

Conversion des pointeurs d'enregistrement sous Delphi 8

par [Nono40](#)

Date de publication : 13/02/2004

Dernière mise à jour : 13/02/2004

Conversion des pointeurs d'enregistrement sous Delphi 8 en partant d'un exemple de conversion d'un source Delphi 7 en programme Delphi8.

Introduction

I - Remplacement du pointeur typé

II - Création d'un élément

III - Accès aux champs de l'enregistrement

IV - Libération d'un élément

V - Gestion des éléments d'une liste

VI - Cas des listes chaînées

Conclusion

Introduction

Les pointeurs ne sont plus d'actualité dans la plateforme .NET : l'adresse des objets est gérée par le JITer à l'exécution du programme et n'est pas forcément fixe.

La conversion va être présentée autour d'un exemple simple de code utilisant une liste de pointeurs. Le source Delphi 7 de cet exemple est ici : [Source 0004](#).

La méthode présentée ici, est une des solutions possibles. Elle permet de rester assez proche du source d'origine sans pour autant utiliser de code non managé. (voir la [FAQ DotNet](#) pour plus d'informations sur ce sujet)

I - Remplacement du pointeur typé

Mais sans utiliser les pointeurs, que faut-il utiliser ?

Une solution simple est de considérer qu'un objet est un pointeur caché. C'était vrai dans le langage objet de Delphi 7 et reste vrai dans Delphi 8. Le pointeur typé va donc être remplacé par une classe simple. Du même coup on n'utilisera plus de Record mais directement une classe.

Voici le code Delphi 7 de déclaration de l'enregistrement :

```
Type
PElementListe=^TElementListe;
TElementListe=Record
  elNom :String[20];
  elPoids :Integer;
  elTaille:Integer;
End;
```

En suivant ce principe décrit on utilisera la classe suivante dans le code Delphi 8 :

```
Type
TElementListe=Class(TObject)
Public
  elNom :String;
  elPoids :Integer;
  elTaille:Integer;
End;
```

Notez que la notion de pointeur n'est plus apparente, mais cela ne modifiera pas le code de façon importante.

Un autre point important dans le choix d'une classe, est que le TList n'est plus une liste de pointeurs mais une liste d'objets.

II - Création d'un élément

L'utilisation de **New** ne permet plus de réserver une zone mémoire pour un pointeur typé. D'autre part la déclaration d'un pointeur de type PElement sera toujours signalé code Unsafe et son utilisation interdite en code managé.

Avec la classe définie précédemment il est simple de remplacer le new. Le code suivant montre la création d'un élément sous Delphi 7 :

```
Var Ele:PElementListe;  
// Création du nouvel élément  
New(Ele);
```

En utilisant la classe, la création d'un élément se fait de la manière suivante :

```
Var Ele:TElementListe;  
// Création du nouvel élément  
Ele:=TElementListe.Create;
```

Le code d'ajout du nouvel élément dans la liste est le même dans les deux cas :

```
// Ajout de l'élément dans la liste  
Liste.Add(Ele);
```

III - Accès aux champs de l'enregistrement

L'utilisation de l'opérateur `^` est considérée comme "Unsafe". Le remplacement de l'enregistrement par une classe supprime l'utilisation de cet opérateur.

Remarque : L'opérateur `^` pouvait déjà être ignoré dans Delphi 7 avec la syntaxe étendue.

Pour accéder aux champs avec Delphi 7 :

```
Var Ele:PElementListe;  
With Ele^ Do  
Begin  
  elNom :=Form2.eNom.Text;  
  elTaille :=Form2.eTaille.Value;  
  elPoids :=Form2.ePoids.Value;  
End;
```

Pour accéder aux champs avec Delphi 7 :

```
Var Ele:TElementListe;  
With Ele Do  
Begin  
  elNom := Form2.eNom.Text;  
  elTaille := StrToIntDef(Form2.eTaille.Text,0);  
  elPoids := StrToIntDef(Form2.ePoids.Text,0);  
End;
```

Remarque : le remplacement de **Form2.eTaille.Value** par **StrToIntDef(Form2.eTaille.Text,0)** vient du fait que le SpinEdit n'était pas présent dans ma version de Delphi8 et n'a rien à voir avec la conversion présentée ici.

