuppPrenom
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur
Ch	Chaine	Chaine auxiliaire

✍ **Analyse de la procédure Décaler**

```
DEF PROC DECALER(N :ENTIER ; VAR T :TAB)
Résultat : T
Pour i de 1 à N faire
Si T[i]= "" alors
  T[i]←""
[J←0]Répéter
          J←j+1
    Jusqu'à (T[j]≠ "")ou(j=N)
    PROC Permuter(T[i], T[j])
Fin Si
Fin Pour
i: compteur
Fin Décaler
```

T.D.O Locaux

Objet	T/N	Rôle
i,j	Entier	Compteur

✍ **Algorithme de la procédure Décaler**

```
0) DEF PROC DECALER(N:ENTIER; VAR T :TAB)
1) Pour i de 1 à N faire
   Si T[i]= "" alors
     T[i]←""
   [J←0]Répéter
           J←j+1
      Jusqu'à (T[j]≠ "")ou(j=N)
   PROC Permuter(T[i], T[j])
   Fin Si
Fin Pour
2) Fin Décaler
```

✍ **Analyse de la procédure Recherche**

```
DEF PROC RECHERCHE(PRENOM :chaine ;N :ENTIER ; T:TAB)
Résultat :Trait
Trait=[]si (prenom=T[i] ) alors
   ecrire("existe ")
Sinon
  ecrire("n'existe pas ")
Fin si
[i←0] répéter
    i←i+1
jusqu'à (prenom=T[i])ou(i=N)
Fin Recherche
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur

✍ **Algorithme de la procédure Recherche**

```
0) DEF PROC RECHERCHE(N:ENTIER;VAR T:TAB)
1) [i←0] répéter
i←i+1
jusqu'à (prenom=T[i])ou(i=N)
2) si (prenom=T[i] ) alors
     ecrire("existe ")
   Sinon
     ecrire("n'existe pas ")
   Fin si
3) Fin Recherche
```

✍ **Traduction Pascal**

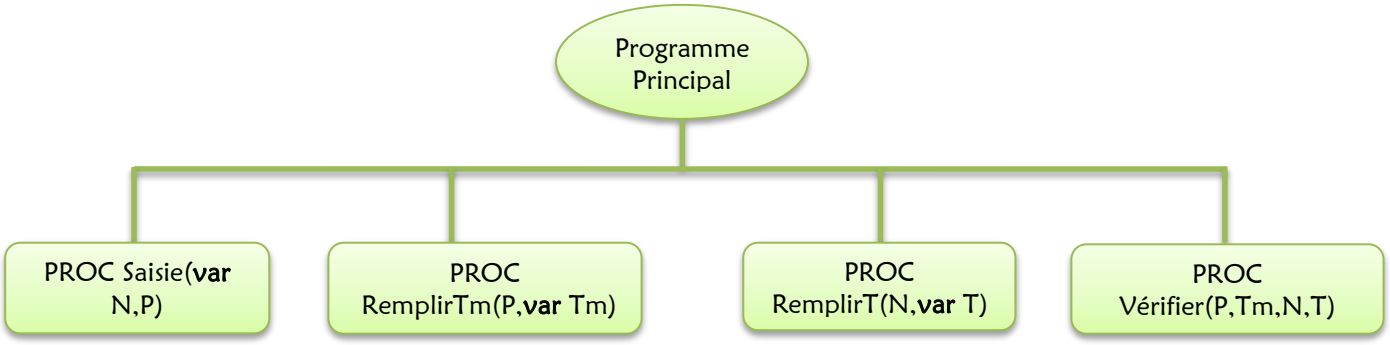
```
PROGRAM Exercice32;
USES wincrt ;
TYPE
     TAB=Array[1..100] of string;
VAR
   L1,L2 :char ;
   N:integer ;
   prenom :string ;
   T :TAB ;
```

```
PROCEDURE Tri(n:integer ; VAR T:TAB) ;
VAR posmin,i :integer ;
FUNCTION Min(pos,n:integer;T:TAB):integer;
VAR   posmin,i :integer ;
BEGIN
      posmin:=pos;
      for i := pos+1 to n do
            if(T[i]<T[posmin]) then
                  posmin:=i;
      Min := posmin;
END;
BEGIN
```

```
PROCEDURE Saisie(VAR N :integer);
BEGIN
 repeat
Write('Donner la taille de tableau:');
Readln(N) ;
 until(N>=10)and(N<100);
END;
PROCEDURE Remplir(n:integer;VAR
T:TAB);
VAR   i :integer ;
BEGIN
for i :=1 to n do
begin
  repeat
     Write('T[',i,']= ');
     Readln(T[i]) ;
  Until length(T[i])<>0 ;
end;
END;
PROCEDURE Majuscule(N :integer ; VAR
T:TAB);
VAR   i,j:integer;  ch :string ;
BEGIN
for i :=1 to n do
   Ch:=T[i];
   for j :=1 to length(ch) do
    ch[j]:=Upcase(ch[j]);
END;
PROCEDURE Permuter(VAR nom1,nom2:
string);
VAR  aux : string;
BEGIN
     aux :=nom1 ;
     nom1 := nom2 ;
     nom2 :=aux ;
END;
PROCEDURE SuppPrenom(L1,L2:char;
N :integer;  VAR T:TAB);
VAR   i,j:integer;  ch :string ;
BEGIN
for i :=1 to n do
   Ch:=T[i];
   if ch[1] in [L1..L2] then
     T[i]:= '';
END;
```

```
for i :=1 to n do
begin
     Posmin := Min(i,N,T) ;
       if posmin <> i then
          Permuter(T[posmin],T[i]) ;
     end;
END;
PROCEDURE Decaler(N :integer; VAR T:TAB);
VAR   i,j :integer ;
BEGIN
for i :=1 to n do
     if T[i]='' then
     begin
          j:=0;
          Repeat
          J:=j+1;
          Until (T[j]<> '')or(j=N);
          Permuter(T[i], T[j]);
     End;
END;
PROCEDURE Recherche (prenom:string; VAR
n:integer; VAR T:TAB);
VAR   i :integer ;
BEGIN
i:=0; Repeat
           i:= i+1;
     until (prenom=T[i]) or (i=n);
     if (prenom=T[i] ) then
      writeln(' existe')
     else
     writeln('n''existe pas');
END;
BEGIN
     Saisie(N);
     Remplir(N,T);
     Majuscule(N,T);
     Tri(N,T);
     Write('Donner L1:');
     Readln(L1);
     Write('Donner L2:');
     Readln(L2);
     SuppPrenom(L1,L2,N,T);
     Decaler(N,T);
     Write('Donner un prènom:');
     Readln(prenom);
     Recherche (prenom,N,T);
END.
```

✍ **Décomposition modulaire du problème**



✍ **Analyse de programme principal:**

Nom de programme : Existance
Résultat:PROC Vérifier (P,Tm,N,T)
PROC RemplirT (N,T)
PROC RemplirTm (P,Tm)
PROC Saisie(N,P)
Fin Existance

✍ **Algorithme de programme principal:**

0) *Début* Existance
1) PROC Saisie(N,P)
2) PROC RemplirTm(p,Tm)
3) PROC RemplirT(N,T)
4) PROC Vérifier(P,Tm,N,T)
5) *Fin* Existance

T.D.N.T

Type
TABmot= Tableau de taille 20 et de type chaine de caractères TAB= Tableau de taille 200 et de type caractère

T.D.O Globaux

Objet	T/N	Rôle
N	Entier	stocker la taille de tableau T
T	TAB	Remplir le tableau par N caractères
P	Entier	stocker la taille de tableau Tm
Tm	TABmot	Remplir le tableau par p mot
Saisie	Procédure	Saisir les tailles de tableau
RemplirT	Procédure	Remplir le tableau T
RemplirTm	Procédure	Remplir le tableau Tm
Verifier	Procédure	Vérifier l'existence des mots du tableau Tm dans le tableau T.

✍ **Analyse de la procédure Saisie**

```
DEF PROC SAISIE(VAR N,P:entier)
Résultat : N,P
(N,P)=[]Répéter
 N=donnée("Donner la taille de tableau T:")
 P=donnée("Donner la taille de tableau Tm:")
jusqu'à (N dans [3..199]) et (P dans [3..19])
Fin Saisie
```

✍ **Algorithme de la procédure Saisie**

