IFT 1179 : Programmation en C#

A) Fichier de type texte :

Jusqu'au premier TP, on est habitude à déclarer et initialiser le contenu d'un tableau (tableau classique ou un tableau d'objets). Cette manière est souvent purement pédagogique afin d'avancer dans certaines matières de base au début d'une session. En pratique, il est très rare qu'on a seulement quelques données. Très souvent, on lit les données dans un ou dans plusieurs fichiers, mémorise ces données dans une structure de données (tableau d'objets, liste linéaire chaînée d'objets, ...). Quand la structure est en mémoire, on la manipule (afficher, rechercher, calculer les statistiques, trier, . . .). Avant de quitter, des fois on met à jour les fichiers de données en recopiant la structure modifiée ou on crée de nouveaux fichiers selon les besoins.

En pratique, un fichier de type texte est un document électronique qui se trouve sur un support magnétique dont le contenu est des lignes de texte

Contenu non aligné : exemple, le fichier Metro.txt

```
PAPINEAU(31)2740237

VIAU(34)2617979

UNIVERSITE-DE-MONTREAL(38)2610922

LANGELIER(37)2547151
```

Chaque ligne contient les informations d'une station de métro.

Contenu aligné : C’est le cas le plus fréquent des dichiers de type texte. Exemples :

Metrique.txt : informations des personnes

```
ROY CHANTAL                F      1.63          54.9        2754

MOLAISON CLAUDE            M      1.57          62.2        1848

BEDARD MARC-ANDRE          M      1.43          80.5        2636

MONAST STEPHANE            M      1.65          61.7        1750

JALBERT LYNE               F      1.63          52.6        2168

DUBE FRANCOISE             F      1.68          67.5        4612

ROBITAILLE SUZANNE         F      1.72          65.4        2325
```

Chaque champ d’information se trouve entre deux colonnes précises (format fixe).

Pays.A05 informations des pays du monde

```
2ETATS-UNIS                WASHINGTON         9629047   291289535

3CHINE                     PEKIN              9596960   1273111290

5RUSSIE                    MOSCOU             17075400  143954573

4AUSTRALIE                 CANBERRA           7686850   19834248
```

etc . . .

On peut créer un fichier de type texte en utilisant un éditeur de texte, la programmation, un logiciel, ...

On verra comment travailler avec la programmation en C#.

On utilise : `using System.IO;`

1) Création d'un fichier de type texte

a) Déclarer, nommer et ouvrir pour écrire son contenu

```
FileInfo fichier = new FileInfo(nomFichier);
StreamWriter aCreer = fichier.CreateText();
```

b) écrire son contenu : souvent dans une boucle

```
aCreer.Write( . . .) ;  ou aCreer.WriteLine( . . .);
```

c) fermer le fichier :

```
aCreer.Close();
```

2) Lecture d'un fichier existant :

a) déclarer et localiser le fichier à lire :

```
StreamReader aLire = File.OpenText(nomFichier);
```

b) lire et traiter son contenu, ligne par ligne :

```
string ligneLue = null;
while ( (ligneLue = aLire.ReadLine()) != null)
          traiter la ligne lue
```

c) fermer le fichier :

```
aLire.Close();
```

3) Premier exemple simple :

