Chapitre6:
Les traitements avancés

Leçon1: **`Méthodes de tri`**

I/ **Introduction**:

- Trier un tableau veut dire organiser ou ordonner ses éléments selon un critère fixé à priori.
- Le tri peut être croissant ou décroissant
- On peut par exemple, lorsqu'on traite deux tableaux chacun de dimension n l'un contient les noms de n élèves et l'autre leurs moyennes:
  - Classer les élèves par ordre alphabétique c'est-à-dire trier n noms du premier tableau dans l'ordre alphabétique croissant ou décroissant.
  - Classer les élèves par ordre de mérite c'est-à-dire trier n réels du deuxième tableau dans l'ordre croissant ou décroissant.
- Dans le domaine informatique, il existe plusieurs algorithmes de tri. Dans cette leçon, nous allons découvrir trois algorithmes les plus utilisés dans le traitement de tri. Il s'agit des algorithmes de:
  - Tri par *sélection*
  - Tri à *bulles*
  - Tri par *insertion*

Pour chaque méthode, on s'intéresse à:
- Son principe
- Une illustration à travers un exemple
- L'analyse du problème, la décomposition en modules, l'élaboration des algorithmes et la traduction en pascal.

Activité1:
On se donne un tableau T de n entiers, et on se propose de les trier par ordre croissant.
Ainsi, Pour n=9

Le tableau T

14	2	47	10	18	13	5

Deviendra

2	5	10	13	14	18	47

II/ **Les méthodes de tri**:

Dans la suite de notre cours, on va utiliser le tri par ordre croissant c'est-à-dire du plus petit au plus grand.

1) **Le tri par sélection:**

a) Principe:

Le principe de tri par sélection consiste à:

1. Sélectionner (trouver) le plus petit élément du tableau (trouver son indice)
2. Le placer à sa position finale (c'est-à-dire à la première position du tableau)
3. Rechercher le second plus petit élément dans le reste du tableau (de T [2] à T[n])
4. Le placer à sa position finale (en seconde position à la place de T [2])
5. Répéter ce traitement jusqu'à ce que la liste soit triée

b) Exemple:

Considérons le tableau T de l'activité1contenant les 7 éléments suivants:

T =

14	2	47	10	18	13	5
1	2	3	4	5	6	7

Etape1: **-** chercher la valeur minimale dans T ($1 \rightarrow 7$): cette valeur étant 2 et son indice est 2

14	2	47	10	18	13	5
1	2	3	4	5	6	7

- permuter cette case avec la case n°1

2	14	47	10	18	13	5
1	2	3	4	5	6	7

Etape2: - chercher la valeur minimale dans T ($2 \rightarrow 7$): cette valeur étant 5 et son indice est 7

2	14	47	10	18	13	5
1	2	3	4	5	6	7

- permuter cette case avec la case n°2

2	5	47	10	18	13	14
1	2	3	4	5	6	7

Partie triée *partie non triée*

Etape3: - chercher la valeur minimale dans T ($3 \rightarrow 7$): cette valeur étant 10 et son indice est 4

2	5	47	10	18	13	14
1	2	3	4	5	6	7

- permuter cette case avec la case n°3

2	5	10	47	18	13	14
1	2	3	4	5	6	7

Etape4: - chercher la valeur minimale dans T (4 → 7): cette valeur étant 13 et son indice est 6

2	5	10	47	18	13	14
1	2	3	4	5	6	7

- permuter cette case avec la case n°4

2	5	10	13	18	47	14
1	2	3	4	5	6	7

Etape5: - chercher la valeur minimale dans T (5 → 7): cette valeur étant 14 et son indice est 7

2	5	10	13	18	47	14
1	2	3	4	5	6	7

- permuter cette case avec la case n°5

2	5	10	13	14	47	18
1	2	3	4	5	6	7

Etape6: - chercher la valeur minimale dans T (6 → 7): cette valeur étant 18 et son indice est 7

2	5	10	13	14	47	18
1	2	3	4	5	6	7

- permuter cette case avec la case n°6

2	5	10	13	14	18	47
1	2	3	4	5	6	7

Remarque:

- Nous n'avons pas besoin de traiter le dernier élément du tableau, puisque si les 6 premiers éléments sont ordonnés alors automatiquement le dernier sera le plus grand et par conséquent il se trouve à sa bonne position.
- Nous allons utiliser pour ces traitements:
  - une *fonction* qui va retourner l'indice du plus petit élément dans le tableau traité. Le minimum peut être répété plusieurs fois, dans ce cas on va décider le premier.
  - L'échange de deux éléments d'un tableau va être assurée par une *procédure* de permutation.

c)Analyse du problème:

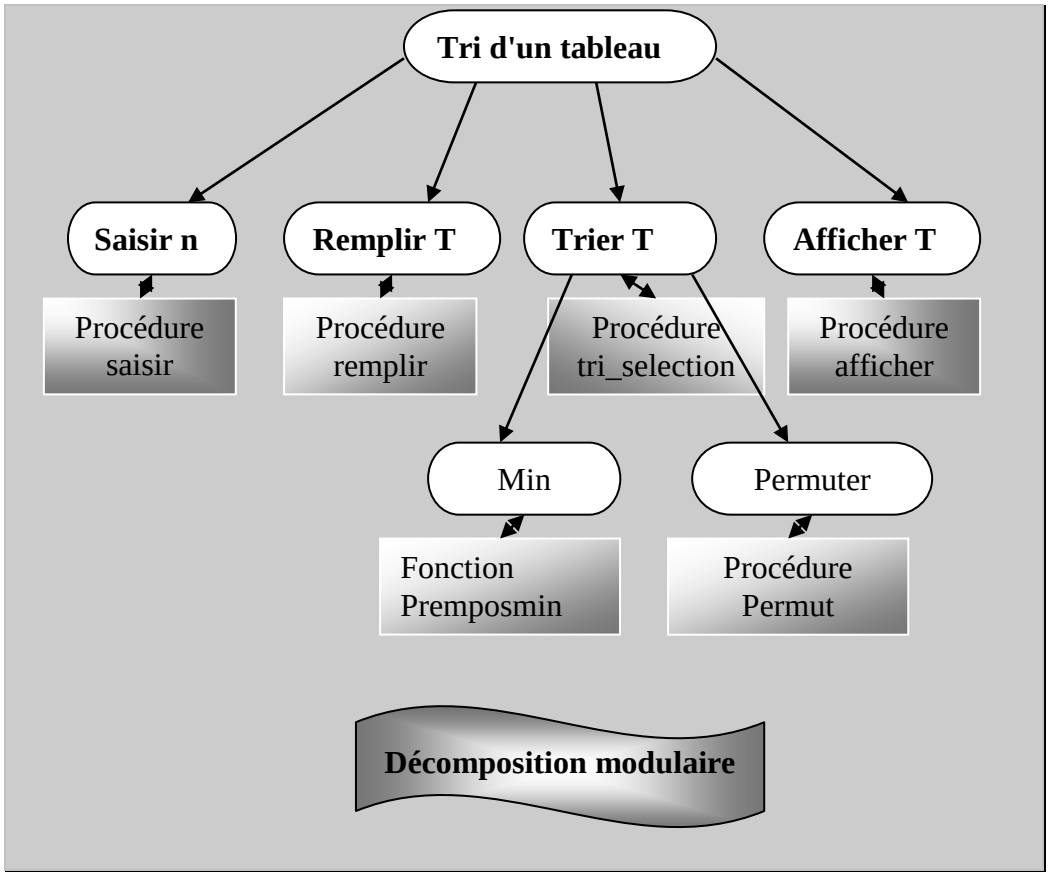
On utilisera l'approche de l'analyse modulaire qui consiste à décomposer le problème en modules.