```
0) DEF PROC SAISIE(VAR N,P :entier)
1) Répéter
      écrire("Donner la taille de tableau T: ")
      lire(N)
      écrire("Donner la taille de tableau Tm: ")
      lire(p)
jusqu'à (N dans [3..199]) et (P dans [3..19])
2) Fin Saisie
```

✍ **Analyse de la procédure RemplirT**

```
DEF PROC REMPLIRT (N:ENTIER; VAR T:TAB)
Résultat : T
T =[]Pour i de 1 à N faire
       T[i]=donnée("T[",i, "]=")
Fin Pour
Fin RemplirT
```

T.D.O Locaux

Objet	T/N	Rôle
I	Entier	Compteur

✍ **Algorithme de la procédure RemplirT**

```
0) DEF PROC REMPLIRT (N:ENTIER; VAR T:TAB)
1) Pour i de 1 à N faire
      écrire("T[",i, "]=")
      lire(T[i])
Fin Pour
2) Fin RemplirT
```

✍ **Analyse de la procédure RemplirTm**

```
DEF PROC REMPLIRTM P:ENTIER; VAR TM:TABmot)
Résultat : T
T =[]Pour i de 1 à p faire
       Répéter
      Tm[i]=donnée("Tm[",i, "]=")
      jusqu'à long(Tm[i])≠0
Fin Pour
Fin RemplirTm
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur

✍ **Algorithme de la procédure RemplirTm**

```
0) DEF PROC REMPLIRTM (P:ENTIER;VAR TM:TABmot)
1) Pour i de 1 à P faire
      Répéter
      écrire("Tm[",i, "]=")
      lire(Tm[i])
jusqu'à long(Tm[i])≠0
Fin Pour
2) Fin RemplirTm
```

✍ **Analyse de la procédure Vérifier**

```
DEF PROC VERIFIER(P:ENTIER; TM:TABmot ;N:ENTIER; VAR T:TAB)
Résultat : T
T =[]
   Pour i de 1 à p faire
   Ch←Tm[i]
   [j←1]
   Répéter
   [ch1←"", k←1]
    Si ch[1] = T[j] alors
     [l←j]Répéter
           Si ch[k]=T[l] alors
             Ch1←ch1+T[l]
             K←k+1
           Fin Si
```

```
        l←l+1
    jusqu'à (l-j+1>long(ch)) ou   (l>N)
     Fin Si
     J←j+1
jusqu'à (ch=ch1)ou(j>N)
si ch=ch1 alors
écrire(ch,"==> existe")
sinon écrire(ch,"==> n'existe
pas")
Fin Si
Fin Pour
Fin Vérifier
```

T.D.O Locaux

Objet	T/N	Rôle
i,j,k,l	Entier	Compteurs
Ch,ch1	chaine	Chaine auxiliaires

✍ **Algorithme de la procédure Vérifier**

```
0) DEF PROC VERIFIER(P:ENTIER; TM:TABmot; N:ENTIER; VAR T:TAB)
1) Pour i de 1 à p faire
   Ch←Tm[i]
   [j←1]
   Répéter
   [ch1←"", k←1]
    Si ch[1] = T[j] alors
     [l←j]Répéter
          Si ch[k]=T[l] alors
            Ch1←ch1+T[l]
            K←k+1
          Fin Si
          l←l+1
       jusqu'à (l-j+1>long(ch))ou(l>N)
     Fin Si
     J←j+1
   jusqu'à (ch=ch1)ou(j>N)
   si ch=ch1 alors écrire(ch,"==> existe")
   sinon écrire(ch,"==> n'existe pas")
   Fin Si
   Fin Pour
2) Fin Vérifier
```

✍ **Traduction Pascal**

```
PROGRAM Existance;
USES wincrt ;
TYPE
     TAB=Array[1..200] of char;
     TABmot=Array[1..20] of string;
VAR
   N,p:integer ;
   T :TAB ;Tm : TABmot ;
PROCEDURE Saisie(VAR N,p :integer);
BEGIN
repeat
Write('Donner la taille de tableau
T:');
Readln(N) ;
```

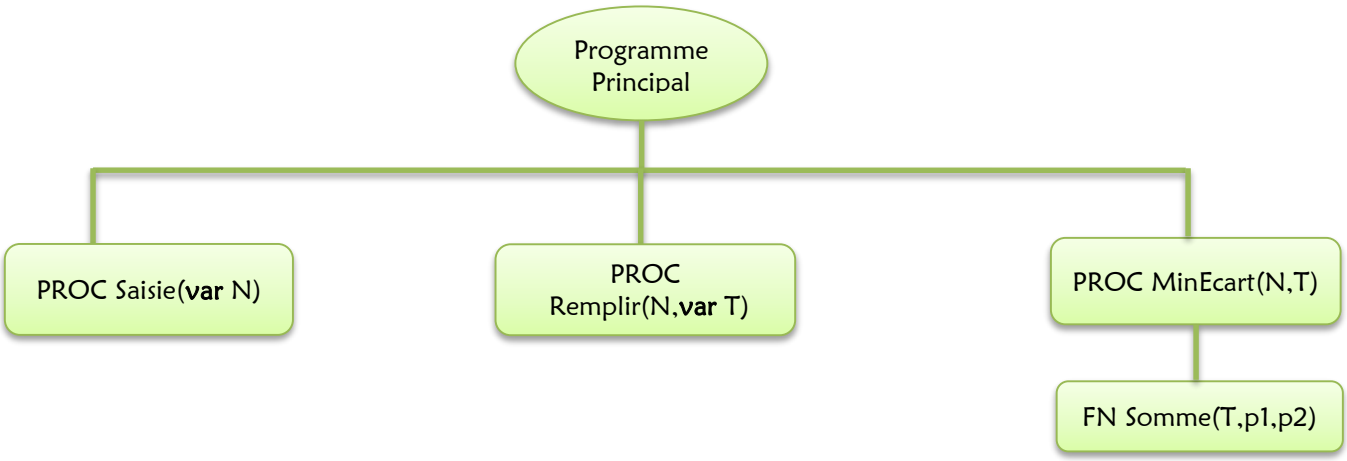
```
PROCEDURE verifier(p:integer; Tm:TABmot;
n:integer ;T:TAB) ;
VAR i,j,k,l :integer ;
Ch,ch1 :string ;
BEGIN
for i :=1 to p do
begin
   Ch:=Tm[i] ;
   j:=1;
   Repeat
   ch1:=''; k:=1;
   if ch[1] = T[j] then
   begin
   l:=j;
```

```
Write('Donner la taille de tableau
Tm:');
Readln(P) ;
until(N in [3..199]) and (P in[3..19])
END;
PROCEDURE RemplirTm(p:integer;VAR
T:TABmot);
VAR   i :integer ;
BEGIN
for i :=1 to p do
begin
  repeat
     Write('Tm[',i,']= ');
     Readln(Tm[i]) ;
  Until length(Tm[i])<>0 ;
end;
END;
PROCEDURE RemplirT(n:integer;VAR
T:TAB);
VAR   i :integer ;
BEGIN
for i :=1 to n do
begin
     Write('T[',i,']= ');
     Readln(T[i]) ;
end;
END;
```

```
   Repeat
        if ch[k]=T[l] then
        begin
            Ch1:=ch1+T[l];
            K:=k+1;
        End;
          l:=l+1;
      until (l-j+1>length(ch))or(l>N);
   End;
   J:=j+1;
   until (ch=ch1)or(j>N);
   if ch=ch1 then write(ch,'==> existe')
   else write (ch,'==> n''existe pas');
end;
END;
BEGIN
     Saisie(N,P);
     RemplirT(N,T);
     RemplirTm(P,Tm);
     verifier (P,Tm,N,T);
END.
```

Exercice 34

✍ **Décomposition modulaire du problème**



✍ **Analyse de programme principal :**

Nom de programme : Ecart
Résultat: PROC MinEcart(N,T)
PROC Remplir(N,T)
PROC Saisie(N)
Fin Ecart

✍ **Algorithme de programme principal:**

0) *Début* Ecart
1) PROC Saisie(N)
2) PROC Remplir(N,T)
3) PROC MinEcart(N,T)
4) *Fin* Ecart

T.D.N.T

Type
TAB= Tableau de taille 20 et de type entier

T.D.O Globaux

Objet	T/N	Rôle
N	Entier	Stocker la taille de tableau
T	TAB	Remplir un tableau par N entiers
Remplir	Procédure	Remplir un tableau par N entiers
Saisie	Procédure	Saisir les valeurs de N
MinEcart	Procédure	Afficher l'indice de l'élément du tableau dont l'écart entre la somme (s1) des éléments qui le précédent et celle des éléments qui le succèdent (s2) soit minimal.

