3.10 Les piles

Les piles et les files sont des exemples de listes chainés.

● Les Piles.

- Dans une pile (stack en anglais): **Le dernier** élément qui a été ajouté **est le premier à sortir** « **LIFO : Last In First Out** ». Prenons un exemple : une pile d'assiettes à laver. On les met les unes sur les autres et pour les laver, on prend celle du dessus. Un deuxième exemple : une pile de dossiers sur un bureau. On les met les uns au-dessus des autres et on retire le premier, donc le dernier arrivé. Les piles répondent à ce principe : *dernier arrivé, premier servi*.

Pile d'assiettes

- L'opération d'ajout dans une pile s'appelle **: « Empiler ».**

- L'opération d'enlever un élément de la liste s'appelle **: « Dépiler ».**

- Le premier élément dans la pile est nommé **« sommet ».**

De nombreuses 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 ».
- La fonction « Annuler la frappe » (en anglais « Undo ») d'un traitement de texte mémorise les modifications apportées au texte dans une pile.
- Vérification de parenthèse d'une chaine de caractères ;
- La récursivité (une fonction qui fait appel à elle-même) ;
- **Etc.**

●Les files

- Dans une file (queue en anglais) : **Le premier élément** qui a été ajouté **est le premier à sortir « FIFO : First In First Out** ».Un exemple tout simple : une file d'attente. Le premier arrivé est le premier à sortir (à être servi).

File d'attente.

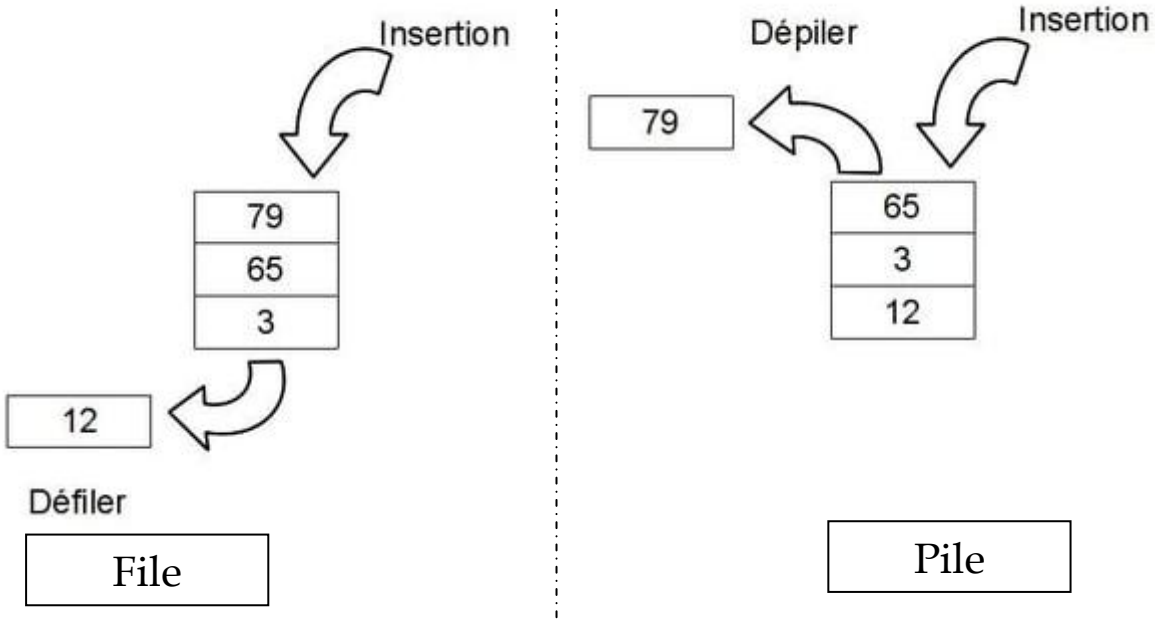
- Ajouter un nouvel élément à la file est appelé **Enfiler**.

- Enlever un élément de la file est appelé **Défiler.**

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

- 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 systèmes d'exploitation, qui doivent exécuter plusieurs tâches dans le même ordre d'arrivées.

