Chapitre 2
Gestion dynamique de la mémoire

Module 20 (info4): Structures de données en C

2éme ANNEE LICENCE D'ENSEIGNEMENT DE MATHEMATIQUES (LEM)
2éme ANNEE LICENCE CRYPTO MATHEMATIQUE ET SECURITE DE L'INFORMATION (LCMSI)

mlahby@gmail.com

10 février 2018

Plan

Rappel sur Les mémoires

- RAM (Random Access Memory) : 32Mo, 64 Mo, 1Go....,16Go
- Le disque dur : plusieurs Go
- La mémoire virtuelle : temps d'accès 1000 fois plus long.

Intérêts des pointeurs

- Gestion de l'espace mémoire en cours d'exécution : cela signifie qu'on peut résever l'espace mémoire en moment d'exécution d'un programme.
- Modifications de variables passées en paramètres de fonction
- Représentation de tableaux d'une manière dynamique

Allocation statique de la mémoire

- Lors de la déclaration de tableaux
  * Les dimensions des tableaux doivent être connu à la compilation.
  * Perte de l'espace mémoire :
    - Impossible : int T[ ] ou int T[N] ;
    - Possible : #define N 10 après on fait int T[N]

- La gestion statique ne se prête pas aisément à la mise en oeuvre de "listes chaînées", d'arbres binaires"

Allocation dynamique de la mémoire

- Gérée par le programmeur :
  * La dimension d'un tableau peut être connu jusqu'à l'exécution de programme
  * Demande d'espace : malloc(), calloc()
  * Manipulation du pointeur indiquant l'espace qu'on veut réserver.
  * Modifier la taille d'une zone préalablement allouée : realloc()
  * Libération de l'espace free()
- Il n' y a pas de perte de l'espace mémoire.

La fonction malloc

- Alloue size octets, et renvoie un pointeur sur la mémoire allouée
- Le contenu de la zone de mémoire n'est pas initialisé
- Si size vaut 0, malloc() renvoie NULL.
  * malloc(0) retourne NULL
- La fonction malloc() renvoie aussi NULL dans le ou il n' y a pas de l'espace mémoire nécessaire.

La fonction malloc

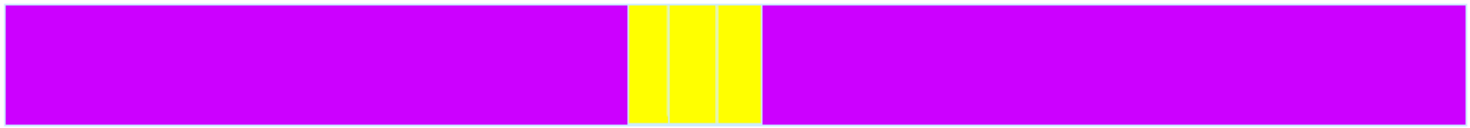
- **Syntaxe :**

  #include <stdlib.h>

  void *malloc(size_t size)

- **Exemple :**

```
malloc(3*sizeof(int));
```



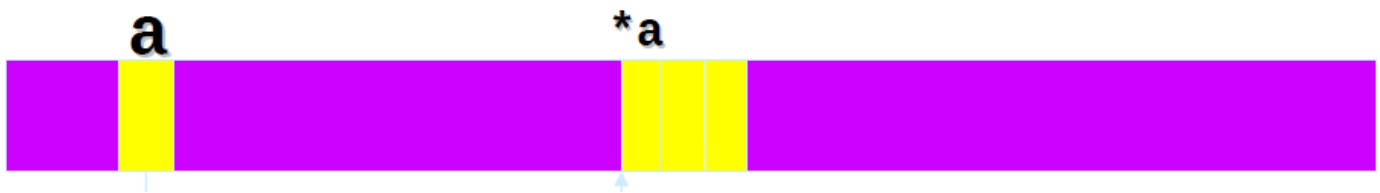
Allocation dynamique de place mémoire(pour 3 entiers)

La fonction malloc

- **Allocation dynamique et assignement**

```
int* a = malloc(3*sizeof(int));
```

```
int* a = (int*)malloc(3*sizeof(int));
```



- **Exemple :**

La déclaration statique d'un tableau d'entiers int T[10] devient:

```
int T ;
T=(int*)malloc(10*sizeof(int));
```

La fonction calloc

- Alloue la mémoire nécessaire pour un tableau de nmemb éléments de taille size octets, et renvoie un pointeur vers la mémoire allouée
- Cette zone est remplie avec des zéros
- Si nmemb ou size vaut 0, calloc() renvoie NULL
- L'allocation par calloc de p blocs de n octets conduira à utiliser un peu moins de mémoire que p allocations de n octets par malloc.

