Programmation C

16 - Allocation dynamique

Exécuter un programme

Disque dure

mon_programme.exe

Chargement

RAM (mémoire vive)

mon programme

autre programme

Mémoires et programme

Zone réservée à d'autres programmes

Zone des instructions et données statique globales

Zone mémoire pour charger dynamiquement les librairies

Données allouées dynamiquement

Mémoire libre, utilisable pour étendre le tas et la pille

Zone mémoire de données spécifiques à une procédure

La Stack (pile)

Stack ou pile

Zone mémoire de données spécifiques à une procédure

Stack

- frame x
- frame ...
- frame 2
- frame 1
- frame 0

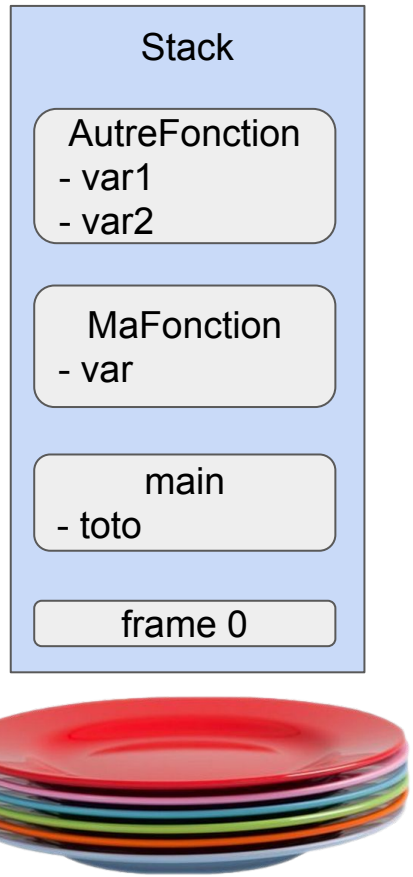
frame X — Procédure ou fonction (paramètres, variables, valeur de retour)

frame 1 — Correspond à la fonction main

La Stack: exemple

```
void MaFonction()
{
   int var = 12;
   var = AutreFonction(2, 63);
}

int AutreFonction(int var1, int var2)
{
    return var1 + var2;
}

int main()
{
   char toto='E';
   MaFonction();
   return 0;
}
```

