ECE PARIS • LYON
ÉCOLE D'INGÉNIEURS

Informatique 2

Cours n°12

Structures de données
Listes Chaînées

Antoine Hintzy

Structure de données

On appelle **structure de données** une façon d'organiser les données afin de les traiter plus facilement.

Une structure de données est généralement composée :

- d'un **type de données**
  - `struct Vaisseau`, `char chaine[50]`, etc.
- de **méthodes** permettant d'exploiter les données
  - `initVaisseau()`, `strcpy()`, etc.

Structures de données déjà vues

- **Structures**
  - méthodes : initialisation, etc.
- **Tableaux** *(à une ou deux dimension(s))*
  - méthodes : tri, recherche, affichage, etc.
- **Chaînes de caractères**
  - méthodes : concaténation, recopie, etc. (**`string.h`**)

Nouvelles structures de données

- Listes
- Files (d'attente) : 🇬🇧 FIFO (*First In First Out*)
- Piles (d'assiettes) : 🇬🇧 LIFO (*Last In First Out*)
- Arbres
- Graphes
- etc.

Ces structures de données visent à regrouper un ensemble de données sous une certaine forme qui les rend plus facilement exploitable.

Complexité

Suivant l'utilisation que l'on fera des données, il convient de choisir la structure de données la plus adaptée.

Par exemple, les tableaux ne sont pas du tout adaptés si l'on souhaite insérer régulièrement des éléments au début de celui-ci. En effet, il faudra décaler tous les éléments de ce tableaux d'une case, et potentiellement le redimensionner si l'espace disponible est insuffisant.

A l'inverse, une liste chaînée est tout à fait adaptée à cet usage puisqu'il ne suffira que de rajouter une cellule au début de la chaîne, sans avoir besoin de toucher au reste des cellules.

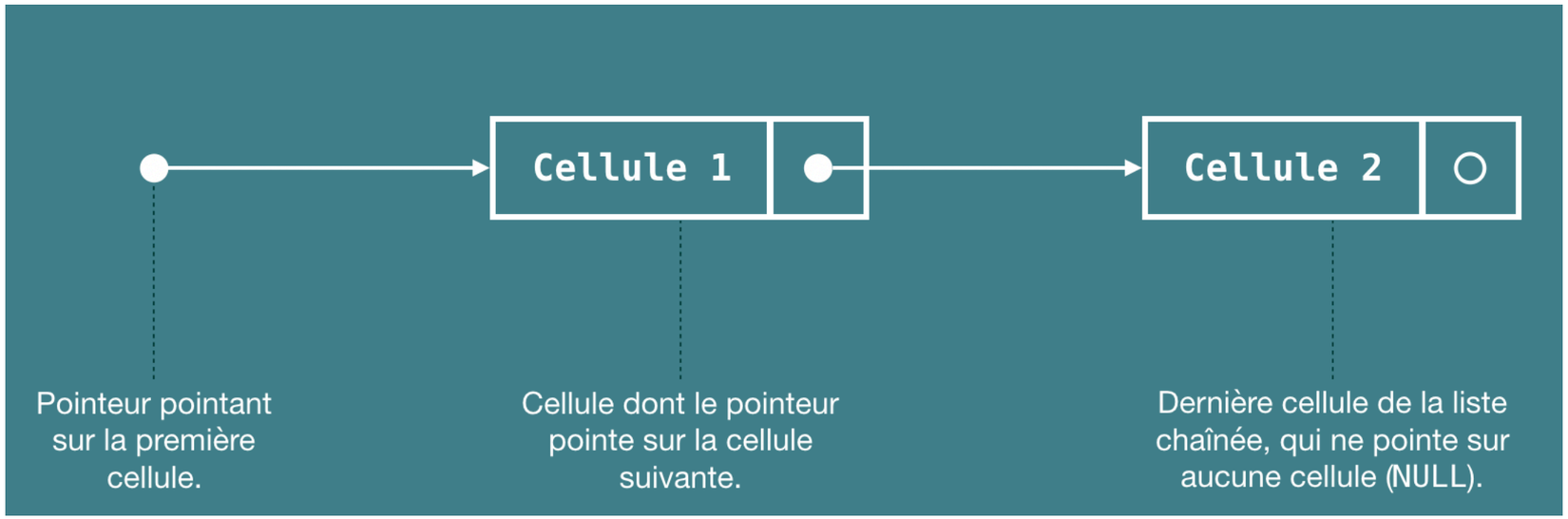
Choix de la structure de données

Selon les tâches à effectuer sur une structure de données, le choix de celle-ci sera différent.

Scénario	Structure de données
Beaucoup d'insertions/suppressions	Liste chaînée
Accès aux éléments par indice	Tableau
etc.	

Liste chaînée

- Ensemble de cellules liées entre elles par des pointeurs.
- Une cellule contient des données, ainsi qu'un pointeur permettant de stocker l'adresse de la cellule suivante.



Liste chaînée - Cellule

- Définition d'une cellule :

```
typedef struct Cell {
    // données :
    char firstname[100];
    char lastname[100];
    // pointeur vers la cellule suivante :
    struct Cell* next;
} Cellule;
```

⚠️ Dans la définition d'une structure, nous ne pouvons pas encore utiliser son alias de type créé avec **`typedef`**. Nous pouvons en revanche utiliser le type complet, qui est en cours de création : ici, **`struct Cell`**.

Liste chaînée - Déclaration

- Déclaration d'une liste chaînée :

```c
int main(void) {
  Cellule* liste = NULL; // contiendra l'adresse de la première cellule
  return 0;
}
```

⚠️ Il est important d'initialiser à **NULL** chaque pointeur, pour éviter de récupérer une ancienne valeur en mémoire. En effet, le parcours d'une liste chaînée se fait de cellules en cellules tant que le pointeur de la cellule parcourue n'est pas **NULL**.

