

Filière: MPI

Module: Programmation II

Plan du cours

Chapitre I. Les structures séquentielles (pile, file)

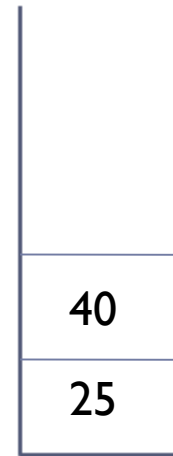
Plan du cours

Partie I. Les piles

I.1- Définition d'une pile

- Une pile est une structure de données de type LIFO (last in first out) : le dernier entré est le premier sorti.
- Caractéristiques d'une pile:
 - L'ajout d'un élément se fait toujours à la tête. Cette opération s'appelle empilement.
 - La suppression d'un élément se fait toujours à la tête. Cette opération s'appelle dépilement

Pile vide

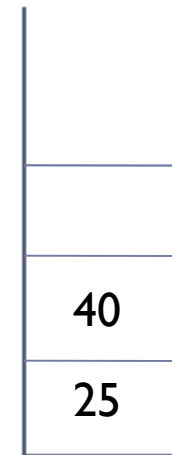
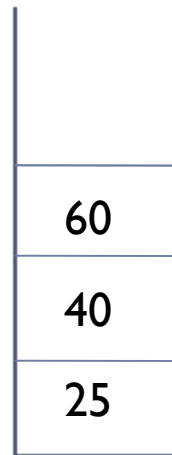
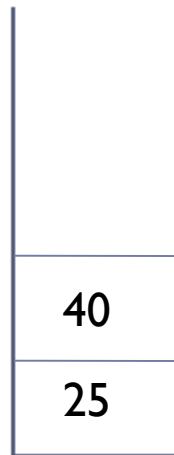


Tête

Queue

Empiler 60

Dépiler



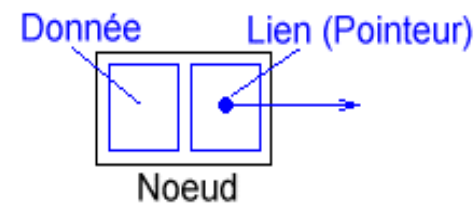
Exemple d'une pile

Les opérations de base qui peuvent être appliquées sont:

- Ajouter un élément : empiler
- Supprimer un élément : dépiler

1.2- Représentation chaînée d'une pile

- L'élément de base d'une liste chaînée s'appelle le nœud. Il est constitué :
- d'un champ de données ;
 - d'un pointeur vers un nœud.



I.3-Définition en C

Définition en C de la structure Nœud

```
typedef struct Nœud  
{  
  int valeur;  
  struct Noeud * suivant;  
} Noeud;
```

Définition en C d'une liste chaînée

```
Typedef Noeud * pile;
```

Les opérations sur une pile

créer une pile : *pile creer(void);*

Tester si une pile est vide : *int estvide(pile pil);*

Empiler un élément dans une pile : *pile empiler(pile pil int val);*

Dépiler un élément d'une pile : *pile depiler(pile pil);*

Déterminer la tête de la pile: *int tete(pile pil)*

- programmes en C des opérations d'une pile

```
pile creer(void)
{
    return NULL ;
}
```

```
int estvide(pile pil)
{
    if (pil==NULL)
        return (1);
    else
        return (0);
}
```

```
pile empiler(pile pil, int valeur)
{ pile pile_i;
  if ((pile_i = (pile)malloc(sizeof(Noeud))) == NULL)
  { printf ("erreur allocation");
    exit(1);
  }
  pile_i->valeur = valeur;
  pile_i->suivant = pil;
  return(pile_i);
}
```

```
pile depiler(pile pil)
{pile pil_enleve;
if (pil==NULL)
{ printf ("erreur la pile est vide");
exit (1);
}
pil_enleve = pil;
pil = pil →suivant;
free(pil_enleve);
return pil;
}
```

```
int tete(pile pil)
{

    int value;

    if (pil == NULL)
    {printf ("erreur la pile est vide");
    exit (1);
    }

    value= pil → valeur;
    return value;
}
```

