Tableaux dynamiques en C et C++.

pile tas new delete[]

calloc free realloc

Enseignement de programmation en langages C et C++.

Olivier Baiplleux

Maitre de conférences HDR
à l'université de Bourgogne.

Version 2021.

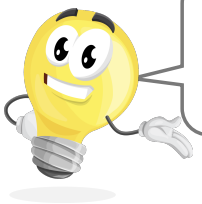
C et C++ Partie E

Olivier Bailleux, version 2021

Prérequis : Être capable de mettre en application les notions des parties A, B, C, D et de prévoir le comportement de programmes utilisant ces notions.

Notions abordées : Tableau dynamique en C et en C++ : créations, utilisation, destruction, affichages en C++.

Étape E01 : Tableaux dynamiques en C



Ce que nous avons vu dans les parties précédentes appartient au langage C et est utilsiable en C++. Nous abordons maintenant les tableaux dynamiques qui se gèrent diféremment en C et en C++.

```
void presentation()
{
    int n = 5;

    int* tab = (int*) calloc(n, sizeof(int));

    for(int i=0; i < n; i++)
    {
        tab[i] = 3 * i;
    }

    afficheTabInt(tab, n);

    tab = (int*) realloc(tab, (n+1) * sizeof(int));

    tab[n] = 1;

    afficheTabInt(tab, n+1);

    free(tab);
}
```

0 3 6 9 12 1

Cet appel à **calloc** réserve un bloc mémoire de **n*sizeof(int)** octets, utilisable comme un tableau de **n** cellules contenant chacune une donnée de type **int**. Ce bloc est réservé dans une zone mémoire appelé **tas**. Tous les octets du bloc sont initialisés à 0. **calloc** retourne l'adresse du bloc réservé sous la forme d'un pointeur de type **void*** qui doit être "casté" (converti) en **int*** pour être placé dans la variable **tab**.

Un tableau dynamique s'utilise exactement comme un tableau automatique ou statique, avec la notation [ ] via un pointeur.

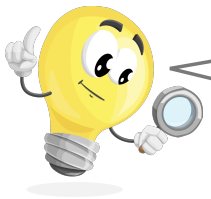
Cette fonction d'affichage d'un tableau de **int** a été définie dans la partie C du cours.

0 3 6 9 12

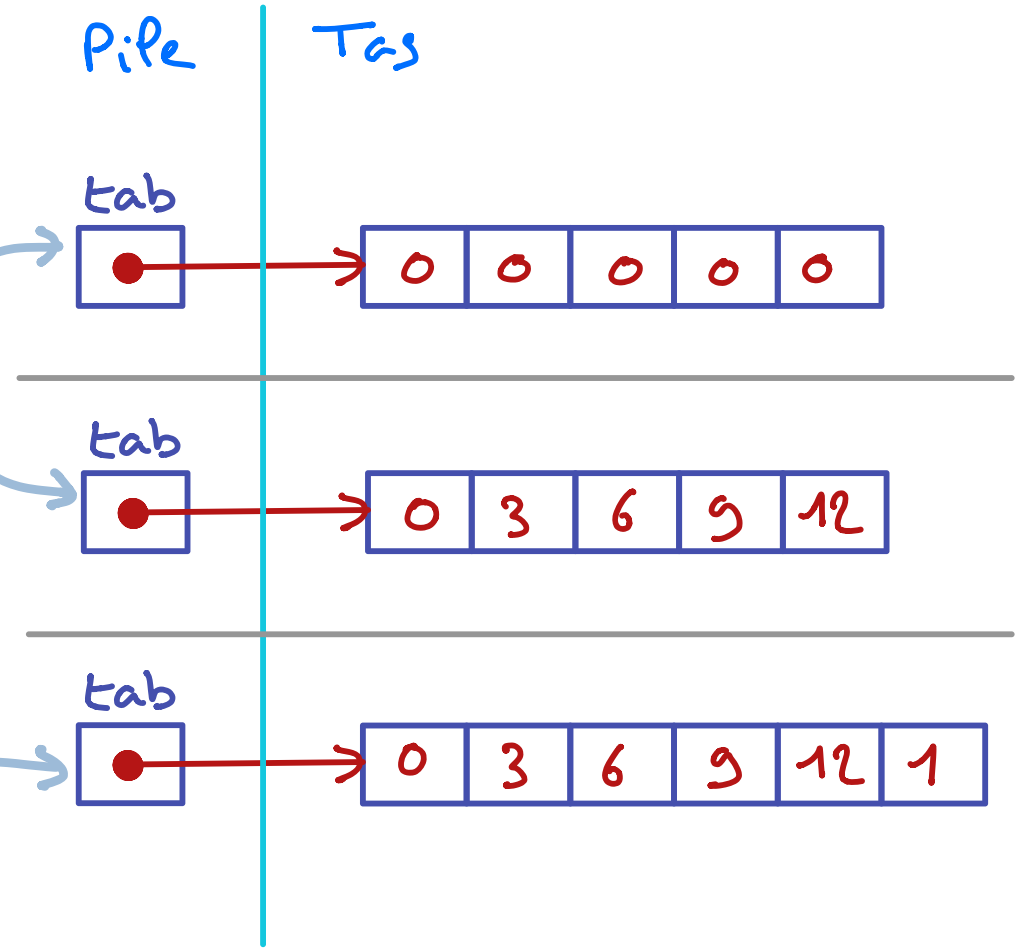
realloc change la taille d'un bloc mémoire. Le premier paramètre est l'adresse du bloc à redimensionner, et le deuxième paramètre la nouvelle taille qui peut être inférieure ou supérieure à la taille précédente. Ici on ajoute une cellule au tableau. Attention, cette opération peut déplacer le bloc. **realloc** retourne l'adresse du nouveau bloc, qui est assignée à la variable **tab** permettant l'accès au tableau.