✍ **Analyse de la procédure Saisie**

DEF PROC SAISIE(VAR N:ENTIER)
Résultat : N
N=[]*Répéter*
N=**donnée**("Donner la taille de tableau:")
jusqu'à (5≤N) ET (N≤20)
Fin Saisie

✍ **Algorithme de la procédure Saisie**

0) DEF PROC SAISIE(VAR N:ENTIER)
1) Répéter
écrire("Donner la taille de tableau : ") lire(N)
jusqu'à (5≤N) ET (N≤20)
2) Fin Saisie

✍ **Analyse de la procédure Remplir**

DEF PROC REMPLIR(N :ENTIER ;*VAR* T :TAB)
Résultat : T
T=[]**Pour i de 1 à N faire**
Répéter
T[i]←**random(100)**
jusqu'à T[i]>0
Fin Pour
i : compteur
Fin Remplir

✍ **Algorithme de la procédure Remplir**

0) DEF PROC REMPLIR(N :ENTIER ;*VAR* T:TAB)
1) Pour i de 1 à N faire
Répéter
T[i] ←**random(100)**
jusqu'à T[i]>0
Fin Pour
2) Fin Remplir

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur

✍ **Analyse de la procédure MinEcart**

```
DEF PROC MINECART(N :ENTIER ;T :TAB)
Résultat: écrire("S1=",FN Somme(1,i-1),"S2=",FN Somme(i+1,N)," ind=",ind)
[e←abs(s1-s2),ind←2]
Pour i de 3 à N-1 faire
S1←FN Somme(1,i-1)
S2←FN Somme(i+1,N)
Si e < abs(S1-S2) alors
    e←S1-S2
    ind←i
Fin Si
Fin Pour
S1←FN Somme(1,1)
S2←FN Somme(3,N)
i : compteur
Fin MinEcart
```

✍ **Algorithme de la procédure MinEcart**

```
0) DEF PROC MINECART(N :ENTIER; T :TAB)
1) S1←FN Somme(T,1,1)
2) S2←FN Somme(T,3,N)
3) [e←abs(s1-s2),ind←2]
4) Pour i de 3 à N-1 faire
   S1←FN Somme(T,1,i-1)
   S2←FN Somme(T,i+1,N)
   Si e > abs(S1-S2) alors
      e←abs(S1-S2)
      ind←i
   Fin Si
Fin Pour
5) écrire("S1=",FN Somme(T,1,ind-1),"S2=",FN
   Somme(T,ind+1,N)," ind=",ind)
6) Fin MinEcart
```

T.D.O Locaux

Objet	T/N	Rôle
I	Entier	Compteur
S1,S2	Entier	Stocker les sommes des deux parties du tableau
E	Entier	Calculer l'écart entre S1 et S2
Ind	Entier	Déterminer l'indice dont l'écart entre les deux sommes est minimal
Somme	Fonction	Calculer la somme des éléments d'un tableau entre deux positions

✍ **Analyse de la fonction somme**

```
DEF FN SOMME(T :TAB ; P1,P2 :ENTIER) :ENTIER
Résultat : Somme←s
S=[s←0]Pour i de p1 à p2 faire
          S←s+T[i]
Fin Pour
i : compteur
Fin somme
```

✍ **Algorithme de la procédure somme**

```
0) DEF FN SOMME(T:TAB ; P1,P2:ENTIER):ENTIER
1) [s←0]
   Pour i de p1 à p2 faire
   S←s+T[i]
  Fin Pour
2) Somme←s
3) Fin somme
```

T.D.O Locaux

Objet	T/N	Rôle
I	Entier	Compteur
S	Entier	Somme des éléments d'un tableau compris entre les positions p1 et p2

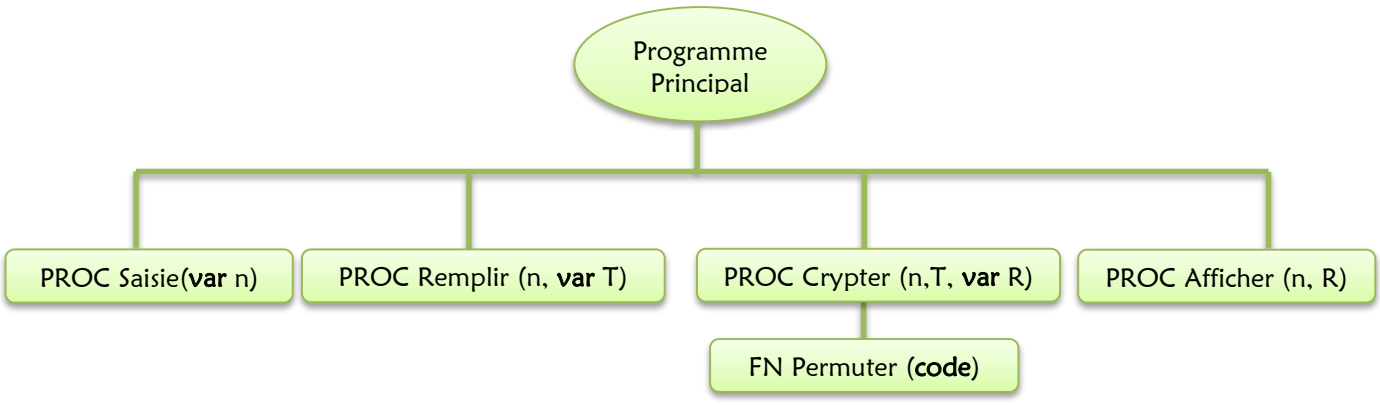
Traduction Pascal

```
PROGRAM Ecart;
USES wincrt ;
TYPE TAB=Array[1..20] of integer ;
VAR  N:integer ;
   T :TAB ;
PROCEDURE MinEcart (N:integer ; VAR T:TAB);
VAR   i,s1,s2,e,ind :integer ;
      FUNCTION Somme(T:TAB;p1,p2:integer)
      :integer;
      VAR i,s:integer ;
      BEGIN
         S :=0 ;
         for i:=p1 to p2 do
            S :=s+T[i] ;
         Somme :=s ;
      END ;
BEGIN
   S1 := Somme(T,1,1) ;
   S2 := Somme(T,3,N) ;
   e:=abs(s1-s2) ;ind:=2 ;
 for i := 3 to n-1 do
 begin
   S1:= Somme(T,1,i-1);
   S2:= Somme(T,i+1,N);
   if e > abs(S1-S2) then
   begin
      e:=abs(S1-S2);
      ind:=i;
    end;
end;
write('S1= ',Somme(T,1,ind-1),' S2= ',
Somme(T,ind+1,N),' ind=',ind);
END;
```

```
PROCEDURE Saisie(VAR N:integer) ;
BEGIN
 repeat
    Write('Donner la taille de
tableau :'); Readln(N) ;
Until (5<=N) and (n<=20);
END;
PROCEDURE Remplir(N :integer; VAR
T:TAB);
VAR   i :integer ;
BEGIN
Randomize ;
for i :=1 to N do
   repeat
      T[i] :=random(100) ;
   until T[i]>0 ;
END;
BEGIN
      Saisie(N) ;
      Remplir(N,T);
      MinEcart(N,T) ;
END.
```

Exercice 35

Décomposition modulaire du problème



Analyse de programme principal :

Nom de programme : Cryptage
Résultat: PROC Afficher(n, R)
PROC crypter (n,T,R)
PROC Remplir (n,T)
PROC Saisie(n)
Fin Cryptage

Algorithme de programme principal:

0) *Début* Cryptage
1) PROC Saisie(n)
2) PROC Remplir (n,T)
3) PROC Crypter (n,T,R)
4) PROC Afficher(n,R)
5) *Fin* Cryptage

T.D.N.T

Type
TAB= Tableau de taille 100 et de type de caractère

T.D.O Globaux

Objet	T/N	Rôle
N	Entier	Stocker la taille de tableau
T	TAB	Stocker le tableau T par n caractères
R	TAB	Contient le résultat de cryptage du tableau T
Saisie	Procédure	Saisir la taille de tableau
Remplir	Procédure	Remplir un tableau par n caractères
Crypter	Procédure	Crypter le tableau T et affecter le résultat au tableau R
Afficher	Procédure	Afficher les éléments d'un tableau

Analyse de la procédure Saisie

```
DEF PROC SAISIE(VAR N:entier)
Résultat : n
n=[]Répéter
 n=donnée("Donner la taille de tableau: ")
jusqu'à n dans [3..20]
Fin Saisie
```