Réalisez un programme C# afin de créer un fichier de type texte nommé Divi100.txt sur le disque réseau. Chaque ligne de ce fichier est un diviseur de 100. On relit le fichier créé et affiche son contenu.

```
/* Fichier1.cs (1er exemple sur fichier de type texte)
 */

using System;
using System.IO;
```

```
class Fichier1
{
      // créer un fichier des diviseurs d'un nombre donné
      static void CreerFichier(int nombre, string nomFichier)
      {
            /* nommer le fichier et ouvrir pour écrire
             * son contenu
             */

            FileInfo fichier = new FileInfo(nomFichier);
            StreamWriter aCreer = fichier.CreateText();

            for (int candi = 1; candi <= nombre; candi++)
                  if (nombre % candi == 0)
                        // écrire dans le fichier
                         aCreer.WriteLine("{0, 5:D}", candi);

            // fermer le fichier
            aCreer.Close();
            Console.WriteLine("Fin de la creation du fichier texte " +
                                        nomFichier);
      }

      static void RelireFichier(string nomFichier)
      {
            Console.WriteLine("\nOn lit le fichier texte " + nomFichier
                   + " venant d'etre cree");
            // localiser et ouvrir our lire
            StreamReader aLire = File.OpenText(nomFichier);
            string ligneLue = null;

            // lire ligne par ligne
            while ( (ligneLue = aLire.ReadLine()) != null)
                  Console.WriteLine(ligneLue);
                  // traiter (ici, l'afficher seulement)

            // fermer le fichier
            aLire.Close();
            Console.WriteLine("\nFin de la lecture du fichier texte " +
                                         nomFichier);
      }

      static void Main(string[] args)
      {
            // créer le fichier des diviseurs de 100
            CreerFichier(100, "R:\\ Divi100.txt");
            RelireFichier("R:\\ Divi100.txt");
      }

}
```

```
/* Exécution:
 Fin de la creation du fichier texte R:\IFT1179\divi100.txt

On lit le fichier texte R:\IFT1179\divi100.txt venant d'etre cree
    1
    2
    4
    5
   10
   20
   25
   50
  100

Fin de la lecture du fichier texte R:\IFT1179\divi100.txt
Press any key to continue
*/
```

4) Fichier + tableau d'objets :

On dispose du fichier de type texte nommé "Metro.txt. Chaque ligne contient 3 informations d'une station de métro : son nom, son rang en 2003 et sa fréquence en 2004. Ces 3 champs sont séparés par (et) :

PAPINEAU(31)2740237

VIAU(34)2617979

UNIVERSITE-DE-MONTREAL(38)2610922

LANGELIER(37)2547151

On a au maximum 70 stations dans le réseau de transports à Montréal.

Réaliser un programme en C# permettant de :

- lire le fichier Metro.txt, remplir le tableau des stations, compter le nombre effectif de stations lues
- trier et afficher le contenu partiel du tableau :

