Les piles et les files

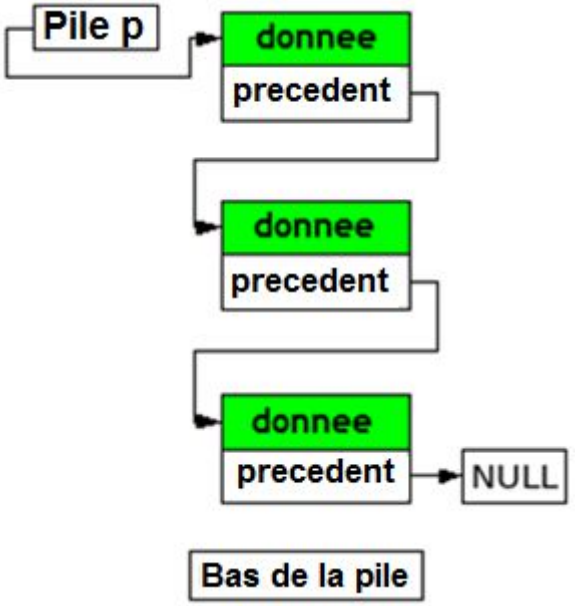
Aziza EL OUAAZIZI
Cours SMIA S4
Faculté Polydisciplinaire de Taza
Université Sidi Mohammed Ben Abdellah

Introduction

- Les piles (*stack*) et les files (*queue*) constituent deux **structures de données** particuliers.
- Elles permettent de stocker diverses données comme pourrait le faire un tableau mais en respectant un critère particulier pour les gérer.
- Les piles sont utilisées pour stocker les valeurs des variables locales et peuvent être utilisées dans des algorithmes d'évaluation d'expressions mathématiques.
- Les files sont utilisées généralement pour mémoriser des données en attente de traitement.

Les piles

- Une pile permet de réaliser une **LIFO** (***L**ast **I**n **F**irst **O**ut*): les derniers éléments ajoutés à la pile sont les premiers à être récupérés.
- Il est possible de comparer cette structure à une pile d'assiettes: lorsqu'on ajoute une assiette en haut de la pile, on retire toujours en premier celle qui se trouve en haut de la pile, sinon tout le reste s'écroule.



Implémentation d'une pile

L'insertion dans une liste se faisant toujours au début de la liste, le 1er élément de la liste sera le dernier élément saisi, donc sa position est en haut de la pile.

```
typedef struct element
{
  int donnee;                    /*La donnée que notre pile stockera*/
  struct element *precedent;  /*Pointeur vers l'élément précédent de la pile*/
} element;
```

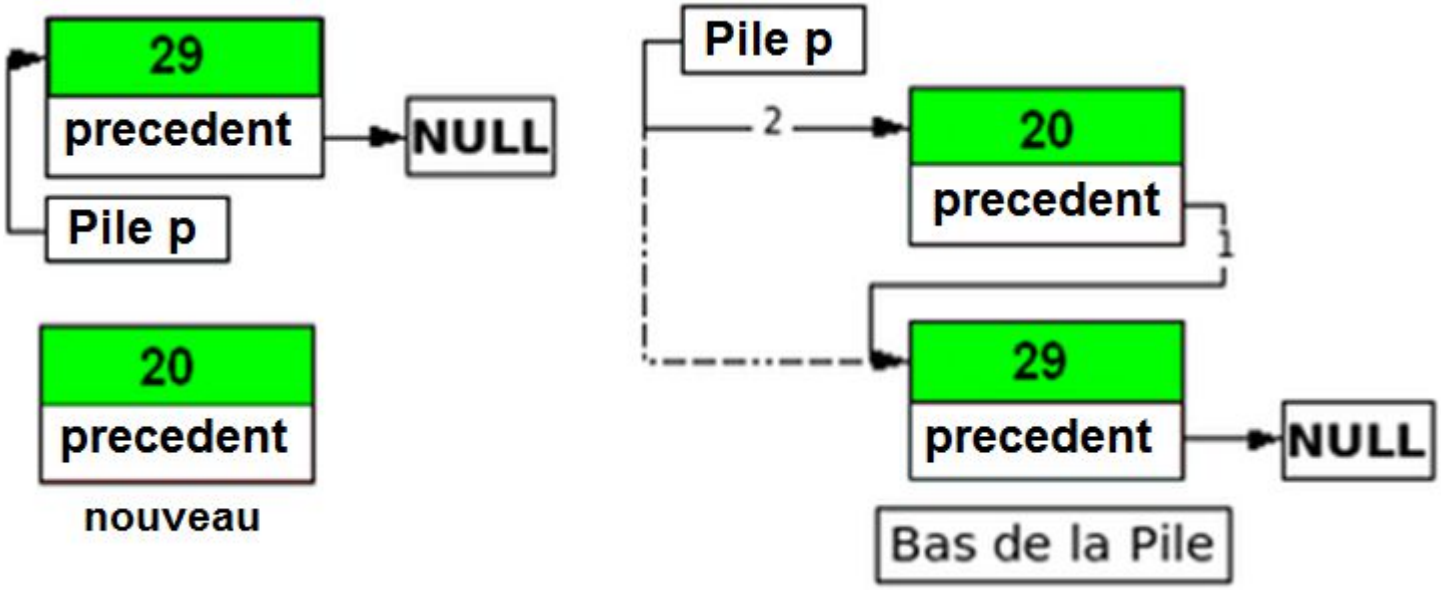
La pile n'est rien qu'un pointeur vers le premier élément de la liste

```
  typedef element* Pile;
  Pile p=NULL;
```

