

Allocation dynamique

1 Pile ou Tas ?

Exercice 1. La mémoire.

Considérons le code suivant.

```
void foo( char *pa, char b )
{
    char tmp = *pa;
    *pa = b;
    b = tmp;
}
int main()
{
    char c = 'z';
    char a = 2, b = -3;
    char *pointeur = &b;
    *pointeur = 5;
    foo( &b, a );
}
```

Complétez le tableau ci-dessous représentant l'état de la mémoire (la pile), de façon simplifiée, **à la fin de l'exécution du code**. Rayez les variables qui ont disparu de la pile à la fin de l'exécution. Une première variable vous est donnée en tant qu'exemple.

Adresse	Valeur de la variable	Nom de variable
@0	'z'	c
@1		
@2		
@3		
@4		
@5		
@6		
@7		
@8		

Exercice 2. La mémoire II, le retour.

Considérons le code suivant.

```
int main()
{
    int i;
    short *tab = NULL, *pa = NULL, a = 1337;

    pa = &a;
    tab = (short*) calloc (4, sizeof(short));

    for ( i = 0; i < 4; i++, a += i )
        tab[i] = *pa;

    pa = &tab[2];
    *pa += *(pa + 1);
}
```

- 1) Décrivez l'effet des lignes de code suivantes, extraites du code ci-dessus :
 - `tab = (short*) calloc (4, sizeof(short));`
 - `for (i = 0; i < 4; i++, a += i)`
 - `*pa += *(pa + 1);`
- 2) Complétez le tableau donné à la fin du TD représentant l'état de la mémoire **à la fin de l'exécution du code**. N'oubliez pas de séparer les différentes variables par un trait dans le tableau. Les deux premières variables vous sont données en tant qu'exemple.

Exercice 3. Adresses des variables.

- 1) Pour chacun des éléments suivants, afficher l'adresse de la variable. Si c'est un pointeur, afficher également l'adresse pointée et si c'est un tableau, afficher l'adresse du premier élément.
 - Une variable de type `float`,
 - un tableau statique de type `char`,
 - un pointeur sur la chaîne "Douglas Power",
 - un pointeur sur un tableau de type `short` alloué dynamiquement.
- 2) Vous devez alors grouper les variables suivant les adresses qu'elles possèdent. Lesquelles sont sur la pile, le tas, autre chose ?

2 Allocation dynamique de tableaux

2.1 Les fonctions

2.2 Allocation de matrices

Exercice 4. Fun with tables (Moodle).

- 1) Créez la fonction `fillRandTab` permettant de remplir le tableau `tab` de taille `n` avec des valeurs aléatoires comprises entre `a` et `b` inclus. Pour tester votre fonction, allouez un tableau de façon dynamique dans le `main()` puis passez son adresse en argument de cette fonction. Pensez à le libérer à la fin du programme !

```
void fillRandTab (int *tab, int n, int a, int b);
```

- 2) Créez la fonction `allocMat` permettant d'allouer une matrice d'entiers formée de `nbRows` lignes et `nbCols` colonnes.

```
int **allocMat (int nbRows, int nbCols);
```

- 3) Créez la fonction de prototype `fillRandMat` permettant de remplir la matrice `mat` de dimensions `l x c` avec des valeurs aléatoires comprises entre `a` et `b` inclus. On suppose que la matrice est correctement allouée.

```
void fillRandMat (int **mat, int l, int c, int a, int b);
```

- 4) Créez la fonction `freeMat` permettant de libérer correctement la mémoire allouée pour le pointeur `mat`.

```
void freeMat (int **mat, int nbRows);
```

Pile			Tas		
Adresse	Valeur	Nom de var.	Adresse	Valeur	Nom de var.
0x0160			0x4AC0		
0x0161			0x4AC1		
0x0162			0x4AC2		
0x0163			0x4AC3		
0x0164			0x4AC4		
0x0165			0x4AC5		
0x0166			0x4AC6		
0x0167			0x4AC7		
0x0168			0x4AC8		
0x0169			0x4AC9		
0x016A			0x4ACA		
0x016B			0x4ACB		
0x016C			0x4ACC		
0x016D			0x4ACD		
0x016E			0x4ACE		
0x016F			0x4ACF		
0x0170			0x4AD0		
0x0171			0x4AD1		
0x0172			0x4AD2		
0x0173			0x4AD3		
0x0174			0x4AD4		
0x0175			0x4AD5		
0x0176			0x4AD6		
0x0177			0x4AD7		

FIGURE 1 – État de la mémoire après l'exécution du code de l'exercice 2.