esiea
ECOLE D'INGENIEURS
DU MONDE NUMERIQUE

TD n°6

INF1031

Allocation dynamique

Arnaud Bannier
Nicolas Bodin
Matthieu Le Berre

1. La théorie

Exercice 1. La mémoire

Considérons le code suivant.

```
1  void foo( char *pa, char b )
2  {
3     char tmp = *pa;
4     *pa = b;
5     b = tmp;
6  }
7  int main()
8  {
9    char c = 'z';
10   char a = 2, b = -3;
11   char *pointeur = &b;
12   *pointeur = 5;
13   foo( &b, a );
14 }
```

Complétez le tableau ci-dessous représentant l'état de la mémoire (la pile) **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.

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

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

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

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

2.1) Décrivez l'effet des lignes de code suivantes, extraites du code ci-dessus :
- `tab = (short*)calloc( sizeof(short), 4 );`
- `for ( i = 0; i < 4; i++, a += i )`
- `*pa += *(pa + 1);`

2.2) 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.

2. La pratique

Exercice 3. Adresses des variables

3.1) 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.

3.2) Vous devez alors grouper les variables suivant les adresses qu'elles possèdent. Lesquelles sont sur la pile, le tas, autre chose ?

Exercice 4. Fun with tables

4.1) Créez la fonction de prototype

```c
int* AllocTab( int n );
```

permettant d'allouer un tableau d'entiers de taille `n` (demandé à l'utilisateur avant l'appel à la fonction).

4.2) Créez la fonction de prototype

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

permettant de remplir le tableau `tab` de taille `n` avec des valeurs aléatoires comprises entre `a` et `b` inclus.

4.3) Créez la fonction de prototype

```c
void FreeTab( int *tab );
```

libérant le tableau `tab` précédemment alloué dynamiquement.

4.4) Faîtes de même pour les tableaux à deux dimensions en codant les fonctions suivantes.

```c
int** AllocMat( int nbRows, int nbCols );
void RandMat( int **mat, int nbRows, int nbCols, int a, int b );
void FreeMat( int **mat, int nbRows );
```

Exercice 5. Le pouvoir void *

Considérons le code suivant.

```c
int main()
{
    void *p = &p;
    printf( "%p\n", p );
    printf( "%p\n",      (char*)p + 1 );
    printf( "%p\n",     (short*)p + 1 );
    printf( "%p\n",       (int*)p + 1 );
    printf( "%p\n",     (float*)p + 1 );
    printf( "%p\n", (long long*)p + 1 );
    printf( "%p\n",    (double*)p + 1 );
}
```

5.1) Qu'affiche le code suivant ? Comment pouvez-vous l'expliquer ?

5.2) Que ce passe-t-il si vous ajoutez la ligne `p++;` au code précédent ? Pourquoi ?

Exercice 6. argc / argv

6.1) Il est possible de donner des arguments au point d'entrée du programme :

```c
int main( int argc, char *argv[] )
```

— `argc` est un entier donnant le nombre d'arguments présents sur la ligne de commande exécutant le programme (`./toto.exe 1 titi` contient 3 arguments).

— `argv` est un tableau de chaînes de caractères donnant la liste des arguments de la ligne de commande (`argv = {"./toto.exe", "1", "titi"}`).

Utilisez cela pour créer un programme qui affiche votre nom et votre âge, arguments que vous aurez saisis en ligne de commande. En cas de manque d'arguments, le programme doit afficher une erreur.

6.2) Créez une fonction permettant d'allouer un tableau 2D de type `char`, copiant les paramètres du `main()` dans ce nouveau tableau et le renvoyant. Comment récupérer les informations importantes telles que les dimensions de ce tableau 2D ? Ajoutez cela au code et créez une fonction permettant d'afficher caractère par caractère ce tableau.

Pile		
Adresse	Valeur de var.	Nom de var.
0x0160	4	i
0x0161		
0x0162		
0x0163		
0x0164	0x4AC0	tab
0x0165		
0x0166		
0x0167		
0x0168		
0x0169		
0x016A		
0x016B		
0x016C		
0x016D		
0x016E		
0x016F		
0x0170		
0x0171		
0x0172		
0x0173		
0x0174		
0x0175		
0x0176		
0x0177		
0x0178		
0x0179		
0x017A		
0x017B		
0x017C		
0x017D		
0x017E		
0x017F		

Tas		
Adresse	Valeur de var.	Nom de var.
0x4AC0		
0x4AC1		
0x4AC2		
0x4AC3		
0x4AC4		
0x4AC5		
0x4AC6		
0x4AC7		
0x4AC8		
0x4AC9		
0x4ACA		
0x4ACB		
0x4ACC		
0x4ACD		
0x4ACE		
0x4ACF		
0x4AD0		
0x4AD1		
0x4AD2		
0x4AD3		
0x4AD4		
0x4AD5		
0x4AD6		
0x4AD7		
0x4AD8		
0x4AD9		
0x4ADA		
0x4ADB		
0x4ADC		
0x4ADD		
0x4ADE		
0x4ADF		