7 premieres et 4 dernieres stations AVANT le tri
3 premieres et 5 dernieres stations APRES le tri

```
using System;
using System.IO;

class Station
{
	private string nom; // nom de la station
	private int rang2003, // rang en 2003
	freq2004; // nb. de passagers aux tourniquets en 2004

		// un constructeur possible
	public Station(string nom, int rang, int freq2004)
	{
		this.nom = nom;
		rang2003 = rang;
		this.freq2004 = freq2004;
	}

		// un autre constructeur possible
	public Station(string chaine)
	{
		char[] separateurs = new char[] {'(', ')'};
		string[] infosMetro = chaine.Split(separateurs, 3);

		nom = infosMetro[0];
		rang2003 = int.Parse(infosMetro[1]);
		freq2004 = int.Parse(infosMetro[2]);
	}

	// un constructeur sans paramètre
	public Station()
	{
	}

	// afficher les infos d'une station
	public void Afficher(string message)
	{
		Console.WriteLine(message + "{0, -35:S} {1,4:D} {2, 12:N0}",
		nom, rang2003, freq2004 );
	}


	public string Nom
	{
		get { return nom; }
		set { nom = value.ToUpper(); }
	}


}
```

```
class Fichier2
{

 static void LireRemplir(string nomFichier, Station[] stat, out int n)
      {
            n = 0; // compteur du nombre de dtations lues

            StreamReader aLire = File.OpenText(nomFichier);
            string ligneLue = null;
            while ( (ligneLue = aLire.ReadLine()) != null)
                  stat[n++] = new Station(ligneLue);

            aLire.Close();
            Console.WriteLine("\nOn vient de creer un tableau de " + n +
                                          " stations");
      }

      static void Permuter(ref Station a, ref Station b)
      {
            Station tempo = a;
            a = b;
            b = tempo;
      }

      static void Trier(Station[] stat, int nbStat)
      {
            for (int i = 0; i < nbStat-1 ; i++)
            {
                  int indMin = i;
                  for (int j = i+1; j < nbStat; j++)
                        if (stat[j].Nom.CompareTo(stat[indMin].Nom) < 0)
                                    indMin = j;

                  if (indMin != i)
                          Permuter(ref stat[i], ref stat[indMin]);
            }

      }

      static void Afficher(Station[] stat, int nbStat, int debut,
                                             int fin, string message)
      {
            Console.WriteLine("\nAffichage partiel du tableau des" +
                        " stations " + message);
            for (int i = 0; i < nbStat; i++)
            if (i < debut || i >= nbStat - fin)
            {
               Console.Write("{0,6:D})  ", i);
               stat[i].Afficher("");
            }
            else if (i == debut+1)
               Console.WriteLine("etc ....  ");

       Console.WriteLine();
      }
```

```
      static void Main(string[] args)
      {
         const int MAX_STAT = 70; // au maximum, 70 stations de métro
       Station[] stat = new Station[MAX_STAT];
         int nbStat;

         LireRemplir("R:\\metro.txt", stat, out nbStat);
         Afficher(stat, nbStat, 7, 4, "avant le tri");

         Trier(stat, nbStat);
         Afficher(stat, nbStat, 3, 5, "apres le tri");
      }
}
/* Exécution:

On vient de creer un tableau de 65 stations

Affichage partiel du tableau des stations avant le tri
     0)  MC-GILL                                1   11 333 531
     1)  BERRI-UQAM                             2   11 067 519
     2)  LONGUEUIL-UNIVERSITE-DE-SHERBROOKE     5    7 375 439
     3)  GUY-CONCORDIA                          4    7 077 669
     4)  ATWATER                                6    6 535 564
     5)  COTE-VERTU                             9    6 387 345
     6)  PEEL                                   8    6 273 357
etc ....
    61)  ASSOMPTION                            62      973 051
    62)  ACADIE                                63      936 535
    63)  GEORGES-VANIER                        64      695 482
    64)  DE LA SAVANE                          65      690 673


Affichage partiel du tableau des stations apres le tri
     0)  ACADIE                                63      936 535
     1)  ANGRIGNON                             15    4 493 057
     2)  ASSOMPTION                            62      973 051
etc ....
    60)  UNIVERSITE-DE-MONTREAL                38    2 610 922
    61)  VENDOME                               12    4 889 583
    62)  VERDUN                                51    1 529 707
    63)  VIAU                                  34    2 617 979
    64)  VILLA-MARIA                           33    2 753 590

Press any key to continue
*/
```

B) Recherche dans un tableau :

1) Recherche séquentielle :

a) Cas de tableau non trié selon la clé de recherche :

a.1 si la clé est unique (exemple NAS, . . .)

On parcours le tableau du début jusqu'à ce qu'on la trouve. Dans l'affirmative, on affiche les informations et quitte la boucle de recherche. Si c'est la fin du tableau, on ne trouve pas la clé recherchée.

a.2 si la clé n'est pas unique (exemple nom et prénom)

On parcours le tableau du début jusqu'à la fin du tableau et on affiche à chaque fois qu'on rencontre la clé.

a) Cas de tableau trié selon la clé de recherche :

L'importance est d'assurer de quitter dès qu'on dépasse la clé : on n'aura plus de chance de trouver la clé après (les valeurs sont de plus en plus grandes)

