Gestion dynamique de la mémoire

Nour-Eddine Oussous, Éric Wegrzynowski

Licence ST-A, USTL - API2

19 octobre 2009

Occupation mémoire de différents types de données

- Déclaration d'une variable = réservation d'un espace mémoire qui est fonction du type de la variable.

Exemples avec Free Pascal (avec l'option `-Mobjfpc`) sur architecture i386

Déclaration	Mémoire réservée
`var somme : INTEGER ;`	4 octets
`var n : CARDINAL ;`	4 octets
`var trouve : BOOLEAN ;`	1 octet
`var moyenne : REAL ;`	8 octets
`var tableau : Array[1..100] of REAL ;`	$100 \cdot 8 = 800$ octets
`var nom : STRING[20] ;`	21 octets

Variables et mémoires

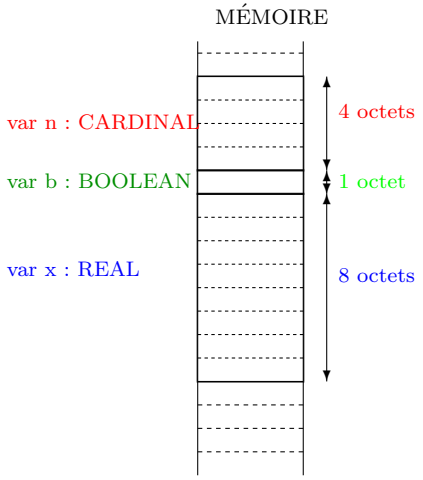

Fig.: Variables en mémoire

Allocation statique/dynamique de mémoire

- Déclaration d'une variable d'un type T = allocation statique d'une zone mémoire
- Possibilité d'allocation dynamique de mémoire

Les pointeurs

Définition

Un pointeur est une variable qui contient l'adresse d'une donnée contenue en mémoire.

- La déclaration d'une variable pointeur
  - réserve 4 octets nécessaires au codage de l'adresse mémoire de la donnée pointée
  - mais ne réserve aucune mémoire pour la donnée pointée
- Quel que soit le type de la donnée pointée, la taille mémoire du pointeur est toujours la même : 4 octets

Pointeur en mémoire

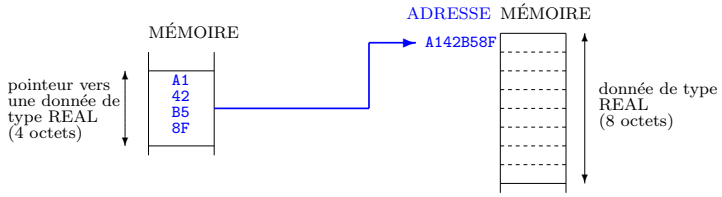

Fig.: Pointeur vers un REAL

Déclaration d'un pointeur

En Pascal, les pointeurs sont des variables dont le type est celui de la donnée pointée précédé d'un ^

```
var P : ^<type> ;
```

Exemples de variables pointeurs

```
var
  P : ^CARDINAL ; // pointeur vers CARDINAL
  R : ^REAL ;     // pointeur vers REAL
```

L'opérateur ^

L'opérateur ^ permet d'obtenir la zone pointée par un pointeur.
P^ peut être considéré comme une variable du type de la zone pointée.

Exemple

```
// affectation a Y de la valeur pointee par P
Y := P^ ;
// affichage de la valeur pointee par P
writeln(P^);
```

Y doit avoir le type des valeurs pointées par P

Autre exemple

```
type
   COMPLEXE = record
                re : REAL;
                im : REAL;
              end {COMPLEXE};
var
  P : ^COMPLEXE;

begin
   ...
   {P pointe vers un complexe}
   // affichage de la partie reelle
   writeln(P^.re);
   // affichage de la partie imaginaire
   writeln(P^.im);
   ...
end
```


La constante NIL

La constante NIL est un pointeur (de tout type) particulier qui ne pointe sur rien.

