Chapitre 6
Structures de données dynamiques: Pile & File

Module Structures de données et programmation C

2éme ANNEE LEESM

mlahby@gmail.com

5 mai 2021

Plan

La structure pile

Définition

- Une pile (stack en anglais) est une structure dynamique dans laquelle l'insertion au la suppression d'un élément s'effectue toujours à partir de la même extrémité de cette structure.
- Cette extrémité est appelée le sommet de la pile

.

Le mécanisme LIFO (last in, first out)

- Une pile permet de modéliser un système régi par le mécanisme « dernier arrivé premier sorti » ; on dit souvent LIFO (last in, first out)
- L'action consistant à ajouter un nouvelle élément au sommet de la pile s'appelle empiler ; celle consistant à retirer l'élément situé au sommet de la pile s'appelle dépiler

.

FIG.: Pile d'd'assiettes

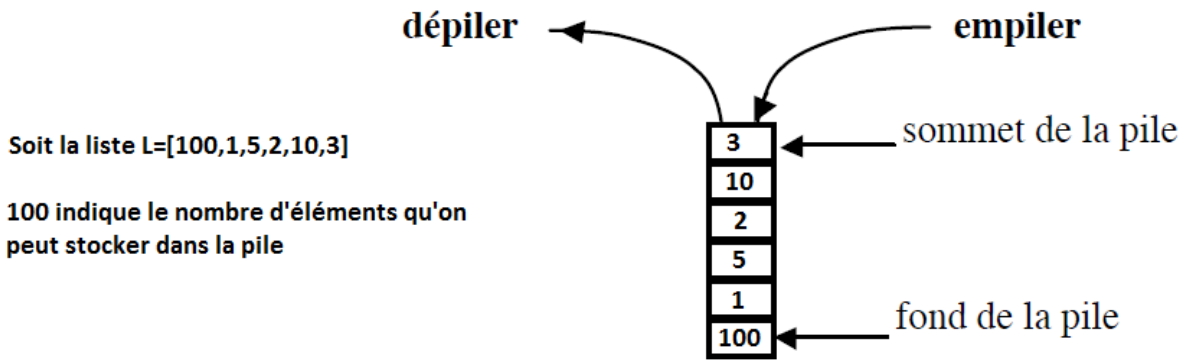
Les applications d'une pile

De nombreuse applications s'appuient sur l'utilisation d'une pile, on peut citer :

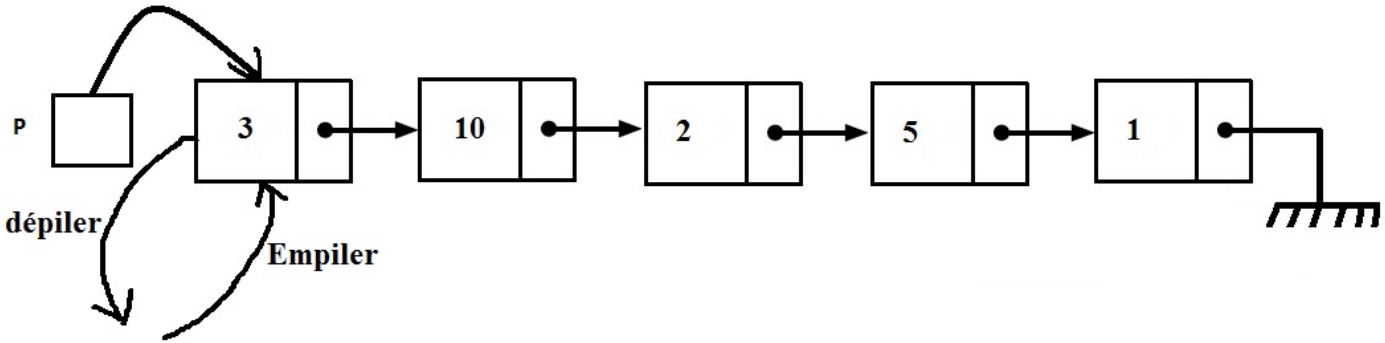
- Dans un navigateur web, une pile sert à mémoriser les pages Web visitées. L'adresse de chaque nouvelle page visitée est empilée et l'utilisateur dépile l'adresse de la page précédente en cliquant le bouton « Afficher la page précédente ».
- L'évaluation des expressions mathématiques en notation post-fixée (ou polonaise inverse) utilise une pile.
- La fonction « Annuler la frappe » (en anglais « Undo ») d'un traitement de texte mémorise les modifications apportées au texte dans une pile.
- Vérification de parenthèsage d'une chaine de caractères ;
- La récursivité (une fonction qui fait appel à elle même) ;
- etc.