Exemple de la recherche séquentielle :

```
/* Fichier : Rec_Seq.cs  (qq méthodes de recherche séquentielle dans un
tableau)
 * À adapter selon vos besoins
 */

using System;
class Rec_Seq // la recherche séquentielle dans un tableau
{
      static void Chercher(int aChercher, int [] tableau , int nbElem)
      {
        for (int i = 0; i < nbElem; i++)
         if ( tableau[i] == aChercher )
            {
                  Console.WriteLine("On trouve {0} a l'indice {1}",
                          aChercher, i);
                  return ; // quitter la recherche car la clé est unique
            }
      Console.WriteLine("{0} n'est pas dans le tableau\n", aChercher);
}
```

```
static void Afficher(int [] tableau , int nbElem, string message)
{
      Console.WriteLine("Contenu du tableau des " + message + ": ");

      for (int i = 0; i < nbElem; i++)
            Console.WriteLine("{0, 3:D})   {1, 5:D}", i, tableau[i]);

            Console.WriteLine();
}

static void Demo1()
{
  Console.WriteLine("Demo 1 : cas de cle unique (ici, numeros " +
                        "d'employes) :\n");

  int[] numEmp = { 6521, 1234, 5555, 2233, 6666, 5432, 3478 };
  int   nbEmp = numEmp.Length;

  Afficher(numEmp, nbEmp, "numeros d'employes");
  Chercher(2233, numEmp, nbEmp);
  Chercher( 777, numEmp, nbEmp);
}

static void Chercher(int [] tableau , int nbElem, int aChercher)
{
  bool trouve = false;
  for (int i = 0; i < nbElem; i++)
      if ( tableau[i] == aChercher )
      {
        Console.WriteLine("On trouve {0} a l'indice {1}",
                    aChercher, i);
        trouve = true;
      }
      if (!trouve)
        Console.WriteLine("{0} n'est pas dans le tableau\n", aChercher);
}

static void Demo2()
{
    Console.WriteLine("Demo 2 : cas de cle non unique (ici, " +
       "consommation de cafe) :\n");

    int[] nbCafe = { 3, 0, 5, 3, 3, 1, 0, 2, 6 , 3};
    int   nbBuveurs = nbCafe.Length;

    Afficher(nbCafe, nbBuveurs, "tasses de cafe consomme par jour");
    Chercher(nbCafe, nbBuveurs, 3); // qui consomme 3 tasses par jour ?
    Chercher( nbCafe, nbBuveurs, 10); // 10 tasses par jour ???
    Chercher(nbCafe, nbBuveurs, 0); // qui consomme 0 tasse par jour ?
}
```

```
static void Chercher(int aChercher, int [] tableau , int nbElem, string
                                quand)
{
  Console.WriteLine("\nRecherche de " + aChercher + " dans un tableau "
                 + " trie:\n");

   for (int i = 0; i < nbElem && aChercher <= tableau[i]; i++)
   if ( tableau[i] == aChercher )
    {
      Console.WriteLine("On trouve {0} a l'indice {1}", aChercher, i);
      return ;
    }
   Console.WriteLine("{0} n'est pas dans le tableau\n", aChercher);

}

static void Demo3()
{
  Console.WriteLine("Demo 3 : Tableau trie, cas de cle unique  :\n");

  int[] numEmp = { 6521, 1234, 5555, 2233, 6666, 5432, 3478 };
  int   nbEmp = numEmp.Length;

  Afficher(numEmp, nbEmp, "numeros d'employes AVANT le tri");

  Array.Sort(numEmp); // à parler en classe

  Afficher(numEmp, nbEmp, "numeros d'employes APRES le tri");
  Chercher(2233, numEmp, nbEmp, "deja trie");
  Chercher(3333, numEmp, nbEmp, "deja trie");
}

  static void Main(string[] args)
    {
            Demo1();
            Demo2();
            Demo3();
    }

}



/* Exécution:
Demo 1 : cas de cle unique (ici, numeros d'employes) :

Contenu du tableau des numeros d'employes:
  0)    6521
  1)    1234
  2)    5555
  3)    2233
  4)    6666
  5)    5432
  6)    3478

On trouve 2233 a l'indice 3
777 n'est pas dans le tableau
```

```
Demo 2 : cas de cle non unique (ici, consommation de cafe) :

Contenu du tableau des tasses de cafe consomme par jour:
  0)       3
  1)       0
  2)       5
  3)       3
  4)       3
  5)       1
  6)       0
  7)       2
  8)       6
  9)       3

On trouve 3 a l'indice 0
On trouve 3 a l'indice 3
On trouve 3 a l'indice 4
On trouve 3 a l'indice 9
10 n'est pas dans le tableau

On trouve 0 a l'indice 1
On trouve 0 a l'indice 6
Demo 3 : Tableau trie, cas de cle unique  :

Contenu du tableau des numeros d'employes AVANT le tri:
  0)    6521
  1)    1234
  2)    5555
  3)    2233
  4)    6666
  5)    5432
  6)    3478

Contenu du tableau des numeros d'employes APRES le tri:
  0)    1234
  1)    2233
  2)    3478
  3)    5432
  4)    5555
  5)    6521
  6)    6666


Recherche de 2233 dans un tableau trie:

2233 n'est pas dans le tableau


Recherche de 3333 dans un tableau trie:

3333 n'est pas dans le tableau

Press any key to continue
*/
```