Exemple



Heap ou tas

Stack

AutreFonction
- var1
- var2

MaFonction
- var

main
- toto

frame 0

Tas

age_utilisateur

mon_tableau

struct client

PILE VS TAS

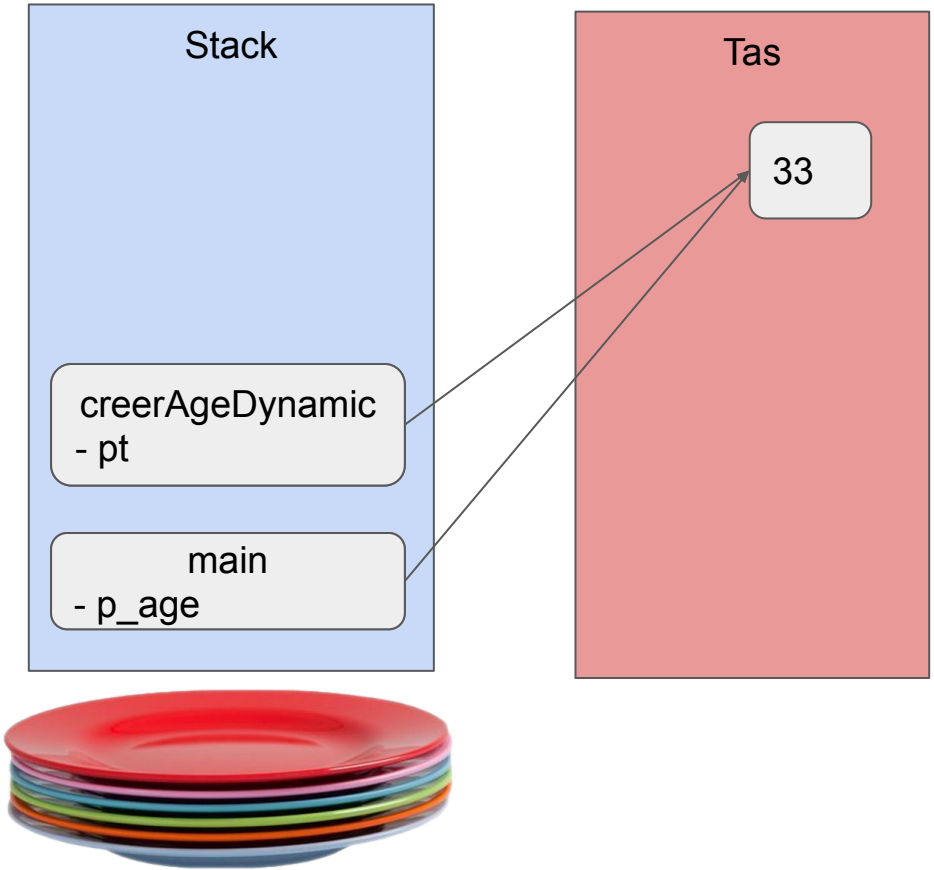
Stack

creerAge
- age=33

main
- p_age=&age

Tas

PILE VS TAS



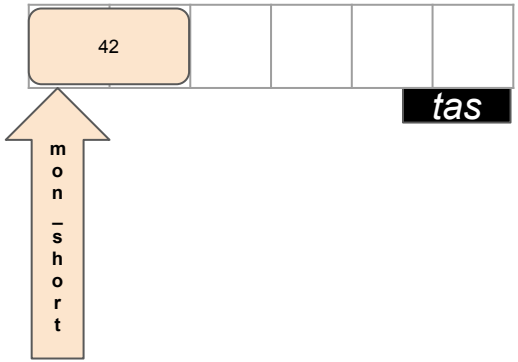
stdlib.h

Allouer de la mémoire avec Malloc

void* malloc (taille en octets)

```
int main()
{
  short* mon_short = NULL;
  mon_short = (short*)malloc( sizeof(short) );

  if(mon_short != NULL)
  {
     *mon_short = 42;
      printf("%d", *mon_short);
  }

  retrurn 0;
}
```

Exemple



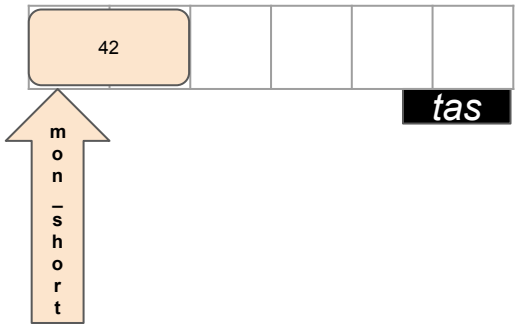
malloc retourne NULL si échec sinon l'adresse

Libérer de la mémoire avec Free

free (pointeur)

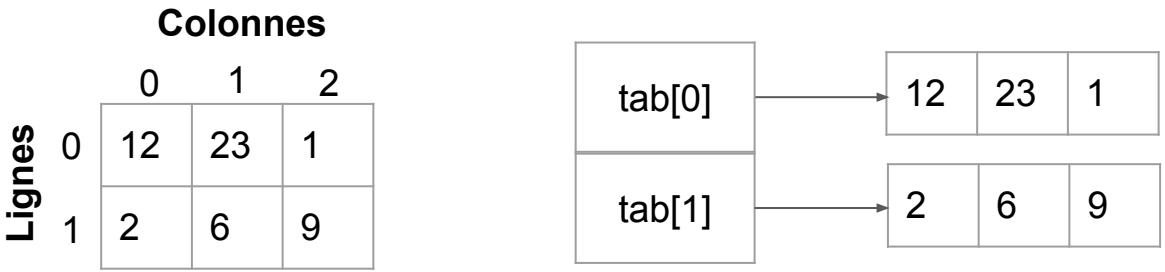
```
int main()
{
  short* mon_short = NULL;
  mon_short = (short*)malloc( sizeof(short) );

  if(mon_short != NULL)
  {
     *mon_short = 42;
      printf("%d", *mon_short);
      free(mon_short);
  }

  retrurn 0;
}
```

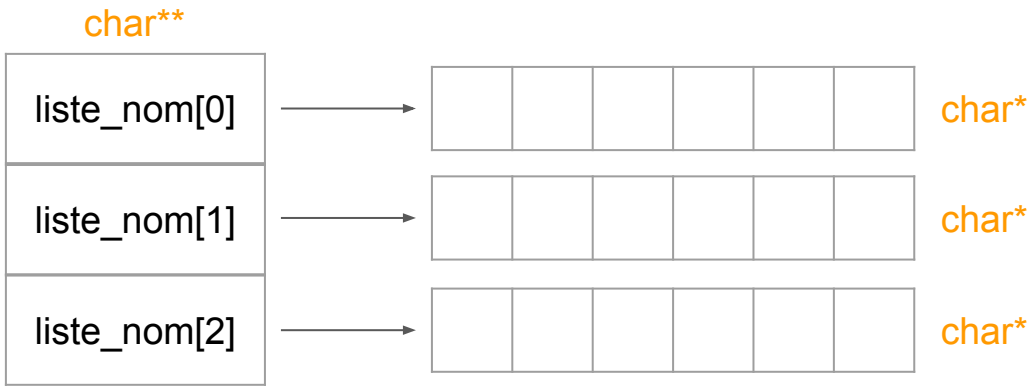
Exemple



Attention ne pas passer de pointeur NULL

Tableaux 2D = tableau de tableaux



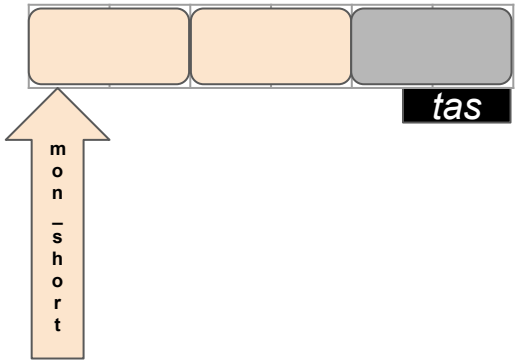


Réallouer de la mémoire avec Realloc

void* realloc (pointeur , taille en octets)

