La mémoire

La mémoire dans un ordinateur est une succession d'**octets** (soit 8 bits), organisés les uns à la suite des autres et accessibles par une adresse. L'unité est le ***byte*** qui est fixée généralement à 8 bits.

En **C/C++**, la mémoire pour stocker des variables est organisée en deux catégories :

- la **pile** (*stack*)
- le **tas** (*heap*)

Remarque : Dans la plupart des langages de programmation compilés, la **pile** (*stack*) est l'endroit où sont stockés les paramètres d'appel et les variables locales des fonctions.

① La **pile** (*stack*) est un espace mémoire réservé au stockage des variables désallouées automatiquement. Sa taille est faible et limitée. La pile est bâtie sur le modèle **LIFO** (*Last In First Out*) ce qui signifie "Dernier Entré Premier Sorti".

② Le **tas** (*heap*) est l'autre segment de mémoire utilisé lors de l'allocation dynamique. Sa taille est souvent considérée comme illimitée mais elle est en réalité limitée. Les fonctions **`malloc`** et **`free`**, ainsi que les opérateurs du langage C++ **`new`** et **`delete`** permettent, respectivement, d'allouer et désallouer la mémoire sur le tas. L'accès à la mémoire est direct et réalisé par adresse.

Attention : La mémoire allouée dans le tas doit être désallouée explicitement (avec **`free`** ou **`delete`**).

```
int* pi = new int;    // alloue un entier, ou :
int* pi = new int(5); // ... et initialisé à 5
int* ti = new int[10]; // alloue 10 entiers
```

pi et ti sont des pointeurs sur des entiers

L'opérateur new réalise une allocation mémoire dans le tas et retourne l'adresse de la première case de cette zone.

```
*pi = 5; cout << *pi << endl;

*(ti + 0) = 10; cout << ti[0] << endl;
*(ti + 1) = 11; cout << ti[1] << endl;

delete pi; // libération de la mémoire

delete [] ti; // libération de la mémoire
```