2) Recherche dichotomique dans un tableau trié :

Chaque tour de recherche, on compare la clé à celle du milieu du tableau :

```
Si clé < tableau[milieu].clé
   on examine dans la moitiée en haut
Sinon
   Si clé > tableau[milieu].clé
       on examine dans la moitiée en bas
   Sinon : on trouve la clé
```

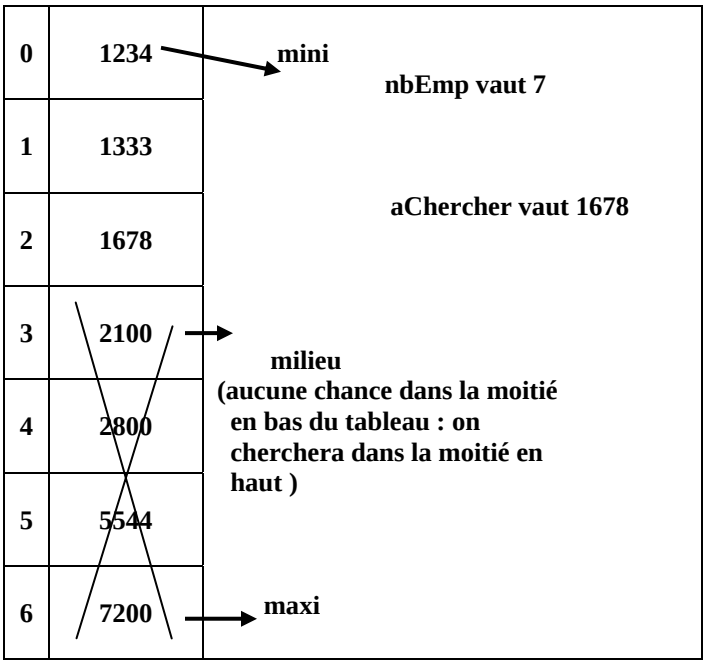
Questions :

- comment détecter si la clé n'existe pas
- comment procéder si la clé n'est pas unique ?

