

Plan

- Algorithmme
- Types abstraits de données
- Pointeurs
- Listes
- **Piles**
- Files
- Tris
- Complexité

01/09/2006

1

Piles > Définition

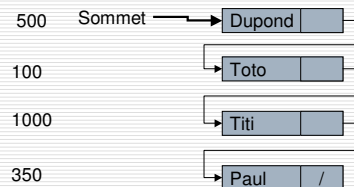
- Une pile est une structure de données telle que :
 - L'ajout d'un élément se fait au sommet de la pile.
 - La suppression d'un élément se fait également au sommet de la pile.
- C'est une structure de données en LIFO « Last In, First out » ou « dernier entré, premier sorti ».

01/09/2006

2

Piles > Représentation en mémoire

Adresses



01/09/2006

3

Piles > Opérations

- Créer_pile : \Rightarrow Pile
- Empiler : Pile \otimes Element \Rightarrow Pile
- Dépiler : Pile \Rightarrow Pile
- Sommet : Pile \Rightarrow Element
- Pile_vide : Pile \Rightarrow Booléen

01/09/2006

4

Piles > Algorithme avec allocation dynamique > Structure de données

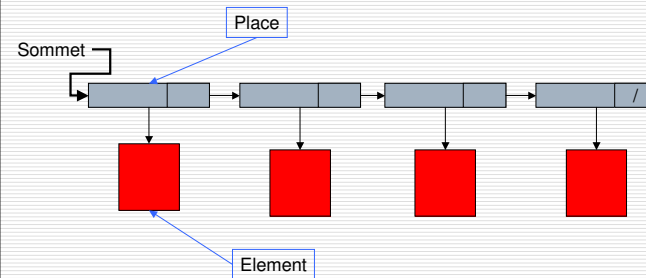
```
Enregistrement Place{  
  Elt : Pointeur[Element];  
  Suivant : Pointeur[Place];  
}
```

```
Enregistrement Pile {  
  Sommet : Pointeur[Place];  
}
```

01/09/2006

5

Piles > Algorithme avec allocation dynamique > Représentation en mémoire



01/09/2006

6

Piles > Algorithme avec allocation dynamique > Créer_pile



01/09/2006

7

Piles > Algorithme avec allocation dynamique > Créer_pile

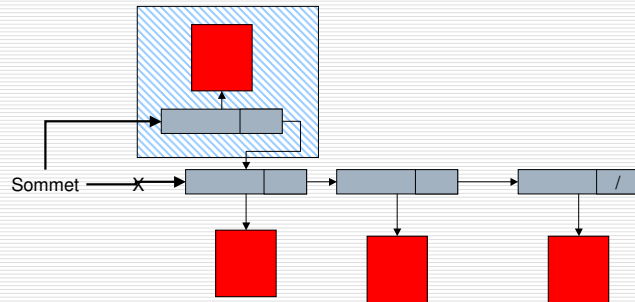
■ Créer_pile : \Rightarrow Pile

```
Fonction créer_pile : Pile;{  
  Resultat : Pile;  
  Pile->Sommet = null;  
}
```

01/09/2006

8

Piles > Algorithme avec allocation dynamique > Empiler



01/09/2006

9

Piles > Algorithme avec allocation dynamique > Empiler

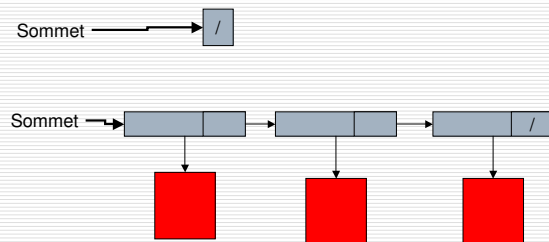
■ Empiler : Pile \otimes Element \Rightarrow Pile

```
Fonction Empiler(P :Pile; Elt : Element) : Pile;{
  Ins : Place;
  Ins->Elt = &Elt;
  Ins->Suivant = P->Sommet;
  P->Sommet = &Ins;
  Retourne P;
}
```

