Découpage des programmes en modules

Les « unités » sous DELPHI

Ricco Rakotomalala
Université Lumière Lyon 2

Pourquoi le découpage du programme en modules ?

(Rappel) Pourquoi les procédures et fonctions ? A mesure que la taille du programme augmente, (1) il faut une meilleure **organisation** pour en maintenir la lisibilité ; (2) des parties du code peuvent être **réutilisées** à plusieurs endroits.

L'étape suivante de l'**organisation** du code est le découpage de l'application en modules qui réunit des procédures et fonctions :

- Relatifs à la même thématique (ex. fonctions financières, etc.) ;
- Ou bien, qui manipulent le même type de données (ex. un type enregistrement, un type matrice, etc.).

Un module (une unité) en DELPHI est un fichier « .pas » avec une structure particulière. Il peut être partagé par différents projets, on va plus loin encore dans la **réutilisation** du code.

Structure d'une unité

Découpage en 2 zones : **interface** et **implementation**

```
unit UnitTableau; //sera stocké dans UnitTableau.pas

interface

//description des éléments destinés à être visibles à l'extérieur de l'unité c.-à-d.

//définition des types
//description des en-têtes de procédures et fonctions

//des variables placées ici sont visibles à l'extérieur (prog. Principal ou autre module)

implementation

//les variables placées ici sont visibles dans cette unité, mais pas à l'extérieur

//programmation des procédures et fonctions

end.
```

Un exemple d'unité

UnitTableau.pas

UnitTableau

```
unit UnitTableau;

interface

//définition du type vecteur
TYPE TVecteur = array of double;

//saisie des valeurs, en-tête
procedure saisie(var t: TVecteur; n: integer);

//moyenne des valeurs
function  moyenne(const t: TVecteur): double;

implementation

//programmation de saisie
procedure saisie(var t: TVecteur; n: integer);
var i: integer;
begin
 SetLength(t,n);
 for i:= 0 to n-1 do
  readln(t[i]);
end;

//programmation moyenne des valeurs
function  moyenne(const t: TVecteur): double;
var i: integer;
    s: double;
begin
 s:= 0.0;
 for i:= 0 to length(t)-1 do
  s:= s + t[i];
 result:= s/length(t);
end;

end.
```

33: 23 Insertion Code

Regrouper ici les définitions de type

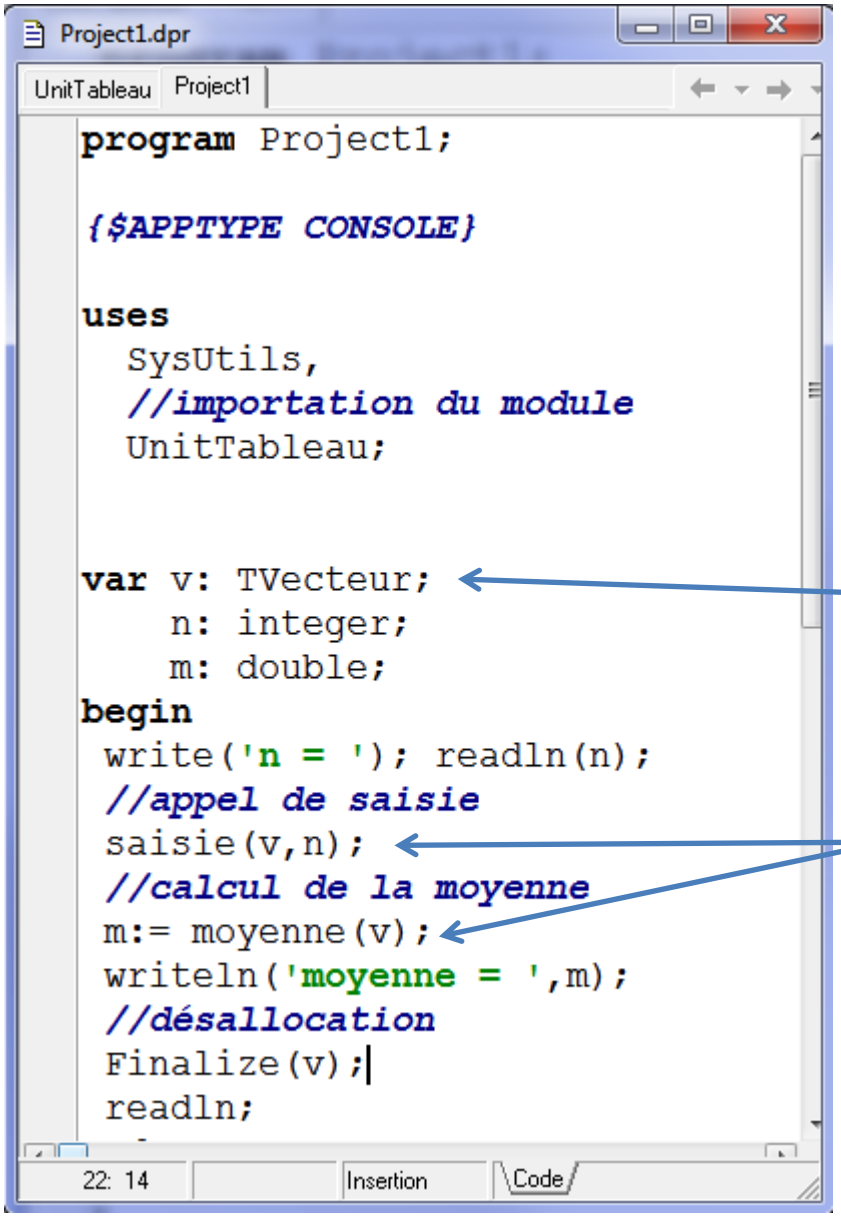
Décrire les en-têtes des procédures et fonctions destinées à être exportées de l'unité (visibles à l'extérieur de l'unité)