La création d'un tableau dynamique par réservation d'un bloc dans le tas crée une "**dette mémoire**", c'est à dire l'obligation de libérer la mémoire réservée, par appel à la fonction **free**, lorsque le bloc n'est plus utilisé.

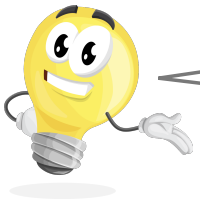
Étape E01 : Tableaux dynamiques en C




```c
void presentation()
{
    int n = 5;

    int* tab = (int*) calloc(n, sizeof(int));

    for(int i=0; i < n; i++)
    {
        tab[i] = 3 * i;
    }

    afficheTabInt(tab, n);

    tab = (int*) realloc(tab, (n+1) * sizeof(int));

    tab[n] = 1;

    afficheTabInt(tab, n+1);

    free(tab);
}
```

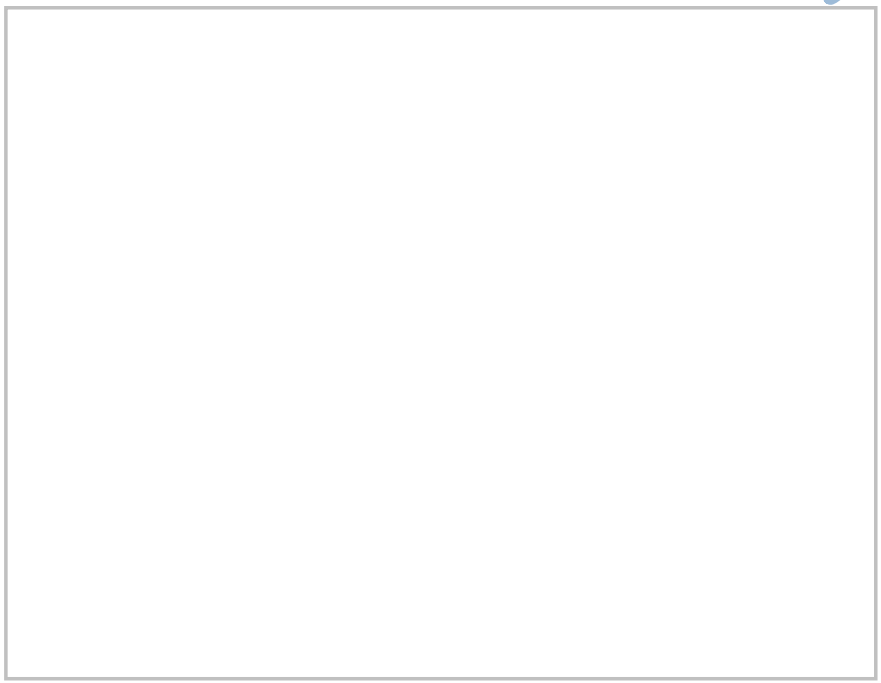


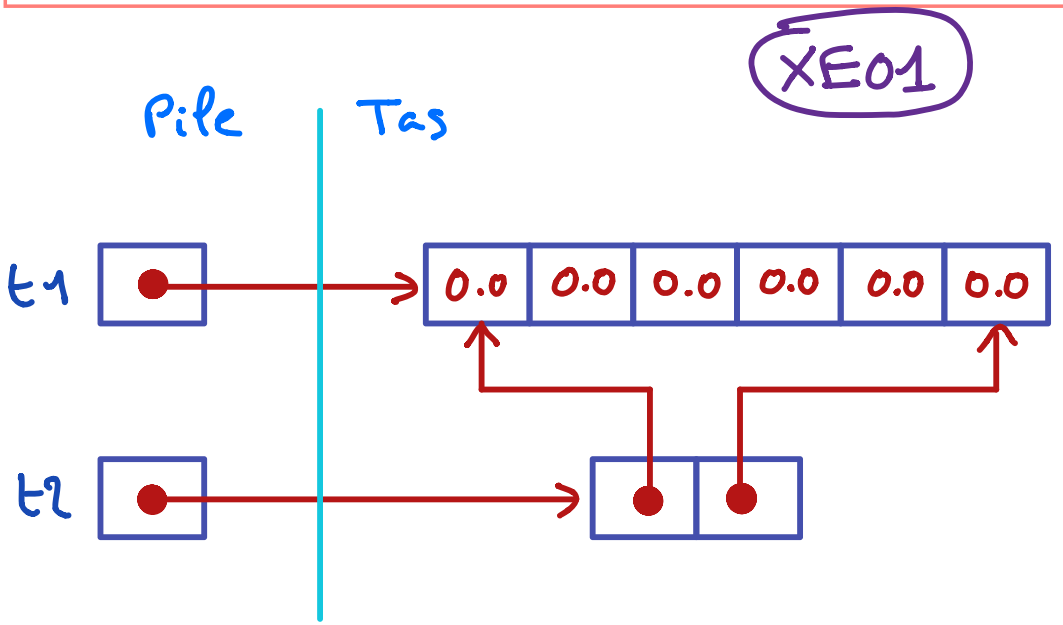
Étape E01 : Tableaux dynamiques en C



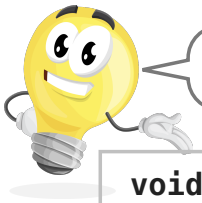
Cet exercice va vous permettre d'assimiler les notions que nous venons de découvrir.

Donnez les lignes de code permettant
- de créer un tableau dynamique de 6 cellules pouvant contenir chacune un **double**.
- de créer un tableau dynamique de 2 cellules contenant chacune un pointeur sur **double**.
- de placer dans la première cellule du deuxième tableau l'adresse de la première cellule du premier et dans la deuxième cellule du deuxième tableau l'adresse de la dernière cellule du premier.
- de libérer la mémoire occupée par les deux tableaux.