Algorithme de la procédure Saisie

```
0) DEF PROC SAISIE(VAR n :entier)
1) Répéter
     écrire("Donner la taille du
     tableau: ")
     lire(n)
   jusqu'à n dans [3..20]
2) Fin Saisie
```

Analyse de la procédure Remplir

```
DEF PROC REMPLIR(N :ENTIER ;VAR T :TAB)
Résultat : T
T=[]Pour i de 1 à N faire
     répéter
T[i]=donnée("T[",i, "]=")
Jusqu'à majus(T[i]) dans ["A".."Z"]
Fin Pour
i : compteur
Fin Remplir
```

Algorithme de la procédure Remplir

```
0) DEF PROC REMPLIR(N :ENTIER ;VAR T: TAB)
1) Pour i de 1 à N faire
     répéter
       T[i]=donnée("T[",i, "]=")
     Jusqu'à majus(T[i]) dans ["A".."Z"]
   Fin Pour
2) Fin Remplir
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur

✍ **Analyse de la procédure Afficher**

```
DEF PROC AFFICHER(N :ENTIER ;R :TAB)
Résultat : Trait
Trait=[]Pour i de 1 à N faire
          Ecrire("R[",i, "]=",R[i])
        Fin Pour
i : compteur
Fin Afficher
```

✍ **Algorithme de la procédure Afficher**

```
0) DEF PROC AFFICHER(N :ENTIER ;R:TAB)
1) Pour i de 1 à N faire
       Ecrire("R[",i, "]=",R[i])
   Fin Pour
2) Fin Afficher
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur

✍ **Analyse de la procédure Crypter**

```
DEF PROC CRYPTER(N:ENTIER; T:TAB; VAR R:TAB )
Résultat : R
Pour i de 1 à N faire
code←    ORD(T[i])
Ncode←FN Permuter(code)
R[i]←CHR(Ncode)
 Fin Pour
i : compteur
Fin Crypter
```

✍ **Algorithme de la procédure Crypter**

```
0) DEF PROC CRYPTER(N:ENTIER; T:TAB; VAR R:TAB)
1) Pour i de 1 à N faire
     code←ORD(T[i])
     Ncode←FN Permuter(code)
     R[i]←CHR(Ncode)
 Fin Pour
2) Fin Crypter
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur
Code	Entier	Code ascii d'un caractère
Ncode	Entier	Nouveau code ascii après permutation des chiffres
Permuter	Fonction	Permuter les deux chiffres de code ascii Code

✍ **Analyse de la Fonction Permuter**

```
DEF FN Permuter(code :entier):ENTIER
Résultat : Permuter ← u*10+d
d←code div 10
u←code mod 10
Fin Permuter
```

✍ **Algorithme de la Fonction Permuter**

```
0) DEF FN Permuter(code :entier): ENTIER
1) d←code div 10
2) u←code mod 10
3) Permuter ← u*10+d
4) Fin Permuter
```

T.D.O Locaux

Objet	T/N	Rôle
D	entier	Déterminer le chiffre de dizaine
U	entier	Déterminer le chiffre d'unité

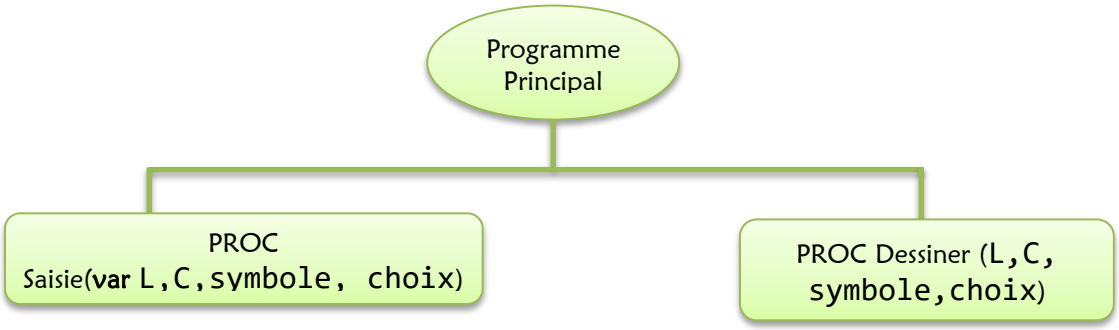
Traduction Pascal

```
PROGRAM Cryptage;
USES wincrt ;
TYPE TAB= array[1..100] of char ;
VAR n :integer ;
    T,R :TAB ;
PROCEDURE Saisie(VAR n :integer) ;
BEGIN
 Repeat
Write('Donner la taille du tableau:');
Readln(n) ;
 until n in [3..20] ;
END;
PROCEDURE Remplir(n:integer;VAR T:TAB);
VAR   i :integer ;
BEGIN
  for i :=1 to n do
   repeat
     Write('T[',i, ']= ') ;
     Readln(T[i] ) ;
   until upcase(T[i]) in ['A'..'Z'];
END;
PROCEDURE Afficher(n:integer;R:TAB) ;
VAR  i :integer ;
BEGIN
for i :=1 to n do
 Writeln(' R[', i ,']=',R[i]) ;
END;
```

```
PROCEDURE Crypter(n:integer; T:TAB ;
var R :TAB);
VAR
    i,code,Ncode :integer ;
  FUNCTION Permuter(code :integer):integer;
  VAR
      d,u :integer ;
  BEGIN
      D :=code div 10 ;
      U :=code mod 10 ;
      Permuter := u*10+d;
  END ;
BEGIN
      for i :=1 to n do
      begin
          code :=ORD(T[i]) ;
          Ncode := Permuter(code) ;
          R[i]:= CHR(Ncode) ;
      end;
END;
BEGIN
      Saisie(n);
      Remplir(n,T);
      Crypter(n,T,R);
      Afficher(n,R) ;
END.
```

Exercice 36

Décomposition modulaire du problème



✍ **Analyse de programme principal:**

Nom de programme : Rectangle
Résultat:
PROC Dessiner(L,C,symbole,choix)
PROC Saisie(L,C, symbole,choix)
Fin Rectangle

✍ **Algorithme de programme principal:**

0) *Début* Rectangle
1) PROC Saisie(L,C,caractere,choix)
2) PROC Dessiner(L,C,caractere,choix)
3) *Fin* Rectangle

T.D.O Globaux

Objet	T/N	Rôle
L,C	entier	Stocker les dimensions du rectangle à dessiner
symbole	caractère	Saisir un caractère parmi (x,+,$,*)
choix	caractère	Saisir le choix de dessin : soit plein (P) ou vide (V)
Saisie	Procédure	Saisir les paramètres nécessaires pour dessiner un rectangle

✍ **Analyse de la procédure Saisie**

DEF PROC SAISIE(VAR L,C:entier ; VAR SYMBOLE,CHOIX:caractére)
Résultat : L,C,symbole,choix
(L,C,symbole,choix)=[]
Répéter
L=**donnée**("Donner le nombre de ligne:")
C=**donnée**("Donner le nombre de colonne:")
jusqu'à (L≠C) et (L dans [2..10])
et (C dans [2..10])
symbole=[]Répéter
symbole=**donnée**("Donner le symbole de dessin (x,+,$,*): ")
jusqu'à symbole dans ["x","+","$","*"]
choix=[]Répéter
choix=**donnée**("Donner le choix de dessin (P ou V): ")
jusqu'à Majus(choix) dans ["P","V"]
Fin Saisie

✍ **Algorithme de la procédure Saisie**

0) DEF PROC SAISIE(VAR L,C:entier; VAR SYMBOLE,CHOIX: caractére)
1) Répéter
écrire("Donner le nombre de ligne:")
Lire(L)
écrire ("Donner le nombre de colonne:")
Lire(C)
jusqu'à (L≠C) et (L dans [2..10]) et (C dans [2..10])
2) Répéter
écrire ("Donner le symbole de dessin (x,+,$,*): ")
Lire(symbole)
jusqu'à symbole dans ["x","+","$","*"]
3) Répéter
écrire ("Donner le choix de dessin (P ou V): ")
lire(choix)
jusqu'à Majus(choix) dans ["P","V"]
4) Fin Saisie

✍ **Analyse de la procédure Dessiner**