La fonction calloc

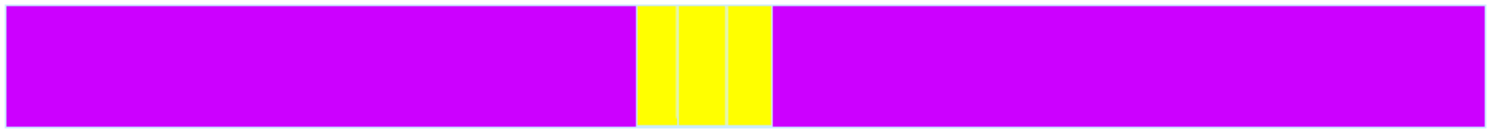
- **Syntaxe :**

  #include <stdlib.h>

  void *calloc(size_t nmemb, size_t size) ;

- **Exemple :**

  calloc(3, sizeof(int))

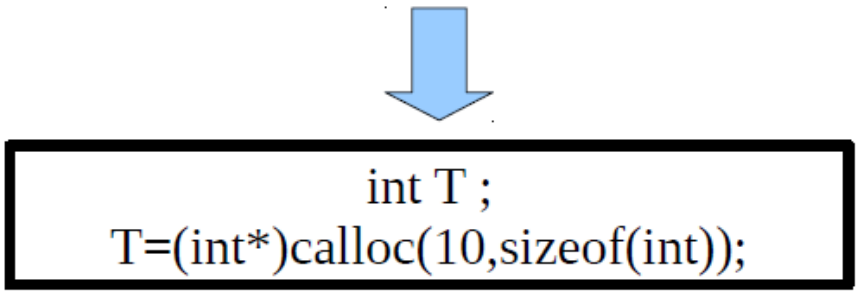


Allocation dynamique de place mémoire(pour 3 entiers)

La fonction calloc

- **Exemple :**

La déclaration statique d'un tableau d'entiers int T[10] devient:



T	0	0	0	0	0	0	0	0	0	0

La fonction free

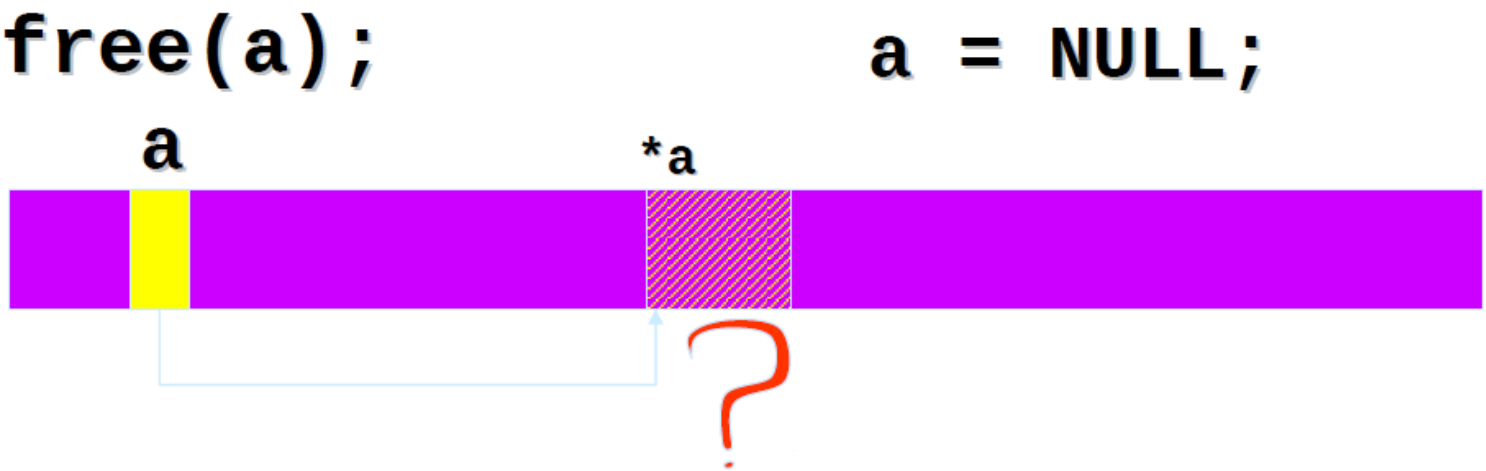
- Cette fonction libère l'espace mémoire pointé par ptr, qui a été obtenu lors d'un appel antérieur à malloc(), calloc() ou realloc()
- Si le pointeur ptr n'a pas été obtenu par l'un de ces appels, ou s'il a déjà été libéré avec free(ptr), le comportement est indéterminé
- Si ptr est NULL, aucune tentative de libération n'a lieu

La fonction free

- **Syntaxe :**

```
#include <stdlib.h>
void free(void *ptr);
```

- **Exemple :**



Libération dynamique

La fonction realloc

- Elle modifie la taille du bloc de mémoire pointé par ptr préalablement allouée (par malloc, calloc ou realloc) pour l'amener à une taille de size octets
- Elle conserve le contenu de la zone mémoire minimum entre la nouvelle et l'ancienne taille
- Le contenu de la zone de mémoire nouvellement allouée n'est pas initialisé

La fonction realloc (suite)

- Si ptr est NULL, l'appel est équivalent à malloc(size),pour toute valeur de size
- Si size vaut zéro, et ptr n'est pas NULL, l'appel est équivalent à free(ptr)
- Si ptr n'est pas NULL, il doit avoir été obtenu par un appel antérieur à malloc(), calloc() ou realloc()

La fonction realloc

- **Syntaxe :**

```
#include <stdlib.h>
void *realloc(void *ptr, size_t size);
```

- **Exemple :**

```
int* a = (int*)malloc(3*sizeof(int));
a=realloc(4*sizeof(int));
```