Objectifs : Ce TP a pour but de compter les nombres d'affectations (a_n) et de comparaisons (c_n) d'éléments d'un tableau pour trier un tableau et de voir comment ces nombres évoluent avec la taille n du tableau et avec l'algorithme utilisé.

Les algorithmes de tri étudiés seront

- le tri par sélection ;
- le tri par insertion ;
- le tri par fusion.

Matériel requis : Pour faire ce TP, il est nécessaire

- d'avoir programmé les trois tris. Il sera supposé dans la suite que vous avez déjà écrit, en suivant les algorithmes décrits en cours , les trois procédures

```
// trie le tableau t
// algo = tri par sélection
procedure trier_select(var t : TABLEAU);

// trie le tableau t
// algo = tri par insertion
procedure trier_insert(var t : TABLEAU);

// trie le tableau t
// algo = tri par fusion
procedure trier_fusion_rec(var t : TABLEAU);
```

  Vous pouvez d'ailleurs démarrer ce TP en utilisant votre fichier `tris.pas`

- de disposer du logiciel (libre) GNUPLOT (ce qui est le cas dans les salles de TP).

1 Compter les affectations et les comparaisons

On souhaite connaître le nombre d'affectations et de comparaisons d'éléments du tableau à trier, éléments de type `ELEMENT`. On veut donc compter le nombre de fois qu'une instruction de la forme `x := y;` est exécutée, et le nombre de fois qu'une expression de la forme `x <= y` est évaluée (`x` et `y` étant des variable ou expressions de type `ELEMENT`).

Pour cela,

1. on déclare deux variables globales qui doivent être visibles de toutes les procédures et fonctions concernées (on les déclare donc avant ces procédures et fonctions) ;

```
var nb_aff  : CARDINAL; // pour compter les affectations
    nb_comp : CARDINAL; // pour compter les comparaisons
```

2. ensuite, dans toutes ces procédures et fonctions, on ajoute l'instruction

```
inc(nb_aff);
```

   après chaque affectation qui nous intéresse, et on ajoute l'instruction

```
inc(nb_comp);
```

   après (ou avant)toute comparaison ;

3. enfin, le programme initialisera à 0 les deux compteurs avant de faire appel à un tri. Par exemple, la procédure suivante effectue le tri par sélection d'un tableau de n éléments initialisé avec des valeurs choisies au hasard entre 0 et `VAL_MAX` (constante préalablement définie et fixée à 20000 par exemple), puis affiche les nombres d'affectations et de comparaisons pour trier ce tableau.

```
procedure evaluer_tri_select_1 ;
var n : CARDINAL;
    t : TABLEAU;
begin
   write('Nombre d''éléments du tableau : ') ; readln(n);
   // une seule des trois lignes suivantes ne doit pas être en commentaire
   t := genereTableau(n,VAL_MAX) ;
   // t := genereTableauTrie(n,VAL_MAX) ;
   // t := genereTableauAntiTrie(n,VAL_MAX) ;
   nb_aff := 0 ; nb_comp := 0 ;
   trier_select(t) ;
   writeln(nb_aff:10,nb_comp:10) ;
end. {evaluer_tri_select_1} ;
```

Exercice 1 : Adaptez vos procédures et fonctions et tester la procédure ci-dessus. (Le programme principal sera réduit à un simple appel à cette procédure, tout le reste, tests des tris etc. est mis en commentaire)

Q 1 . Que vaut le nombre c_n lorsque $n = 10$?, $n = 20$?, $n = 30$? Les valeurs obtenues dépendent–elles du contenu du tableau ?

Q 2 . Que vaut a_n pour $n = 10$, 20 et 30 ? Les valeurs obtenues dépendent–elles du contenu du tableau ?

Exercice 2 : Modifiez la procédure précédente pour calculer a_n et c_n avec un tableau déjà trié, puis avec un tableau trié dans l'ordre inverse.
(On utilise les procédures `genereTableauTrie` et `genereTableauAntiTrie` de l'unité `U_TABLEAU`).