Soit le tableau trié suivant :

```
int[]  numEmp = { 1234, 1333, 1678, 2100, 2800, 5544, 7200 };
```

Schéma pour la recherche du numéro 1678 dans le tableau :



a) Une version possible pour programmer cette recherche :

Cette méthode détermine et retourne :
. l'indice de l'élément trouvé
. -1 si non trouvé

int Dicho1(int aChercher, int numero[], int nbEmp)

```
 {   int mini = 0, // 1er indice du tableau
         maxi = nbEmp-1, //dernier indice du tableau
         milieu; // indice au milieu du tableau

     bool trouve = false; /* pas encore recherché =>
                             pas encore trouvé */

     while (!trouve && mini <= maxi)
      {
         milieu = (mini + maxi) / 2;

         if (aChercher < numero[milieu])
             maxi = milieu - 1; /* première moitiée */
         else
           if (aChercher > numero[milieu])
             mini = milieu + 1; /* dernière moitié */
           else
             trouve = true; /* on l'a trouvé */
     }

           return trouve? milieu:-1;
}
```

b) Une autre version possible pour programmer cette recherche :

int Dicho1(int aChercher, int numero[], int nbEmp)

```
 {   int mini = 0, // 1er indice du tableau
         maxi = nbEmp-1; //dernier indice du tableau

     bool trouve = false; /* pas encore recherché =>
                             pas encore trouvé */

     while (!trouve && mini <= maxi)
      {
         int milieu = (mini + maxi) / 2;

         if (aChercher < numero[milieu])
             maxi = milieu - 1; /* première moitiée */
```

```
            else
              if (aChercher > numero[milieu])
                mini = milieu + 1; /* dernière moitié */
              else
                return milieu;
        }

        return -1;
    }
```

Question : Peut-on aussi enlever la variable booléenne trouve ?

c) Version récursive :

C'est un sujet possible d'une question d'un numéro du TP2.

C) Le tri d'un tableau :

1) Le tri par sélection:

Dans le premier cours de programmation, on a déjà enseigné une méthode de tri : le tri par sélection (chaque tour de boucle, on détermine l'indice de l'élément le plus petit associé : **indMin** puis on place l'élément à la bonne position.

```
static void Permuter(int[] tableau, int i, int indMin)
    { int tempo = tableau[i];
      tableau[i] = tableau[indMin];
      tableau[indMin] = tempo;
    }

static void Trier(int[] tableau, int nbElem)
{
            for(int i = 0; i < nbElem-1; i++)
             { int indMin = i;
               for(int j = i+1; j < nbElem; j++)
                       if (tableau[j] < tableau[indMin])
                               indMin = j;

              if (indMin != i)
                      Permuter(tableau, i, indMin);
             }
}
```

2) Le tri rapide (Quick Sort) :

Une des applications la plus célèbre de la récursivité est le tri rapide (Quick : rapide, Sort : trier).

Veuillez noter qu'il existe des livres sur le tri et la recherche (celui de Knuth avec plus de 700 pages réservées Uniquement pour le tri et la recherche). Il n'est pas important de connaître plusieurs. Il suffit de mieux connaître quelques unes. Dans l'exemple qui suit, on présente une version (sur plusieurs possibles) du tri rapide. En classe, on fait aussi la simulation.

Le programme suivant permet de :

