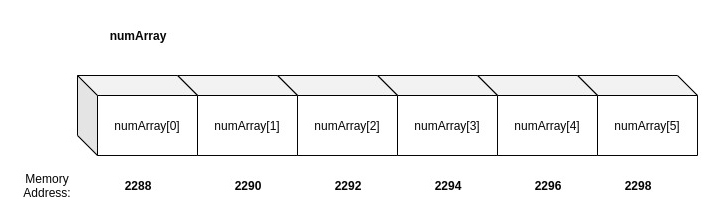
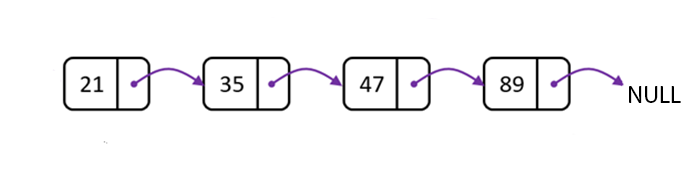
|  |  |
| --- | --- |
| ***Cours Liste chaînée*** | ***[Enseignement supérieur](https://fr.wikipedia.org/wiki/Enseignement_sup%C3%A9rieur)*** |

1. *Introduction :*

Les tableaux peuvent être utilisés pour stocker des données de mêmes types et de manière linéaire mais ils ont leurs limites:

1. La taille des tableaux est fixe : Il est donc nécessaire de connaître à l'avance le nombre d'éléments quelle que soit l'utilisation.
2. L'insertion et la suppression d'un nouvel élément dans un tableau est coûteuse car les éléments doivent être décalés et un espace doit être créé ou tranché.



Nous avons recours à une autre alternative : La liste chaînée est une structure de données représentant une collection ordonnée et de taille arbitraire d'éléments de même type, dont la représentation en mémoire de l'ordinateur est une succession de cellules faites d'un contenu et d'un lien vers une autre cellule.

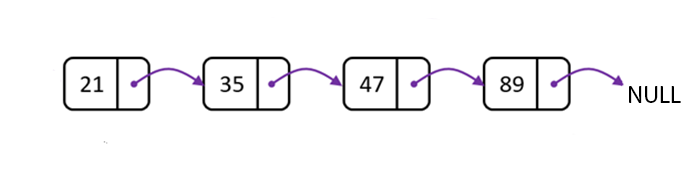
De façon imagée, l'ensemble des cellules ressemble à une chaîne dont les maillons seraient les cellules.

Le principal avantage d'une liste chaînée par rapport à un tableau est que les éléments de la liste peuvent être facilement insérés ou supprimés sans réaffectation de l'ensemble de la structure car les éléments n'ont pas besoin d'être stockés de manière contiguë en mémoire ou sur disque.

*2-types et variantes :*

Il existe trois types populaires de liste chaînée :

1. Liste chaînée simple :La navigation dans les éléments se fait uniquement vers l'avant.

typedef struct noeud {

    type val;

    struct noeud\* next;

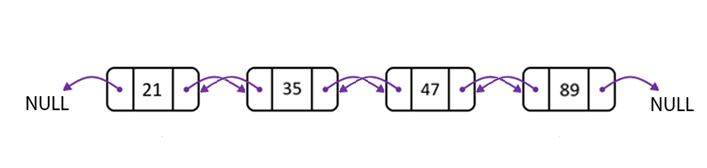
}noeud;

Typedef struct {

noeud *\*head;*

*}liste;*

1. Liste doublement chaînée : Les éléments peuvent être parcourus en avant et en arrière.

typedefstructnoeud {

    type val;

    struct noeud \*next;

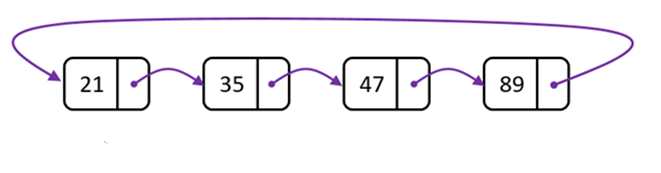
struct noeud \*previous;

}noeud;

Typedef struct {

    noeud \*tete;

}liste;

1. Liste chaînée circulaire : La dernière cellule contient le lien de la première cellule ou la première cellule a un lien vers la dernière cellule et autre, une liste circulaire peut être simple ou doublement chaînée.

On peut parcourir toute la liste en partant de n'importe quel point.

*3- Opérations de base d’une liste chaînée:*

Voici les opérations de base prises en charge par une liste chaînée:

* Insertion : Ajoute un élément à la liste.
* Suppression : Supprime un élément de la liste.
* Affichage : Affiche la liste complète.
* Recherche : Recherche un élément à l'aide d’une clé donnée.

