Elements d'Algorithmique
Files

Daniela Petrişan
Université de Paris, IRIF

Files

Les files sont utilisées pour représenter des ensembles dynamiques dans lesquels, s'ils ne sont pas vides, nous ne pouvons supprimer que l'élément qui est présent depuis le plus de temps dans l'ensemble.

La file met en œuvre le principe premier entré, premier sorti, ou FIFO (first in, first out).

Files

Les files sont utilisées pour représenter des ensembles dynamiques dans lesquels, s'ils ne sont pas vides, nous ne pouvons supprimer que l'élément qui est présent depuis le plus de temps dans l'ensemble.

La file met en œuvre le principe premier entré, premier sorti, ou FIFO (first in, first out).

Une file est une structure de données abstraite sur laquelle sont définies trois opérations :

Files

Les files sont utilisées pour représenter des ensembles dynamiques dans lesquels, s'ils ne sont pas vides, nous ne pouvons supprimer que l'élément qui est présent depuis le plus de temps dans l'ensemble.

La file met en œuvre le principe premier entré, premier sorti, ou FIFO (first in, first out).

Une file est une structure de données abstraite sur laquelle sont définies trois opérations :

- `empty(F)` qui teste si la file `F` est vide

Files

Les files sont utilisées pour représenter des ensembles dynamiques dans lesquels, s'ils ne sont pas vides, nous ne pouvons supprimer que l'élément qui est présent depuis le plus de temps dans l'ensemble.

La file met en œuvre le principe premier entré, premier sorti, ou FIFO (first in, first out).

Une file est une structure de données abstraite sur laquelle sont définies trois opérations :

- `empty(F)` qui teste si la file `F` est vide
- `put(x,F)` qui ajoute un élément `x` à la queue de la file `F`. Cette opération est également appelée `enfiler`.

Files

Les files sont utilisées pour représenter des ensembles dynamiques dans lesquels, s'ils ne sont pas vides, nous ne pouvons supprimer que l'élément qui est présent depuis le plus de temps dans l'ensemble.

La file met en œuvre le principe premier entré, premier sorti, ou FIFO (first in, first out).

Une file est une structure de données abstraite sur laquelle sont définies trois opérations :

- `empty(F)` qui teste si la file `F` est vide
- `put(x,F)` qui ajoute un élément `x` à la queue de la file `F`. Cette opération est également appelée `enfiler`.
- `get(F)` qui supprime l'élément en tête de file et le renvoie. Cette opération est aussi appelée `défiler`.

Files: exemple

```
f:= new File();
f.put(1);
f.put(2);
print(f.get());
f.put(3);
f.put(4);
f.put(1);
while(!f.empty()) {
    print(f.get())
}
```

f: >

Files: exemple

```
f:= new File();
f.put(1);
f.put(2);
print(f.get());
f.put(3);
f.put(4);
f.put(1);
while(!f.empty()) {
    print(f.get())
}
```

f: 

>

Files: exemple

```
f:= new File();
f.put(1);
f.put(2);
print(f.get());
f.put(3);
f.put(4);
f.put(1);
while(!f.empty()) {
    print(f.get())
}
```

f: 

>

Files: exemple

```
f:= new File();
f.put(1);
f.put(2);
print(f.get());
f.put(3);
f.put(4);
f.put(1);
while(!f.empty()) {
    print(f.get())
}
```

f:

> 1

Files: exemple

```
f:= new File();
f.put(1);
f.put(2);
print(f.get());
f.put(3);
f.put(4);
f.put(1);
while(!f.empty()) {
    print(f.get())
}
```

f:

> 1

Files: exemple

```
f:= new File();
f.put(1);
f.put(2);
print(f.get());
f.put(3);
f.put(4);
f.put(1);
while(!f.empty()) {
    print(f.get())
}
```

f:

> 1

Files: exemple

```
f:= new File();
f.put(1);
f.put(2);
print(f.get());
f.put(3);
f.put(4);
f.put(1);
while(!f.empty()) {
    print(f.get())
}
```

f:

> 1

Files: exemple

```
f:= new File();
f.put(1);
f.put(2);
print(f.get());
f.put(3);
f.put(4);
f.put(1);
while(!f.empty()) {
    print(f.get())
}
```

f:

> 1

Files: exemple

```
f:= new File();
f.put(1);
f.put(2);
print(f.get());
f.put(3);
f.put(4);
f.put(1);
while(!f.empty()) {
    print(f.get())
}
```

f:

> 1 2

Files: exemple

```
f:= new File();
f.put(1);
f.put(2);
print(f.get());
f.put(3);
f.put(4);
f.put(1);
while(!f.empty()) {
    print(f.get())
}
```

f: 

> 1 2

Files: exemple

```
f:= new File();
f.put(1);
f.put(2);
print(f.get());
f.put(3);
f.put(4);
f.put(1);
while(!f.empty()) {
    print(f.get())
}
```

f: 

> 1 2 3

Files: exemple

```
f:= new File();
f.put(1);
f.put(2);
print(f.get());
f.put(3);
f.put(4);
f.put(1);
while(!f.empty()) {
    print(f.get())
}
```

f: 

> 1 2 3

Files: exemple

```
f:= new File();
f.put(1);
f.put(2);
print(f.get());
f.put(3);
f.put(4);
f.put(1);
while(!f.empty()) {
    print(f.get())
}
```

f: 

> 1 2 3 4

Files: exemple

```
f:= new File();
f.put(1);
f.put(2);
print(f.get());
f.put(3);
f.put(4);
f.put(1);
while(!f.empty()) {
    print(f.get())
}
```

f: 


```
> 1 2 3 4
```

Files: exemple

```
f:= new File();
f.put(1);
f.put(2);
print(f.get());
f.put(3);
f.put(4);
f.put(1);
while(!f.empty()) {
    print(f.get())
}
```

f: 

> 1 2 3 4 1

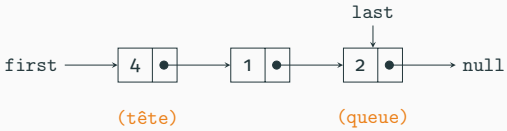
Files: exemple

```
f:= new File();
f.put(1);
f.put(2);
print(f.get());
f.put(3);
f.put(4);
f.put(1);
while(!f.empty()) {
    print(f.get())
}
```

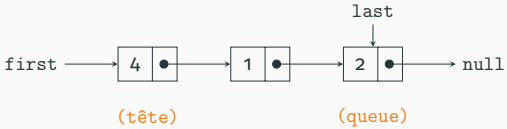
f: 

> 1 2 3 4 1

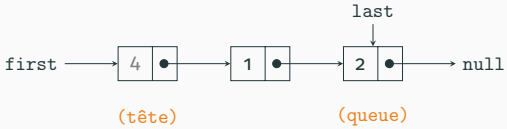
Files : implémentation à l'aide de listes chaînées



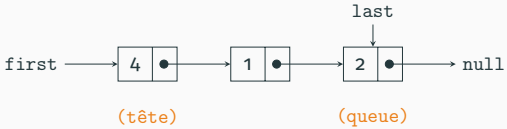
Files : implémentation à l'aide de listes chaînées



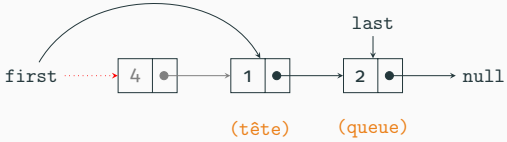
Défiler ou `get()`



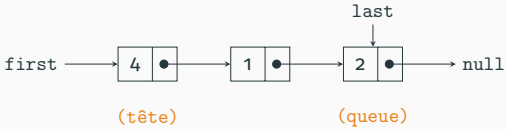
Files : implémentation à l'aide de listes chaînées



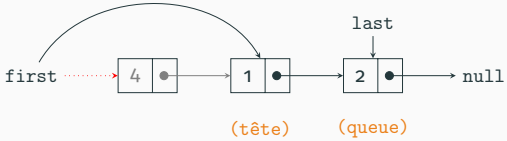
Défiler ou `get()`



Files : implémentation à l'aide de listes chaînées

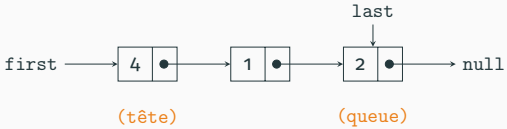


Défiler ou `get()`

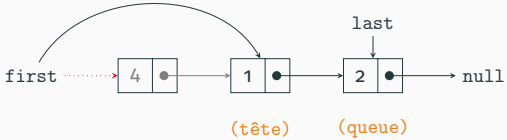


renvoie 4

Files : implémentation à l'aide de listes chaînées



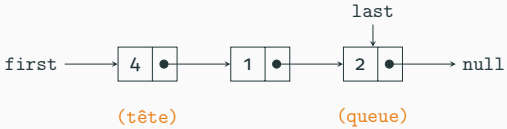
Défiler ou `get()`



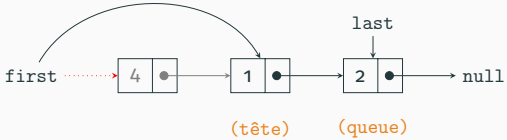
`renvoie 4`

Si `first = last` ?