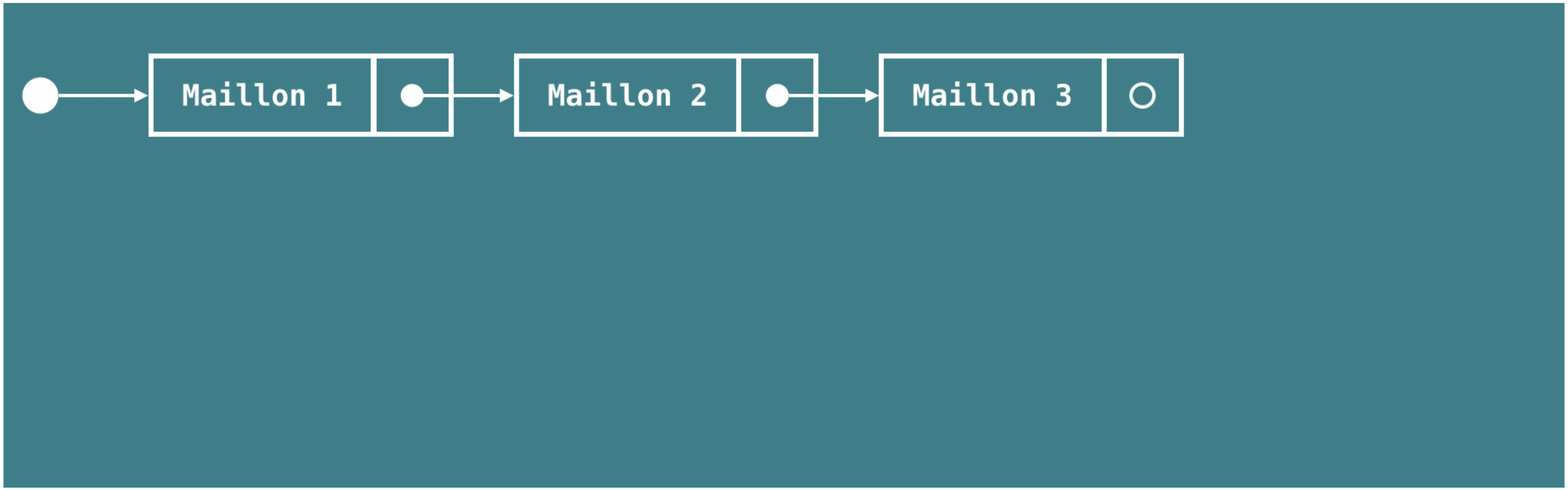
Liste chaînée - Création de la première cellule

```c
int main(void) {
   Cellule* liste = NULL;

   liste = (Cellule*) malloc(sizeof(Cellule));
   strcpy(liste->firstname, "Toto");
   strcpy(liste->lastname, "Titi");
   liste->next = NULL;

   return 0;
}
```

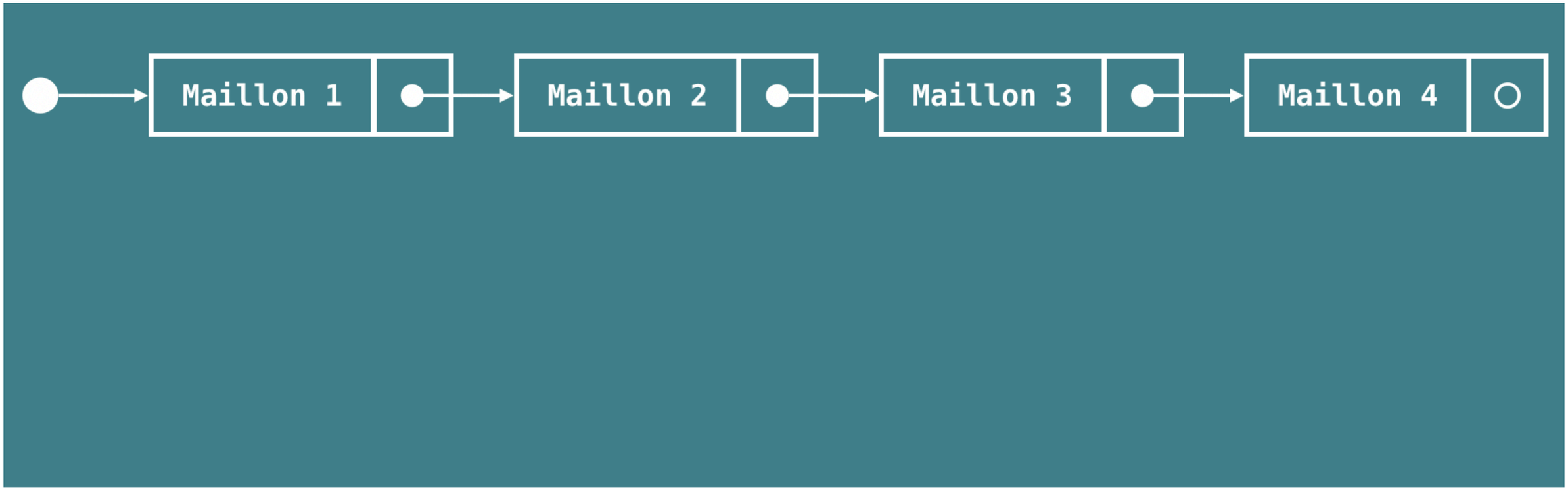
Liste chaînée - Création de la deuxième cellule

```c
int main(void) {
  Cellule* liste = NULL;

  // Création de la première cellule...

  liste->next = (Cellule*) malloc(sizeof(Cellule));
  strcpy(liste->next->firstname, "Tata");
  strcpy(liste->next->lastname, "Tutu");
  liste->next->next = NULL;

  return 0;
}
```