♣ ***Pré analyse et décomposition en modules du programme principal:***

Résultat: afficher le tableau trié

Traitements:

- L'affichage du tableau trié est la tâche de la procédure **afficher**
- On doit donc trier le tableau ce qui sera la tâche de la procédure **tri_selection.** Ce module fera appel à deux modules qui sont:
  - --La fonction **premposmin** qui retourne l'indice du plus petit élément du tableau
  - --La procédure **permut** qui, dans le cas du besoin, réalisera l'action de permutation.
- Pour trier le tableau, il faut tout d'abord le remplir ce qui a été accordé à la procédure **remplir**
- Pour remplir le tableau nous avons besoin de connaitre le nombre d'éléments à saisir ce qui est la tâche de la procédure **saisir**.



Dans ce qui suit, nous allons détailler l'analyse de la procédure tri_selection et on va considérer T de type TAB.

♣ ***Analyse de la procédure tri_séléction:***

DEF PROC tri_selection (n:entier; var T:tab)

Résultat = t

T=[] pour i de 1 à n-1 faire

Ppm ← FN posmin (t,n,i)

Si (t[ppm] <> t[i]) alors

Proc permut (t[ppm],t[i])

Fin si Fin pour

**** Tableau de déclaration d'un nouveau type:***

Type
Tab= tableau de 20 entiers

**** Tableau de déclaration des objets de la procédure tri_selection:***

Objet	Nature / type	Rôle
i	Entier	Compteur
ppm	Entier	Première position du min
posmin	Fonction	Renvoie 1ère position du min
permut	Procedure	Permute le contenu de 2 variables

♣

**** Analyse de la fonction posmin:***

DEF FN posmin (t :tab ; n,m :entier) : entier
Resultat : posmin ← pm
Pm=[pm←m] pour j de m+1 à n faire
Si t[pm] > **t[j] alors**
pm ←j
fin si
fin pour

**** Tableau de déclaration des objets de la :***

Objet	Nature / type	Rôle
pm j	Entier Entier	Contient la position du min compteur

♣

- ***Analyse de la procedure permut:***

DEF PROC permut (var x,y :entier)
Resultat : (x,y)
z←x
x ←y
y ←z

**** Tableau de déclaration des objets de la :***

Objet	Nature / type	Rôle
z	Entier	Variable intemediaire

d)Algorithmes du problème:

♣ **Algorithme du programme principal:**

0) début tri_tableau
1) saisir (n)
2) remplir (t,n)
3) tri_selection (n,t)

4) afficher (t,n)
5) fin tri_ tableau

♣ **Algorithme de la procédure tri_selection:**

0) DEF PROC tri_selection (n:entier; var T:tab)
1) pour i de 1 à n-1 répéter
[ppm ← FN posmin (T,n,i)] Si T[ppm] ≠ T[i] alors
PROC permut (T[ppm], T[i])
Fin si
Fin pour
2) Fin PROC tri_selection

♣ **Algorithme de la fonction posmin:**

0) **DEF FN** posmin (A:tab; n,m:entier):entier

1) [pm ← m] pour j de m+1 à n répéter

Si A[pm] > A[j] alors

Pm ← j

Fin si

Fin pour

2) posmin ← pm

3) **fin** posmin

♣ **Algorithme de la procedure permut:**

0) DEF PROC permut (var x,y:entier)

1) aux ← x

2) x ← y

3) y ← aux

4) fin permut

e) Traduction en pascal:

2) **Le tri par insertion:**

a) Principe:

Le tri par insertion est le tri le plus efficace sur les listes de petite taille. C'est pourquoi il est utilisé par d'autres méthodes de tri.

Le principe de ce tri est simple: c'est le tri que toute personne utilise quand elle a des dossiers à classer. On prend un dossier et on le met à sa place parmi les dossiers déjà triés. Puis on recommence avec le dossier suivant.

Le principe général est le suivant:

1. considérer que les (i-1) premiers éléments de la liste sont triés et placer le $i^{ème}$ élément à sa place parmi les (i-1) déjà triés.
2. répéter cette action jusqu'à atteindre la fin de la liste.
3. l'action d'insertion se traduit par:
   - utiliser une variable intermédiaire tmp pour conserver la valeur à insérer,
   - déplacer les éléments T [i-1], T [i-2], …vers la droite tant que leur valeur est supérieure à celle de tmp.
   - Affecter alors à l'emplacement dans le tableau laissé libre par ce décalage la valeur tmp.

b) Exemple:

Considérons le tableau T de l'activité1contenant les 7 éléments suivants:

T =

2	14	4	3	18	13	5
1	2	3	4	5	6	7

On commence par l'élément n°2 puisque si un tableau contient un seul élément, il est considéré comme trié.

