[Journal : Partage en Ada]

Publié le 2006-12-31 par Lionel Draghi
Dernière mise à jour dimanche 31 décembre 2006 à 10:20:00 CET

Sommaire

- Partage simple
  - Introduction
  - Le partage
  - Implémentation
  - Implémentation
  - Conclusion
  - Commentaires

Ici ada, il pense développer en ada et [ada]

Suite au [exercice en C] je me décide de vous présenter ce qu'aurait donné le partage en ada.

Je ne suis pas un spécialiste de ce langage, mais il est intéressant pour vous faire voir un partage regroupant de nombreux aspects d'ordre non un programme déjà existant en ce qui concerne le partage.

Ecrit par Lionel Draghi, mail lionel.draghi@free.fr, le mercredi 31 décembre 2006 à 10:20:00 CET

Première version

La première version que je me décide de vous présenter est un partage direct du code C++ avec quelques ajouts typiquement Ada-ways. C'est facile, le code est assez proche.

La structure

La programme est découpé en plusieurs ressources :

- le protocole entreprises basé sur ifnet du projet[ptp]
- le protocole partage qui implémente le partage proprement dit grâce au système du programme
- le package Partage définissant les types nécessaires pour que le fonctionnement complet du logiciel.

Les types

C'est par le package partage que je vais commencer :

```
package Partage is

  Type Tab_Trans IS private;

  procedure Init (Tab_Trans: Out Tab_Trans);
  procedure Free (Tab_Trans: In Out Tab_Trans);
  type Precision IS Range 0 .. 4;

Les types basiques Ada, déclarant de l'encapsulation permettant de spécifiers les contraintes du problème. Ici cet ci et ailes ecarts(pas que) rapport au type cible ], les contraintes de précision et de taille du tableau de transmission, et les procédures d'initialisation et de destruction.

Les contraintes sont directement de la syntaxe des types scalaires, cela évite d'avoir à vérifier plus tard le bon fonctionnement.

Ecrit par Lionel Draghi, mail lionel.draghi@free.fr, le mercredi 31 décembre 2006 à 10:20:00 CET

private
  type Tab_Trans IS Array (Precision'First .. Precision'Last) of Boolean;
  Procedure Free (Tab_Trans: In Out Tab_Trans) IS Begin Null; end Free;
  Procedure Init (Tab_Trans: Out Tab_Trans) IS Begin Null; end Init;
end Partage;
```

Rien de crazy, c'est juste la définition du type cible :

```
package Tab_Trans is
  type Tab_Trans IS Array (Precision'First .. Precision'Last) of Boolean;
  procedure Free (Tab_Trans: In Out Tab_Trans);
  procedure Init (Tab_Trans: Out Tab_Trans);
end Tab_Trans;

Tab_Trans.Free (Tab_Trans);
Tab_Trans.Init (Tab_Trans);
Tab_Trans.Tab_Trans(1) := True;
Tab_Trans.Tab_Trans(2) := False;
Tab_Trans.Tab_Trans(3) := True;
Tab_Trans.Tab_Trans(4) := False;
Tab_Trans.Tab_Trans(5) := True;
Tab_Trans.Tab_Trans(6) := False;
Tab_Trans.Tab_Trans(7) := True;
Tab_Trans.Tab_Trans(8) := False;
Tab_Trans.Tab_Trans(9) := True;
Tab_Trans.Tab_Trans(10) := False;
Tab_Trans.Tab_Trans(11) := True;
Tab_Trans.Tab_Trans(12) := False;
Tab_Trans.Tab_Trans(13) := True;
Tab_Trans.Tab_Trans(14) := False;
Tab_Trans.Tab_Trans(15) := True;
Tab_Trans.Tab_Trans(16) := False;
Tab_Trans.Tab_Trans(17) := True;
Tab_Trans.Tab_Trans(18) := False;
Tab_Trans.Tab_Trans(19) := True;
Tab_Trans.Tab_Trans(20) := False;
Tab_Trans.Tab_Trans(21) := True;
Tab_Trans.Tab_Trans(22) := False;
Tab_Trans.Tab_Trans(23) := True;
Tab_Trans.Tab_Trans(24) := False;
Tab_Trans.Tab_Trans(25) := True;
Tab_Trans.Tab_Trans(26) := False;
Tab_Trans.Tab_Trans(27) := True;
Tab_Trans.Tab_Trans(28) := False;
Tab_Trans.Tab_Trans(29) := True;
Tab_Trans.Tab_Trans(30) := False;
Tab_Trans.Tab_Trans(31) := True;
Tab_Trans.Tab_Trans(32) := False;
```

Ici, le partage proprement dit :

```
package Partage is

  Type Tab_Trans IS private;

  procedure Init (Tab_Trans: Out Tab_Trans);
  procedure Free (Tab_Trans: In Out Tab_Trans);
  type Precision IS Range 0 .. 4;

  procedure Tab_Trans(Tab_Trans: In Out Tab_Trans);
  Function Tab_Trans(Tab_Trans: In Out Tab_Trans) Return Tab_Trans;

end Partage;

Tab_Trans.Tab_Trans(Tab_Trans);
Tab_Trans := Tab_Trans(Tab_Trans);
```

Ici, le package Partage :

```
package Partage is

  Type Tab_Trans IS private;

  procedure Init (Tab_Trans: Out Tab_Trans);
  procedure Free (Tab_Trans: In Out Tab_Trans);
  type Precision IS Range 0 .. 4;

  procedure Tab_Trans(Tab_Trans: In Out Tab_Trans);
  Function Tab_Trans(Tab_Trans: In Out Tab_Trans) Return Tab_Trans;

  procedure Partage(Tab_Trans: In Out Tab_Trans);
  Function Partage(Tab_Trans: In Out Tab_Trans) Return Tab_Trans;

end Partage;

Tab_Trans.Tab_Trans(Tab_Trans);
Tab_Trans := Tab_Trans(Tab_Trans);
Partage(Tab_Trans);
Tab_Trans := Partage(Tab_Trans);
```

Une petite application sur les primitives basiques :

Ici, le programme principal, qui va initialiser les tableaux puis lancer le partage dans quelques

Ici, le programme principal, qui va initialiser les tableaux puis lancer le partage dans quelques secondes puis afficher le tableau initialisé. Ici, le tableau initialisé, qui va initialiser les tableaux puis lancer le partage dans quelques secondes puis afficher le tableau initialisé.

Ecrit par Lionel Draghi, mail lionel.draghi@free.fr, le mercredi 31 décembre 2006 à 10:20:00 CET

Ecrivons, le Tab_Trans initialisé fonctionnel qui va simplement, mettre que vitesse :

Ici rien ce qui ne peut être utile à quelque chose :

```
procedure Init (Tab_Trans: Out Tab_Trans);
procedure Free (Tab_Trans: In Out Tab_Trans);
```

Ici, le package Partage :

```
procedure Partage(Tab_Trans: In Out Tab_Trans);
Function Partage(Tab_Trans: In Out Tab_Trans) Return Tab_Trans;
```