

Listes chaînées

La notion de structure autoréférentielle

Une *structure autoréférentielle* (parfois appelée *structure récursive*) correspond à une **structure** dont au moins un des champs contient un pointeur vers une structure de même type. De cette façon on crée des éléments (appelés parfois *noeuds* ou *liens*) contenant des données, mais, contrairement à un tableau, celles-ci peuvent être éparpillées en mémoire et reliées entre elles par des liens logiques (des pointeurs), c'est-à-dire un ou plusieurs champs dans chaque structure contenant l'adresse d'une ou plusieurs structures de même type.

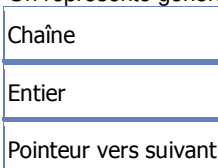
- Lorsque la structure contient des données et un pointeur vers la structure suivante on parle de **liste chaînée**
- Lorsque la structure contient des données, un pointeur vers la structure suivante, et un pointeur vers la structure précédente on parle de **liste chaînée double**
- Lorsque la structure contient des données, un pointeur vers une première structure suivante, et un pointeur vers une seconde, on parle d'**arbre binaire**

Qu'est-ce qu'une liste chaînée ?

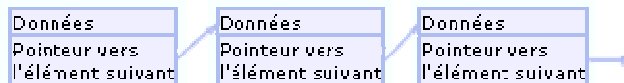
Une liste chaînée est une structure comportant des champs contenant des données et un pointeur vers une structure de même type. Ainsi, la structure correspondant à une liste chaînée contenant une chaîne de 15 caractères et un entier ressemble à ceci :

```
struct Nom_de_la_liste {  
    char Chaîne[16];  
  
    int Entier;  
  
    struct Nom_de_la_liste * pSuivant;  
};
```

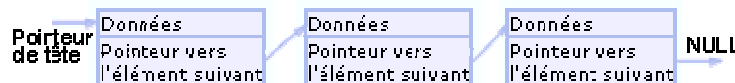
On représente généralement cette structure de la manière suivante :



Une liste chaînée se représente donc de la façon suivante :



En réalité la déclaration de la structure et la récursivité de celle-ci grâce à des pointeurs est nécessaire car cela crée une chaîne d'enregistrements liés par des liens logiques, mais cela n'est pas suffisant. En effet, il est nécessaire de conserver une « trace » du premier enregistrement afin de pouvoir accéder aux autres, c'est pourquoi un pointeur vers le premier élément de la liste est indispensable. Ce pointeur est appelé *pointeur de tête*. D'autre part, étant donné que le dernier enregistrement ne pointe vers rien, il est nécessaire de donner à son pointeur la valeur *NULL* :



Pour créer une liste chaînée en langage C, il s'agit dans un premier temps de définir la structure de données, ainsi qu'un pointeur vers une structure du type de celle définie précédemment, afin de pointer vers la tête de la liste, c'est-à-dire le premier enregistrement :

```
struct Liste {
```

```

char Chaine[16];

struct Liste * pSuivant;

};

struct Liste *Nouveau;

struct Liste *Tete;

Tete = NULL;

```

Ajout d'un premier élément

Une fois la structure et les pointeurs définis, il est possible d'ajouter un premier maillon à la liste chaînée, puis de l'affecter au pointeur *Tete*. Pour cela il est nécessaire :

- d'allouer la mémoire nécessaire au nouveau maillon grâce à la fonction *malloc*, selon la syntaxe suivante :

```
Nouveau = (Liste*)malloc(sizeof(struct Liste));
```

- d'assigner au champ « pointeur » du nouveau maillon, la valeur du pointeur vers le maillon de tête :

```
Nouveau->pSuivant = Tete;
```

- définir le nouveau maillon comme maillon de tête :

```
Tete = Nouveau;
```

- *Il est conseillé de tester la valeur renvoyée en retour par la fonction malloc() afin de savoir si l'allocation dynamique de mémoire s'est déroulée correctement.*

Ajout d'un élément en fin de liste

L'ajout d'un élément à la fin de la liste chaînée est similaire, à la différence près qu'il faut définir un pointeur (appelé généralement *pointeur courant*) afin de parcourir la liste jusqu'à atteindre le dernier maillon (celui dont le pointeur possède la valeur *NULL*). Les étapes à suivre sont donc :

- la définition d'un pointeur *courant* :

```
struct Liste * pCourant;
```

- le parcours de la liste chaînée jusqu'au dernier noeud :

- ```
if (Tete != NULL) {
```
- ```
    pCourant = Tete;
```
- ```
 while (pCourant->pSuivant != NULL) pCourant = pCourant->pSuivant;
```
- ```
}
```

```
}
```

- l'allocation de mémoire pour le nouvel élément :

```
Nouveau = (Liste*)malloc(sizeof(struct Liste));
```

- faire pointer le pointeur courant vers le nouveau noeud, et le nouveau noeud vers NULL :

```
pCourant->pSuivant = Nouveau;
```

-

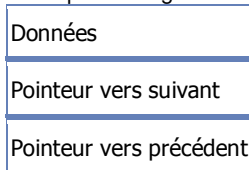
```
Nouveau->pSuivant = NULL;
```

Qu'est-ce qu'une liste chaînée double ?

Une **liste chaînée double** est basée sur le même principe que la liste chaînée simple, à la différence près qu'elle contient non seulement un pointeur vers le maillon suivant mais aussi un pointeur vers le maillon précédent, permettant de cette façon le parcours de la liste dans les deux sens. Ainsi, la structure correspondant à une liste chaînée double contenant une chaîne de 15 caractères et un entier ressemble à ceci :

```
struct Liste {  
    char Chaîne[16];  
  
    int Entier;  
  
    struct Liste * pSuivant;  
  
    struct Liste * pPrecedent  
};
```

On représente généralement cette structure de la manière suivante :



Une liste chaînée double se représente donc de la façon suivante :