```
DEF PROC DESSINER(L,C:entier; SYMBOLE,CHOIX: caractére)
Résultat : Trait
Trait=[]
Pour i de 1 à L faire
    Pour j de 1 à C faire
    Si Majus(choix)="P" alors
       Ecrire(symbole)
    Sinon
       Si (i=1)ou(i=L)ou(j=1) ou(j=C) alors
          Ecrire(symbole)
       Sinon
          Ecrire(" ")
       Fin Si
    Fin Si
    Fin Pour
    Ecrire() {écrire un retour à la ligne}
Fin Pour
i,j : compteur
Fin Afficher
```

T.D.O Locaux

Objet	T/N	Rôle
i,j	Entier	Compteurs

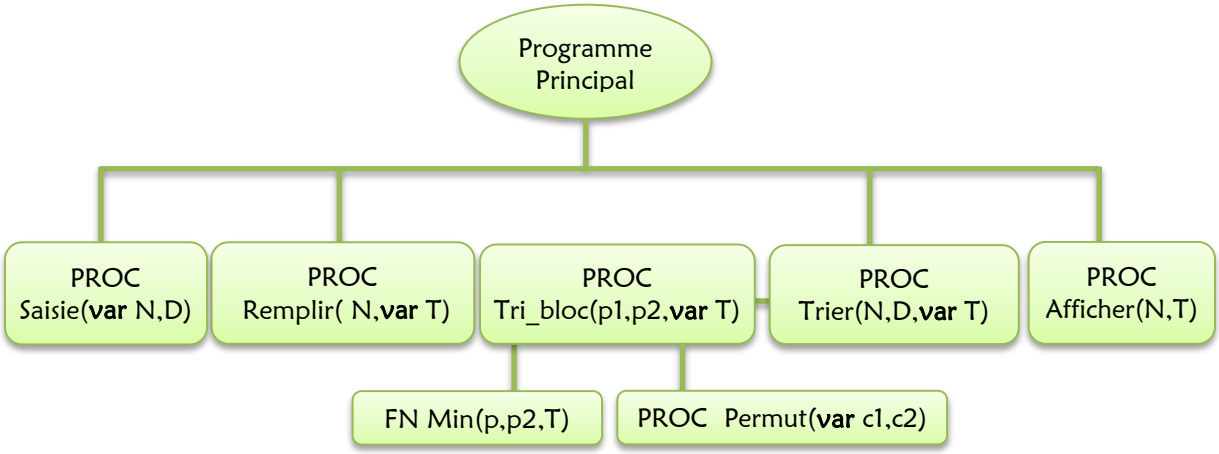
✍ **Algorithme de la procédure Afficher**

```
0) DEF DESSINER(L,C:entier; SYMBOLE,CHOIX: caractére)
1) Pour i de 1 à L faire
   Pour j de 1 à C faire
   Si Majus(choix)="P" alors
      Ecrire(symbole)
   Sinon
     Si (i=1)ou(i=L)ou(j=1)ou(j=C) alors
      Ecrire(symbole)
     Sinon
      Ecrire(" ")
     Fin Si
   Fin Si
   Fin Pour
   Ecrire()
   Fin Pour
2) Fin Afficher
```

✍ **Traduction Pascal**

```
PROGRAM Rectangle ;
USES wincrt ;
VAR L,C :integer ;
Choix ,symbole:char ;
PROCEDURE Saisie(VAR l,c:integer; VAR symbole,choix: char) ;
BEGIN
     REPEAT
        Write('Donner le nombre de ligne:');
        Readln(L) ;
        Write(Donner le nombre de colonne:');
        Readln(L) ;
     UNTIL (L<>C) and (L in [2..10]) and (C in [2..10]);
     REPEAT
       Write('Donner le symbole de dessin (x,+,$,*):');
       Readln(symbole) ;
     UNTIL symbole dans ['x','+','$','*'];
     REPEAT
       Write('Donner le choix de dessin (P ou V): ');
       Readln(choix) ;
     UNTIL Upcase(choix) in ['P','V'];
END;
PROCEDURE Dessiner(l,c:integer; symbole,choix: char) ;
VAR    i,j :integer;
BEGIN
for i:=1 to L do
begin
     for j:=1 to C do
     begin
          if(upcase(choix) = 'P') then
             write(symbole)
          else
          begin
               if(i=1)or(i=L)or(J=1)or(j=C) then
                  write(symbole)
               else
                  write(' ') ;
          end;
     end; writeln;
end;
END;
BEGIN
    Saisie(L,C,symbole,choix) ;
    Dessiner(L,C,symbole,choix) ;
END.
```

✍ **Décomposition modulaire du problème**



✍ **Analyse de programme principal :**

Nom de programme : Tri_par_bloc
Résultat : PROC Afficher(n,T)
PROC Trier(n,T)
PROC Remplir(n,T)
PROC Saisie(n)
Fin Tri_par_bloc

✍ **Algorithme de programme principal:**

0) *Début* Tri_par_bloc
1) PROC Saisie(n)
2) PROC Remplir(n,T)
3) PROC Trier(n,T)
4) PROC Afficher(n,T)
5) *Fin* Tri_par_bloc

T.D.N.T

Type
TAB= Tableau de taille 100 et de type caractère

T.D.O Globaux

Objet	T/N	Rôle
N	Entier	stocker la taille du tableau
D	Entier	stocker la taille du bloc
T	TAB	Remplir le tableau par N caractères
Afficher	Procédure	Afficher un tableau
Remplir	Procédure	Remplir le tableau par n caractères.
Saisie	Procédure	Saisir la taille du tableau
Trier	Procédure	Trier le tableau T dans l'ordre croissant par bloc

✍ **Analyse de la procédure Saisie**

```
DEF PROC SAISIE(VAR N,D :ENTIER)
Résultat : N,D
(N,D)=[]Répéter
 N=donnée("Donner la valeur de N :  ")
 D=donnée("Donner la valeur de D :  ")
jusqu'à (N≥6)ET(N≤100)et
        (N mod D=0)et(D>1)
Fin Saisie
```

✍ **Algorithme de la procédure Saisie**

```
0) DEF PROC SAISIE(VAR N,D  :ENTIER)
1) Répéter
     écrire("Donner la valeur de N: ")
     lire(N)
     écrire("Donner la valeur de D: ")
     lire(D)
jusqu'à (N≥6)ET(N≤100)et(N mod D=0)
et(D>1)
2) Fin Saisie
```

✍ **Analyse de la procédure Remplir**

```
DEF PROC REMPLIR(N :ENTIER ;VAR T :TAB)
Résultat : T
T =[]Pour i de 1 à N faire
Répéter
T[i]=donnée("T[",i, "]=")
jusqu'à T[i] dans ["a".."z"]
Fin Pour
i : compteur
Fin Remplir
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur

✍ **Algorithme de la procédure Remplir**

```
0) DEF PROC REMPLIR(N :ENTIER ;VAR T:TAB)
1) Pour i de 1 à N faire
   Répéter
        écrire("T[",i, "]=")
        lire(T[i])
   jusqu'à T[i] dans ["a".."z"]
  Fin Pour
2) Fin Remplir
```

✍ **Analyse de la procédure Afficher**

```
DEF PROC AFFICHER(N :ENTIER ;T :TAB)
Résultat : Trait
Trait=[]Pour i de 1 à N faire
          Ecrire("T[",i, "]=",T[i])
         Fin Pour
i : compteur
Fin Afficher
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur

✍ **Algorithme de la procédure Afficher**

```
0) DEF PROC AFFICHER(N:ENTIER ;T :TAB)
1) Pour i de 1 à N faire
      Ecrire("T[",i, "]=",T[i])
   Fin Pour
2) Fin Afficher
```

✍ **Analyse de la procédure Trier**

```
DEF PROC TRIER(N,D :ENTIER ;VAR T :TAB)
Résultat : T
T =[]Pour i de 1 à N div D faire
       P1←(i-1)*D+1
       P2←i*D
     Tri_bloc(p1,p2,T)
     Fin Pour
i : compteur
Fin Trier
```

T.D.O Locaux

Objet	T/N	Rôle
P1,p2	Entier	Les bornes de chaque bloc du tableau de taille D
i	Entier	compteur
Tri_bloc	Procédure	Trier un bloc de taille D du tableau T

✍ **Algorithme de la procédure Trier**

```
0) DEF PROC TRIER(N,D:ENTIER ;VAR T:TAB)
1) Pour i de 1 à N div D faire
    P1←(i-1)*D+1
    P2←i*D
    Tri_bloc(p1,p2,T)
  Fin Pour
2) Fin Trier
```

✍ **Analyse de la procédure Tri_bloc**

