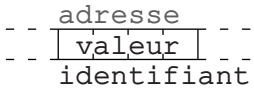
AP2
Travaux dirigés sur les pointeurs
Partie 1

Important pour chaque exercice, une schématisation des variables en mémoire devra être faite poursuivant la disposition exposée en cours.



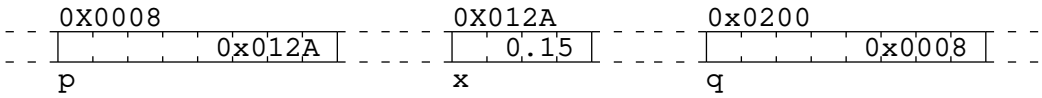
1 Manipulation basique de pointeur

Exercice 1. Schématiser l'état de mémoire à chaque ligne du programme ci-dessous en choisissant l'adresse de chaque variable déclarée.

```c
int main()
{
    int x = 4, y;
    int* p;
    p = &x;
    *p = 5;
    y = *p + 2;
    p = &y;
    *p = 12;
    return 0;
}
```

Exercice 2.

Soit trois variables : `x` de type `float`, `p` de type `float*` et `q` de type `float**` (pointeur sur pointeur) qui suivent la représentation ci-dessous.



(a) Schématiser les liens entre ses variables

(b) Écrire un programme qui déclare 3 variables qui respectent les mêmes liens.

(c) Pour chaqu'une de ces variables que vaut `variable`, `*variable`, `&variable`

Exercice 3. Pour chaque question, vous écrirez le un programme qui appelle chaque fonction.

(a) Écrire une fonction `triple` permettant de multiplier par 3 la valeur d'un réel passé en paramètre. Aucune valeur ne sera retournée, la variable d'origine sera modifiée

(b) Écrire une fonction `saisie_entier` réalisant la saisie d'un entier naturel passé en paramètre (attention on doit s'assurer que l'entier saisi est bien positif). Aucune valeur ne sera retournée, la variable d'origine sera modifiée.

(c) Écrire une fonction `permutation` réalisant la permutation circulaire de trois entiers passés en paramètre. Aucune valeur ne sera retournée, les variables d'origine seront modifiées.

Exercice 4.

(a) Écrire un algorithme `min_max` qui accepte en entrée un tableau statique et qui calcule le minimum et le maximum de ce tableau. La fonction ne rendra aucune valeur de retour mais et utilisera le passage par pointeurpour renvoyer les valeurs calculées.

(b) Le programme suivant (si on suppose que la fonction `min_max` est correctement implémentée) compile mais ne s'execute pas correctement, expliquer à travers des schémas de l'état de la mémoire pourquoi.

```
1  void min_max(float tab[], int taille, float* ptr_min, float* ptr_max){
2     /* définition de la question précédente */
3  }
4
5  int main()
6  {
7     float tab[] = {1.2, -5.3, 6.4, 3.2, 4.2, 41.2, 42.};
8     float* ptr_min;
9     float* ptr_max;
10    min_max(tab, 7, ptr_min, ptr_max);
11    AFFICHER("le minimum du tableau est", *ptr_min);
12    AFFICHER("le maximum du tabealu est", *ptr_max);
13    return 0;
14 }
```

(c) Écrire le programme précédent en rectifiant l'erreur.

2 Arithmétique des pointeurs

Exercice 5.

(a) Écrire un programme qui demande à l'utilisateur de saisir une variable entière `x` puis affiche sa valeur ainsi que son adresse.

(b) Récrire le programme précédent en utilisant un pointeur initialisé sur `x`. La saisie ainsi que l'affichage manipuleront uniquement le pointeur `px` et non la variable `x`.

(c) Écrire un programme demandant à l'utilisateur de saisir un tableau d'entiers puis affiche la valeur de chaque élément ainsi que son adresse. Vous utiliserez l'opérateur de référencement (`&`) et l'opérateur d'accès (`[]`)

(d) Réécrire le programme précédent sans utiliser les opérateurs de référencement et d'accès.

Exercice 6. Donnez la sortie à l'écran des programmes suivants :

Programme 1 :

```
1  #include <stdio.h>
2  int main()
3  {
4     int tab[]={0,1,2,3};
5     int taille=4;
6     int* p1;
7     int valeur1,valeur2;
8     p1 = tab+2;
9     valeur1 = *p1-tab[0];
10    valeur2 = 5+*(tab+1);
11    printf("valeur1 = %i, valeur2 = %i\n",valeur1,valeur2);
12    p1++;
13    printf("p1-tab = %i\n",p1-tab);
14    printf("*p1-tab[1] = %i\n",*p1-tab[1]);
```

```
    p1 = tab+taille-1;
    printf("*p1 = %i",*p1);
    return 0;
}
```

Programme 2 :

```
#include <stdio.h>
int main()
{
    int tab[10]={0,1,2,3,4,5,6,7,8,9};
    int *p;
    int *q;
    int a=2;
    int b=5;
    p=tab;
    q=p+2;
    printf("\n1. *p=%i,*q=%i",*p,*q);
    p=&a;
    if(p==q){
    printf("\n2. p==q");
    }
    else{
    printf("\n2. p!=q");
    }
    p=tab+a;
    q=tab+b;
    printf("\n3. *p=%i,*q=%i",*p,*q);
    p=&tab[8]-*q;
    q=p;
    p++;
    printf("\n4. *p=%i,*q=%i",*p,*q);
    return 0;
}
```

Exercice 7. Compiler et executer le programme suivant. Expliquer l'affichage du programme à travers des schémas

```
#include <stdio.h>
#define TAILLE 5

void mystere1(int* tab)
{
    int* q = tab;
    while (q - tab < TAILLE){
        printf("%li:%d\n", q - tab, *q - *tab);
        q++;
    }
}


void mystere2(int** copie, int* source)
{
    *copie=source;
}


```

```
20 int main(void)
21 {
22    int tab[TAILLE] = {1,3,2,4,5};
23    int* ptr;
24    mystere1(tab);
25    printf("adresse de tab : %p\n", tab);
26    mystere2(&ptr, tab);
27    printf("valeur de ptr : %p\n", ptr);
28    mystere1(ptr);
29    return 0;
30 }
```