Allocation mémoire en C

Statique, pile, tas

Allocation statique

- Concerne les variables globales
- Gestion simple:

  la variable est allouée

  pour toute la durée

  de vie du programme

```c
#include <stdio.h>
#include <stdlib.h>

#define N 100

// Variables globales
// Allocation statique
// Durée de vie = celle du programme
int g_tableau[100];
int g_variable;

int main(void)
{
    printf("Hello World\n");
    return EXIT_SUCCESS;
}

```

Allocation sur la pile

- Concerne les variables locales et les paramètres d'une fonction
- La mémoire est allouée sur la pile à l'appel de la fonction puis est libérée à la sortie.

```c
#include <stdio.h>
#include <stdlib.h>

#define N 100

// Variables globales
// Allocation statique
// Durée de vie = celle du programme
int g_tableau[100];
int g_variable;

int fonction(int a)
{
    // Variables locales
    // Allocation sur la pile
    // Durée de vie = celle de la fonction
    int l_tableau[100];
    int l_variable;

    printf("fonction");
    return 1;

    return l_tableau;
}

int main(void)
{
    printf("Hello World\n");
    return EXIT_SUCCESS;
}
```

Allocation dynamique

- Permet d'allouer ou de libérer de l'espace mémoire quand on en a besoin.
- Permet d'allouer uniquement la taille dont on a besoin.
- La mémoire allouée n'est libérée que sur un appel à free.

```
25  int* fonction2(int taille)
26  {
27      int i;
28      int *l_pointeur;
29
30      //Allocation dynamique d'un tableau de taille variable
31      l_pointeur = (int *) malloc(taille*sizeof(int));
32      for (i=0; i<taille; i++)
33      {
34          l_pointeur[i] = 0;
35      }
36      return l_pointeur;
    }
37
38  int main(void)
39  {
40      int *l_point;
41
42      l_point = fonction2(10);
43      //Ne pas oublier de libérer la mémoire
44      free(l_point);
45      return EXIT_SUCCESS;
46  }
```