```
DEF PROC TRI_BLOC(P1,P2 :ENTIER ;VAR T :TAB)
Résultat : T
T =[]Pour i de p1 à p2-1 faire
       posmin←FN Min(i,p2,T)
       si posmin ≠ i alors
        PROC Permut(T[posmin],T[i])
       Fin Si
     Fin Pour
i : compteur
Fin Tri_bloc
```

✍ **Algorithme de la procédure Tri_bloc**

```
0) DEF PROC TRI_BLOC(P1,P2:ENTIER;VAR
   T:TAB)
1) Pour i de p1 à p2-1 faire
       posmin←FN Min(i,p2,T)
      si posmin ≠ i alors
        PROC Permut(T[posmin],T[i])
      Fin Si
  Fin Pour
2) Fin Tri_bloc
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur
Min	Fonction	Déterminer la position de minimum dans un bloc d'éléments d'un tableau à partir d'une position pos
Permut	Procédure	Permuter deux entiers dans un tableau

✍ **Analyse de la Fonction Min**

```
DEF FN Min(pos,p2:ENTIER;T:TAB):ENTIER
Résultat : Min ← posmin
posmin =[posmin←pos]
Pour i de pos+1 à p2 faire
     si(T[i]<T[posmin]) alors
          posmin←i
     Fin Si
Fin pour
i:compteur
Fin Min
```

✍ **Algorithme de la Fonction Min**

```
0) DEF FN Min(pos,n:ENTIER ;T:TAB): ENTIER
1) [posmin←pos]
Pour i de pos+1 à p2 faire
    si(T[i]<T[posmin]) alors
      posmin←i
    Fin Si
Fin pour
2) Min ← posmin
3) Fin Min
```

T.D.O Locaux

Objet	T/N	Rôle
I	Entier	Compteur
posmin	entier	Déterminer la position de minimum dans un tableau à partir d'une position pos

✍ **Analyse de la procédure Permut**

```
DEF PROC PERMUT(VAR C1,C2 :CARACTERE)
Résultat : c1,c2
          aux←c1
          c1←c2
          c2←aux
Fin Permut
```

✍ **Algorithme de la procédure Permut**

```
0) DEF PROC PERMUT(VAR C1,C2 :CARACTERE)
1) aux←c1
2) c1←c2
3) c2←aux
4) Fin Permut
```

T.D.O Locaux

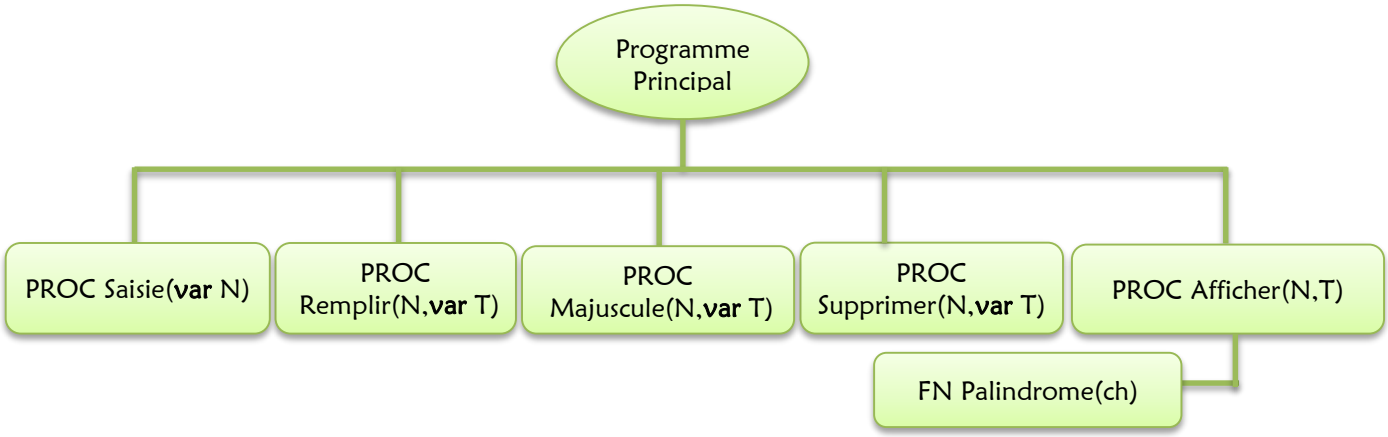
Objet	T/N	Rôle
aux	caractère	Variable auxiliaire

Traduction Pascal

```
PROGRAM Tri_par_bloc;
USES wincrt ;
TYPE TAB=Array[1..100] of char ;
VAR
   N,D:integer ;
   T :TAB ;
PROCEDURE Saisie(VAR N,D :integer) ;
BEGIN
 repeat
    Write('Donner la valeur de N: ');
    Readln(N) ;
    Write('Donner la valeur de D: ');
    Readln(D) ;
 until(N>=6)and(N<=100)and(N mod D=0)
and(D>1);
END;
PROCEDURE Remplir(n:integer;VAR T:TAB);
VAR   i :integer ;
BEGIN
for i :=1 to n do
begin
     repeat
          Write('T[',i, ']= ') ;
          Readln(T[i]) ;
     Until T[i] in ['a'..'z'] ;
end;
END;
PROCEDURE Afficher(n:integer ;T :TAB) ;
VAR  i :integer ;
BEGIN
write('T=') ;
for i :=1 to n do
    Write(T[i],' ') ;
END;
```

```
PROCEDURE Tri_bloc(p1,p2:integer ; VAR
T :TAB) ;
VAR posmin,i :integer ;
FUNCTION Min(pos,p2:integer;T:TAB):
integer;
VAR   posmin,i :integer ;
BEGIN
      posmin:=pos;
      for i := pos+1 to p2 do
            if(T[i]<T[posmin]) then
                  posmin:=i;
      Min := posmin;
END;
PROCEDURE Permut(VAR c1,c2:char);
VAR  aux :char ;
BEGIN
      aux :=c1 ;
      C1 :=c2 ;
      C2 :=aux ;
END;
BEGIN
      for i :=p1 to p2-1 do
      begin
        Posmin := Min(i,p2,T) ;
           if posmin <> i then
             Permut(T[posmin],T[i]) ;
      end;
END;
PROCEDURE Trier(N,D:integer;VAR T:TAB);
VAR   i,p1,p2 :integer ;
BEGIN
For i :=1 to N div D do
      Begin
            P1 :=(i-1)*D+1 ;
            P2 := i*D ;
            Tri_Bloc(p1,P2,T) ;
      End ;
End ;
BEGIN
 Saisie(N,D) ;
 Remplir(n,T);
 Trier(N,D,T) ;
 Afficher(n,T) ;
END.
```

✍ **Décomposition modulaire du problème**



✍ **Analyse de programme principal:**

Nom de programme : Chaine_Palindrome
Résultat:PROC Afficher(N,T)
PROC Supprimer(N,T)
PROC Majuscule(N,T)
PROC Remplir (N,T)
PROC Saisie(N)
Fin Chaine_Palindrome

✍ **Algorithme de programme principal:**

0) *Début* Chaine_Palindrome
1) PROC Saisie(N)
2) PROC Remplir(N,T)
3) PROC Majuscule (N, T)
4) PROC Supprimer(N,T)
5) PROC Afficher(N,T)
6) *Fin* Chaine_Palindrome

T.D.N.T

Type
TAB= Tableau de taille 30 et de type chaine de caractères

T.D.O Globaux

Objet	T/N	Rôle
N	Entier	stocker la taille de tableau
T	TAB	Remplir le tableau par N chaines
Saisie	Procédure	Saisir la taille de tableau
Remplir	Procédure	Remplir le tableau T par N chaines
Supprimer	Procédure	Supprimer tous les caractères non alphabétiques des chaines du tableau.
Majuscule	Procédure	Convertir tous les prénoms en majuscule
Afficher	Procédure	Afficher tous les chaines palindromes du tableau.

