Considérations sur les types du langage Pascal

par Paul Toth (Point Virgule)

Date de publication : 2001-2003

Dernière mise à jour : 09/11/2010

Cet article rassemble une série de considérations de Paul Toth sur divers types du langage Pascal. Il aborde tout d'abord les **objets**, depuis leur introduction dans le langage avec Turbo Pascal 5.5, puis les **classes**, depuis leur introduction avec Delphi 1. Il décortique ensuite les **chaînes de caractères** et détaille leurs diverses implémentations dans Delphi. Il détaille avec clarté les **pointeurs** - et listes de pointeurs - et les **records complexes**. Il termine par une analyse limpide du problème des **références circulaires**.

Les objets

Avec Turbo Pascal 5.5 (TP55), Borland introduit la notion d'**objet** dans le langage Pascal. Cette première implémentation est limitée mais puissante.
En étudiant le code ci-dessous, vous verrez qu'un Objet est au départ très semblable à un Record; mais le seul fait de déclarer des procédures ou fonction (appelées **méthodes**) à l'intérieur de l'objet ajoute automatiquement à celles-ci un paramètre nommé **SELF** qui pointe sur l'objet lui même.

```
{ Déclaration d'un object simple en Turbo Pascal }
Type
 TMonObjet=Object
  X:integer;
  procedure Initialise(I:integer);
  function Calcul(Y:integer):integer;
 end;


procedure TMonObjet.Initialise(I:integer);
begin
 X:=I;
end;



function TMonObjet.Calcul(Y:Integer):integer;
begin
 Calcul:=X*Y;
end;




Var
 Obj:TMonObjet;
begin
 Obj.Initialise(2);
 WriteLn(Obj.Calcul(4));
end.
```

```
{ Cet objet est en tout point similaire au code suivant }
Type
 TMonRecord=Record
  X:integer;
 end;



procedure Initialise(Var SELF:TMonRecord;
 I:integer);
begin
 With SELF do begin

  X:=I;
 end;
end;

function Calcul(Var SELF:TMonRecord;
 Y:integer):boolean;
begin
 With SELF do begin

  Calcul:=X*Y;
 end;
end;

Var
 Obj:TMonRecord
begin
 Initialise(Obj,2);
 WriteLn(Calcul(Obj,4);
end;
```

Là où l'objet devient intéressant, c'est qu'il peut être enrichi.
C'est-à-dire qu'on va partir d'un objet existant et qu'on va lui ajouter des fonctionnalités... Notez que ce nouvel objet est entièrement compatible avec l'ancien; on peut donc l'utiliser partout où l'on utilise son **ancêtre**... notamment en paramètre d'une procédure.

```
{ Déclaration d'un object dérivé en Turbo Pascal }
Type
 TMonObjet2=Object(TMonObjet)
  A:integer;
  procedure Calcul2(Y,Z:integer); virtual;
 end;


procedure TMonObjet2.Calcul2(Y,Z:integer);
begin
 A:=X*Y*Z;
end;

Var
 Obj:TMonObjet2;
begin
 Obj.Initialise(2);
 WriteLn(Obj.Calcul(4));
```

```
 Obj.Calcul2(4,6);
 WriteLn(Obj.A);

end.
```

J'ai glissé le mot clé **virtual** dans l'exemple pour introduire le **polymorphisme**. C'est une autre particularité des objets; comme je le disais plus haut, on peut manipuler un objet "comme s'il sagissait de son ancêtre"... Mais on peut personnaliser le comportement de l'object ancêtre avec des méthodes **virtuelles**.
Pour comprendre le mécanisme, le plus simple est d'expliquer comment il est mis en oeuvre...

```
{ Déclaration d'une méthode virtuelle en Turbo Pasc
Type
 TMonObjet3=Object(TMonObjet2)
  s:string;
  procedure Calcul2(Y,Z:integer); virtual;
 end;


procedure TMonObjet3.Calcul2(Y,Z:integer);
begin
 inherited
 Calcul2; { appel la méthode ancètre de TMonObjet2
 A:=A*A;
end;

procedure exemple(obj:TObject2);
begin

 obj.Calcul2(2,3);
end;

Var
 Obj2:TMonObjet2;
 Obj3:TMonObjet3;
begin
 Exemple(Obj2); { c'est TMonObjet2.Calcul2 qui sera
 Exemple(Obj3); { c'est TMonObjet3.Calcul2 qui sera

end.
```

```
{ Implémentation des méthodes virtuelles sans Object }
Type
 TMonRecord2=Record {TMonRecord}
  X:integer;
  A:integer;
  Calcul2:procedure(Y,Z:integer);
 end;
 TMonRecord3=Record {TMonRecord2}

  X:integer;
  A:integer;
  Calcul2:procedure(Y,Z:integer);
  s:string;
 end;

procedure Record2Calcul2(Var SELF:TMonRecord2;
 Y,Z:integer);
begin
 With SELF do begin
  A:=X*Y*Z;
 end;
end;


procedure Record3Calcul2(Var SELF:TMonRecord3;
 Y,Z:integer);
begin
 With SELF do begin
  Record2Calcul2(TMonRecord2(Self),X,Z);
  A:=A*A;
 end;
end;

procedure exemple(obj:TMonRecord2);

begin
 obj.Calcul2(obj,2,3);
end;

Var
 Obj2:TMonRecord2;
 Obj3:TMonRecord3;
begin
 Obj2.Calcul2:=Record2Calcul2;
 Obj3.Calcul2:=Record3Calcul2;
 Exemple(Obj2);
 Exemple(TMonRecord2(Obj3));
end.
```

N'hésitez pas à **me contacter** si mes explications sont insuffisantes... mais notez que je considère que la programmation Pascal standard est acquise :)
TP6 introduit les méthodes virtuelles indexées; elles répondent à un besoin technique plus qu'à une nécessité de programmation. Je ne donnerai pas le détail ici mais le but du jeu était de réduire la taille mémoire nécessaire au stockage des objets.
L'usage intensif des objets sous TP ne présentait qu'un seul inconvénient majeur; il fallait gérer manuellement les allocations mémoire.

```
{ voici un exemple de déclaration d'objet dynamique,
  notez les méthodes spéciales de creation et destruction d'objet. }

Type
 PMonObject=^TMonObject;
 TMonObject=Object
  x:integer;
  Constructor Create(I:integer);
  Destructor Destroy;
 end;

Constructor TMonObjet.Create(I:integer); begin
 x:=i; end;

Destructor TMonObjet.Destroy; begin end;

Var
 Obj:PMonObject;
begin
 New(Obj,Create);
 Obj^.Initialise(1);
 Dispose(Obj,Destroy);
end.
```

Les classes

Les composants

Les contrôles

Les chaînes de caractères (String)

Les chaînes à 0 terminal

Les pointeurs (Pointer)

Allocation dynamique (GetMem et FreeMem)

Liste de pointeurs (TList)

Les records complexes (Union en C)

Le Case...of

Le transtypage

Les références circulaires

Méthode 1, le sous-typage

Méthode 2, l'ancêtre virtuel

Méthode 3, l'interface