01/09/2006

10

Piles > Algorithme avec allocation dynamique > Pile_vider



01/09/2006

11

Piles > Algorithme avec allocation dynamique > Pile_vider

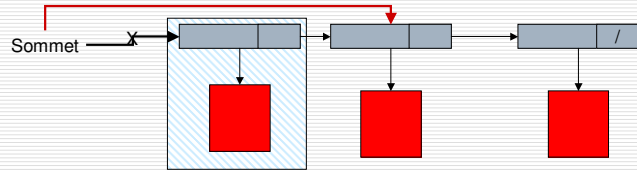
■ Pile_vider : Pile \Rightarrow Booléen

```
Fonction pile_vider (P: Pile) : Booléen;{
  Resultat : Booléen;
  Resultat = (P->Sommet == null);
  Retourne Resultat;
}
```

01/09/2006

12

Piles > Algorithme avec allocation dynamique > Dépiler



01/09/2006

13

Piles > Algorithme avec allocation dynamique > Dépiler

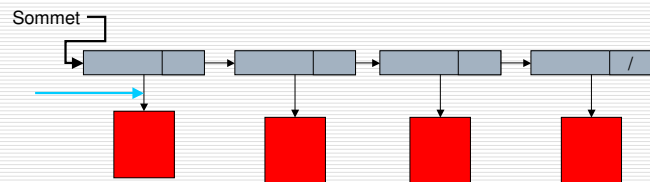
■ Dépiler : Pile \Rightarrow Pile

```
Fonction Dépiler (P : Pile) : Pile;{  
  Si (!Pile_vide(P)) Alors {  
    P->Sommet = Valeur[Pile->Sommet]-  
      >Suivant;  
  }  
  FinSi  
  Retourne P;  
}
```

01/09/2006

14

Piles > Algorithme avec allocation dynamique > Sommet



01/09/2006

15

Piles > Algorithme avec allocation dynamique > Sommet

■ Sommet : Pile \Rightarrow Element

```
Fonction Sommet (P :Pile) : Pointeur [Element];{  
  Resultat : Pointeur[Element];  
  Si (Pile_vide(P)) Alors {  
    Resultat = null  
  }  
  Sinon {  
    Resultat = valeur[P->Sommet]->Elt;  
  }  
  FinSi  
  Retourne resultat  
}
```

01/09/2006

16

Plan

- Algorithmme
- Types abstraits de données
- Pointeurs
- Listes
- Piles
- Files**
- Tris
- Complexité

01/09/2006

17

Files > Définition

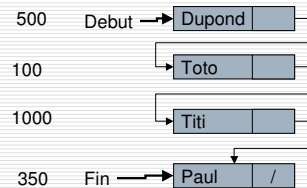
- Une file est une structure de données telle que :
 - L'ajout d'un élément se fait à la fin de la file.
 - La suppression d'un élément se fait au début de la file.
- C'est une structure de données en FIFO « First In, First out » ou « premier entré, premier sorti ».

01/09/2006

18

Files > Représentation en mémoire > Représentation en mémoire

Adresses



01/09/2006

19

Files > Opérations

- Créer_file : \Rightarrow file
- Ajouter : file \otimes Element \Rightarrow file
- Retirer : file \Rightarrow file
- Premier : file \Rightarrow Element
- File_vide : File \Rightarrow Booléen

01/09/2006

20

Files > Algorithme avec allocation dynamique > Structure de données

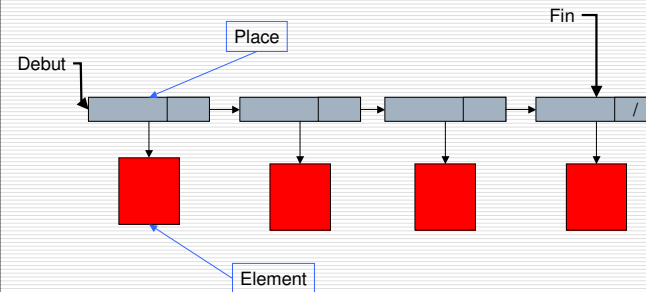
```
Enregistrement Place{  
  Elt : Pointeur[Element];  
  Suivant : Pointeur[Place];  
}
```

```
Enregistrement Liste {  
  Debut : Pointeur[Place];  
  Fin : Pointeur[Place];  
}
```