✍ **Analyse de la procédure Saisie**

DEF PROC SAISIE(VAR N:entier)
Résultat : N
N=[]*Répéter*
N=**donnée**("Donner la taille de tableau:")
jusqu'à (N ≥ 2) et (N≤30)
Fin Saisie

✍ **Algorithme de la procédure Saisie**

0) DEF PROC SAISIE(VAR N :entier)
1) Répéter
écrire("Donner le nombre de tableau: ")
lire(N)
jusqu'à (N ≥ 2) et (N<30)
2) Fin Saisie

✍ **Analyse de la procédure Remplir**

```
DEF PROC REMPLIR (N:ENTIER; VAR T:TAB)
Résultat : T
T =[]Pour i de 1 à N faire
     Répéter
       T[i]=donnée("T[",i, "]=")
     jusqu'à long(T[i])≠0
Fin Pour
Fin Remplir
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur

✍ **Algorithme de la procédure Remplir**

```
0) DEF PROC REMPLIR (N:ENTIER; VAR T:TAB)
1) Pour i de 1 à N faire
   Répéter
     écrire("T[",i, "]=")
     lire(T[i])
jusqu'à long(T[i])≠0
Fin Pour
2) Fin Remplir
```

✍ **Analyse de la procédure Majuscule**

```
DEF PROC MAJUSCULE(N:ENTIER; VAR T:TAB)
Résultat : T
Pour i de 1 à N faire
Ch←T[i]
     Pour j de 1 à long(ch) faire
        ch[j]←Majus(ch[j])
     Fin Pour
     T[i]←ch
Fin Pour
Fin Majuscule
```

T.D.O Locaux

Objet	T/N	Rôle
i,j	Entier	Compteur
ch	chaine	Chaine auxiliaire

✍ **Algorithme de la procédure Majuscule**

```
0) DEF PROC MAJUSCULE(N:ENTIER; VAR T:TAB)
1) Pour i de 1 à N faire
   Ch←T[i]
   Pour j de 1 à long(ch) faire
   ch[j]←Majus(ch[j])
   Fin Pour
Fin Pour
2) Fin Majuscule
```

✍ **Analyse de la procédure Supprimer**

```
DEF PROC SUPPRIMER(N:ENTIER; VAR T:TAB)
Résultat : T
Pour i de 1 à N faire
Ch←T[i]
Pour j de 1 à long(ch) faire
  Si Non(Majus(ch[j])dans["A".."Z"])
    Alors    Efface(ch,j,1)
  Fin si
Fin pour
Fin Pour
i : compteur
Fin Supprimer
```

T.D.O Locaux

Objet	T/N	Rôle
i,j	Entier	Compteur

✍ **Algorithme de la procédure Supprimer**

```
0) DEF PROC SUPPRIMER (N :ENTIER; VAR
   T:TAB)
1) Pour i de 1 à N faire
   Ch←T[i]
   Pour j de 1 à long(ch) faire
     Si Non(Majus(ch[j])dans["A".."Z"])
      Alors
        Efface(ch,j,1)
     Fin si
   Fin pour
Fin Pour
2) Fin Supprimer
```

ch	chaine	Chaine auxiliaire

✍ **Analyse de la procédure Afficher**

```
DEF PROC AFFICHER(N :ENTIER ; T:TAB)
Résultat :Trait
Trait=[] Pour i de 1 à n faire
     Si FN Palindrome(T[i]) alors
          Ecrire(T[i])
     Fin Si
Fin pour
Fin Afficher
```

✍ **Algorithme de la procédure Afficher**

```
0) DEF PROC AFFICHER(N:ENTIER; T:TAB)
1) Pour i de 1 à n faire
     Si FN Palindrome(T[i]) alors
          Ecrire(T[i])
     Fin Si
Fin pour
2) Fin Afficher
```

T.D.O Locaux

Objet	T/N	Rôle
I	Entier	Compteur
Palindrome	Fonction	Vérifier si une chaine est palindrome ou non.

✍ **Analyse de la Fonction Palindrome**

```
DEF FN Palindrome (ch :chaine):BOOLEEN
Résultat : Palindrome ← ch=ch1
Ch1 =[ch1←""]
Pour i de long(ch) à 1(pas=-1) faire
Ch1←ch1+ch[i]
Fin pour
i:compteur
Fin Palindrome
```

✍ **Algorithme de la Fonction Palindrome**

```
0) DEF Palindrome (ch :chaine):BOOLEEN
1) Ch1 =[ch1←""]
Pour i de long(ch) à 1(pas=-1) faire
     Ch1←ch1+ch[i]
Fin pour
2) Palindrome← ch=ch1
3) Fin Palindrome
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur
Ch1	chaine	Déterminer l'inverse de la chaine ch

✍ **Traduction Pascal**

```
PROGRAM Chaine_Palindrome;
USES wincrt ;
TYPE TAB=Array[1..30] of string;
VAR
   N:integer ;
   T :TAB ;
PROCEDURE Saisie(VAR N :integer);
BEGIN
 repeat
Write('Donner la taille de tableau:');
Readln(N) ;
```

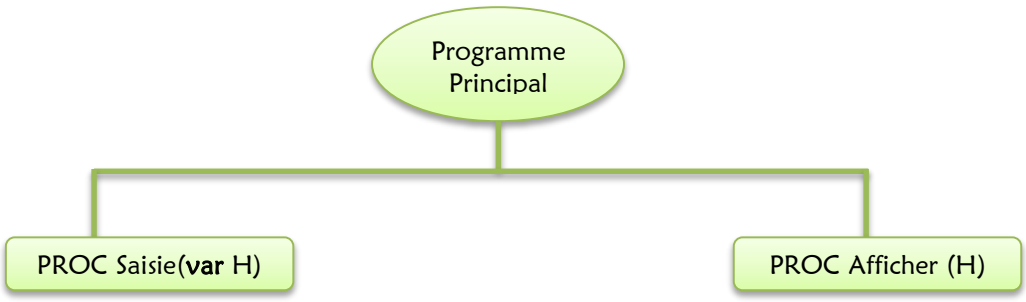
```
PROCEDURE Supprimer(n:integer ; VAR
T:TAB) ;
VAR i,j :integer ; ch :string ;
BEGIN
for i :=1 to n do
begin
ch :=T[i] ;
for j :=1 to length(ch) do
     if NOT(Upcase(Ch[j]) in ['A'.. 'Z'])
     then
          delete(ch,j,1) ;
```

```
 until(N>=2)and(N<=30);
END;
PROCEDURE Remplir(n:integer;VAR
T:TAB);
VAR   i :integer ;
BEGIN
for i :=1 to n do
begin
  repeat
     Write('T[',i,']= ');
     Readln(T[i]) ;
  Until length(T[i])<>0 ;
end;
END;
PROCEDURE Majuscule(N :integer ; VAR
T:TAB);
VAR   i,j:integer;  ch :string ;
BEGIN
for i :=1 to n do
   Ch:=T[i];
   for j :=1 to length(ch) do
    ch[j]:=Upcase(ch[j]);
END;
```

```
      end;
END;

PROCEDURE Afficher (n:integer; T:TAB);
VAR   i :integer ;
FUNCTION Palindrome(ch :string):boolean;
VAR   i :integer ; ch1 :string ;
BEGIN
      Ch1:='';
      for i := length(ch) downto 1 do
         ch1:= ch1+ ch[i];
      palindrome := ch=ch1;
END;
BEGIN
      For i:=1 to N do
        If Palindrome(T[i]) then
           Writeln(T[i]);
END;
BEGIN
      Saisie(N);
      Remplir(N,T);
      Majuscule(N,T);
      Supprimer(N,T);
      Afficher (N,T);
END.
```

Exercice 39

✍ **Décomposition modulaire du problème**



✍ **Analyse de programme principal :**

Nom de programme : Triangle
Résultat: PROC Afficher(H)
PROC Saisie(H)
Fin Triangle

✍ **Algorithme de programme principal:**

0) *Début* Triangle
1) PROC Saisie(H)
2) PROC Afficher(H)
3) *Fin* Triangle

T.D.O Globaux

Objet	T/N	Rôle
H	entier	stocker la hauteur d'une chaine
Saisie	Procédure	Saisir la hauteur d'une chaine
Afficher	Procédure	Un triangle isocèle d'hauteur H

✍ **Analyse de la procédure Saisie**

```
DEF PROC SAISIE(VAR H :ENTIER)
Résultat : H
H=[]Répéter
 H=donnée("Donner la hauteur :  ")
jusqu'à H dans [5..10]
Fin Saisie
```

✍ **Algorithme de la procédure Saisie**

```
0) DEF PROC SAISIE(VAR H :ENTIER)
1) Répéter
     écrire("Donner la hauteur : ")
     lire(H)
     jusqu'à H dans [5..10]
2) Fin Saisie
```

✍ **Analyse de la procédure Afficher**

```
DEF PROC AFFICHER(H :ENTIER)
Résultat : Trait
     Trait=[L←2*H-1]
Pour i de 1 à 2*H-1 faire
     Pour j de 1 à L faire
          Si j<=i-1 alors
              Écrire(" ")
          Sinon
              Écrire("*")
          Fin Si
     Fin Pour
     L←L-1
Fin Pour
i : compteur
Fin Afficher
```