1. créer, par l'ordinateur, un tableau dont
   - le nombre d'éléments est aléatoire entre 119 000 et 120 000
   - les valeurs sont des entiers avec valeurs varient entre 0 et 1 000 000
2. trier ce tableau avec QuickSort
3. afficher le contenu partiel du tableau avant et après le tri
4. afficher l'heure avant et après le tri pour constater la rapidité

```
using System;


public class TriRapide1
{
      // fournir un nombre aléatoire entre 2 bornes
      static int Aleatoire(Random arbitraire, int borne1, int borne2)
      {
            return (arbitraire.Next() % (borne2 - borne1) + borne1);
      }

      static void  CreerAleatoire(int[] tableau, int nbElem,
                              int MAX_VAL)
      {
            Random arbitraire = new Random();
            for (int i = 0; i < nbElem; i++)
                        tableau[i] = Aleatoire(arbitraire, 0, MAX_VAL);

      }
```

```
/* Afficher PARTIELLEMENT le contenu du tableau :
   les 7 ers indices de 0 à 6, les 4 derniers
   On affiche des entiers alignés
*/
static void  Afficher(int[] tableau, string message)
{
      Console.WriteLine("Contenu Partiel du tableau " + message);
      for (int i = 0; i < tableau.Length; i++)
            if (i <= 6 || i >= tableau.Length - 4)
            {
                  Console.Write("{0,6:D})  ", i);
                  Console.WriteLine("{0, 10:N0}", tableau[i]);

            }
            else if (i == 7)
                  Console.WriteLine("etc ....  ");

      Console.WriteLine();
}

/* Partitionner un tableau en deux sous tableaux :
- placer la valeur du pivot à sa position finale (après le tri)
- sa position finale est l'indice du pivot à retourner
- en Partionnant:
les valeurs à gauche de l'indice du pivot <= valeur du pivot
les valeurs à droite de l'indice du pivot >  valeur du pivot
vous n'avez pas besoin de comprendre techniquement cette fonction.
Il faut retenir surtout son but.
*/

/* J'écris mais je n'appelle pas :
Essayez de remplacer deux permutations dans Partitionner
par 2 appels de cette fonction et COMPARER le temps d'exécution.
*/
static void  Permuter(int[] T, int g, int d)
{
      int tempo = T[g];
      T[g] = T[d];
      T[d] = tempo;
}

static int Partitionner(int[] T, int debut, int fin)
{
      int g = debut, d = fin;
      int valPivot = T[debut];

      do
      {
      while (g <= d && T[g] <= valPivot)
            g++;
      while (T[d] > valPivot)
            d--;

            if (g < d) Permuter(T, g, d);
                        }
      while (g <= d);
```

```
            Permuter(T,debut, d);

            return d;
      }

      /* Cette fonction est récursive car dans son corps on appelle
      cette même fonction (2 fois)
      */
      static void  QuickSort(int[] T, int gauche, int droite)
      {
            int indPivot;
            if (droite > gauche)
                  /* au moins 2 éléments */
            {
                  indPivot = Partitionner(T, gauche, droite);
                  QuickSort(T, gauche, indPivot - 1);
                  QuickSort(T, indPivot + 1, droite);
            }
      }



      public static void  Main(string[] args)
      {
            const int MAX_ELEM = 120000,
                      MAX_VAL = 1000000;
            /* valeurs varient entre 0 et un million */

            Random arbitraire = new Random();
            int nbElem = Aleatoire(arbitraire, MAX_ELEM-1000, MAX_ELEM);

            int[] tableau = new int[nbElem];


            CreerAleatoire(tableau, nbElem, MAX_VAL);
            Afficher(tableau, "apres la creation");
            DateTime d = DateTime.Now;

            Console.WriteLine("Debut du tri de " + nbElem + " entiers "
                              + "(" + d.ToString("r") + ")");

            QuickSort(tableau, 0, nbElem - 1);

            d = DateTime.Now;

            Console.WriteLine("Fin du tri de " + nbElem + " entiers " +
                              "(" + d.ToString("r") + ")");

            Afficher(tableau, "\nApres le tri");
      }
}


/* Exécution :


Contenu Partiel du tableau apres la creation
```

```
     0)     888 253
     1)     838 104
     2)     994 325
     3)     770 988
     4)     247 727
     5)     338 539
     6)     342 247
etc ....
119249)     775 288
119250)     266 570
119251)     548 576
119252)     282 174

Debut du tri de 119253 entiers (Sat, 01 Oct 2005 09:29:08 GMT)
Fin du tri de 119253 entiers (Sat, 01 Oct 2005 09:29:08 GMT)
Contenu Partiel du tableau
Apres le tri
     0)           6
     1)          12
     2)          19
     3)          21
     4)          43
     5)          47
     6)          57
etc ....
119249)     999 980
119250)     999 992
119251)     999 992
119252)     999 995

Press any key to continue
*/
```

D) Le tri et la recherche avec System.Array :

À venir.