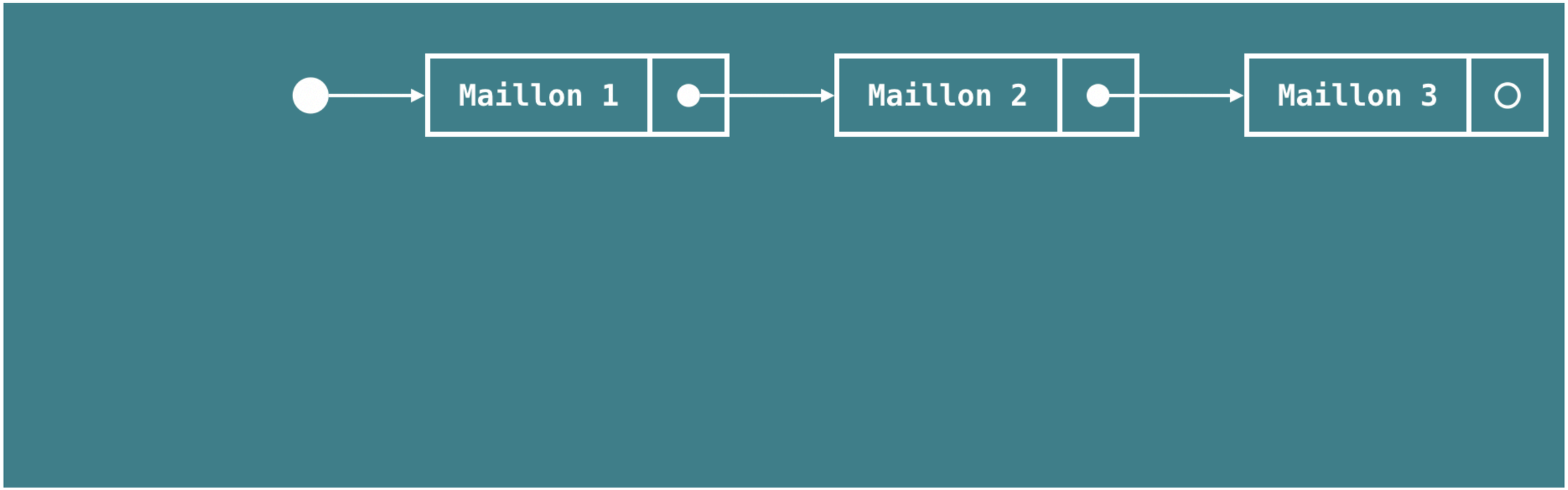
Insertion en fin de liste



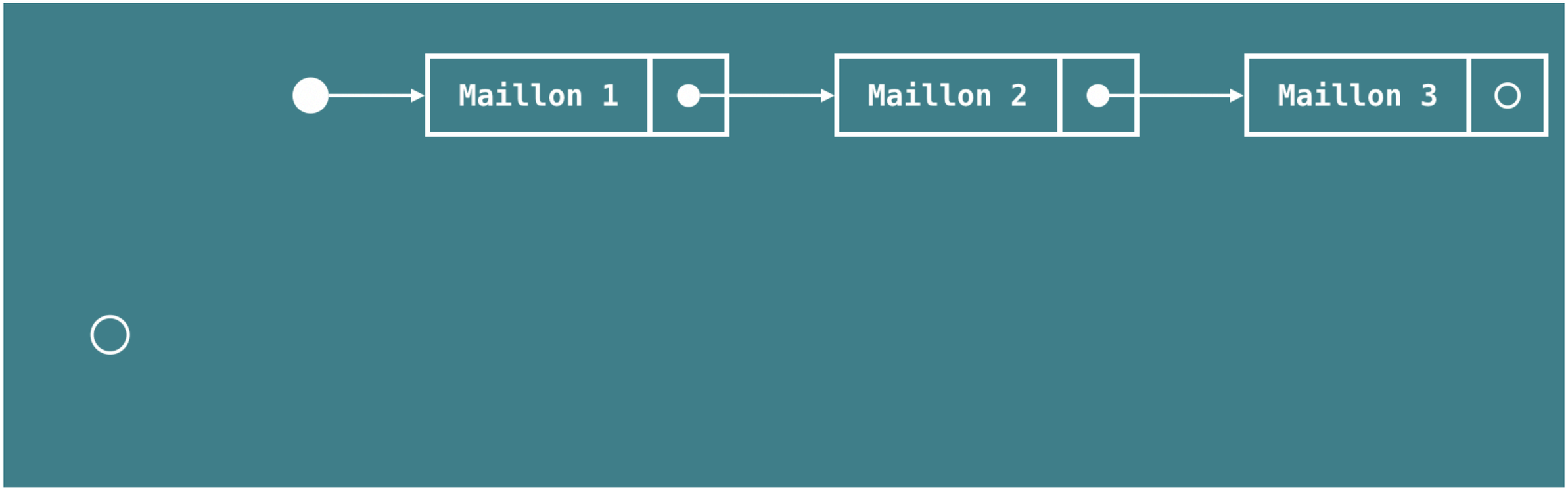
Insertion en fin de liste



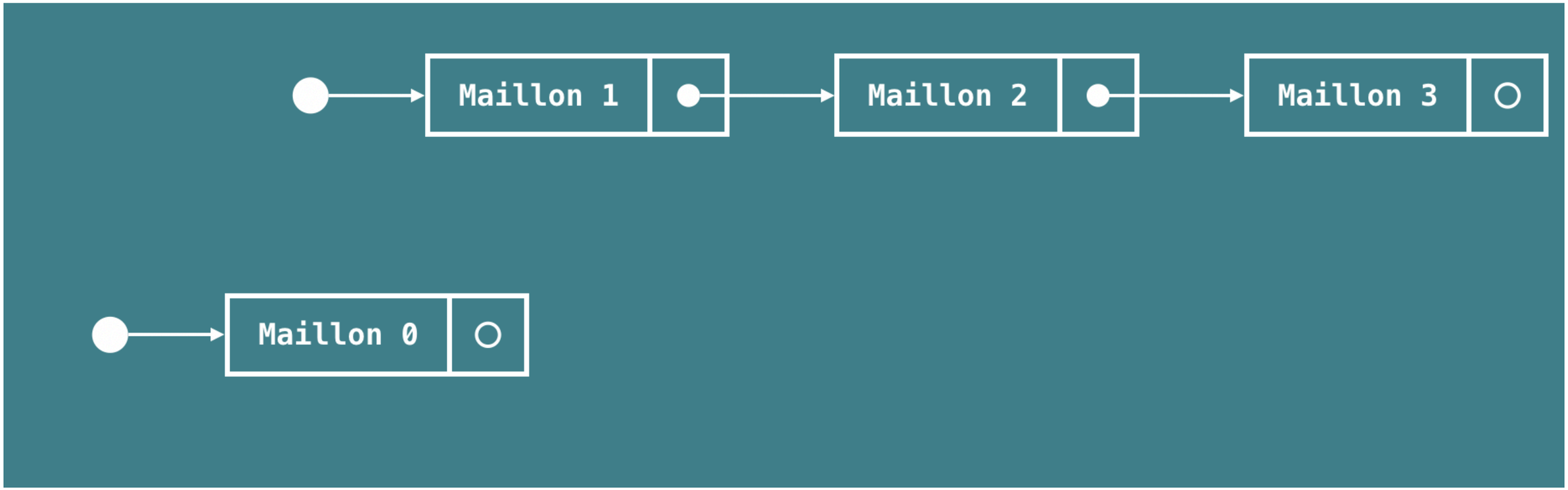
Insertion en tête de liste



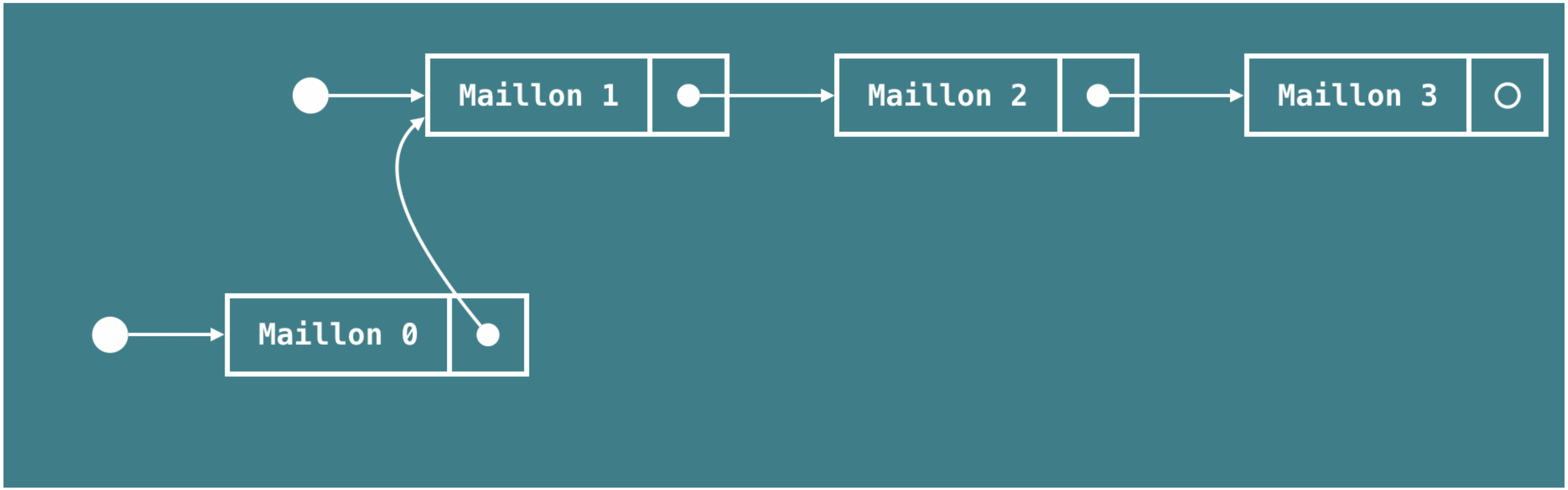