Implémentation d'une pile

- Il existe deux façons pour implémenter une pile :
  1. Soit on utilise la structure tableau pour réaliser une pile ;



  2. Soit on utilise une liste simplement chaînée pour réaliser une pile.



Implémentation d'une pile avec une liste chaînée

Syntaxe pour définifir le type Pile

```
//Définir la strcuture cellule

typedef struct cellule{
  int info ; //le champ info peut avoir n'importe quel type
  struct cellule *suiv ; //pointeur contenant l'adresse de la cellule suivante
}liste ;

//definir le type Pile
typedef liste * Pile ;
```

Déclaration d'une pile

- **Pour déclarer une variable de type Pile, il existe deux possibilités :**
  - En utilisant une variable statique : Pile P.
  - En utilisant une variable dynamique : Pile *P.

Les primitives

- Afin de manipuler une pile, on doit programmer un ensemble des fonctions de gestion d'une pile (primitives).
- Voici les primitives communément utilisées :
  1. PileVide() : créer une liste vide
  2. EstVide() : renvoie vrai si la pile est vide, faux sinon ;
  3. SommetPile(P) : renvoie l'élément sommet de la pile P ;
  4. Empiler(P,v) : ajoute au sommet de la pile P l'élément v ;
  5. Depiler(P) : supprime de la pile le sommet.

Les primitives

Prototype : Pile PileVide()

```
Pile PileVide()
    {
        return(NULL);
    }
```

Les primitives

Prototype : int EstVide(Pile p)

```
int EstVide(Pile p)
  {
   if(p==NULL)
     return (1);
   else
     return (0);
  }
```

Les primitives

Prototype : int SommetPile(P)

```
int Sommet(Pile p)
  {
   if(p==NULL)
     {
       printf("La pile est vide.");
       getchar();
       exit(-1);
     }
   else
     return (p->info);
  }
```

Les primitives

Prototype : Pile Empiler(P,v)

```
Pile Empiler(Pile P,int v)
  {
   Pile c ;
   c=(Pile)malloc(sizeof(liste)) ;
   c− >info=v ;
   c− >suiv=p ;
   p=c ;
   return(p) ;
  }
```

Les primitives

Prototype : Pile Depiler(P)

```
Pile Depiler(P)
  {
   Pile c=p;
   if(c==NULL)
     return(NULL);
   else
     {
       p=p->suiv;
       free(c);
       return(p);
     }
  }
```

La structure file

Définition

- Une file (queue en anglais) est une structure de données dans laquelle l'insertion se fait à la fin et la suppression d'un élément s'effectue à partir de début de cette structure.
- Le fonctionnement ressemble à une file d'attente : les premières personnes à arriver sont les premières personnes à sortir de la file.

.

Le mécanisme FIFO (first in, first out)

- Une file permet de modéliser un système régi par le mécanisme "premier arrivé premier sorti" ; on dit souvent FIFO (first in, first out)
- L'action consistant à ajouter un nouvelle élément s'appelle enfiler ; celle consistant à retirer l'élément situé au début de la file s'appelle défiler

.

