1. Organisation en modules – Tableau de réels

Ecrire une application qui permet de manipuler un tableau de réels. **Attention, votre programme doit être organisé en modules** (unités). Le programme doit :

1. Demander le nombre de valeurs du tableau
2. Saisir les valeurs
3. Afficher les valeurs
4. Trier les valeurs de manière croissante (cf. tri à bulles sur wikipédia - http://fr.wikipedia.org/wiki/Tri_à_bulles)
5. Afficher les valeurs triées
6. Calculer et afficher la médiane des valeurs (attention, la formule n'est pas la même selon que le nombre d'éléments est pair ou impair – Cf Wikipédia / Mode de Calcul - http://fr.wikipedia.org/wiki/Médiane_(centre)

Le programme principal devrait ressembler à ceci :

```
program Projet_Tableau_Reels;

//importation de l'unité UTableau
uses UTableau;

//variables
var n: integer; //taille du tableau
    t: TTableau; //variable tableau

begin
  //saisie taille du tableau
  write('Taille du tableau = '); readln(n);
  setLength(t,n);//allouer le tableau
  //>>> début coeur des calculs (appel des fonctions de UTableau)
  Saisie(t);//saisie des données du tableau
  Afficher(t);//affichage des valeurs du tableau
  Tri_bulles(t);//trier le tableau cf. http://fr.wikipedia.org/wiki/Tri_à_bulles
  Afficher(t);//afficher le tableau des valeurs triées
  write('Mediane = '); writeln(Mediane(t));//calcul et affichage de la médiane
  //<<< fin cœur des calculs
  finalize(t);//désallouer le tableau
  //blocage console
  readln;
end.
```

Toutes les procédures et fonctions relatives à la structure **TTableau** doivent être programmées dans une unité « UTableau » (dans le fichier UTableau.pas) que vous devez rajouter à votre projet. Voici la description de l'interface de l'unité :

```
unit UTableau;

interface

//type tableau de données à une dimension
TYPE TTableau = array of double;

//saisie des valeurs
procedure Saisie(var t: TTableau);

//affichage des valeurs
procedure Afficher(const t: TTableau);

//trier le tableau
procedure Tri_Bulles(var t: TTableau);

//médiane du tableau - on considère qu'il est déjà trié
function Mediane(const t: TTableau): double;

implementation

à vous de jouer ...
```

Vous aurez à programmer ces procédures et fonctions dans la partie implémentation de l'unité.

2. Collection de véhicules

Ecrire une application qui permet de gérer une collection de voitures. Dans le programme principal :

1. On doit demander le nombre de voitures
2. Saisir les caractéristiques de chaque voiture
3. Afficher les voitures avec leurs caractéristiques (marque, age, cv [*champs de la structure*] et taxe [*obtenue à l'aide d'une fonction appliquée à la structure*])
4. Trier les voitures selon leur taxe
5. Afficher de nouveau la liste des voitures triées selon leur taxe

Voici le programme principal de votre application :

```
program Projet_Tableau_Voitures;

//appel des unités associées au projet
uses UVoiture, UTableauVoiture;

//variables
```

```
var t: TTabVoiture;//var. tableau de voitures
    n: integer;//nombre de voitures à gérer
begin
 Write('Taille du tableau = '); readln(n);//demander le nombre de voitures
 SetLength(t,n);//initialiser la collection
 writeln('>> saisie <<');
 SaisieTab(t);
 writeln('>> affichage <<');
 AfficherTab(t);
 writeln('>> trier <<');
 TrierTab(t);
 writeln('>> affichage apres tri <<');
 AfficherTab(t);
 Finalize(t);//supprimer la collection
 readln;
end.
```

Structure voiture. La structure voiture doit être déclarée dans l'unité **UVoiture.pas**. Nous y déclarons et implémentons également les procédures et fonctions qui lui sont associées. Voici l'interface de l'unité.

```
unit UVoiture;

interface

TYPE
//*********************************
//structure voiture avec ses champs
TVoiture = record
    marque : shortstring;
    age : integer;
    cv : integer;
    end;

//*************************************************************
//*** procédures et fonctions manipulant la structure TVoiture ***

//saisie des champs d'une voiture (marque, age, cv)
procedure saisieVoiture(var v: TVoiture);

//calcul de la taxe associée à une voiture
function taxeVoiture(v: TVoiture): double;

//affichage des caractéristiques d'une voiture (marque, age, cv et taxe)
procedure affichageVoiture(v: TVoiture);
```

La fonction **taxe()** renvoie la taxe associée à une voiture (structure TVoiture). Elle est définie de la manière suivante :

```
//taxe d'une voiture
function  taxeVoiture(v: TVoiture): double;
begin
 result:= 30.0 * v.cv + 5.0 * v.age;
end;
```

Structure Collection de voitures. Le tableau de voiture est défini dans **UTableauVoiture.pas**, dont voici l'interface, on notera qu'elle fait appel au module UVoiture (uses UVoiture ;), ce qui permet de reconnaître la structure « TVoiture » dans cette unité.

```
unit UTableauVoiture;

interface

//importation du module UVoiture
//indispensable pour avoir accès à la structure TVoiture
USES UVoiture;

TYPE
//*****************************
//structure tableau de voiture
TTabVoiture = array of TVoiture;

//*************************************************************************
//*** procédures et fonctions de manipulation de la structure TTabVoiture ***

//saisie
procedure saisieTab(var t: TTabVoiture);

//affichage des voitures du tableau
procedure afficherTab(const t: TTabVoiture);

//trier selon la taxe()
procedure trierTab(var t: TTabVoiture);
```