Etape1: **-** chercher la position d'insertion de l'élément n°2 dans la première partie du tableau T avec souci de la garder triée.

Puisque 14>2, donc les deux premiers éléments sont déjà en ordre.

2	14	4	3	18	13	5
1	2	3	4	5	6	7

Etape2: - chercher la position d'insertion de l'élément n°3 dans la première partie du tableau T avec le souci de la garder triée.

- On affecte à tmp la valeur de l'élément n°3
- On décale d'un cran à droite l'élément n°2, ...jusqu'à avoir un élément inférieur à 4
- Affecter à la dernière case décalée la valeur de tmp

2	4	14	3	18	13	5
1	2	3	4	5	6	7

tmp | **4** |

Partie triée *partie non triée*

Etape3: - chercher la position d'insertion de l'élément n°4 dans la première partie du tableau T avec le souci de la garder triée.

- On affecte à tmp la valeur de l'élément n°4
- On décale d'un cran à droite l'élément n°3, … jusqu'à avoir un élément inférieur à 3
- Affecter à la dernière case décalée la valeur de tmp.

2	3	4	14	18	13	5
1	2	3	4	5	6	7

tmp | **3**

Etape4: - chercher la position de l'élément n°5 dans la première partie du tableau T avec souci de la garder triée.
Puisque 18 est supérieure à tous les éléments de la partie triée donc elle doit rester à sa position initiale.

2	3	4	14	18	13	5
1	2	3	4	5	6	7

tmp | **18**

Etape5: - chercher la position d'insertion de l'élément n°6 dans la première partie du tableau T avec le souci de la garder triée.
- On affecte à tmp la valeur de l'élément n°6
- On décale d'un cran à droite l'élément n°5, ...jusqu'à avoir un élément inférieur à 13
- Affecter à la dernière case décalée la valeur de tmp

2	3	4	13	14	18	5
1	2	3	4	5	6	7

tmp | **13**

Etape6: - chercher la position d'insertion de l'élément n°7 dans la première partie du tableau T avec le souci de la garder triée.
- On affecte à tmp la valeur de l'élément n°7
- On décale d'un cran à droite l'élément n°6, ...jusqu'à avoir un élément inférieur à 5
- Affecter à la dernière case décalée la valeur de tmp