Étape E02 : Tableaux dynamiques en C++



Voyons maintenant comment gérer des tableaux dynamiques en C++

```
void presentation()
{
    int n = 5;

    int* tab = new int[n];

    for(int i=0; i < n; i++)
    {
        tab[i] = 3 * i;
    }
    afficheTabInt(tab, n);

    int* tmp = new int[n+1];

    for(int i=0; i < n; i++)
    {
        tmp[i] = tab[i];
    }

    delete[] tab;
    tab = tmp;
    tmp[n] = 1;

    afficheTabInt(tab, n+1);

    delete[] tab;
}
```

Pour créer un tableau dynamique en C++, la syntaxe est plus simple qu'en C. le bloc mémoire réservé est situé au même endroit, dans le tas. On contracte également une "dette mémoire" qu'il faudra rembourser en libérant la mémoire réservée lorsqu'on en aura plus besoin.

Le tableau créé avec **new** est en tous points semblable à celui créé avec **calloc** en C. La fonction d'affichage **afficheTab** pour les tableaux d'entier est utilisable.

0 3 6 9 12

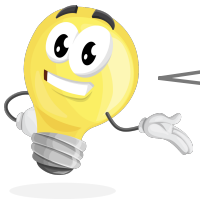
Le C++ ne dispose pas de moyen de changer la taille d'un tableau dynamique. Pour avoir un tableau plus grand, il faut en créer un nouveau, recopier le contenu de l'ancien dedans, et effacer l'ancien, ce que la fonction **realloc** du C permettait de réaliser en une seule ligne.

En C++, la destruction d'un tableau dynamique se fait à l'aide de l'instruction **delete[]**.