Insertion en tête de liste



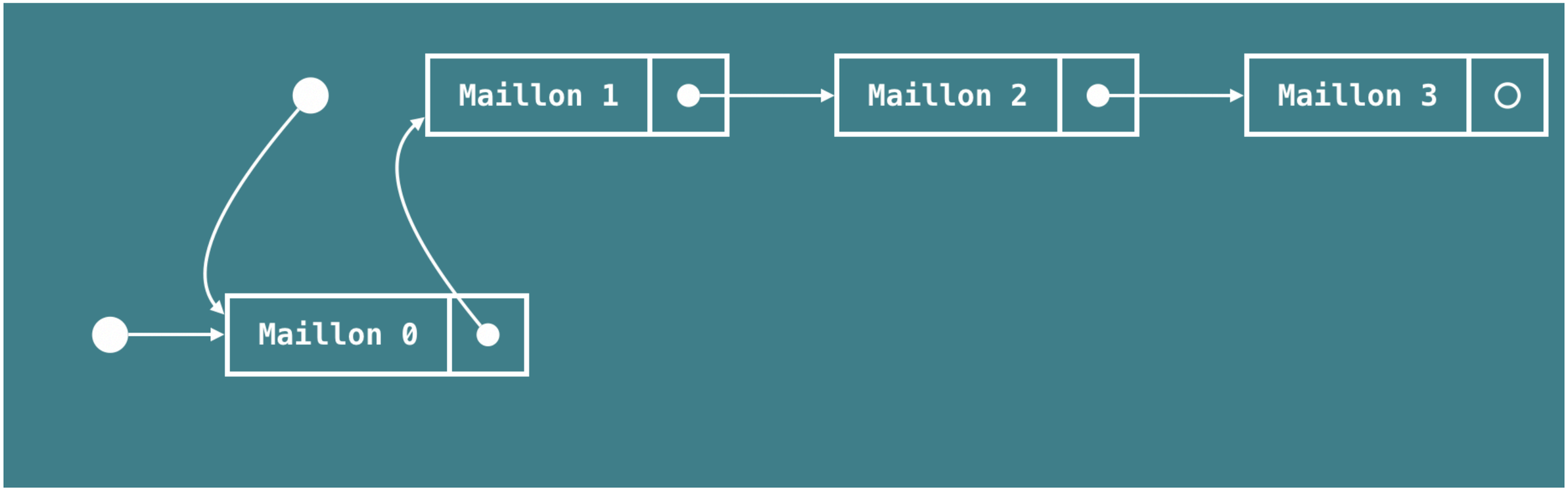
Insertion en tête de liste



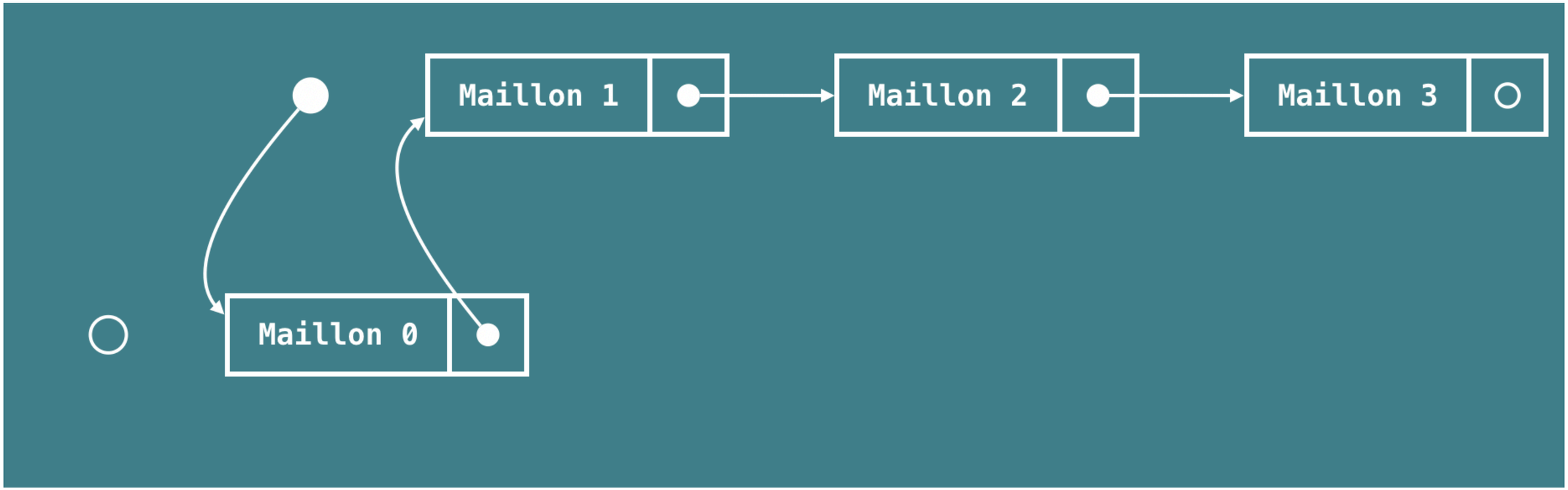
Insertion en tête de liste



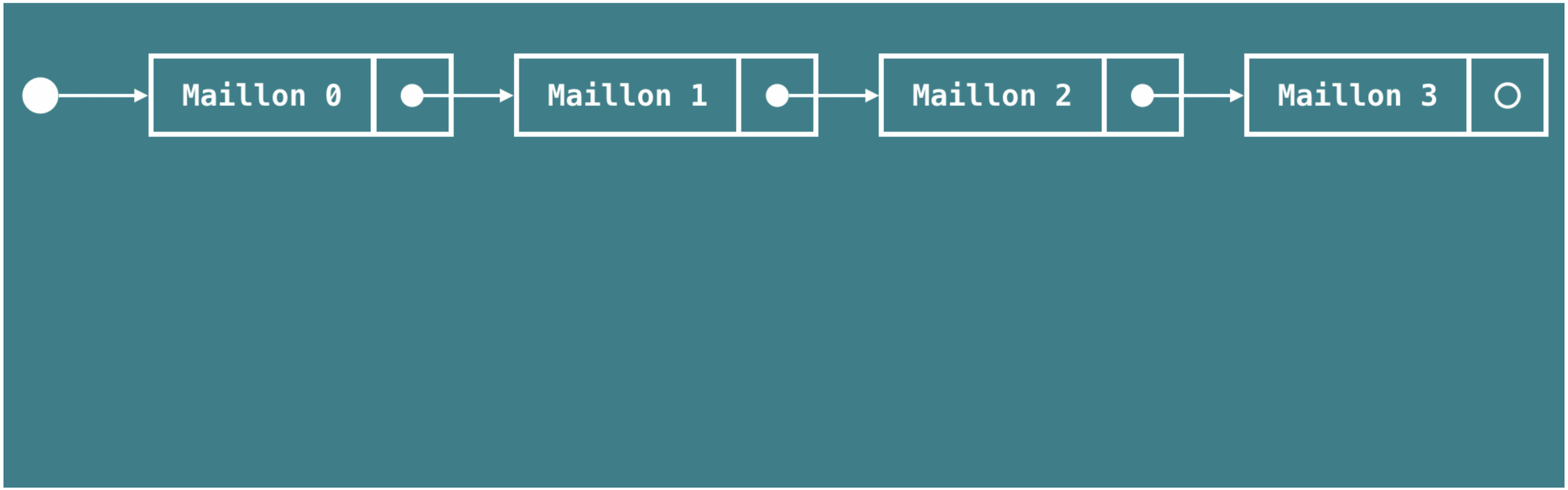
Insertion en tête de liste



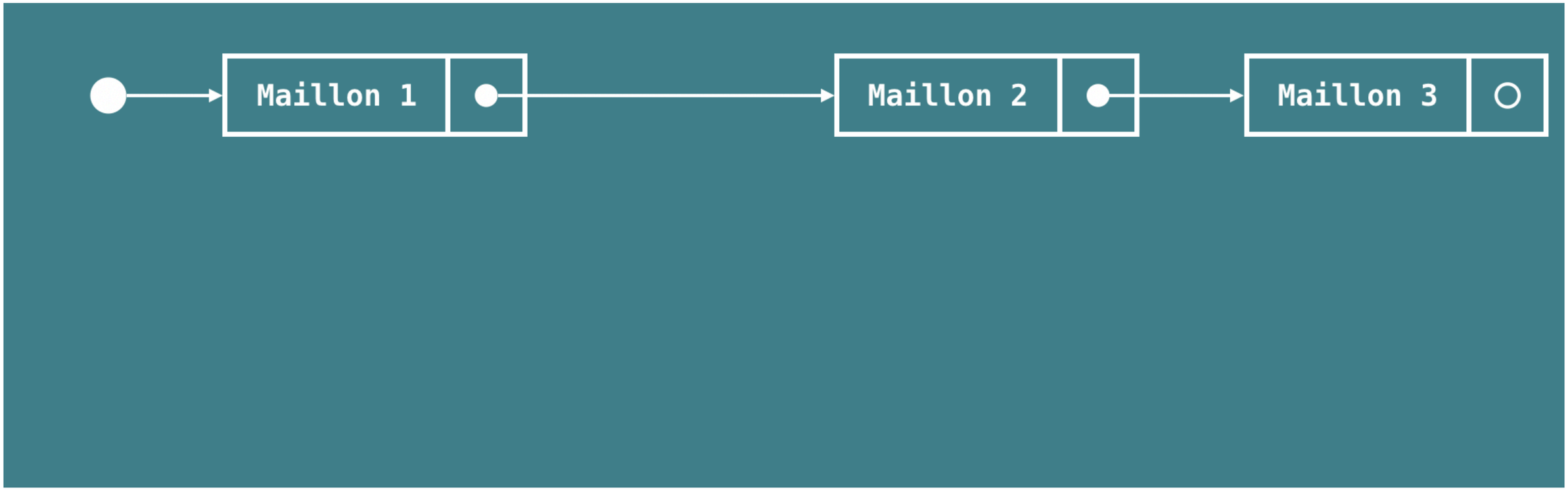