**#include<stdio.h>**

**#include<stdlib.h>**

**Typedef struct noeud{**

**int valeur;**

**structnoeud\*next;**

**} noeud;**

**typedefstruct{**

**noeud \*tete**

**} liste;**

**voidinitListe(liste\*l){**

**l->tete = NULL;**

**}**

**voidinserTete(liste\*l, intx){**

**noeud\*nouveau = (noeud \*)malloc(sizeof(noeud)) ;**

**nouveau->valeur = x;**

**nouveau->next = l->tete;**

**l->tete = nouveau;**

**}**

**voidinserFin(liste\*l, intx){**

**noeud\*p, \*nouveau = (noeud\*)malloc(sizeof(noeud));**

**nouveau->valeur = x;**

**nouveau->next = NULL;**

**if (l->tete == NULL)**

**l->tete = nouveau;**

**else**

**{**

**p = l->tete;**

**while (p->next != NULL) p = p->next;**

**p->next = nouveau;**

**}**

**}**

**int recherche(listel, intx){**

**noeud \*p = l.tete;**

**while ((p != NULL) && (p->valeur != x))**

**p = p->next;**

**returnp != NULL;**

**}**

**voidaffich(listel){**

**noeud \*p = l.tete;**

**while (p != NULL)**

**{**

**printf("%d-->", p->valeur);**

**p = p->next;**

**}**

**printf("\n");**

**}**

**void supprimer(liste\*l, intx){**

**noeud \*courant, \*prec, \*p;**

**if (l->tete != NULL)**

**{**

**if (x == l->tete->valeur)**

**{**

**p = l->tete;**

**l->tete = l->tete->next;**

**free(p);**

**}**

**else**

**{**

**prec = l->tete;**

**courant = l->tete->next;**

**while ((courant != NULL) && (courant->valeur != x))**

**{**

**prec = courant;**

**courant = courant->next;**

**}**

**if (courant != NULL)**

**{**

**prec->next = courant->next;**

**free(courant);**

**}**

**else**

**printf("valeur inexistante\n");**

**}**

**}**

**}**

**voidinsertionTriee(liste\*l, intx){**

**// on suppose que la liste est triee**

**noeud \*courant, \*prec, \*nouveau = (noeud \*)malloc(sizeof(noeud));**

**nouveau->valeur = x;**

**if (l->tete == NULL)**

**{**

**l->tete = nouveau;**

**l->tete->next = NULL;**

**}**

**else**

**{**

**if (x<l->tete->valeur)**

**{**

**nouveau->next = l->tete;**

**l->tete = nouveau;**

**}**

**else**

**{**

**prec = l->tete;**

**courant = l->tete->next;**

**while ((courant != NULL) && (courant->valeur <= x))**

**{**

**prec = courant;**

**courant = courant->next;**

**}**

**prec->next = nouveau;**

**nouveau->next = courant;**

**}**

**}**

**}**

**int min(listel){**

**int m = l.tete->valeur;**

**noeud \*p = l.tete->next;**

**while (p != NULL)**

**{**

**if (p->valeur < m)**

**m = p->valeur;**

**p = p->next;**

**}**

**returnm;**

**}**

**void main(){**

**int choix, x, trouve;**

**liste l;**

**initListe(&l);**

**do{**

**printf("Opérations sur les listes\n");**

**printf("1- Insérer une valeur en tête de liste\n");**

**printf("2- Insérer une valeur en fin de liste\n");**

**printf("3- Rechercher une valeur\n");**

**printf("4- Supprimer une valeur\n");**

**printf("5- afficher le minimum\n");**

**printf("6- afficher la liste\n");**

**printf("7- Quitter l'application\n");**

**printf("veuillez taper votre choix:");**

**scanf("%d", &choix);**

**switch (choix)**

**{**

**case 1:**

**printf("saisir une valeur\n");**

**scanf("%d", &x);**

**inserTete(&l, x);**

**break;**

**case 2:**

**printf("saisir une valeur\n");**

**scanf("%d", &x);**

**inserFin(&l, x);**

**break;**

**case 3:**

**printf("saisir une valeur à rechercher\n");**

**scanf("%d", &x);**

**trouve = recherche(l, x);**

**if (trouve)**

**printf("%d existe\n", x);**

**else**

**printf("%d n'existe pas\n", x);**

**break;**

**case 4:**

**printf("saisir une valeur à supprimer\n");**

**scanf("%d", &x);**

**supprimer(&l, x);**

**break;**

**case 5:**

**printf("min=%d\n", min(l));**

**break;**

**case 6:**

**printf("contenu de la liste\n");**

**affich(l);**

**break;**

**case 7:**

**printf("Fin\n");**

**break;**

**default:**

**printf("choix non valide\n");**

**}**

**}while (choix != 7);**

**}**