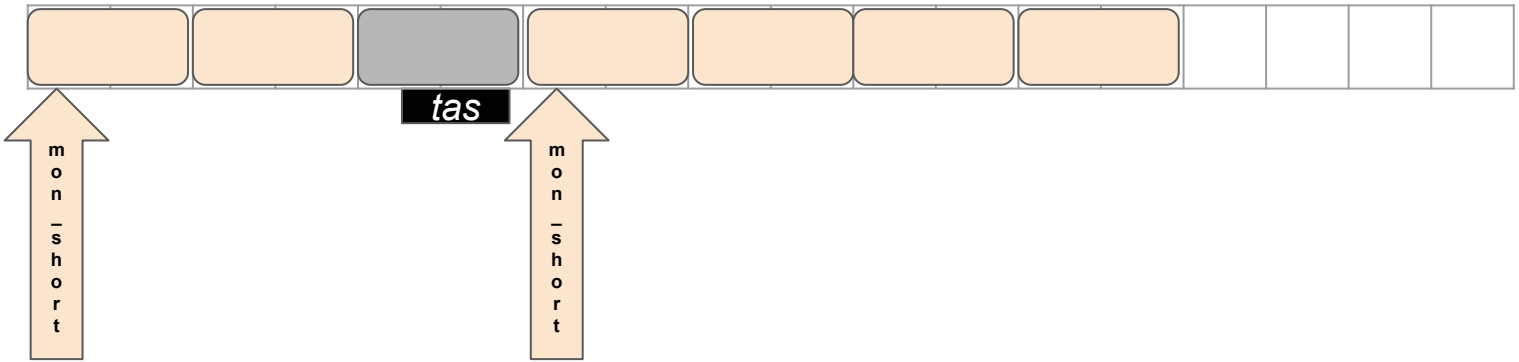
```
int main()
{
  short* mon_short = NULL;

  mon_short = (short*)malloc( sizeof(short) );
  mon_short = (short*)realloc(mon_short, 2 * sizeof(short) );

  retrurn 0;
}
```

Exemple



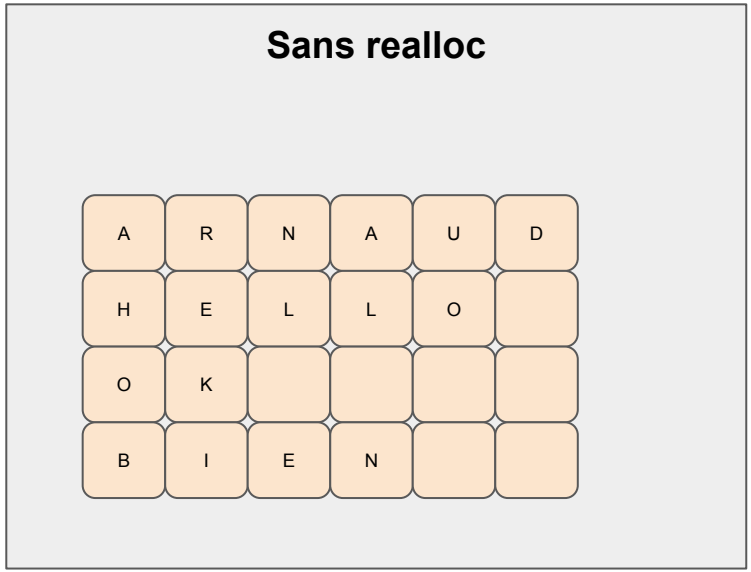
Adresse retournée par realloc peut être différente de l'entrée

Réallouer de la mémoire avec Realloc




```
...
  mon_short = (short*)realloc(mon_short, 4 * sizeof(short) );
...
```

Réallouer de la mémoire avec Realloc

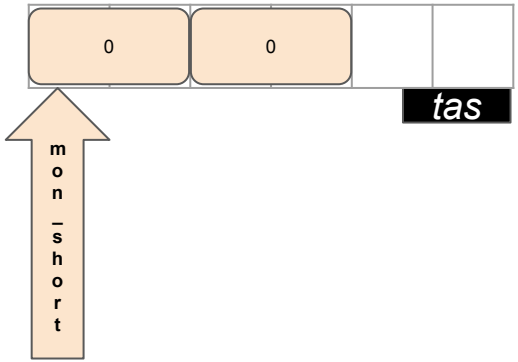


Allouer et initialiser à 0 la mémoire avec calloc

void* calloc(nombre elements **,** taille element **)**

```
int main()
{
  short* tab = NULL;
  tab = (short*)calloc( 2, sizeof(short) );

  retrurn 0;
}
```

Exemple



écrit la valeur binaire “0000 0000” dans chaque octet