2	3	4	5	13	14	18
1	2	3	4	5	6	7

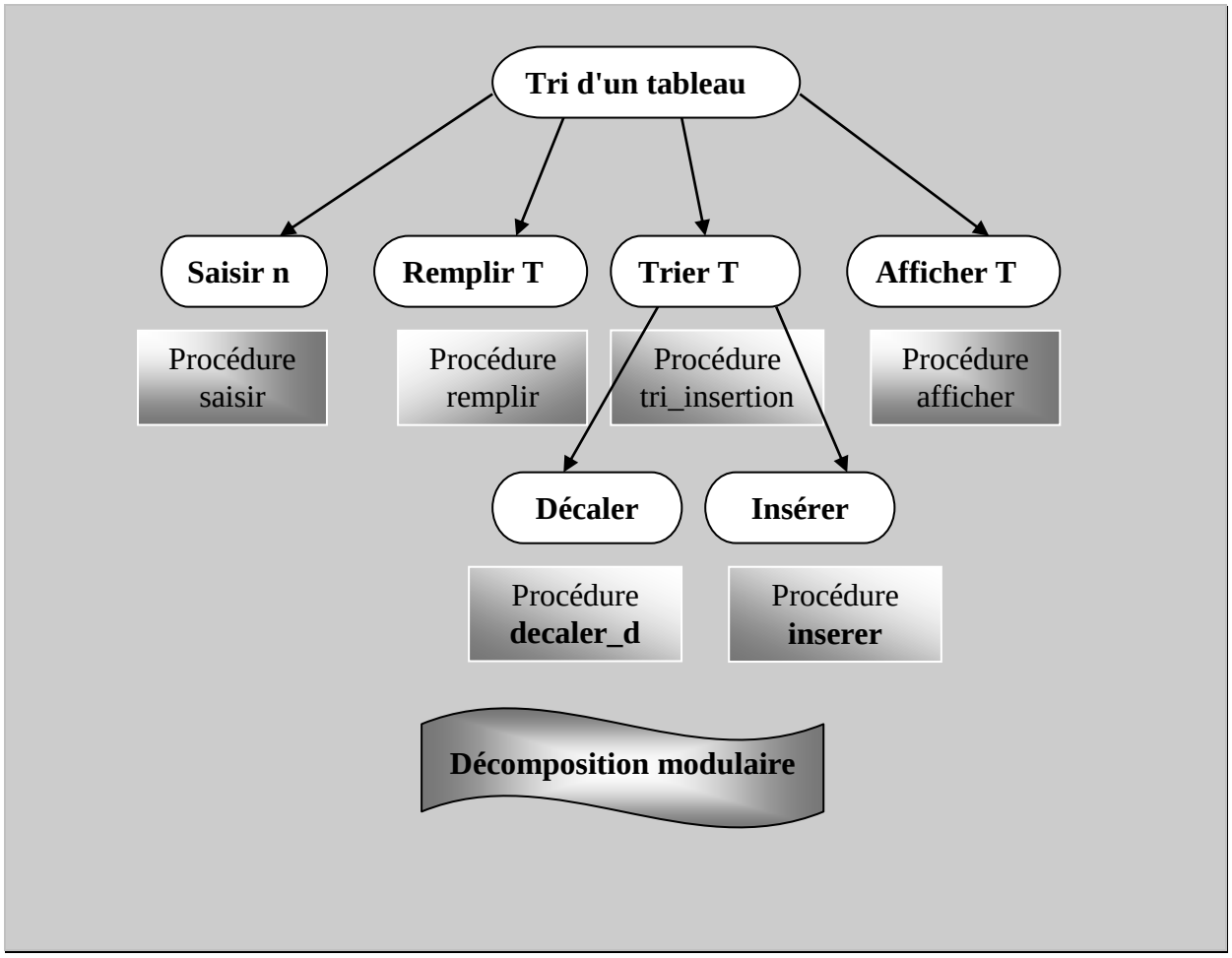
tmp | **5**

c)Analyse du problème:

♣ ***Pré analyse et décomposition en modules du programme principal:***

Résultat: afficher le tableau trié
Traitements:
- L'affichage du tableau trié est la tâche de la procédure **afficher**
- On doit donc trier le tableau ce qui sera la tâche de la procédure **tri_insertion.** Ce module fera appel à deux modules qui sont:
  --La procedure **decaler_d**
  --La procédure **inserer**
- Pour trier le tableau, il faut tout d'abord le remplir ce qui a été accordé à la procédure **remplir**

- Pour remplir le tableau nous avons besoin de connaitre le nombre d'éléments à saisir ce qui est la tâche de la procédure **saisir**.



♣ ***Analyse de la procédure tri_insertion:***

DEF PROC tri_insertion (n:entier; var T:tab)
Resultat : t
T=[] pour i de 2 à n faire
V ← t[i]
J ← i
Proc decaler (t,j,v)
Proc inserer (t,j,v)
Fin pour

**** Tableau de déclaration d'un nouveau type:***

Type
Tab= tableau de 20 entiers

**** Tableau de déclaration des objets de la procédure tri_insertion:***

Objet	Nature / type	Rôle
I	Entier	Compteur
V	Entier	Element à inserer
J	Entier	Position de l'element à inserer
Decaler	Procedure	Decale les elements d'un tableau à droite
inserer	procedure	Insere un element dans un tableau dans une position bien définie

