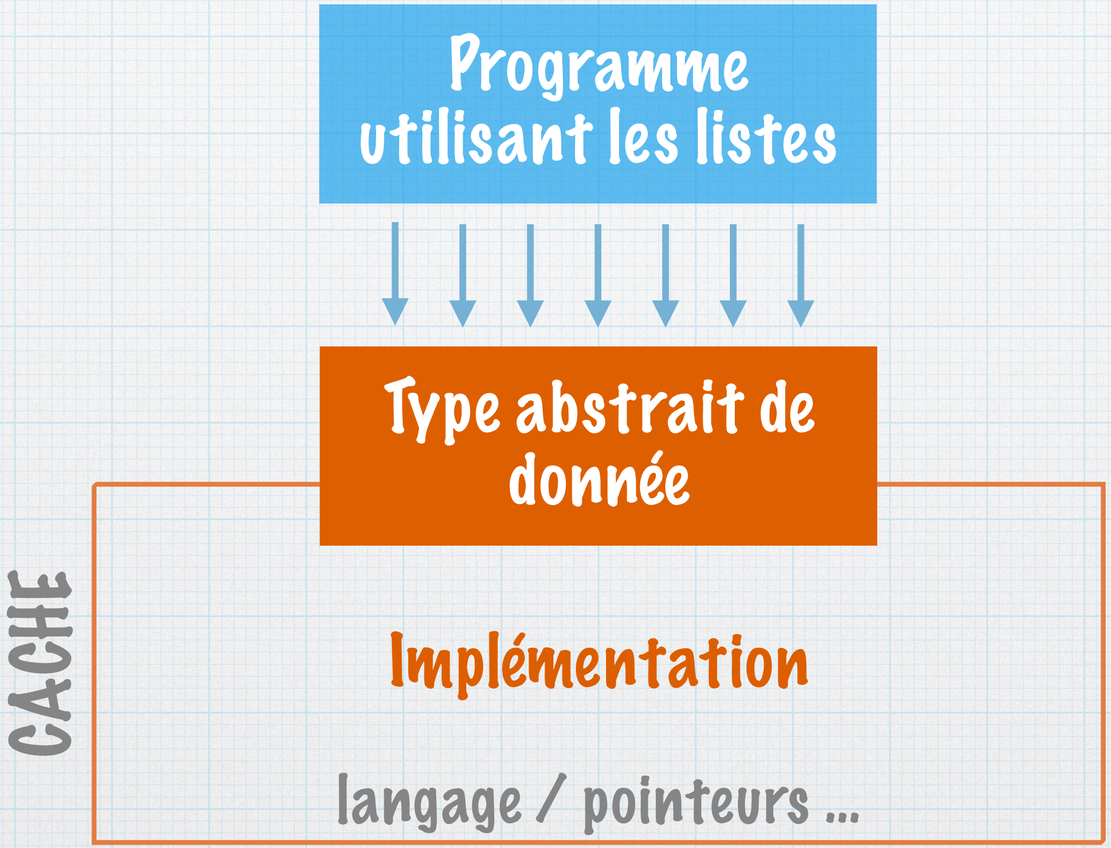
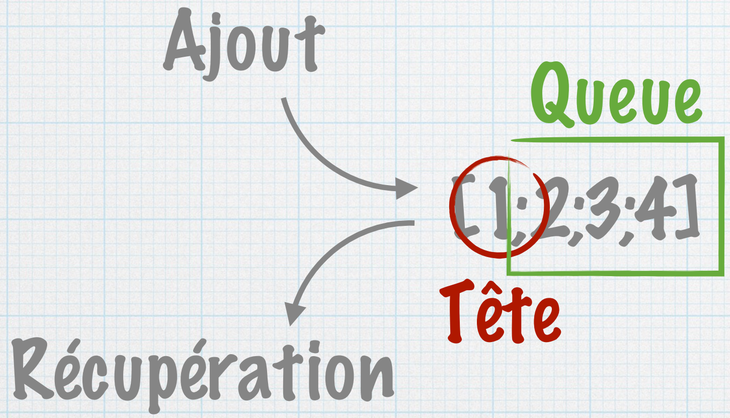
Algorithmique et structures de données

Cours 2 : listes

Structures de données : type abstrait et implémentation



Listes (intuition)

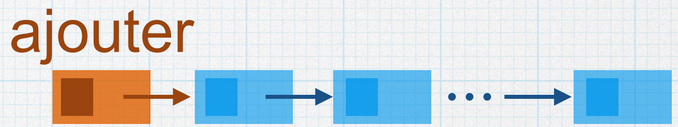


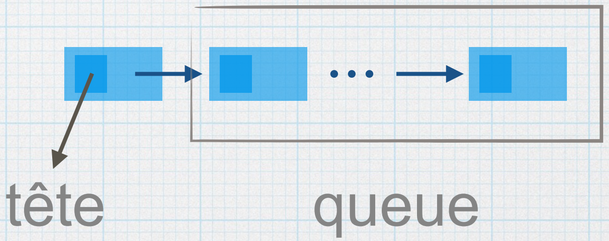
Données stockées séquentiellement
Accès par la tête

Type abstrait de données Listes

Type abstrait Liste

- Nom : Liste
- Utilise : Element, Bool
- Opérations
  - Constructueurs
    - Creer_liste_vide : void -> Liste
  - Transformateurs
    - Ajouter : Element, Liste -> Liste
    - Tête : Liste -> Element
    - Queue : Liste -> Liste
- Observateurs
  - Est_liste_vide : Liste -> Bool
- Axiomes
  - Est_liste_vide(Creer_liste_vide()) == VRAI
  - Est_liste_vide(Ajouter(e,l)) == FAUX
  - Tête(Ajouter(e,l)) == e
  - Queue(Ajouter(e,l)) == l
- Préconditions
  - Tête(l) : NON(est_liste_vide(l)
  - Queue(l) : NON(est_liste_vide(l)