Files : implémentation à l'aide de listes chaînées



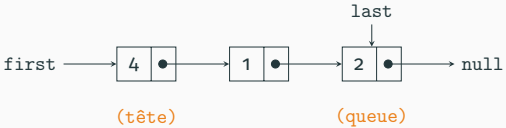
Défiler ou `get()`



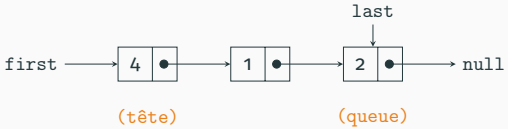
`renvoie 4`

Si `first = last` ?

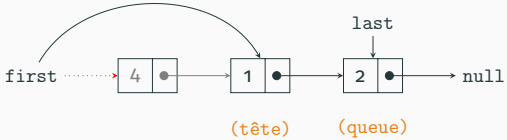
Enfiler ou `put(x)`



Files : implémentation à l'aide de listes chaînées



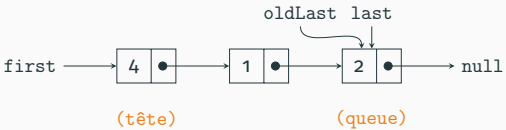
Défiler ou `get()`



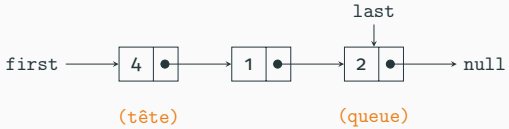
`renvoie 4`

Si `first = last` ?

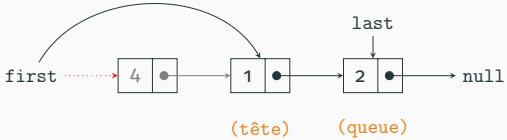
Enfiler ou `put(x)`



Files : implémentation à l'aide de listes chaînées



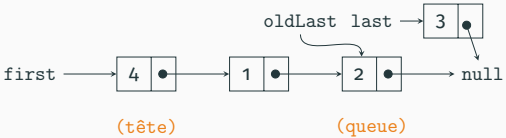
Défiler ou `get()`



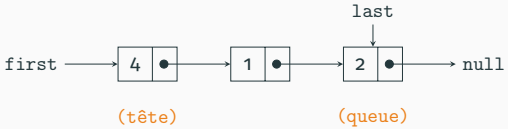
`renvoie 4`

Si `first = last` ?

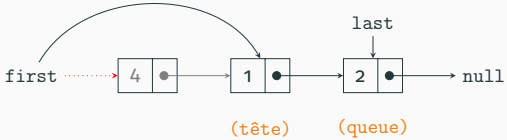
Enfiler ou `put(x)`



Files : implémentation à l'aide de listes chaînées



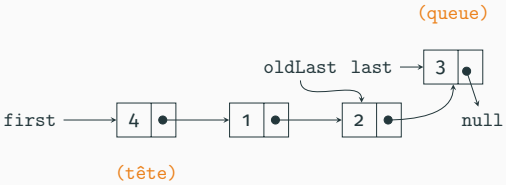
Défiler ou `get()`



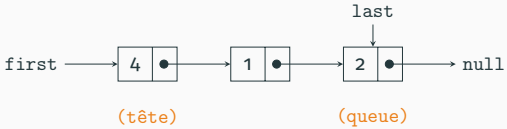
`renvoie 4`

Si `first = last` ?

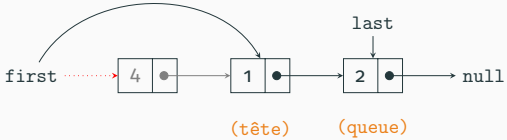
Enfiler ou `put(x)`



Files : implémentation à l'aide de listes chaînées



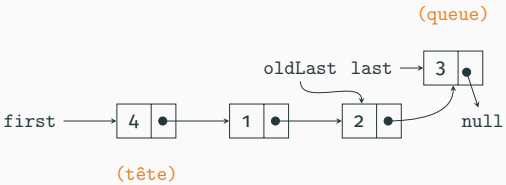
Défiler ou `get()`



`renvoie 4`

Si `first = last` ?

Enfiler ou `put(x)`



Si la liste initiale était vide ?