01/09/2006

21

Files > Algorithme avec allocation dynamique > Représentation en mémoire



01/09/2006

22

Files > Algorithme avec allocation dynamique > Créer_file



01/09/2006

23

Files > Algorithme avec allocation dynamique > Créer_file

■ Créer_file : \Rightarrow file

```
Fonction Créer_file : File; {  
  Resultat : File;  
  Resultat ->Debut = null;  
  Resultat ->Fin = null;  
  Retourne Resultat;  
}
```

01/09/2006

24

Files > Algorithme avec allocation dynamique > File_vider



01/09/2006

25

Files > Algorithme avec allocation dynamique > File_vider

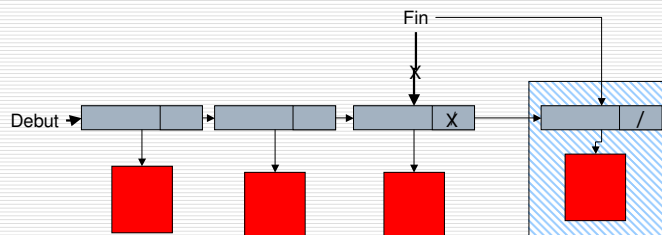
- File_vider : File \Rightarrow Booléen

```
Fonction File_Vide(F: File) : Booléen;{  
  Retourne F->(Debut == null);  
}
```

01/09/2006

26

Files > Algorithme avec allocation dynamique > Ajouter



01/09/2006

27

Files > Algorithme avec allocation dynamique > Ajouter

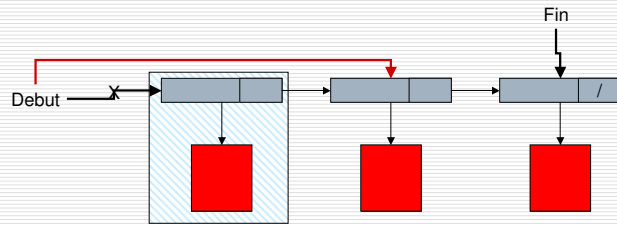
- Ajouter : file \otimes Element \Rightarrow file

```
Fonction Ajouter(F : File; Elt : Element) : File;{  
  Ins : Place;  
  Ins->Elt = &Elt;  
  Ins->Suivant = null;  
  
  Si (File_Vide(F)) Alors{  
    F->Debut = &Ins  
  }  
  Sinon {  
    Valeur[F->Fin]->Suivant = &Ins;  
  }  
  FinSi  
  
  F->Fin = &Ins;  
  
  Retourne F;  
}
```

01/09/2006

28

Files > Algorithme avec allocation dynamique > Retirer



01/09/2006

29

Files > Algorithme avec allocation dynamique > Retirer

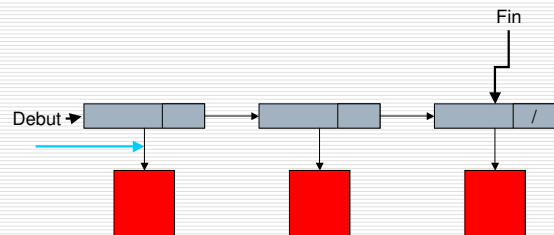
- Retirer : file \Rightarrow file

```
Fonction Retirer(F :File) : File;{  
  Si (!File_Vide(F)){  
    F->Debut = Valeur[F->Debut]->Suivant;  
    Si (F->Debut == null){  
      F->Fin = null;  
    }  
    FinSi  
  }  
  FinSi  
  Retourne F;  
}
```

01/09/2006

30

Files > Algorithme avec allocation dynamique > Premier



01/09/2006

31

Files > Algorithme avec allocation dynamique > Premier

- Premier : file \Rightarrow Element

```
Fonction Premier (F : File) : Pointeur[Element];{  
  Resultat:Pointeur[Element];  
  Si (File_vide(F)) Alors {  
    Resultat = null;  
  }  
  Sinon {  
    Resultat = Valeur[F->Debut]->Elt;  
  }  
  Retourne Resultat;  
}
```

01/09/2006

32