Bienvenue!

Conférence BRIEF-42

#01

Introduction Data Structure and Bitwise operators.

Une introduction aux listes chaînées et aux opérations binaires.

Introduction à l'utilisation :

1 - Les opérateurs binaires, avec *Sam*.

2 - Les listes chaînées, avec *Gabriel*.

3 - FAQ.

Durée estimée : 1h00

Les opérateurs binaires avec Sam

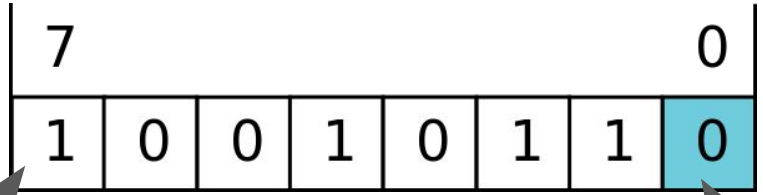
Pourquoi utiliser les opérateurs binaires ?

- Une nouvelle perspective.
- Pour des problèmes orientés bas niveau.
- Les opérations binaires plus rapides, plus légères et plus simples.
- Emballage des données, Encodage des données.
- Utiliser beaucoup en réseau et cryptographie et bas niveau.
- Également souvent utilisé pour obscurcir du code ;).

Les opérations binaires :

Opérateur	Description
&	ET (AND) : opérateur logique de comparaison de bits
|	OU (OR) : opérateur logique de comparaison de bits
^	OU exclusif (XOR) : opérateur logique de comparaison de bits
~	NON (NOT) : complémentaire par bit
<<	décalage de tous les bits vers la gauche
>>	décalage de tous les bits vers la droite

MSB et LSB



MSB : Most Significant Bit, ou bit de poids fort
“le plus à gauche”

LSB : Least Significant Bit, ou bit de poids faible
“le plus à droite”

Opérateur binaire **ET (**bitwise **AND) : &**

x	**y**	**x & y**
0	0	0
0	1	0
1	0	0
1	1	1

Opérateur binaire **OU (**bitwise **OR) :** |

x	**y**	**x | y**
0	0	0
0	1	1
1	0	1
1	1	1

Opérateur binaire **OU exclusif (**bitwise **XOR) : ^**

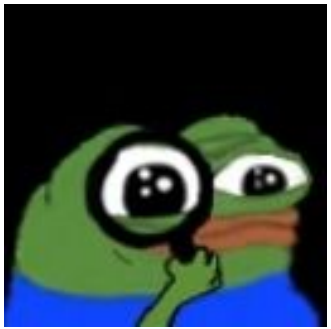
x	**y**	**x ^ y**
0	0	0
0	1	1
1	0	1
1	1	0