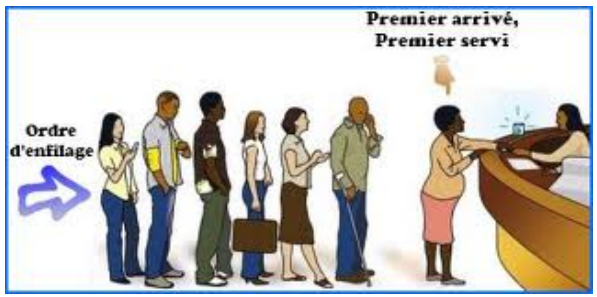


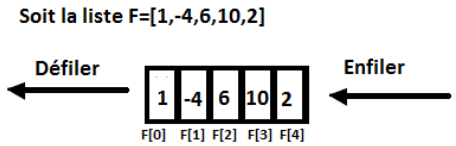
FIG.: file d'attente

Les applications d'une file

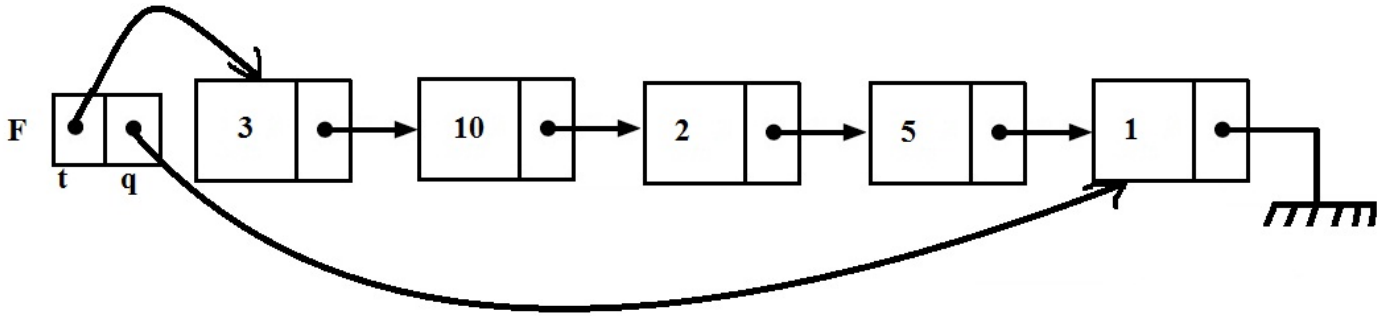
- En général, on utilise des files pour mémoriser temporairement des transactions qui doivent attendre pour être traitées ;
- Les serveurs d'impression, qui doivent traiter les requêtes dans l'ordre dans lequel elles arrivent, et les insèrent dans une file d'attente (ou une queue) ;
- Certains moteurs multitâches, dans un système d'exploitation, qui doivent accorder du temps-machine à chaque tâche, sans en privilégier aucune ;
- Un algorithme de parcours en largeur utilise une file pour mémoriser les noeuds visités ;
- On utilise aussi des files our créer toutes sortes de mémoires tampons (en anglais buffers).
- etc.

Implémentation d'une file

- Il existe deux façons pour implémenter une file :
  1. Soit on utilise un tableau pour réaliser une file ;



  2. Soit on utilise une liste simplement chaînée



Implémentation d'une file avec une liste chaînée

Syntaxe pour définifir le type File

```
//Définir la strcuture cellule

typedef struct cellule{
  int info ; //le champ info peut avoir n'importe quel type
  struct cellule *suiv ; //pointeur contenant l'adresse de la cellule suivante
}liste ;

//definir le type File
typedef struct {
  liste* t ;
  liste* q ;
}File_t ;
```

Déclaration d'une file

- **Pour déclarer une variable de type Pile, il existe deux possibilités :**
  - En utilisant une variable statique : File_t F.
  - En utilisant une variable dynamique : File_t *F.

Les primitives

- Pour résoudre un problème donné qui repose sur la structure file, il est nécessaire de programmer un ensemble des primitives pour la gestion d'une file.
- Voici les primitives communément utilisées :
  1. FileVide() : créer une file vide
  2. EstVide() : renvoie vrai si la file est vide, faux sinon ;
  3. PrmierElement(F) : renvoie le premier élément de la file F ;
  4. Enfiler(P,v) : ajoute à la fin de la file F l'élément v ;
  5. Defiler(F) : supprime de la file F le premier élément.

Les primitives

Prototype : File_t FileVide()

```
File_t FileVide()
  {
    File_t F ;
    F.t=NULL ; ;
    F.q=NULL ;
    return(F) ;
  }
```

Les primitives

Prototype : int EstVide(File_t F)

```
int EstVide(File_t F)
  {
   if(F.t==NULL)
     return (1);
   else
     return (0);
  }
```

Les primitives

Prototype : int premierElement(File_t F)

```
int premierElement(File_t F)
  {
   if(EstVide(F))
     {
       printf("La file est vide.");
       getchar();
       exit(-1);
     }
   else
     return (F.t->info);
  }
```

Les primitives

Prototype : File_t Enfiler(int n,File_t F)

```
File_t Enfiler(int n,File_t F)
   { liste *c;
     c=(liste*)malloc(sizeof(liste));
     c->info=n;
     c->suiv=NULL;
     if(EstVide(F))
       { F.q=c;
         F.t=c; }
     else
       {F.q->suiv=c;
         F.q=c; }
     return(F);
   }
```

Les primitives

Prototype : File_t Defiler(File_t F)

```
File_t Defiler(File_t F)
  { liste *c;
   if(EstVide(F))
     { printf("La file est vide.");
       exit(-1);
     }
   else
     { c=F.t;
       F.t=F.t->suiv;
       free(c);
       return(F);
     }
  }
```