Exercice 3 : Écrivez deux procédures analogues pour le tri par insertion, puis pour le tri par fusion, puis refaites les deux exercices précédents pour ces autres algorithmes de tri.

2 Fabrication de tables de mesures

On va maintenant construire des tables avec les valeurs des nombres a_n et c_n pour n compris entre 2 et une constante `TAILLE_MAX` que l'on fixera à 512.

```
const TAILLE_MAX = 512;
```

Cette table sera produite par un procédure qui triera des tableaux de taille n comprise entre 2 et `TAILLE_MAX` et affichera pour chacune de ces tailles les trois nombres n, a_n et c_n séparés par des espaces et sur une seule ligne (instruction `writeln(n:3,aff:10,comp:10);`).

Voici un extrait de l'affichage produit

```
  2         3         1
  3         6         3
  4         9         6

  ...

510      1527    129795
511      1530    130305
512      1533    130816
```

Exercice 4 : Réalisez une procédure sans paramètre nommée `evaluer_tri_select` pour le tri par sélection de tableaux remplis d'entiers choisis aléatoirement entre 0 et `VAL_MAX` (constante préalablement définie et fixée à 20000 par exemple). Comme pour la procédure `evaluer_tri_select_1`, vous mettrez les trois "générations" possibles de tableau sachant qu'une seule n'est pas en commentaire.
(Comme précédemment, dans le programme principal, tout sera mis en commentaires avec `(*  *)` ou `{  }`, sauf bien sûr l'appel à cette procédure `evaluer_tri_select`)
L'exécution de votre programme donne alors l'affichage de cette table dans le terminal (pas très lisible ni exploitable).

Afin de pouvoir exploiter ultérieurement les nombres a_n et c_n ainsi calculés, il est intéressant de *rediriger* l'affichage produit par ce dernier programme dans un fichier.

Exercice 5 : Après avoir testé votre programme dans un terminal, si votre programme s'appelle `evaluer_tris` tapez dans un terminal la commande

```
....$ ./evaluer_tris  >tri_select_qcque.txt
```

(`....$` est le *prompt* vous ne l'écrivez pas ($\ddot\smile$))

où `tri_select_qcque.txt` sera le nom du fichier dans lequel les nombres seront écrits.

Ouvrez ensuite ce fichier ainsi réalisé avec un éditeur de textes (KATE par exemple), et assurez-vous que le nombre de lignes est égal à la valeur de la constante `TAILLE_MAX`, et que chacune d'elles contient trois nombres espacés comme souhaité.

Exercice 6 : Créez de la même façon les fichiers

1. `tri_select_trie.txt`
2. `tri_select_antitrie.txt`
3. `tri_insert_qcque.txt`
4. `tri_insert_trie.txt`
5. `tri_insert_antitrie.txt`
6. `tri_fusion_qcque.txt`
7. `tri_fusion_trie.txt`
8. `tri_fusion_antitrie.txt`

3 Représentations graphiques

Il est possible de faire une représentation graphique des tables numériques ainsi construites. Un programme utile pour cela est le programme GNUPLOT.

GNUPLOT est accessible dans les salles de TP. Lorsqu'on lance l'exécution de cette application, il est alors possible d'écrire des commandes afin de produire des graphiques. Nous allons voir quelques unes de ces commandes.

Pour lancer cette application, il suffit de taper dans un terminal la commande

```
....$ gnuplot
```

Faites-le en étant dans le répertoire où se trouvent vos fichiers `.txt` créés précédemment.

