# 5IPRO

# Principes algorithmiques et programmation

Benjamin Delbar

Benjamin Delbar

# Cours 16

Listes chaînées

- Exercices

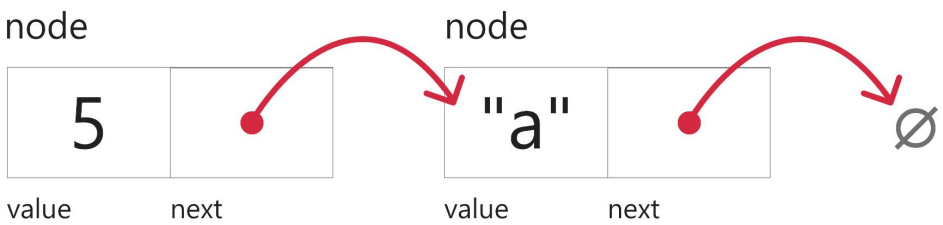
# Les Listes chaînées

Structure de donnée

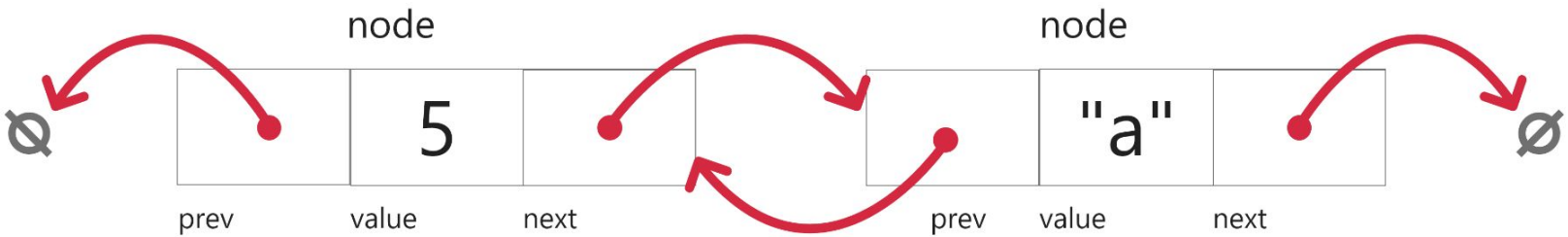
Séquences d’éléments (nodes) , ou chaque node lie le node suivant

Comme dans un tableau, les éléments peuvent être de tout type (string, number, boolean, etc...)

# Les Listes chaînées



# Les Listes chaînées



# Listes chaînées

```
let myLinkedList = {
    head: {
        value: “je suis le premier noeud”,
        next: {
            value: “je suis le 2eme noeud”,
            next: null
        }
    }
}
```

# Listes chaînées

Comment parcourir et afficher ma linkedlist?

```
function print(linkedlist) {
    let current = linkedlist.head; //on commence par la head
    while (current) {
        console.log(current.value);
        current = current.next;
    }
}
```

# Listes chaînées

Comment ajouter un élément au début de ma linkedlist?

```
function prepend (node, linkedlist) {
  node.next = linkedlist.head; //l'ancienne head devient le next
  linkedlist.head = node; //ce node est maintenant la head
  return linkedlist;
}
```

# Listes chaînées

Comment ajouter un élément n'importe où dans ma liste?

```
function insert_at(index, node, linkedlist) {
  if (index == 0) {
    return prepend(node, linkedlist);
  }
  //… a vous de jouer :)
}
```

# Listes chaînées

Comment attribuer les fonctions à ma liste chaînée ?

On peut ajouter une fonction comme attribut de notre objet linkedlist

```
linkedlist.print = function() {
    let current = this.head; //this fait référence à la linkedlist
    while (current) {
        console.log(current.value);
        current = current.next;
    }
}
```

# Listes chaînées

```
function print(linkedlist) {
  let current = linkedlist.head;
   while (current) {
        console.log(current.value);
        current = current.next;
    }

}

print(linkedlist);
```

```
linkedlist.print = function () {
    let current = this.head;
    while (current) {
        console.log(current.value);
        current = current.next;
    }

}

linkedlist.print();
```

# Listes chaînées

Réalisez la fonction **create_node** : qui retourne un nouveau noeud

Réalisez la fonction **insert_at** : pour insérer un nouveau noeud dans votre liste à l'endroit choisi

Réalisez la fonction **remove_first** : pour enlever le premier noeud de votre liste

Réalisez la fonction **remove_at** : pour enlever un noeud dans votre liste à l'endroit choisi

Réalisez la fonction **size** : pour compter le nombre des noeuds dans votre liste

Réalisez la fonction **read** : pour lire un élément de la liste à l'endroit choisi

Réalisez la fonction **append** : pour insérer un nouveau noeud à la fin de votre liste

Réalisez la fonction **remove_last** : pour enlever le dernier noeud de votre liste

# Listes chaînées

Réaliser une fonction permettant de **rechercher** un noeud par valeur et non par index

Réaliser une fonction permettant de faire un **tri** (à bulle/insertion/sélection) sur votre liste

Réaliser une fonction permettant de **fusionner** 2 listes chaînées

Réaliser une fonction permettant **d'enlever** les noeuds contenant une **valeur** donnée

# Listes doublement chaînées

Réaliser une liste **doublement chaînée** pouvant profiter de vos fonctions

=> Ajouter une tail, qui garde en mémoire le noeud en dernière position
=> Ajouter un attribut prev sur chaque noeud, qui garde en mémoire le noeud précédent

- Modifier vos fonctions append & remove_last pour utiliser le tail (fin de la liste)

- Modifier vos autres fonctions pouvant profiter de votre double lien