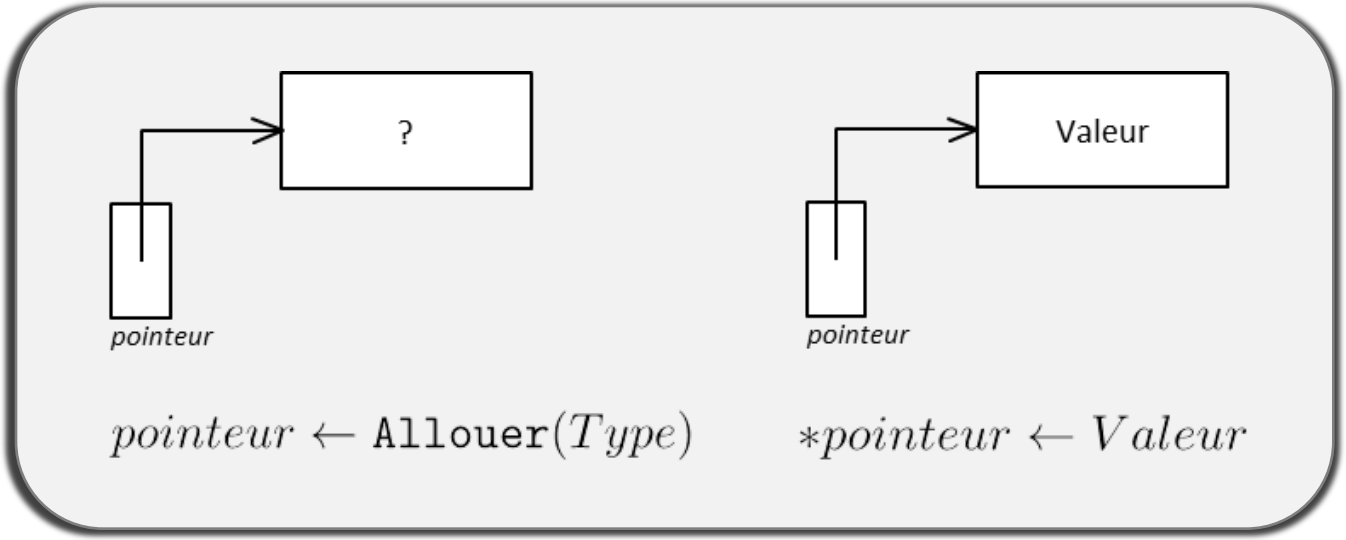
Algorithmique

Allocation dynamique



INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

Eric Guérin
INSA de Lyon – Département Informatique
3IF

Structures séquentielles

- Structure séquentielle = tableau
- Temps d'accès constant
- Limitations nombreuses
  - Nombre d'éléments fixe
  - Pas de mélange de types

Structures séquentielles

- Limitation
  - Nombre d'éléments fixe
- Solution
  - Allocation dynamique
    - De nombreuses différences avec l'allocation statique
    - Allocation/libération explicites

Structures séquentielles

- Limitation
  - Utilisation d'un seul type de données
- Solution
  - Structures de données
    - Regroupement de plusieurs types en un seul
    - Base de la construction des structures de données élémentaires
      - Piles
      - Files
      - Arbres
      - *Etc.*

Variables statiques

- Possède
  - Un nom
  - Un type
  - Une adresse
  - Une valeur
- Le compilateur réserve un espace mémoire directement dans le segment de l'exécutable
- Tout se fait au moment de la compilation

Variables statiques

- Limitations
  - En général pour les constantes
  - Quasi constantes
    - Valeur déterminée avant l'exécution
  - Variables globales
    - Ne sont pas déclarées dans une fonction