Vous entrez alors dans un nouvel interprêteur de commande et le *prompt* devient `gnuplot>`
(pour quitter cet interprêteur il suffit de taper la commande `quit`)

3.1 Représentation graphique de données dans un fichier

La commande permettant de visualiser l'évolution de a_n en fonction de n pour le tri par sélection de tableaux quelconques est

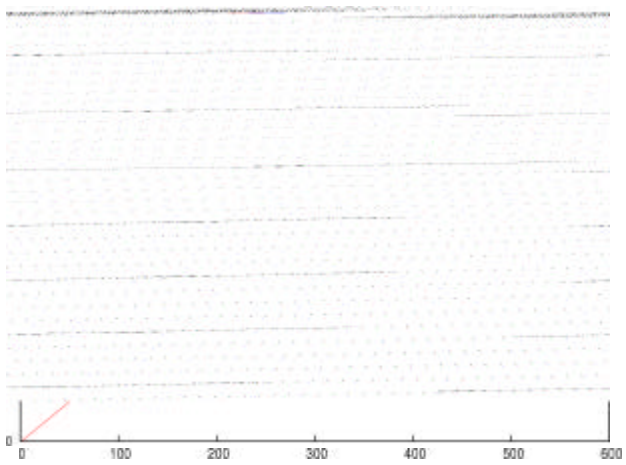
```
plot 'tri_select_qcque.txt' using 1:2 \
     title 'tri select, affectations, tableaux qcques' with lines
```

dont voici quelques explications

- `plot` est la commande de production de graphique ;
- `'tri_select_qcque.txt'` est le fichier dans lequel se trouvent les données à afficher (notez les ' ' entourant le nom du fichier) ;
- `using 1:2` indique que les points servant à construire le graphique ont leur abscisse dans la première colonne, et leur ordonnée dans la deuxième ;
- `title '...'` est la légende apportée au graphique ;
- `with lines` précise que le tracé doit être continu.

(notez l'utilisation du caractère `\` pour pouvoir écrire la commande sur plusieurs lignes, à chaque début de ligne apparaît le signe `>`)

Cette commande donne le graphique illustré par la figure suivante



Exercice 7 : Produisez le graphique du nombre de comparaisons dans le tri par sélection pour des tableaux quelconques.

Nous allons voir maintenant comment superposer et sauvegarder des graphiques pour pouvoir faire des comparaisons entre les courbes

3.2 Superposition de graphiques et sauvegarde

Il est possible de superposer plusieurs courbes sur un même graphique. Voici un exemple de commande superposant la courbe des a_n du tri par insertion pour des tableaux quelconques et des tableaux triés dans l'ordre inverse

```
plot 'tri_insert_qcque.txt' using 1:3 \
     title 'Tri insert, comparaisons, tab qcques' with lines,\
     'tri_insert_anti-trie.txt' using 1:3 \
     title 'Tri insert, comparaisons, tab anti-tries' with lines
```

On peut en suivant ce schéma superposer autant de courbes que l'on veut.

On peut aussi sauvegarder un tel graphique dans un fichier au lieu de l'afficher à l'écran. Il suffit d'effectuer préalablement à la commande qui produit le graphique, les deux commandes suivantes :

```
set terminal <format>
set output <fichier>
```

où

- `<format>` est le format de sauvegarde de l'image. Il peut être (entre autres) : *png*, *eps*, *svg*, ...
- `<fichier>` est le nom du fichier de sauvegarde.

Voici un exemple complet de sauvegarde dans un fichier au format *png* nommé `tri_select_aff_qcque.png`.

```
set terminal png
set output 'tri_select_aff_qcque.png'
plot 'tri_select_qcque.txt' using 1:2 \
     title 'Tri select, affectations, tableaux qcques' with lines
```

Exercice 8 : Comparaison en fonction du type de tableau
Q 1 . Superposez (et sauvegardez) les courbes des a_n pour les trois types de tableaux avec le tri par sélection.
Q 2 . Idem avec le tri par insertion.
Q 3 . Idem avec le tri par fusion.

Exercice 9 : Même exercice avec les courbes des c_n.

Exercice 10 : Enfin pour comparer entre eux les trois algorithmes de tris
Q 1 . Superposez (et sauvegardez) les courbes des a_n pour les trois tris avec des tableaux quelconques.
Q 2 . Idem avec des tableaux triés.
Q 3 . Idem avec des tableaux anti-triés

Exercice 11 : Même exercice avec les courbes des c_n.