N.B: les donnes de la pile peuvent être de n'importe quel type: entier, réel, caractère, structure, tableau ,…

Ajouter un élément à la pile (empiler)

- Dans une pile l'insertion d'un élément se fait toujours en haut de la pile (au début de la liste).

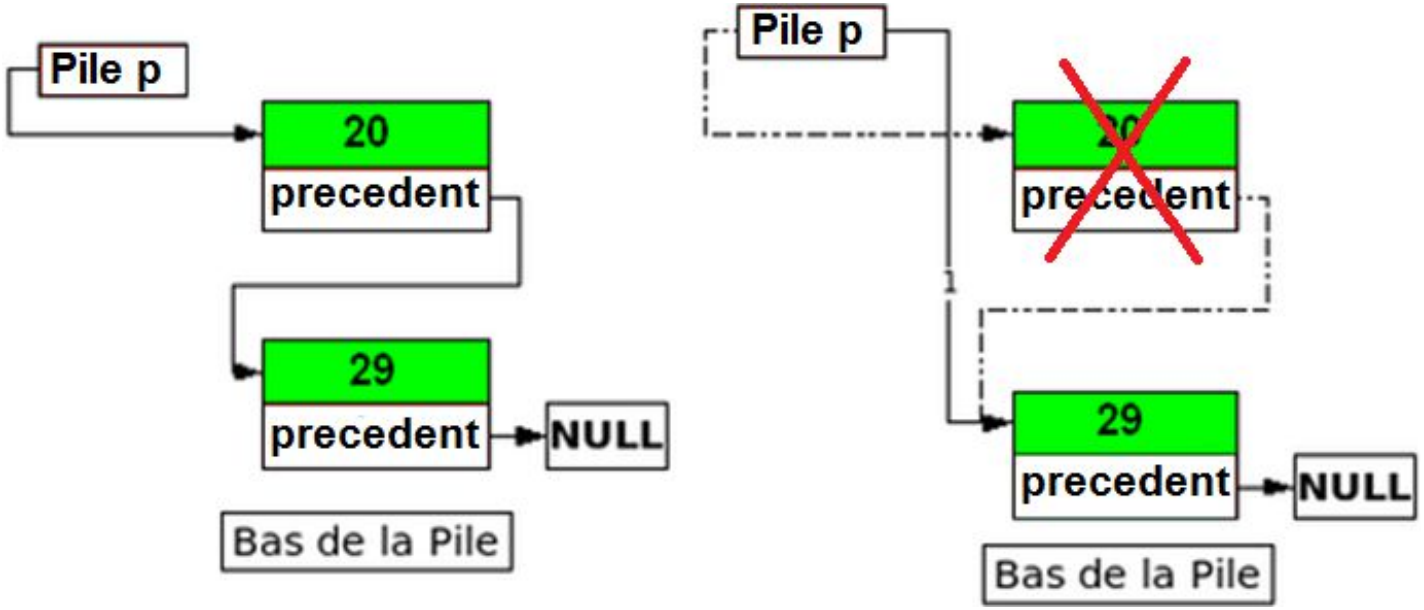


Fonction Empiler()

```
void Empiler(Pile *p, int valeur)
{
  /*On crée un nouvel élément*/
  element* nouveau = (element*)malloc(sizeof(element));
  /*Si le nouvel élément est crée*/
  if (nouveau!=NULL)
  {
    /* On assigne la valeur au nouvel élément */
    nouveau->donnee = valeur;

    /*On attache le nouvel élément au premier élément de la pile*/
    nouveau->precedent=*p;

    /*on pointe la pile vers le nouvel élément*/
    *p=nouveau;
  }
}
```

Retirer un élément de la pile (dépiler)

L'élément qui sera retiré sera le dernier élément que l'on a ajouté, c'est-à-dire l'élément se trouvant au sommet de la pile.