Exemples de fonctions sur les listes

Longueur d'une liste

```
int longueur_iterative (Liste l)
{
    int cpt = 0 ;
    while (!est_liste_vide(l))
    {
        l = queue(l) ;
        cpt = cpt+1 ;
    }
    return cpt ;
}
```

```
int longueur_recursive (Liste l)
{
    if (!est_liste_vide(l))
        return 0 ;
    else
        return 1+longueur_recursive(queue(l)) ;
}
```

Insertion d'un élément en ième position

```
Liste insertion (Data e, Liste l, int i)
{
    Liste tmp ;
    if (i == 0)
        return ajouter(e,l) ;
    else
    {
        assert(!est_liste_vide(l)) ;
        tmp = insertion(e, queue(l),i-1) ;
        return ajouter(tete(l),tmp) ;
    }
}
```

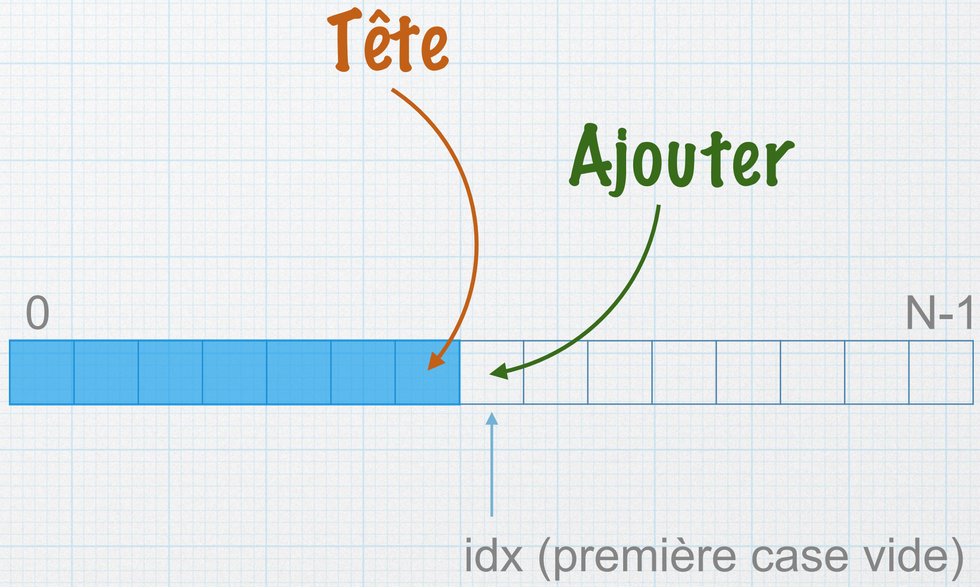
Implémentations du TAD

Liste

Implémentation par tableau (listes bornées)

Liste de taille maximale N

```
typedef int Data ;
struct z_liste {
    Data elt[N] ;
    int idx ; } ;
typedef struct z_liste liste ;
```

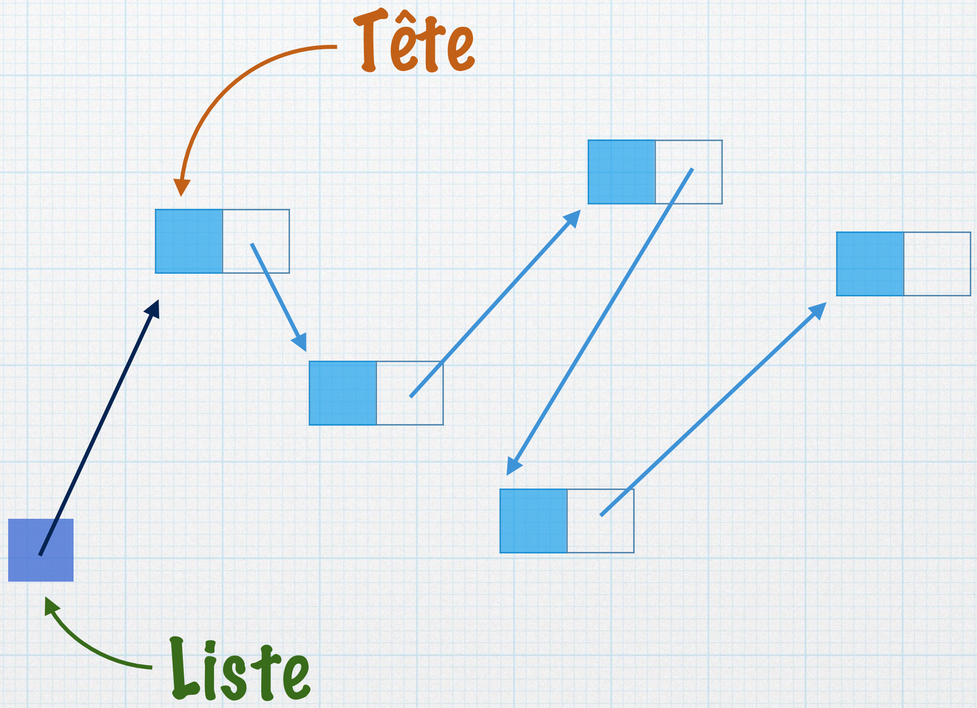


Ajout/suppressions très rapides

Longueur bornée

Implémentation par listes chaînées

```
typedef int Data ;
struct z_maillon {
    Data elt ;
    struct z_maillon * next ; } ;
typedef struct z_maillon maillon ;
typedef maillon * Liste ;
```



Souples, longueur variable

Allocation à chaque ajout