Opérateur binaire **NON (**bitwise **NOT) : ~**

x	**~x**
0	1
1	0

Opérateur binaire de décalage (bitshift): **<<** et **>>**

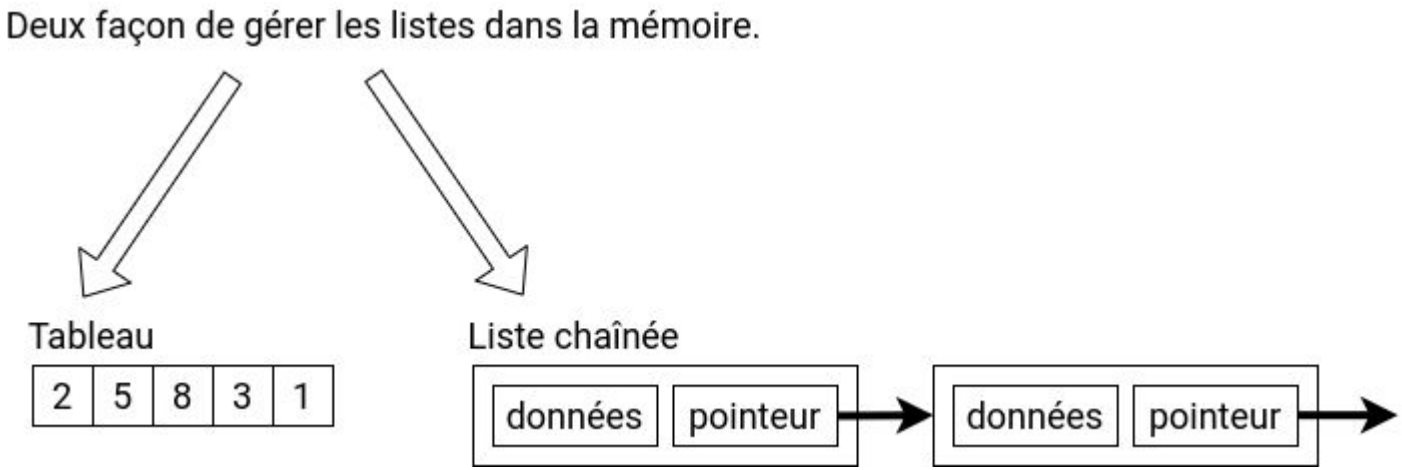
Quelques exemples



Les listes chaînées avec Gabriel

Rappel :

Il faut d'abord comprendre la différence entre un tableau et une liste chaînée de structures en C.



Différences principales

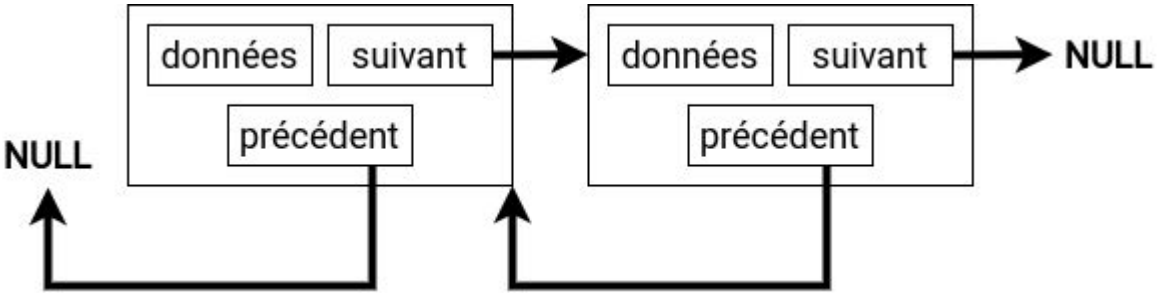
Tableau	Liste chaînée
• Travailler avec une liste de données de même type. • Le nombre d'éléments est connu et fixe. • Pas d'allocation mémoire	• Travailler avec une liste de données de différents types. • Le nombre d'éléments peut changer de manière dynamique. • Il faut tout malloc

La liste chaînée simple :

On ne peut qu'avancer dans la liste jusqu'au dernier élément.

La liste chaînée double :

On peut avancer et reculer dans la liste entre le premier et le dernier élément.



Un exemple de liste chaînée double :

Objectif : avoir une liste de contacts

```
typedef struct s_friends    t_friends;

struct s_friends
{
    char        *tel;
    char        *nom;
    t_friends   *suiv;
    t_friends   *prec;
};
```

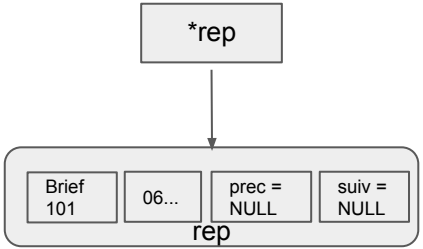
Définition dans le fichier header

Un exemple de liste chaînée double :

Objectif : avoir une liste de contacts

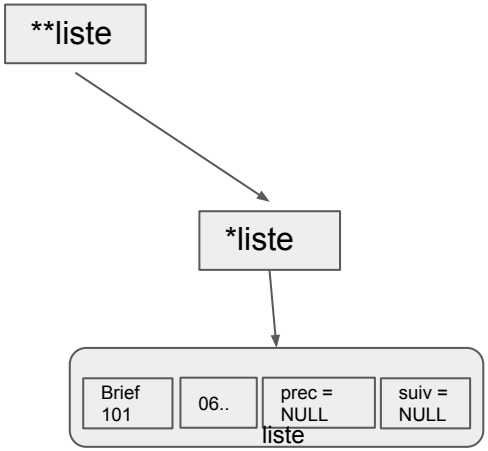
```c
int main(void)
{
    t_friends   *rep;

    rep = calloc(1, sizeof(t_friends));
    rep->nom = "Brief 101";
    rep->tel = "0642424242";
    rep->suiv = NULL;
    rep->prec = NULL;
    ajouter_contact_debut(&rep, "gab", "0610011001");
    ajouter_contact_fin(&rep, "sam", "0690898786");
}
```