Plan du cours

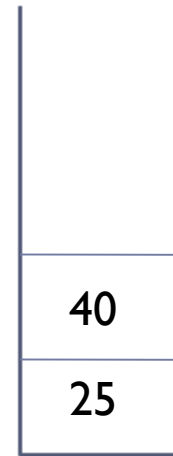
Partie 2. Les files

2.1- Définition

Une file est une structure de données de type FIFO (first in first out) : le premier entré est le premier sorti.

- Caractéristiques d'une file:
 - L'ajout d'un élément se fait toujours à la queue. Cette opération s'appelle enfilement.
 - La suppression d'un élément se fait toujours à la tête. Cette opération s'appelle défilement

File vide

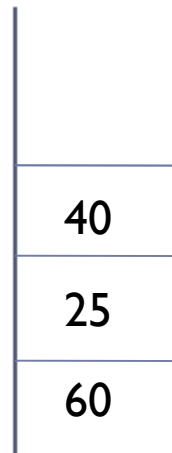
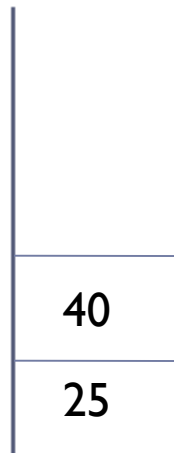


Tête

Queue

Enfiler 60

Défiler



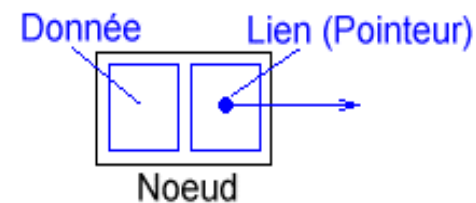
Exemple d'une file

Les opérations de base qui peuvent être appliquées sont:

- Ajouter un élément : enfiler
- Supprimer un élément : défiler

2.2- Représentation chaînée d'une file

- L'élément de base d'une liste chaînée s'appelle le nœud. Il est constitué :
- d'un champ de données ;
 - d'un pointeur vers un nœud.



I.3-Définition en C

Définition en C de la structure Nœud

```
typedef struct Nœud  
{  
  int valeur;  
  struct Noeud * suivant;  
} Noeud;
```

Définition en C d'une liste chaînée

```
Typedef Noeud * file;
```

Les opérations sur une file

créer une file : file creer(void);

Tester si une file est vide : int estvide(file fil);

Enfiler un élément dans une file : file enfiler(file fil int val);

Défiler un élément d'une file : file defiler(file fil);

Déterminer la tête de la file: int tete(file fil)

- programmes en C des opérations d'une file

```
file creer(void)
{
    return NULL ;
}
```

```
int estvide(file fil)
{
    if (fil==NULL)
        return (1);
    else
        return (0);
}
```

```

file enfiler(file fil int, valeur)
{
file file_i, file_move;
if((file_i = (file )malloc(sizeof(Noeud))) == NULL)
{ printf(" erreur allocation "); exit(1); }
file_i → valeur = valeur;
file_i → suivant= NULL;
if(fil == NULL)
{
return(file_i);
}
else {
file_move= fil;
while (file_move → suivant!=NULL)
file_move=file_move → suivant;
file_move → suivant = file_i;
return(fil);
}
}

```

```
file defiler(file fil)
{file fil_enleve;
if (fil==NULL)
{ printf ("erreur la file est vide");
exit (1);
}
fil_enleve = fil;
fil = fil →suivant;
free(fil_enleve);
return fil;
}
```

```
int tete(file fil)
{

    int value;

    if (fil == NULL)
    {printf ("erreur la pile est vide");
    exit (1);
    }

    value = fil → valeur;
    return value;
}
```