Insertion en tête de liste



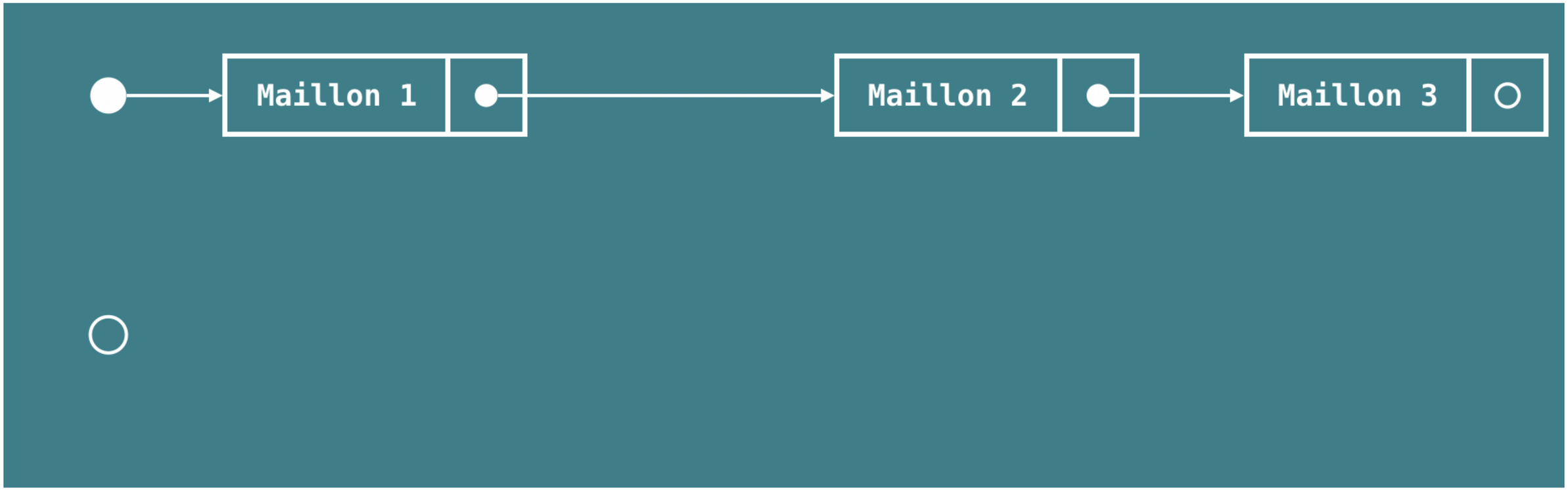
Insertion en tête de liste



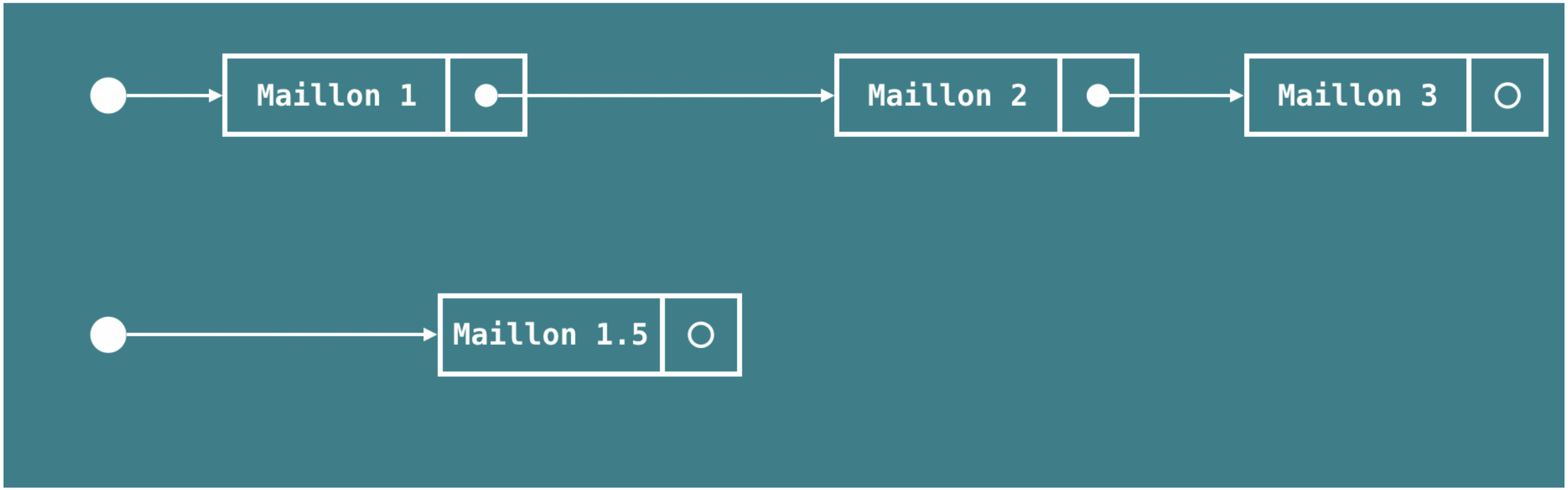
Insertion en milieu de liste



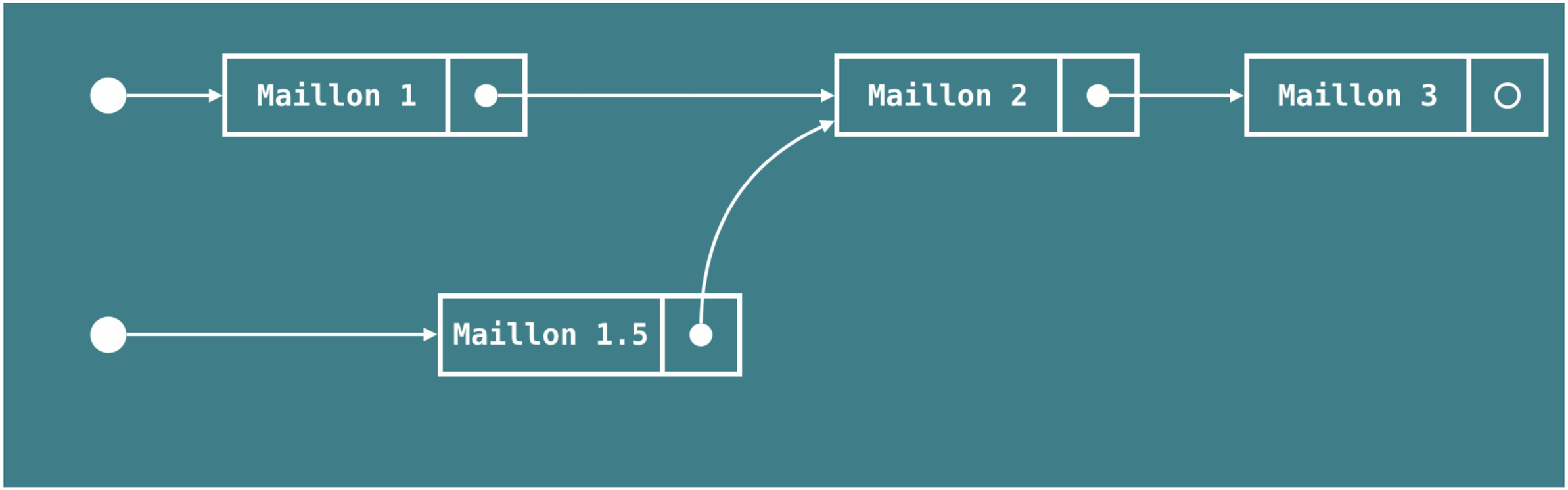
Insertion en milieu de liste