Déclaration et initialisation de la liste



Un exemple de liste chaînée double :

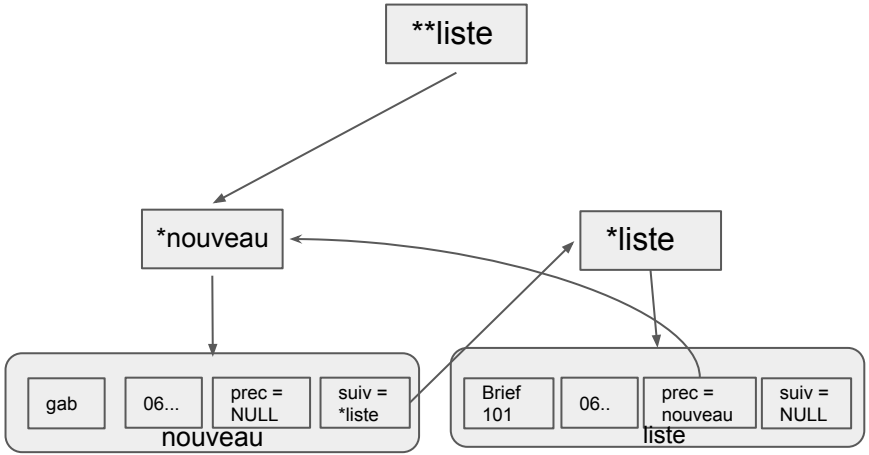
Ajout d'un maillon au début




```c
void    ajouter_contact_debut(t_friends **liste, char *tel, char *nom)
{
    t_friends   *nouveau;

    nouveau = calloc(1, sizeof(t_friends));
    nouveau->nom = nom;
    nouveau->tel = tel;
    nouveau->suiv = *liste;
    (*liste)->prec = nouveau;
    *liste = nouveau;
}
```

Un exemple de liste chaînée double :

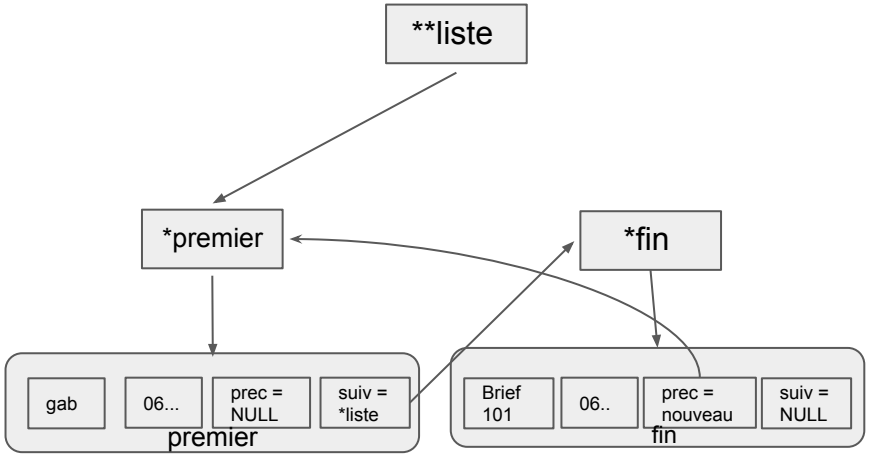
Ajout d'un maillon au début




```c
void    ajouter_contact_debut(t_friends **liste, char *tel, char *nom)
{
    t_friends   *nouveau;

    nouveau = calloc(1, sizeof(t_friends));
    nouveau->nom = nom;
    nouveau->tel = tel;
    nouveau->suiv = *liste;
    (*liste)->prec = nouveau;
    *liste = nouveau;
}
```

Un exemple de liste chaînée double :

