

Le type access et les structures dynamiques

Le Type access, ou "Qu'est-ce qu'un pointeur ?"

Un *pointeur* est simplement une variable dont la valeur est égale à l'adresse d'une case mémoire.

```
type pointeur is access Integer;
P: pointeur;
```

Ainsi défini, P est un *pointeur* sur un objet de type `Integer`. Autrement dit, on pourra affecter à P l'adresse d'une case mémoire contenant un entier. La valeur de l'objet pointé par P peut être obtenue par *déréférencement*: `P.all`

Le Pointeur vide.

Par défaut, uen fois déclaré, un pointeur est initialisé à la constante `null`, qui ne correspond à aucune adresse.

```
P:=null;
```

Allocation dynamique de mémoire.

La déclaration `P:Pointeur` a pour effet d'attribuer de la mémoire à la variable P, comme n'importe quelle déclaration de variable. Mais, il n'y a pas d'allocation de mémoire pour l'objet pointé, ici un entier. Il faut une allocation explicite, avec l'instruction `new` qui prend en argument le type de l'objet.

```
P:=new Integer;
P.all:=6;
```

il est également possible de combiner l'allocation de mémoire et l'initialisation de l'objet abrité dans cette case mémoire en une seule instruction, avec la syntaxe suivante :

```
Q:=new Integer'(5); -- Q.all=5
```

Qui dit *allocation* dit ensuite *désallocation*, ou *libération* de mémoire. Cela se fait par instantiation de la procédure générique `Unchecked_Deallocation`.

```
with Unchecked_Deallocation;
...
procedure Free is new Unchecked_Deallocation(Integer, Pointeur);

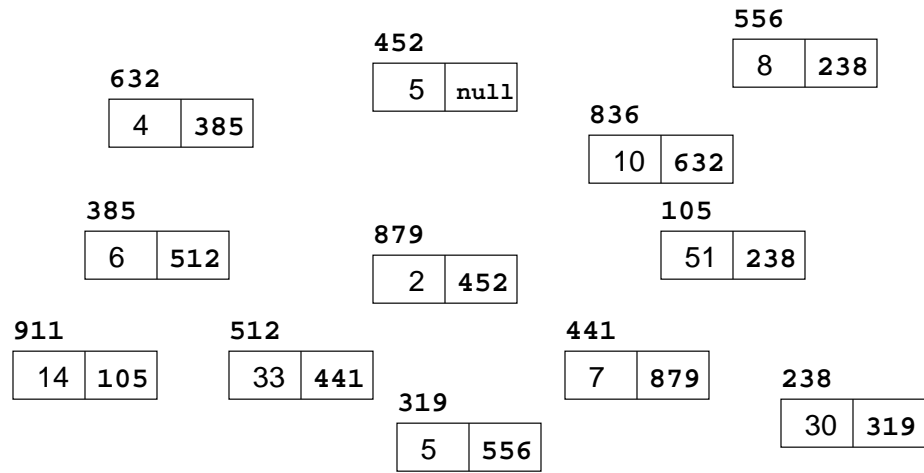
Free(P); -- la valeur pointee par P est perdue
```

Les Listes chaînées

Une liste chaînée est une structure de données dans laquelle les objets sont rangés linéairement. Toutefois, contrairement au tableau pour lequel l'ordre linéaire est déterminé par les indices, l'ordre d'une liste chaînée est déterminé dans chaque objet par un pointeur: l'adresse de l'objet suivant. L'avantage d'une liste chaînée sur un tableau est sa souplesse : on peut l'agrandir, intercaler des objets, en supprimer etc. La contrepartie est qu'il n'y a plus d'accès direct aux éléments, comme dans un tableau. Accéder au *i*ème élément nécessite de parcourir la liste depuis le début.

```
type Cellule;
type Liste is access Cellule;
type Cellule is record
  Valeur:Integer;
  Suivant:Liste;
end record;
```

Ce type permet de définir une liste d'entiers (`Integer`). Chaque cellule est en fait un enregistrement, constitué d'un champs `Valeur` qui contient la valeur de l'élément de la liste, et d'un champ `Suivant` qui contient l'adresse de l'élément suivant dans la liste. La liste est assimilée à l'adresse de la cellule de tête.



Chaque rectangle correspond à une cellule : le premier compartiment est le champ **Valeur** et le second le champ **Suivant**. L'adresse de chaque cellule est précisée au dessus.

Question 1: Quelle est la liste chaînée correspondant à L1=836 ?

Question 2: Quelle est le troisième élément de la liste L2= 385 ?

Question 3: Que pensez-vous de la liste L3 = 911 ?

La solution dans le numéro de la semaine prochaine.