Fonction Depiler()

```
int Depiler(Pile *p)
{
     int temp=-1;
     if(*p!= NULL)
     {
          /*on memorise la valeur à dépiler*/
          temp=(*p)->donnee;
          /*on memorise l'adresse de l'avant dernière élément de la pile*/
          element* ptemp = (*p)->precedent;
          /* On libère le premier élément */
          free(*p);
          /* On pointe la pile vers le deuxième élément */
          *p=ptemp;
     }
    /*on retourne la valeur dépilée*/
     return temp;
}
```

Vidage de la pile

Il s'agit de la fonction permettant d'effacer la pile. Cette fonction se réalise en seulement deux étapes :

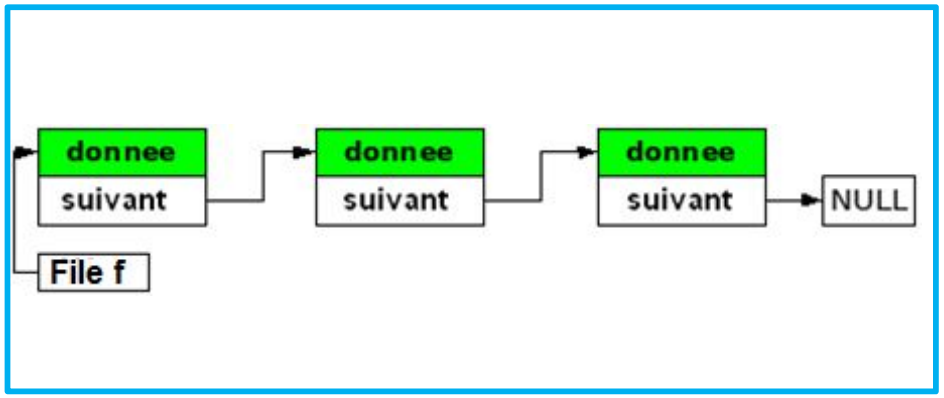
1) Tant que la pile n'est pas vide;

2) Effacer le dernier élément de la pile grâce à la fonction Depiler().

```
void vider_pile(Pile *p)
{
     while (*p != NULL)
     {
          Depiler(p);
     }
}
```

Les files

Une file, permet de réaliser une FIFO (First In First Out): les premiers éléments ajoutés à la file sont les premiers à être récupérés.

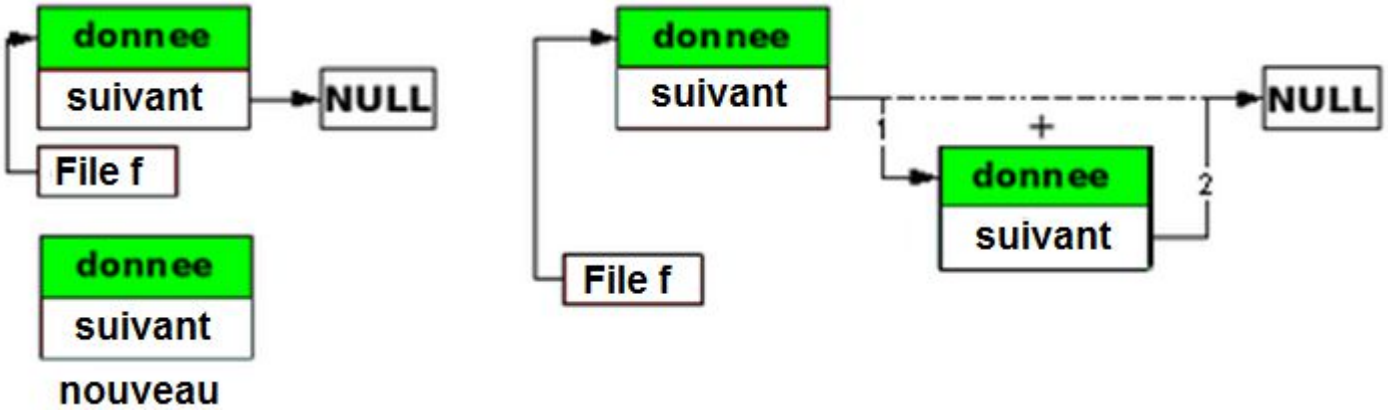



```
typedef struct element
{
    int donnee;
    struct file *suivant;
} element;

Typedef element* File;

/* Déclaration d'une file */
File f=NULL;
```