IV - Libération d'un élément

De même que New n'est plus utilisable, Dispose ne l'est plus non plus. Le remplacement de l'enregistrement par une classe permet aussi un remplacement simple de la procédure Dispose.

Dans Delphi 7 la suppression d'un élément est :

```
Var Ele:PElementListe;  
Dispose(Ele);
```

Dans Delphi 8 la suppression d'un élément devient :

```
Var Ele:TElementListe;  
Ele.Free;
```

V - Gestion des éléments d'une liste

La gestion des éléments dans la liste (de type TList) est la même dans les deux cas.

Les méthodes Add(), Delete(), etc utilisent les classes là où celles de Delphi 7 utilisaient les pointeurs.

Pour accéder à un élément de la liste il convient de transtyper l'élément pour pouvoir accéder à ses champs.

Dans le cas de Delphi 7 le transtypage pouvait être fait de deux manières : sur le pointeur ou sur l'enregistrement. Exemple :

```
// Transtypage de l'enregistrement
With TElementList(Liste.Items[i]^) Do...
// Transtypage du pointeur
With PElementList(Liste.Items[i])^ Do...
```

Dans Delphi 8 la première méthode est interdite directement. La deuxième n'est autorisée que dans l'écriture de code non managé.

L'utilisation d'une classe permet encore de convertir facilement cet accès vu que la notion de pointeur est cachée :

```
// Transtypage de la classe :
With TElementList(Liste.Items[i]) Do...
```

La comparaison avec **Nil** pour tester l'assignation du pointeur est toujours possible avec la classe. La comparaison suivante sera donc toujours la même dans les deux cas.

```
If Liste.Items[i]=Nil Then ...
```

Remarque : dans cet exemple le TList est conservé, donc il ne faut oublier de libérer manuellement tous les éléments de la liste avant sa destruction ou la fermeture du programme. Dans le cas de l'écriture d'une liste d'objet il serait plus judicieux d'utiliser directement un TObjectList qui peut être propriétaire de ses éléments et gérer leur destruction.

VI - Cas des listes chaînées

Les listes chaînées ne sont pas utilisées ici, mais peuvent aussi être aisément converties en classes.

```
Type
TElementListe=Class(TObject)
Public
  elNom :String;
  elPoids :Integer;
  elTaille:Integer;
  Suivant:TElementListe;
End;
```

La fin de liste peut aussi être testée par Nil sur le dernier élément. La création, destruction, parcours des éléments peuvent être convertis en suivant l'exemple ci-dessus.

Dans les nouvelle applications il est plus simple d'utiliser un TList ou descendant.

Conclusion

La conversion des pointeurs d'enregistrement n'est donc pas très difficile. Pour les nouvelles applications, même écrites avec Delphi 7, il est préférable d'utiliser tout de suite les classes. Le code présenté ici pour Delphi 8 fonctionne aussi directement sous Delphi 7. La conversion future vers Delphi 8 n'en sera que facilitée.

Fichiers sources de cet article :

Miroir 1 : [Source Delphi 7 de l'exemple](#)

Miroir 1 : [Source Delphi 8 de l'exemple](#)

Dans le cas où le miroir 1 ne fonctionne pas :

Miroir 2 : [Source Delphi 7 de l'exemple](#)

Miroir 2 : [Source Delphi 8 de l'exemple](#)

Version PDF de cet article :

Miroir 1 : [Version PDF](#)

Dans le cas où le miroir 1 ne fonctionne pas :

Miroir 2 : [Version PDF](#)

Source du projet sous Delphi 7 : [Source Delphi 7](#)

Source du projet sous Delphi 8 : [Source Delphi 8](#)

Merci à [Anomaly](#) pour la correction orthographique.