✍ **Algorithme de la procédure Afficher**

```
0) DEF PROC AFFICHER(H :ENTIER)
1) [L←2*H-1]Pour i de 1 à 2*H-1 faire
     Pour j de 1 à L faire
          Si j<=i-1 alors
              Écrire(" ")
          Sinon
              Écrire("*")
          Fin Si
     Fin Pour
     L←L-1
Fin Pour
2) Fin Afficher
```

T.D.O Locaux

Objet	T/N	Rôle
i,j	Entier	Compteurs
L	Entier	Variable auxiliaire

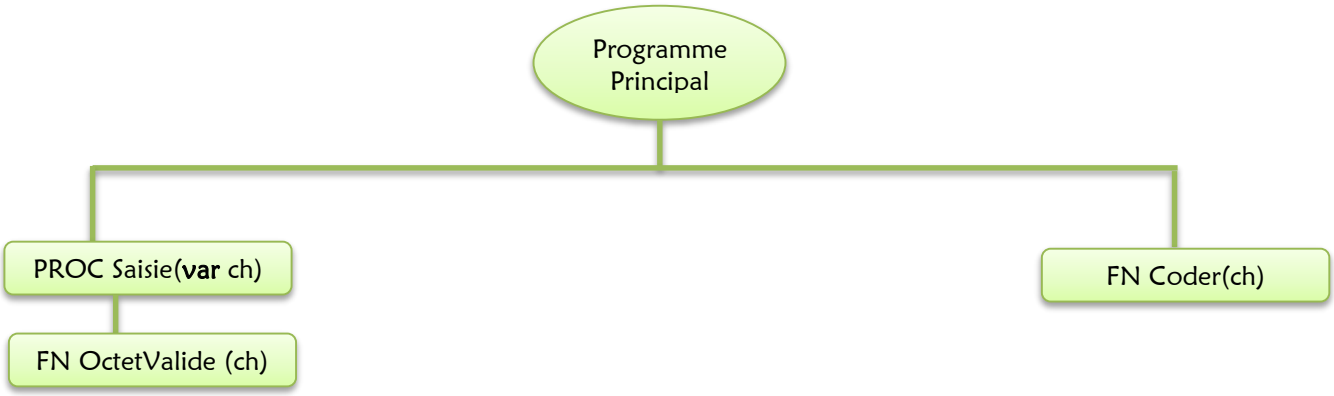
✍ **Traduction Pascal**

```
PROGRAM Triangle ;
USES wincrt ;
VAR H :integer ;
PROCEDURE Afficher(H :integer) ;
VAR    i,j,L :integer;
BEGIN
L :=2*H-1 ;
for i:=1 to 2*H-1 do
begin
      for j:=1 to L do
      begin
            if j<=i-1 then
                  write(' ')
            else
                  write('*');
      End;
      L:=L-1;
      Writeln;
end;
END;
```

```
PROCEDURE Saisie(VAR H :integer) ;
BEGIN
 REPEAT
    Write('Donner la hauteur: ');
    Readln(H) ;
 UNTIL H in [5..10] ;
END;
BEGIN
    Saisie(H) ;
    Afficher(H) ;
END.
```

Exercice 40

✍ **Décomposition modulaire du problème**



✍ **Analyse de programme principal :**

Nom de programme : Codec
Résultat:
écrire("Octet résultat : ",Res)
Res← Fn Coder(ch)
PROC Saisie(ch)
Fin Codec

✍ **Algorithme de programme principal:**

0) *Début* Codec
1) PROC Saisie(ch)
2) **Res**← Fn Coder(ch)
3) écrire("Octet résultat : ",Res)
4) *Fin* Codec

T.D.O Globaux

Objet	T/N	Rôle
Ch	Chaine de caractère	stocker un octet
Saisie	Procédure	Saisir un octet
Coder	Fonction	compresser un octet

✍ **Analyse de la procédure Saisie**

```
DEF PROC SAISIE(VAR CH:chaine de caractères)
Résultat : ch
ch=[]Répéter
 ch=donnée("Donner un octet :  ")
jusqu'à (long(ch) =8) et
        (FN OctetValide(ch))
Fin Saisie
```

✍ **Algorithme de la procédure Saisie**

```
0) DEF PROC SAISIE(VAR ch:chaine
   de caractères)
1) Répéter
   écrire("Donner un octet : ")
   lire(ch)
  jusqu'à (long(ch) =8) et (FN
OctetValide(ch))
2) Fin Saisie
```

T.D.O Locaux

Objet	T/N	Rôle
OctetValide	Fonction	Tester si une chaine donnée est composée seulement par les chiffres 0 et 1 ou non

✍ **Analyse de la fonction OctetValide**

```
DEF FN OCTETVALIDE(CH :chaine de caractères) :BOOLEEN
Résultat : OctetValide←test
Test=[test←vrai,i←1]
Répéter
  Si Non(Ch[i] dans["0".."1"])
    Alors
          Test← faux
  Fin Si
i←i+1
jusqu'à (test=faux) ou (i>long(ch))
Fin OctetValide
```

✍ **Algorithme de la fonction OctetValide**

```
0) DEF FN OCTETVALIDE(CH :chaine de
   caractères) :BOOLEEN
1) [test←vrai,i←1]
   Répéter
  Si Non(Ch[i] dans["0".."1"])
     Alors
           Test← faux
  Fin Si
   i←i+1
   jusqu'à (test=faux) ou
   (i>long(ch))
2) OctetValide←test
3) Fin OctetValide
```

T.D.O Locaux

Objet	T/N	Rôle
i	Entier	Compteur
Test	booléen	Tester si une chaine donnée est composée seulement par les chiffres 0 et 1 ou non

Analyse de la fonction Coder

```
DEF FN CODER(CH :chaine de caractères) : chaine de caractères
Résultat : Coder← ch1
[ch1←"",i←1]
répéter
     j←i+1
     Tant que (CH[i]=CH[j]) et (j<=long(ch)) faire
          J←j+1
     Fin tant que
     si j-i≠1 alors
          Convch(j-1,b)
          Ch1←ch1+b+CH[i]
     Sinon
          Ch1←ch1+CH[i]
     Fin si
     i←j
Jusqu'à i>long(ch)
i :compteur
Fin Coder
```

Algorithme de la fonction Coder

```
0) DEF FN CODER(CH :chaine de
   caractères):chaine de caractères
1) [ch1←"",i←1]
   répéter
        j←i+1
        Tant que (CH[i]=CH[j]) et (j<=long(ch))
        faire
          J←j+1
        Fin tant que
        si j-i≠1 alors
             Convch(j-i,b)
             Ch1←ch1+b+CH[i]
        Sinon
             Ch1←ch1+CH[i]
        Fin si
        i←j
   Jusqu'à i>long(ch)
2) Coder ← ch1
3) Fin Coder
```

T.D.O Locaux

Objet	T/N	Rôle
i,j	Entier	Compteurs
B	chaine	Chaine auxiliaire
Ch1	chaine	Contient le résultat de compression

Traduction Pascal

```
PROGRAM Codec ;
USES wincrt ;
VAR Ch,res :string ;
PROCEDURE Saisie(VAR ch :string) ;
FUNCTION OctetValide(ch :string):Boolean;
VAR
   i :integer ;    test :boolean ;
BEGIN
    Test :=true ; i :=1 ;
Repeat
     If NOT(Ch[i] in['0'..'1']) then
         Test :=false ;
    i :=i+1 ;
```

```
FUNCTION Coder(ch:string) :String;
VAR i,j :integer ;
Ch1,b:string ;
BEGIN
   Ch1:=" ; i:=1 ;
Repeat
J :=i+1 ;
While (ch[i]=ch[j])and(j<=length(ch))do
   J :=j+1 ;
if j-i>1 then
begin
   str(j-i,b);
```

```
Until (test=false)OR(i>length(ch)) ;
 OctetValide :=test ;
END ;
BEGIN
 REPEAT
    Write('Donner un octet: ');
    Readln(ch) ;
 UNTIL (length(ch)=8) AND
       (OctetValide(ch)) ;
END;
```

```
   Ch1:=ch1+b+CH[i];
End
else
    Ch1:=ch1+CH[i];
i:=j;

Until i>length(ch);
Coder := ch1;
END ;
BEGIN
    Saisie(ch) ;
    Res :=  Coder(ch);
    write('Octet résultat : ',Res) ;

END.
```