0 3 6 9 12 1

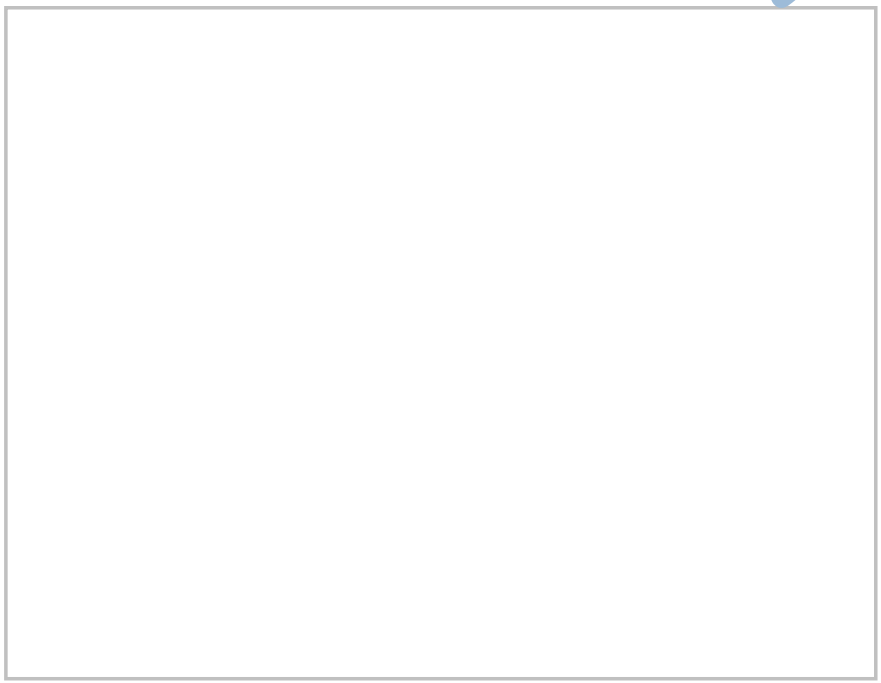
Les tableaux dynamiques C et C++ s'utilisent de la même manière mais il ne faut PAS utiliser **free** ni **realloc** avec un tableau créé par **new** et il ne faut pas utiliser **delete** avec un tableau créé avec **calloc**.

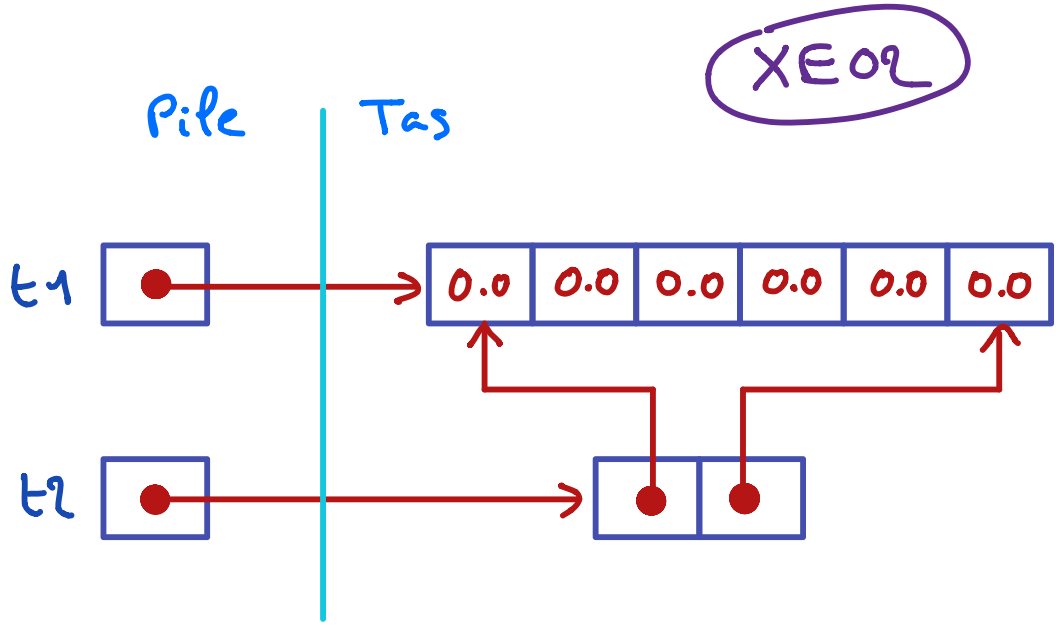
Étape E02 : Tableaux dynamiques en C++



Vous avez déjà traité cet exercice en C et vous devez maintenant le faire en C++, si possible sans consulter la solution en C.