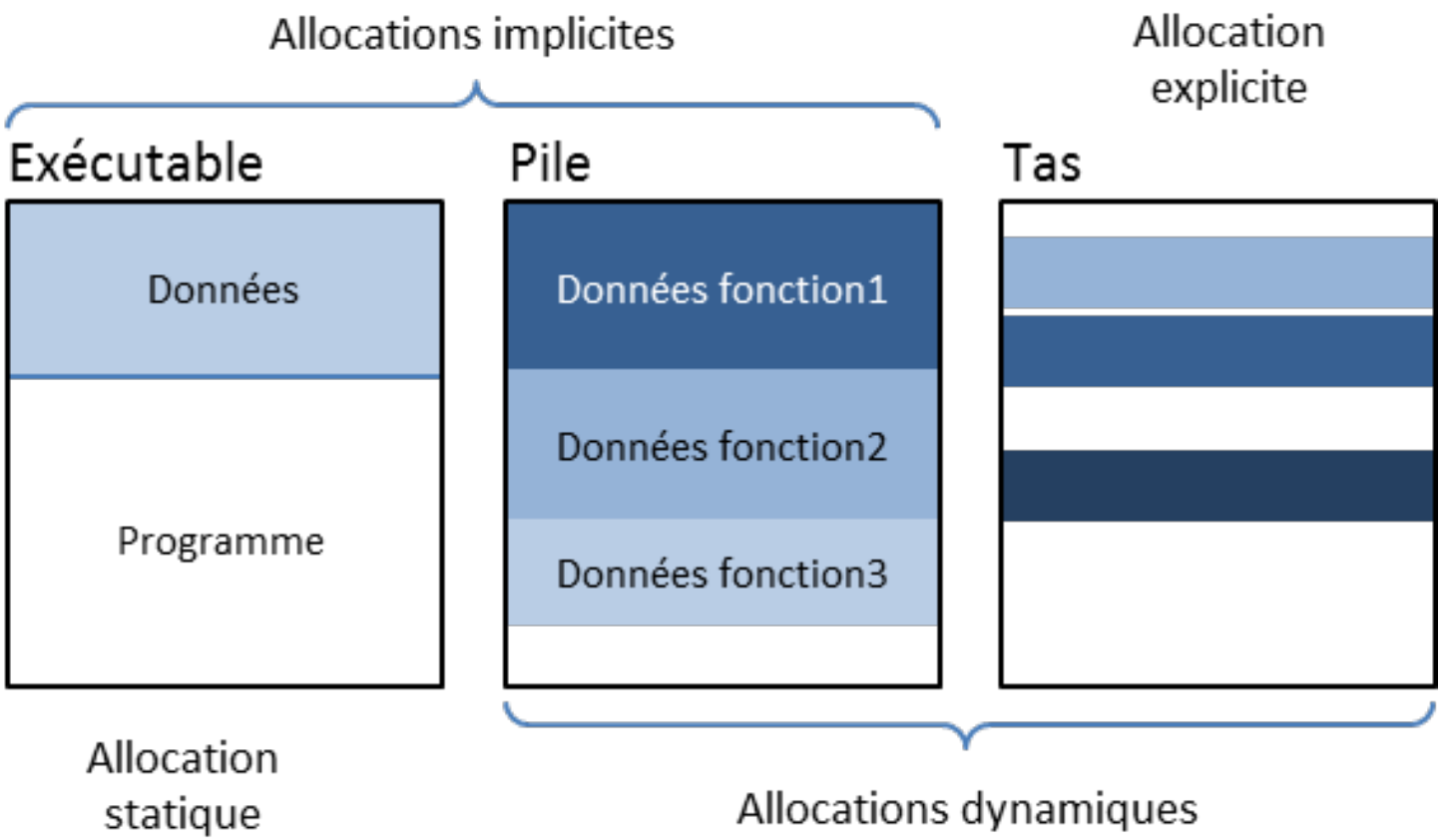
Variables dynamiques sur la pile

- Variables de fonction
  - Impossible de déterminer combien de fois elle sera allouée
    - Exemple typique : récursivité
- Allocation dynamique
  - Dans la zone de la pile
  - Allocation/libération automatiques
    - Générées par le compilateur
  - Transparent pour l'utilisateur

Variables dynamiques sur le tas

- Un maximum de contrôle pour l'utilisateur
- Allocation/libération explicites
- Espace mémoire dédié
- Impossible de passer par une déclaration standard
  - Nécessité d'utiliser des pointeurs : variable représentant l'adresse d'une autre variable
  - Opérateur d'indirection ou de déréférencement *