Reprendre chaque en-tête et programmer effectivement chaque procédure et fonction.

Il est possible de programmer ici des procédures et fonctions à usage interne au module. Elles sont utilisables à l'intérieur du module, mais ne seront pas visibles (non utilisables) à l'extérieur.

Importation d'une unité dans le programme principal :

Uses Nom_Unité;



Appel des unités. Si plusieurs, on les met à la suite en les séparant avec des « , »

Le type « Tvecteur » est reconnu parce qu'on a importé le module « UnitTableau »

On peut appeler directement les procédures et fonctions exportées de UnitTableau.

Ca ne se fait pas trop, mais il est possible de préfixer la fonction par le nom d'unité. Ex.
UnitTableau.saisie(v,n);
m:= UnitTableau.moyenne(v);

Appel entre unités

On peut faire appel aux procédures et fonctions d'un module dans un autre module.

UVecteur.pas

Project2Units | UVecteur | UStat

```
unit UVecteur;

interface

//tableau d'entiers
Type TVecteur = array of double;

//saisie
procedure saisie(var t: TVecteur);

//tri des valeurs du vecteur
procedure trier(var t: TVecteur);

implementation

//saisie
procedure saisie(var t: TVecteur);
begin
 //écriture du prog.
end;

//tri des valeurs
procedure trier(var t: TVecteur);
begin
 //écriture du prog.
end;

end.
```

Définition du type TVecteur et de qqs proc., dont **trier()**.

UStat.pas

Project2Units | UVecteur | UStat

```
unit UStat;

interface

//importation du module
uses UVecteur;

//moyenne des valeurs
function VecMoyenne(v: TVecteur): double;

//minimum
function VecMin(v: TVecteur): double;

implementation

//moyenne des valeurs
function VecMoyenne(v: TVecteur): double;
var s: double; i: integer;
begin
 s:= 0.0;
 for i:= 0 to length(v)-1 do s:= s + v[i];
 result:= s/length(v);
end;

//minimum
function VecMin(v: TVecteur): double;
begin
 //on trie les valeurs
 trier(v);
 //on renvoie la première valeur -> min.
 result:= v[0];
end;

end.
```

5: 14

Exploitation du type TVecteur et de la procédure **trier()**.

Le type TVecteur n'est pas visible s'il n'y avait pas le « uses »

La procédure **trier()** n'est pas utilisable s'il n'y avait pas le « uses ».

On aurait pu écrire UVecteur.trier(v);

FIN…

Les mêmes concepts sont – à peu de choses près – présents dans tous les langages de programmation…