♣ ***Analyse de la procédure décaler :***

DEF PROC decaler (var t :tab ; p,e :entier)
Resultat= t
T=[] tant que (t[p-1] > e) faire
T[p] ← t[p-1]
p ← p-1
Fin tant que

♣ ***Analyse de la procédure inserer***
DEF PROC inserer (var t :tab ; p,e :entier)
Resultat= t
T[p] ← e ***:***

d) Algorithmes:

♣ **Algorithme de la procédure tri_insertion:**

3) **Le tri à bulles:**

a) Principe:

Le tri à bulles est appelé aussi le tri par propagation, Son principe consiste à:

1) Comparer le 1er et le 2ème élément et les échanger (ou permuter) s'ils sont désordonnés (tenir compte de cette action)
2) Refaire l'action précédente jusqu'à l'avant dernière place
3) Refaire les deux actions précédentes jusqu'à ne pas faire d'échanges au dernier passage.

Autrement dit:

Comparer chaque élément du tableau avec son suivant et permuter si l'ordre n'est pas correct; après chaque permutation on reprend à partir du premier élément du tableau. Le tri s'arrête lorsqu'il n'y a plus aucune permutation.
Chaque élément est déplacé vers le début du tableau à la manière d'une bulle qui remonte à la surface.

b) Exemple:

Considérons le tableau T de l'activité1contenant les 5 éléments suivants:

T =

3	1	9	0	3

On se propose d'utiliser la méthode de tri à bulles pour trier T en ordre croissant.
Soit une variable booléenne **echange** qu'on initialise à faux et qui devient vrai à chaque fois que nous réalisons une permutation

echange | faux |

PREMIER PASSAGE

Etape1: - On se pointe à la 1ère case du tableau et on compare T [1] et T [2]
- puisque 3>1, on les permute

T=

1	3	9	0	3
1	2	3	4	5

echange | vrai |

- ***Etape2***: - on compare T [2] et T [3]
  - puisqu'ils sont dans le bon ordre on ne fait rien

T=

1	3	9	0	3
1	2	3	4	5

echange | vrai |

- ***Etape3***: - on compare T [3] et T [4]
  - puisque 9>0, on les permute

T=

1	3	0	9	3
1	2	3	4	5

echange | vrai |

- ***Etape4***: - on compare T [4] et T [5]
  - puisque 9>3, on les permute

T=

1	2	3	4	5
1	3	0	3	9

echange | vrai |

Puisqu'on a atteint la fin du tableau et la variable **echange** a comme valeur vrai alors on recommence un nouveau passage jusqu'à ce qu'on fasse un passage complet du tableau sans effectuer aucune permutation.