Ajout d'un maillon à la fin

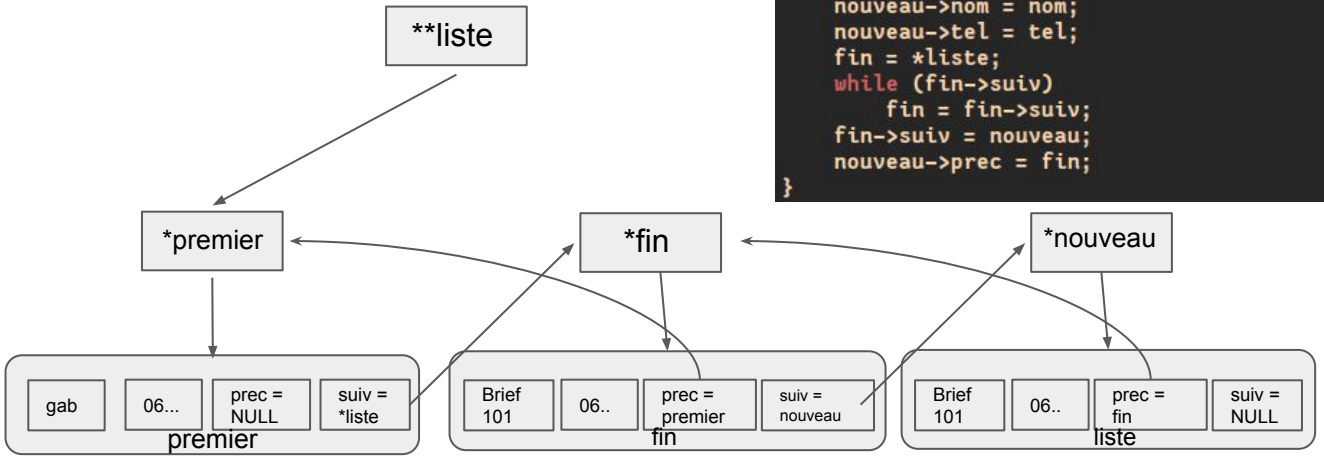



```c
void    ajouter_contact_fin(t_friends **liste, char *nom, char *tel)
{
    t_friends   *nouveau;
    t_friends   *fin;

    nouveau = calloc(1, sizeof(t_friends));
    nouveau->nom = nom;
    nouveau->tel = tel;
    fin = *liste;
    while (fin->suiv)
        fin = fin->suiv;
    fin->suiv = nouveau;
    nouveau->prec = fin;
}
```

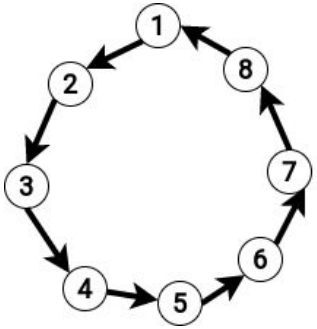
Un exemple de liste chaînée double :

Ajout d'un maillon au début

```c
void    ajouter_contact_fin(t_friends **liste, char *nom, char *tel)
{
    t_friends   *nouveau;
    t_friends   *fin;

    nouveau = calloc(1, sizeof(t_friends));
    nouveau->nom = nom;
    nouveau->tel = tel;
    fin = *liste;
    while (fin->suiv)
        fin = fin->suiv;
    fin->suiv = nouveau;
    nouveau->prec = fin;
}
```



D’autres usages de structures reliées:

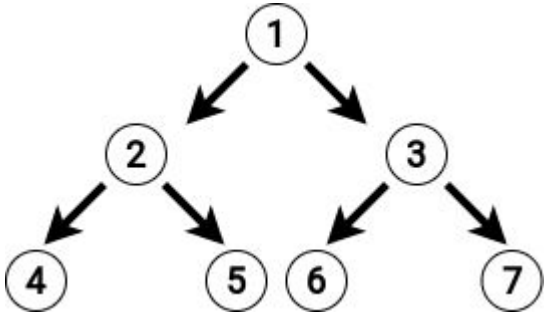


La liste circulaire :

On peut avancer et reculer dans la liste sans qu’il y ait de premier ou de dernier élément.
(Le premier élément est relié au dernier)

L’arbre binaire :

Une structure qui pointe vers deux autres structures.



Au revoir!

FAQ

Pour ceux qui le souhaitent.