Donnez les lignes de code permettant
- de créer un tableau dynamique de 6 cellules pouvant contenir chacune un **double**.
- de créer un tableau dynamique de 2 cellules contenant chacune un pointeur sur **double**.
- de placer dans la première cellule du deuxième tableau l'adresse de la première cellule du premier et dans la deuxième cellule du deuxième tableau l'adresse de la dernière cellule du premier.
- de libérer la mémoire occupée par les deux tableaux.





Qu'avons nous appris ?

Création d'un tableau dynamique en C	avec calloc
Destruction d'un tableau dynamqieu en C	avec free
Redimensionnement d'un tableau dynamique en C	avec realloc
Création d'un tableau dynamique en C++	avec new
Destruction d'un tableau dynamique en C++	avec delete[]
Redimentionnement d'un tableau dynamique en C++	necessite de créer un nouveau tableau, recopier le contenu de l'ancien, puis détruire l'ancien.

Exercices d'assimilation

Ces exercices vous permettront de monter en compétence et de vérifier vos acquis. Ne vous précipitez pas. Si possible, testez vos solutions sur machine. Essayez de comprendre vos erreurs. Au besoin demandez des indices ou explications complémentaires.

XE03

Réalisez une fonction nommée **fusion** en accepte en paramètres :
- un pointeur sur **int tab1** désignant un tableau d'entiers,
- un entier **n1** représentant la longueur du tableau désigné par **tab1**,
- un pointeur sur **int tab2** désignant un tableau d'entiers,
- un entier **n2** représentant la longueur du tableau désigné par **tab2**,

et qui retourne un tableau dynamique de longuer **n1** + **n2** contenant les éléments de **tab1** suivis de ceux de **tab2**.

En C — XE03C

En C++ — XE03C++

XE04

Réalisez en C (uniquement) une fonction **concat** qui accepte en paramètres deux pointeurs sur **char** désignant des chaînes de caractères (au format C) et retournant l'adresse d'une nouvelle chaîne, stockée dans un tableau dynamique, qui contient la concaténation des chaînes désignées par les deux pointeurs passés en paramètres.

XE05

Réalisez en C une fonction **main** qui :
- crée un chaîne "Tim " (au format C) dans un tableau dynamique dont l'adresse sera placée dans une variable s1,
- crée un chaîne "et Tom" (au format C) dans un tableau dynamique dont l'adresse sera placée dans une variable s2,
- appelle la fonction **concat** pour créer une chaîne dynamique contenant la concaténation des chaînes désignées par s1 et s2 et place l'adresse de cette nouvelle chaîne dans une variable s3.
- affiche avec **printf** la chaîne désignée par s3.
- libère toute la mémoire occupée par les chaînes ayant été créées.

Attention, pour placer une chaîne dans un tableau créé dans le tas, il faut utiliser **strcpy**. On ne peut pas directement initialiser un tableau dynamique avec une chaîne comme on peut le faire avec un tableau automatique.

Exercice de consolidation

Cet exercice est plus compliqué que les précédents. Ne le faites que si vous avez un bon niveau en algorithmique et que vous maîtrisez la récursivité.

XE06

Vous devez réaliser une fonction **`void fusion(int* t1, int n1, int* t2, int n2, int* t)`** qui, en supposant que **`t1`** désigne un tableau de taille **`n1`** contenant des entiers en ordre croissant et que **`t2`** désigne un tableau de taille **`n2`** contenant des entiers en ordre croissant, place dans le tableau désigné par **`t`**, supposé de taille **`n1 + n2`**, les valeurs de **`t1`** et **`t2`** classées en ordre croissant.

Vous devez réaliser en C++ une fonction **`void triFusion(int* tab, int n)`** qui, en supposant que **`tab`** désigne un tableau de taille **`n`**, trie en ordre croissant les valeurs situées dans ce tableau en appliquant l'algorithme suivant :
Si **`n`** est inférieur ou égal à 1, le tableau est déjà trié et il n'y a rien à faire.
Dans le cas contraire, on crée deux tableaux (dynamiques) **`t1`** et **`t2`** de tailles respectives **`n1 = n/2`** et **`n2 = n-n/2`**. On place les **`n1`** premiers élements de **`tab`** dans **`t1`** et les **`n2`** éléments suivant dans **`t2`**. Puis on utilise deux appels récursifs de la fonction **`triFusion`** pour trier **`t1`** et **`t2`**. Ensuite on utilise la fonction **`fusion`** pour fusionner les tableaux triées **`t1`** et **`t2`** dans le tableau **`tab`**. Enfin, on détruit les tableaux **`t1`** et **`t2`** pour libérer la mémoire utilisée par ces tableaux.

XE07