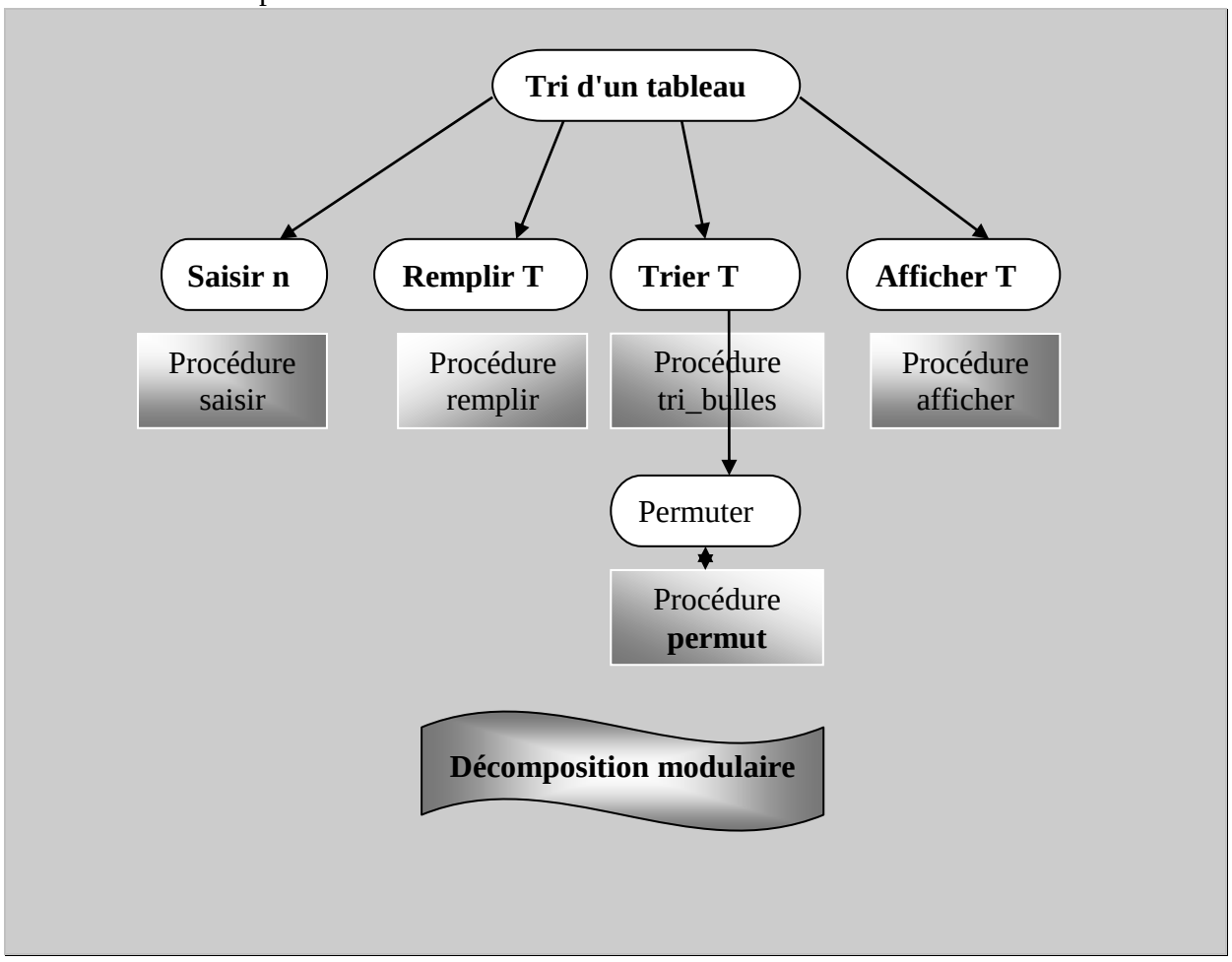
c)Analyse du problème:

4) ***Pré analyse et décomposition en modules du programme principal:***

Résultat: afficher le tableau trié

Traitements:

- L'affichage du tableau trié est la tâche de la procédure **afficher**
- On doit donc trier le tableau ce qui sera la tâche de la procédure **tri_insertion.** Ce module fera appel à un module qui est:
  --La procedure **permut**
- Pour trier le tableau, il faut tout d'abord le remplir ce qui a été accordé à la procédure **remplir**
- Pour remplir le tableau nous avons besoin de connaitre le nombre d'éléments à saisir ce qui est la tâche de la procédure **saisir**.



♣ ***Analyse de la procédure tri_bulles:***

DEF PROC tri_bulles (n:entier; var T:tab)

Resultat= t

T= [] répéter
Echange ← faux
Pour i de 1 à n-1 faire
Si (t[i] > t[i+1]) alors
Proc permut (t[i], t[i+1])
Echange ← vrai
Fin si
Fin pour
n ← n-1
jusqu'à (n=1) ou non (echange)

**** Tableau de déclaration d'un nouveau type:***

Type
Tab= tableau de 20 entiers

**** Tableau de déclaration des objets de la procédure tri_bulles:***

Objet	Nature / type	Rôle
Echange	Booléen	Reçoit la valeur vrai si permutation a eu lieu
I	Entier	Compteur
T	Tab	Tableau d'entiers
Permut	procédure	Permute le contenu de deux variables

Remarque:

La procédure **permut** est déjà traitée précédemment.

d) Algorithmes:

♣ **Algorithme de la procédure tri_bulles:**