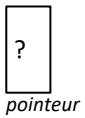
Récapitulatif



Fonctionnement

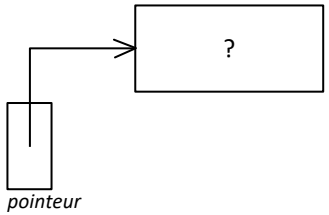
- En quatre temps
  - Déclaration d'une variable de type pointeur
  - Allocation dynamique explicite
  - Utilisation de la variable
  - Libération de la zone mémoire

1. Déclaration



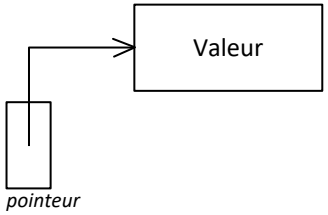
$Type * pointeur$

2. Allocation



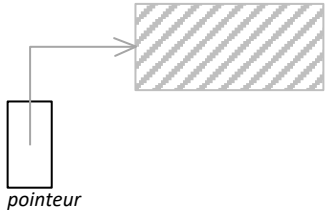
$pointeur \leftarrow \texttt{Allouer}(Type)$

3. Utilisation



$*pointeur \leftarrow Valeur$

4. Libération



$\texttt{Liberer}(pointeur)$

Fonctionnement

- Allocation implicite sans danger
- Allocation dynamique = danger
  - Tout est explicite
  - Un oubli et le programme peut planter
- Exemple
  - Indirection d'un pointeur qui a été libéré

En langage C

- Allocation : `malloc()`
  - Renvoie le pointeur vers la zone allouée
    - Renvoie un pointeur nul si l'allocation n'a pas pu être faite (dépendant du système)
  - Prend en argument le nombre d'octets à allouer
  - Utilisé avec `sizeof()`
- Libération : free()
  - Prend en argument le pointeur de la zone allouée avec `malloc()`

En langage C

- Exemple

```
#include <stdlib.h>
Type * pointeur;
pointeur = (Type*) malloc(sizeof(Type));
if (pointeur)
{
        *pointeur = valeur;
        free(pointeur);
}
```

Exercice

- Représenter les états successifs de la mémoire pour l'algorithme suivant :

entier * a
entier * b
$a \leftarrow$ Allouer(*entier*)
$b \leftarrow a$
$*a \leftarrow 1$
$b \leftarrow$Allouer(*entier*)
$*b \leftarrow *a$
Libérer(a)
Libérer(b)

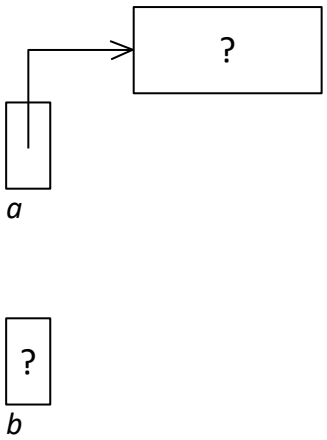
Exercice

entier * a
entier * b

$a \leftarrow$ Allouer(*entier*)
$b \leftarrow a$
$*a \leftarrow 1$
$b \leftarrow$ Allouer(*entier*)
$*b \leftarrow *a$
Libérer(a)
Libérer(b)

Exercice

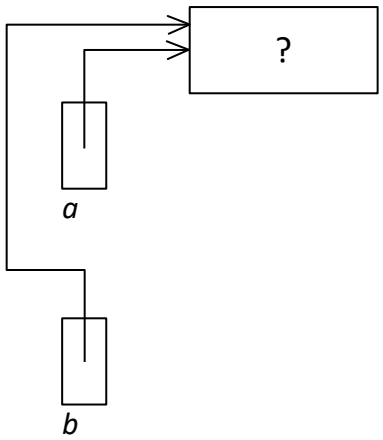
entier * a
entier * b
$a \leftarrow$ Allouer(*entier*)
$b \leftarrow a$
$*a \leftarrow 1$
$b \leftarrow$ Allouer(*entier*)
$*b \leftarrow *a$
Libérer(a)
Libérer(b)