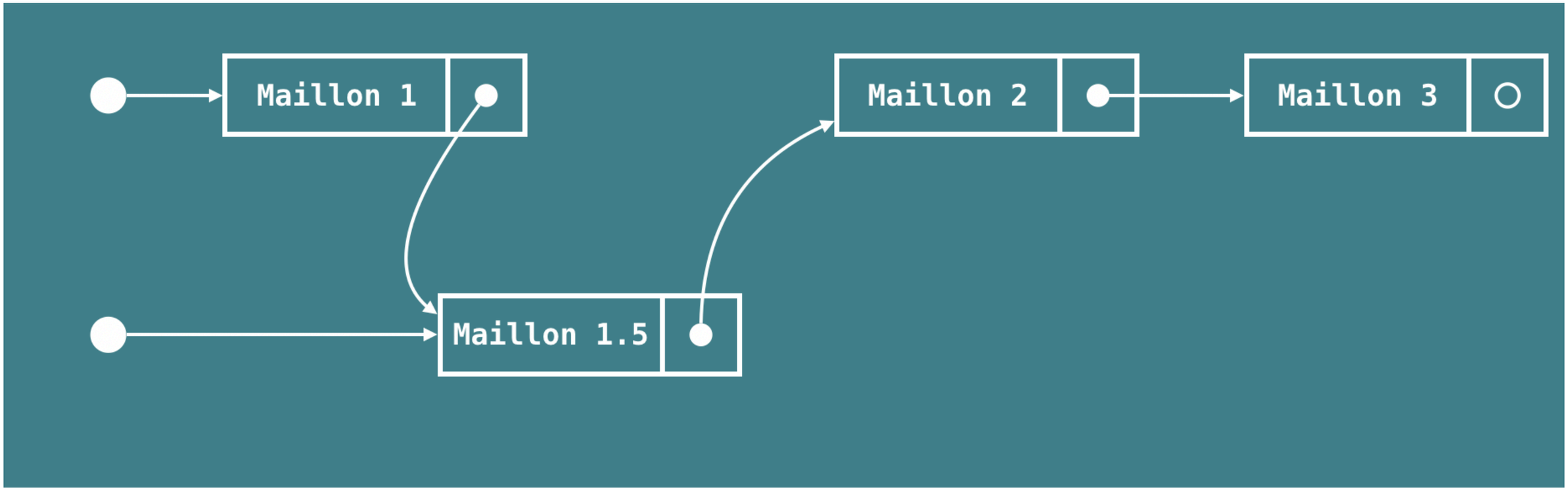
Insertion en milieu de liste



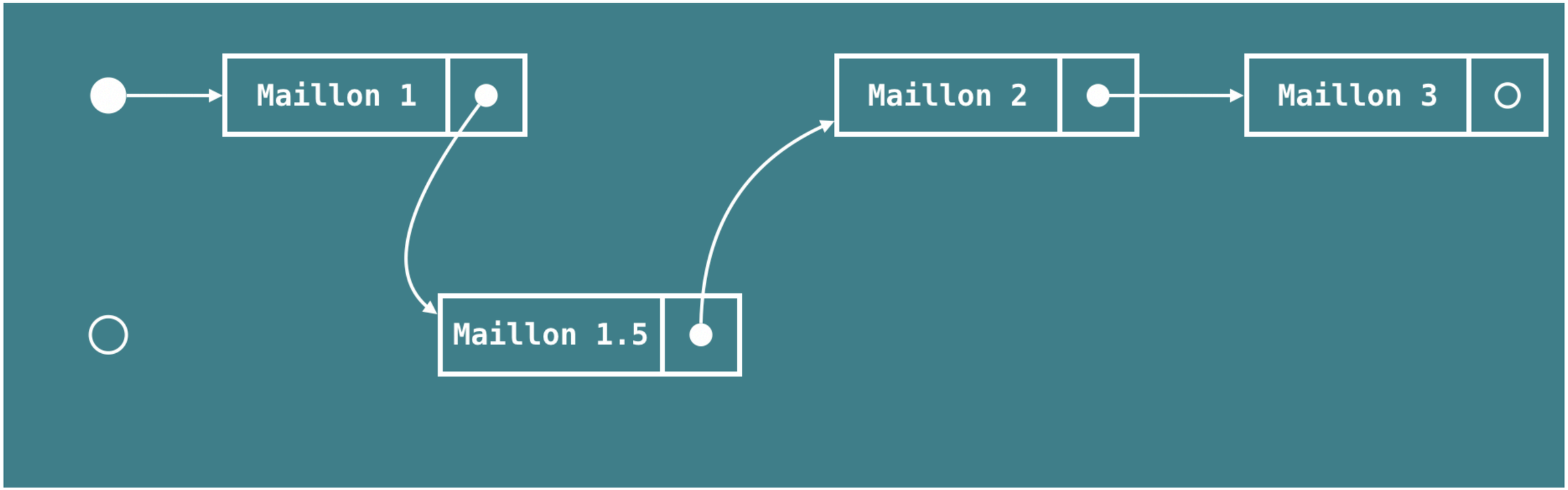
Insertion en milieu de liste



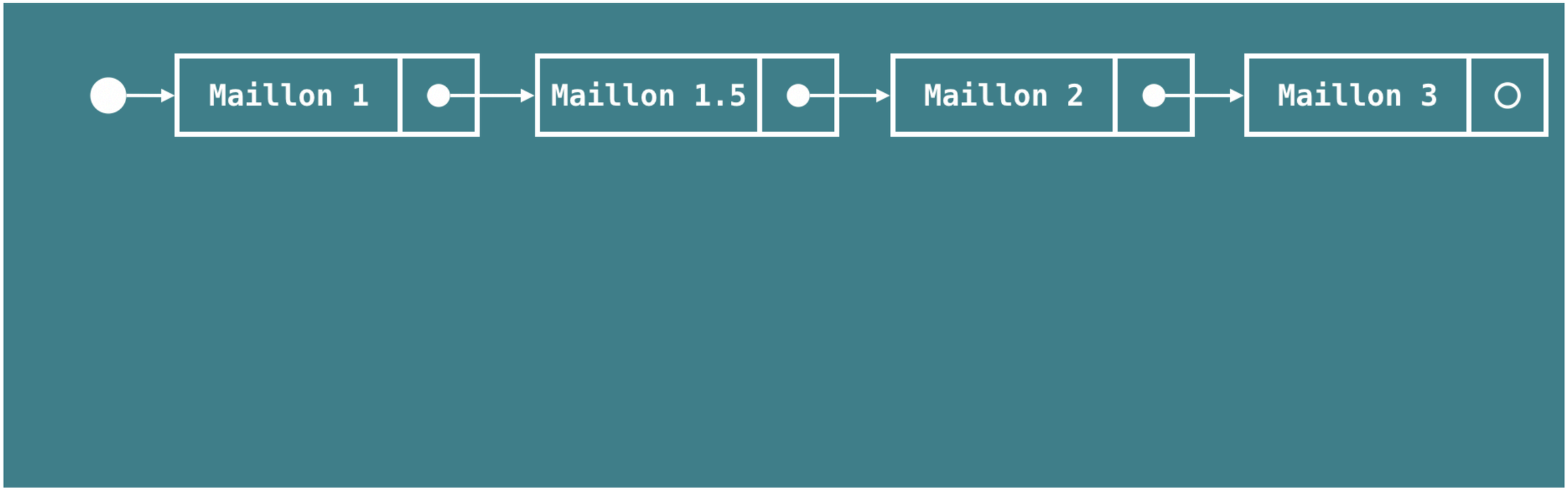
Insertion en milieu de liste



Insertion en milieu de liste



Insertion en milieu de liste



Ce qu'il faut retenir

Attention à :

- Faire le bon choix de structure de données (tableau, liste chaînée...)
- Ne pas perdre de cellules (mauvaise manipulation des pointeurs) 🦮
- Libérer toute la mémoire
- Savoir si on s'arrête quand `curseur == NULL` ou quand `curseur->next == NULL`.

Solution :

-  Faire des schémas !

Astuces

Pour éviter de reparcourir à chaque fois toutes les cellules pour accéder à la dernière, on peut créer un pointeur qui pointe toujours directement sur celle-ci.