●**Comparaison entre la Pile et la File.**



- **Dans une pile on ajoute à la tète de la pile** « sommet » **et on enlève à partir de la tête,** ce qui fait que l'élément ajouté en dernier lieu est le premier à récupérer.
- **Dans une liste on ajoute à la queue et on enlève à partir de la tête**, ce qui fait que le premier élément ajouté est le premier à récupérer.

●Implémentation des piles et des files

- Il y **a deux méthodes pour créer des piles et des files**:
  - Méthode 1 : Utilisation des tableaux (piles et files statiques).
  - Méthode 2 : Utilisation des listes chainées (piles et files dynamiques).

- Dans ce cours on s'intéresse aux piles et files dynamiques (Utilisation des listes chainées).

❖ ***Pour créer une pile*** : créer une liste chainée ; ajouter les éléments à la tête de la liste « Empiler » et récupérer les éléments à partir la tête (sommet) « Dépiler ».

❖ ***Pour créer une file*** : créer une liste chainée ; ajouter les éléments à la fin de la liste « Enfiler » et récupérer les éléments à partir de la tête « Défiler».

❖ **Exemple de gestion d'une pile :**

- **La déclaration de l'élément de base dans la pile**

```
typedef struct Maillon
{
    int donnee;
    struct Maillon* suivant;

}Maillon;
```

- **La déclaration la fonction empiler (ajouter au début).**

```
Maillon * empiler(Maillon * sommet, int valeur)
{
    /* On crée un nouvel élément */
  Maillon * nouvelElement = malloc(sizeof(Maillon));
  if (nouvelElement == NULL)
    {         printf("Allocation échouée");
              exit(1);
    }

  nouvelElement->donnee = valeur;
```

```
   /* On ajoute le nouvel élément à la tête de la pile */

   nouvelElement->suivant = sommet;

    /* On retourne le pointeur sur le premier élément */
    return nouvelElement;
}
```

- **La déclaration la fonction dépiler (retirer un élément).**

Nous envoyons à la fonction dépiler le sommet de la pile (*passage par référence d'un pointeur*, on cherche à changer le sommet et non la valeur sur laquelle il pointe, d'où l'utilisation de Maillon ** sommet). Lors de l'appel de cette fonction, il faut envoyer l'adresse du pointeur. La fonction d'épiler retourne la valeur dépilée ou -1 si la pile est vide.

```
int depiler(Maillon **sommet)
{
       /* Si la liste est vide :  Returner -1 */
       if (*sommet == NULL ) {  return -1;
                              }

      /* Créer un pointeur   sur le sommet */
        Maillon *temp = *sommet;

        /* Modifier le sommet */
        *sommet = temp->suivant;

      /* Créer une copie des valeurs à dépiler */
       int valeur = temp->donnee;

      /* Libérer le sommet */
        free(temp);

      /* retourner le nouveau sommet */
        return valeur;
}
```

- **La déclaration de la fonction d'affichage**

```
void afficherPile(Maillon* sommet)
{
  /* creer un Maillon temporaire qui pointe sur le debut de la liste */
    struct Maillon *temp = sommet;

    /* verifier si c'est pas null */
    while (temp != NULL)
```

```
    {
      printf(" %d ", temp->donnee);
      /* pointer sur le maillon suivant */
      temp = temp->suivant;
     }
}
```

- **Exemple de programme principal.**

```
int main()
{
  /* déclaration d'un pointeur sur le sommet de la pile*/
    Maillon * sommet = NULL;
  /* Empiler */
    sommet =  empiler(sommet,10);
    sommet =  empiler(sommet,20);
    sommet =  empiler(sommet,30);
    sommet =  empiler(sommet,40);
    sommet =  empiler(sommet,50);



    // Afficher
    printf("Affichage de la pile :");
    afficherPile(sommet);

     // Dépiler
     int n =   depiler(&sommet);
      // Afficher
    printf("\nAffichage de la pile: ");
    afficherPile(sommet);

}
```

❖ **La gestion d'une file.**

- La gestion d'une file c'est une gestion d'une liste chainée simple :
  - L'ajout c'est à la fin de la liste (voir listes chainées).
  - L'extraction c'est à partir de la tête de la liste (comme pour la pile).