Exercice

entier * a
entier * b
$a \leftarrow$ Allouer(*entier*)
$b \leftarrow a$

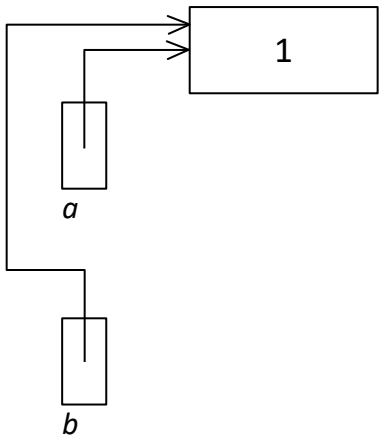
$*a \leftarrow 1$
$b \leftarrow$ Allouer(*entier*)
$*b \leftarrow *a$
Libérer(a)
Libérer(b)



Exercice

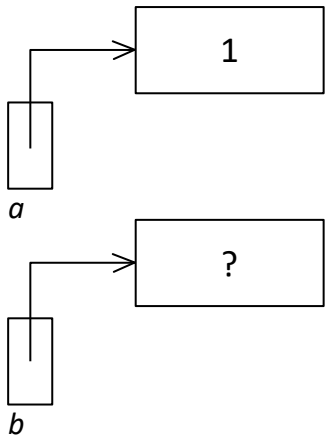
entier * a
entier * b
$a \leftarrow$ Allouer(*entier*)
$b \leftarrow a$
$*a \leftarrow 1$

$b \leftarrow$Allouer(*entier*)
$*b \leftarrow *a$
Libérer(a)
Libérer(b)



Exercice

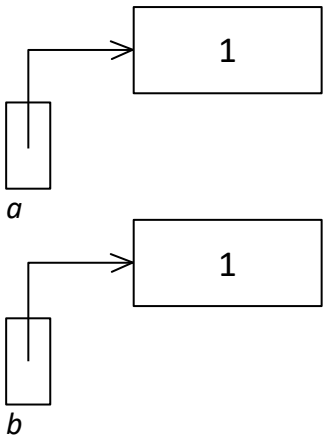
entier * a
entier * b
$a \leftarrow$ Allouer(*entier*)
$b \leftarrow a$
$*a \leftarrow 1$
$b \leftarrow$ Allouer(*entier*)
$*b \leftarrow *a$
Libérer(a)
Libérer(b)



Exercice

entier * a
entier * b
$a \leftarrow$ Allouer(*entier*)
$b \leftarrow a$
$*a \leftarrow 1$
$b \leftarrow$ Allouer(*entier*)
$*b \leftarrow *a$

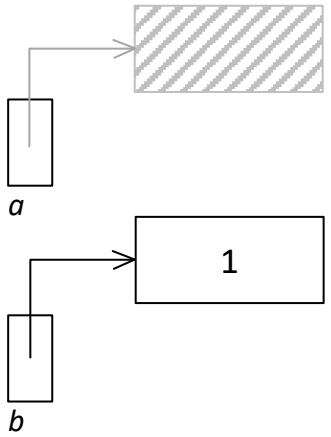
Libérer(a)
Libérer(b)



Exercice

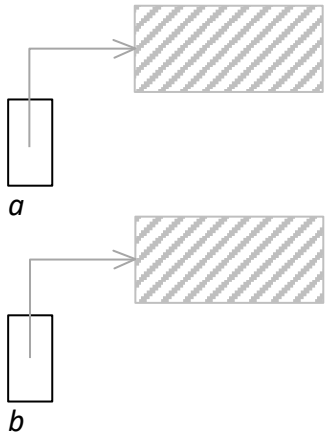
entier * a
entier * b
$a \leftarrow$ Allouer(*entier*)
$b \leftarrow a$
$*a \leftarrow 1$
$b \leftarrow$ Allouer(*entier*)
$*b \leftarrow *a$
Libérer(a)

Libérer(b)



Exercice

entier * a
entier * b
$a \leftarrow$ Allouer(*entier*)
$b \leftarrow a$
$*a \leftarrow 1$
$b \leftarrow$ Allouer(*entier*)
$*b \leftarrow *a$
Libérer(a)
Libérer(b)



A retenir

- Allocation dynamique
  - Allocation dynamique = contrôle total du cycle de vie de la variable
  - Contrôle = surcoût
    - Plus de précautions
    - Manipulations bas niveau (pointeurs)
  - Segment de mémoire séparé (tas)
  - Accès à la variable dynamique par déréférencement (opérateur *)