

## TD 1 : Pointeurs, chaînes de caractères, mémoire

## ★ Exercice 1. Pour commencer simplement...

▷ Question 1. Quelles sont les valeurs des variables après exécution du programme suivant ? (faire un dessin de la mémoire).

```
int f(int x, int *y, int z) {
    int a;
    x = x + *y;
    a = 3;
    *y = *y - x;
    z = x - *y;
    return z;
}

int main(int argc, char *argv[]){
    int a, b, d, e;
    int *c;

    a = 5; b = 9;
    c = &e;
    *c = 23;
    d = f(a, &b, *c);
    return 0;
}
```

▷ Question 2. Ecrire une fonction `void echange(...x, ...y)` qui échange les valeurs de deux entiers passés en argument et la fonction `main()` qui utilise cette fonction.

▷ Question 3. Pourquoi la séquence suivante est-elle incorrecte ? Proposez une correction.

```
int *f(int a){
    int val[TAILLE_MAX];
    ...
    return(val);
}
```

▷ Question 4. Dans le manuel de référence du langage C on trouve la définition suivante :

```
char* gets(char *buffer)
```

La fonction `gets()` permet de lire une chaîne de caractères jusqu'à rencontrer le caractère `\n` ou fin de fichier ; elle recopie cette chaîne à l'adresse fournie en paramètre et remplace le caractère `\n` par le caractère de fin de chaîne `\0`.

En quoi le code suivant est-il incorrect ? Proposez une correction.

```
char *ptr;
gets(ptr);
```

★ Exercice 2. Quelles sont les valeurs des variables après chaque instruction de programme suivant ?

```
int a[] = {10, 20, 30, 40, 50};

main(){
1 int i, *pi, *pk, *b[2], **pl;
2 pi = &a[0];
3 pk = &a[1];
4 pl = &pk;
5 (*pl)--;
6 **pl = 0;
7 b[0] = &a[4];
8 b[1] = b[0];
9 b[0]--;
10 b[0]--;
11 *(b[0]) = 3;
12 pi++;
13 *pi = 4;
14 a[a[2]] = 1;
15 for(i=0; i<=4; i++)
16     printf(" a[%d] = %d ", i, a[i]);
17 printf("\n");
18 }
```

★ Exercice 3. Commentez toutes les étapes de ce programme en décrivant le contenu des différentes variables (faire un schéma du tableau et des variables). Indiquez également ce qu'il imprime.

```
#include <stdio.h>

int main() {
    char mot[] = "VACANCE";
    char *ptr, **ptr2;

    mot[1] = '0';
    ptr = mot + 2;
    *ptr = mot[0] + 3;
    ptr++;
    ptr2 = &ptr;
    **ptr2 = *(mot + 3);
    *(&ptr2) = 'G';
    *(ptr + 1) = *(&ptr2 + 2);
    *(ptr + 2) = 'S';
    printf("Nouveau mot %s \n", mot);
    *(&ptr2++) = mot[7];
    printf("Nouveau mot %s \n", mot);
}
```



- ★ **Exercice 4.** On considère les déclarations suivantes d'un programme C : écrivez l'instruction correspondant au commentaire.

```
int main(int argc, char *argv[]){
    char msg[9] = "bonjour!" ;
    char *adrMsg, *ptrCar, c;
    short longueur;

    /* initialise adrMsg à l'adresse du 1er caractère 'b' de msg (adresse
    de début) */

    /* instruction identique à adrMsg = &msg[0]; */

    /* initialise ptrCar à l'adresse du dernier caractère (nul) de msg */

    /* ramène le pointeur sur le caractère '!' */

    /* stocke la longueur de la chaîne "jour" */

    /* stoke le caractère pointé par ptrCar dans la variable c */

    /* décrémente ptrCar, obtient le caractère 'r' et le range dans la
    variable c */

    /* ramène le pointeur sur le caractère '!' */

    /* stocke le caractère pointé par ptrCar ('!') dans c, puis incrémente
    ptrCar */

    /* stocke le caractère contenu par la variable c à l'adresse du
    caractère 'j' */

}
```