Ajouter un élément à la file (enfiler)

Dans une file, l'insertion d'un élément se fait en fin de la liste. Pour ajouter un élément:

1) On crée un nouvel élément
2) On vérifie que le nouvel élément a bien été créé
3) On fait pointer cet élément vers NULL
4) On assigne à la donnée de cet élément la donnée que l'on veut ajouter
5) Si la file est vide, alors on fait pointer la file vers l'élément que l'on vient de créer
6) Sinon, on crée un pointeur temporaire pour parcourir la file jusqu'au dernier élément.
8) On fait pointer l'élément temporaire vers le nouvel élément créé pour le rattacher à la liste.

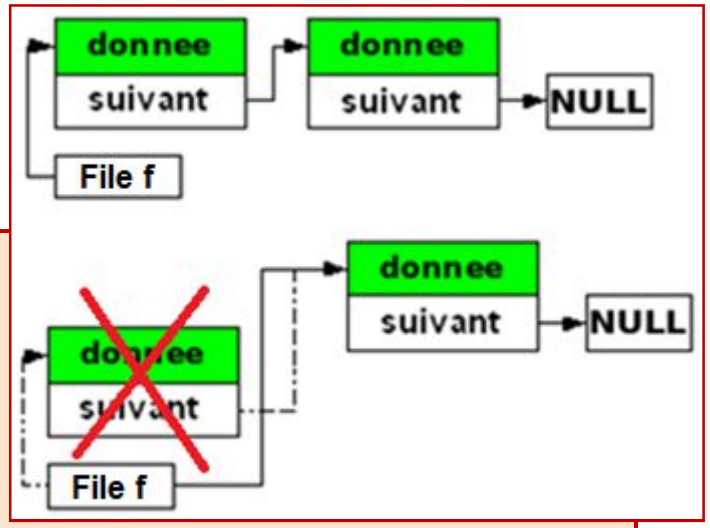


Fonction Enfiler()

```
void Enfiler(File *f, int valeur)
{
    /*On crée un nouvel élément*/
    element* nouveau = (element*)malloc(sizeof(element));
    /*Si le nouvel élément est crée*/
    if (nouveau!=NULL)
    {
        /*On ajoute en fin, donc aucun élément ne va suivre*/
        nouveau->suivant = NULL;
        /*On assigne la valeur au nouvel élément*/
        nouveau->donnee = valeur;
        if(*f== NULL)
        {
          /*Si la liste est vidée il suffit de renvoyer l'élément créé*/
          *f=nouveau;
        }
        else
        {
           /*Sinon, on parcourt la liste à l'aide d'un pointeur
           temporaire et on relit le dernier élément de la file au
           nouvel élément*/
           element* ptmp=*f;
           while(ptmp->suivant != NULL)
           {
              ptmp = ptmp->suivant;
           }
           ptmp->suivant=nouveau;
        }
     }
}
```

Retirer un élément de la file (Défiler)

L'élément qui sera enlevé sera le premier élément de la file.




```
int Defiler(File *f)
{
    int tmp=-1;
    if(*f!= NULL)
    {
        /*on mémorise la valeur à défiler*/
        tmp=(*f)->donnee;
        /*on mémorise l'adresse de l'élément suivant de la file */
        element* ptmp = (*f)->suivant;
        /* On libère le premier élément */
        free(*f);
        /* On pointe la file vers l'élément suivant */
        *f=ptmp;
    }
    return tmp;
}
```

Vidage de la file

Il s'agit de la fonction permettant d'effacer la file. Cette fonction se réalise en seulement deux étapes :

1) Tant que la file n'est pas vide;

2) Effacer le dernier élément de la file grâce à la fonction Defiler().

```
void vider_pile(File *f)
{
     while (*f != NULL)
     {
          Defiler(f);
     }
}
```

Conclusion

- Les piles (*stack*) et les files (*queue*) constituent des **structures de données** particuliers.
- Elles peuvent être réalisées soit par les tableaux ou les listes chainées.
- Elles permettent de contrôler la manière dont sont ajoutés les nouveaux éléments.
- Les piles et les files sont très utiles pour les programmes qui doivent traiter des données qui arrivent au fur et à mesure.