Exemple

```
P := NIL;  // P ne pointe sur rien
```


Attention

Lorqu'un pointeur vaut NIL, il n'y a aucun sens de tenter d'accéder à la zone pointée

Exemple

Si P vaut NIL, à l'exécution l'instruction

```
X := P^;
```

produira un comportement imprévisible.

Affectation de pointeurs

Il est possible d'affecter la valeur d'un pointeur à un autre pointeur du même type.

Exemple

```
P := Q; // P pointe vers la meme zone que Q
```

CU : P et Q pointent tous deux vers le même type de données.

L'opérateur @

L'opérateur @ appliqué à une variable donne l'adresse de cette variable. Cette adresse peut être affectée à un pointeur.

Exemple

```
var
  X : CARDINAL  ;
  P : ^CARDINAL ;
begin
  X := 3 ;
  P := @X ; // P pointe vers X
end
```


Affectation de pointeurs

Attention

Aucune différence entre une adresse et une adresse !
⇒ possibilité d'affecter à un pointeur vers une donnée d'un certain type l'adresse d'une variable d'un autre type ! !
⇒ résultats imprévisibles ! ! !

Exemple

```
var
  X : REAL;
  P : ^CARDINAL;
begin
  X := 3.141592 ;
  P := @X; // autorise, mais que pointe P ???
end
```


Allocation dynamique de mémoire

Définition

L'allocation dynamique de mémoire est la possibilité de réserver une zone mémoire à l'exécution d'un programme.

⇒ nécessité de disposer d'un espace mémoire dans lequel faire l'allocation : cette zone est nommée TAS.

La procédure new

La procédure new, appliquée à un pointeur P,

1. réserve une zone mémoire dans le TAS d'une taille correspondant à la taille des données pointées par P,
2. et attribue à P l'adresse de cette zone.

Exemple

```
var
  P : ^CARDINAL;
  R : ^REAL;
begin
  new(P); // 4 octets alloues dans le TAS,
          // P pointe vers cette zone
  new(R); // 8 octets alloues dans le TAS,
          // R pointe vers cette zone
```

(à suivre ...)

Allocation de mémoire $\neq$ affectation d'une valeur.
$\Rightarrow$ nécessité après allocation, d'attribuer une valeur à la zone allouée

Exemple

(...suite)

```
P^ := 4;
R^ := 3.141592;
```


Attention

L'affectation d'une zone mémoire par une instruction du type

```
P^ := ...
```

peut avoir des effets de bord.

Exemple

```
{ X =?? }
X := 5;
{ X = 5}

P := @X;
{ X = 5, P =adresse de X}

P^ := 6;
{ X = 6, P =adresse de X }
writeln(X); // affiche 6 !!
```


Allocation dynamique de mémoire

Définition

La désallocation de mémoire est l'opération inverse de l'allocation : elle libère une zone du TAS qui a été allouée.

- opération nécessaire pour ne pas épuiser le TAS,
- à réaliser dès qu'une zone n'a plus d'utilité.

La procédure `dispose`

La procédure `dispose`, appliquée à un pointeur P,

1. libère la zone mémoire du TAS pointée par P
2. et rend indéterminée la valeur de P.

Exemple

```
var
  P : ^CARDINAL;
begin
  new(P) ;  // allocation de 4 octets
  P^ := 6 ; // attribution d'une valeur
  dispose(P); // desallocation de la zone
end
```


Attention

La procédure `dispose` ne doit être appliquée qu'à des pointeurs pointant sur une zone allouée dynamiquement (par un `new`).

Attention

La désallocation d'une zone pointée peut avoir un effet de bord lorsque cette zone est pointée par d'autres pointeurs, ou correspond à une variable.

Exemple

```
new(P); // allocation d'une zone dans le TAS
P^:= 10;
Q := P ;  // Q pointe vers la meme zone que P

dispose(P) ; // liberation de la zone pointee
             // par P
writeln(Q^); // affiche n'importe quoi
```


La désallocation n'est pas simple

- Gérer la désallocation n'est pas une opération simple.
- Certains langages de programmation (Lisp, Java,...) la gèrent automatiquement.
- La gestion automatique de récupération de mémoire est nommée ramasse-miettes (garbage collector en anglais).