

TD4

Langage C (LC4)

semaine du 27 février

1 Échauffement

Exercice 1 Complétez le tableau en indiquant les valeurs des différentes variables au terme de chaque instruction du programme suivant. On peut aussi indiquer sur quoi pointent les pointeurs :

| programme | a | b | c | p1, *p1 | p2, *p2 |
|------------------------|---|---|---|---------|---------|
| int a, b, c, *p1, *p2; | × | × | × | × | × |
| a = 1, b = 2, c = 3; | | | | | |
| p1 = &a, p2 = &c; | | | | | |
| *p1 = (*p2)++; | | | | | |
| p1 = p2; | | | | | |
| p2 = &b; | | | | | |
| *p1 -= *p2; | | | | | |
| ++*p2; | | | | | |
| *p1 *= *p2; | | | | | |
| a = ++*p2 * *p1; | | | | | |
| p1 = &a; | | | | | |
| *p2 = *p1 /= *p2; | | | | | |

Solution:

Les valeurs finals de a, b, et c sont 6, 6, et 6.

2 Passage de paramètres par adresse

2.1 Avec des pointeurs...

Exercice 2 Soient `adra` et `adrb` les adresses de deux variables `a` et `b` de type `int`. Écrivez une fonction `void echange(int *adra, int *adrb)` qui échange les valeurs de `a` et `b`.

Solution:

```
void echange (int *adra, int *adrb){
    int temp = *adra;
    *adra = *adrb;
    *adrb = temp;
}
```

Exercice 3 L'instruction `scanf("%d %d", &n, &m);` lit deux valeurs entières écrites en base décimale et les place dans les variables `n` et `m`. À quoi servent les `&` ?

Exercice 4 Écrivez une fonction `int read_int(int *adr)` qui lit sur l'entrée standard (le clavier, par défaut) une valeur entière écrite en décimal et la stocke à l'adresse `adr`. En pratique, la fonction lira (au moyen de `getchar()`) une suite de caractères entre `'0'` et `'9'` et convertira cette suite en un entier. La valeur de retour de la fonction servira à signaler s'il y a eu une erreur de saisie. La fonction renverra 1 si la valeur entrée était bien un entier et 0 sinon.

Solution:

```
int read_int(int *adr){
    int res=-1;
    *adr = 0;
    while(1){
        int i = getchar();
        if(48 <=i && i <= 57){
            res=1;
            *adr = *adr * 10 + i - 48;
            printf("*adr: %d\n", *adr);
        } else {
            return res;
        }
    }
}
```

Exercice 5 Écrivez une fonction : `int set_max(int tab[], int taille, int *adrmax)` qui parcourt le tableau, stocke à l'adresse `adrmax` la valeur la plus grande du tableau, et renvoie le nombre d'occurrences de cette valeur. Écrivez ensuite un exemple d'instruction qui appelle cette fonction.

Solution:

```
#include <stdio.h>

int set_max(int tab[], int taille, int *adrmax){
    int i;
    int occ = 0;
    *adrmax = tab[0];
    for(i = 0; i < taille; i++){
        if(tab[i] > *adrmax){
            *adrmax = tab[i];
            occ = 0;
        }
        if(tab[i] == *adrmax) occ++;
    }
    return occ;
}

int main(){
    int a[] = {5,2,3,3,3,5,1,3};
    int max;
    int occ = set_max(a, 8, &max);
    printf("maximum: %d\noccurences: %d\n", max, occ);
    return 0;
}
```

Exercice 6 Écrire une fonction

```
char *recherche(char *s, char c)
```

qui renvoie un pointeur vers la première occurrence dans la chaîne `s` du caractère `c` passé en argument. Si ce caractère n'apparaît pas dans la chaîne, la fonction devra renvoyer `NULL`.

Solution:

```

char *recherche(char *s, char c) {
    if (s != NULL) {
        while (*s != '\0') {
            if (*s == c)
                return s;
            s++;
        }
    }
    return NULL;
}

```

Exercice 7 À l'aide de la fonction précédente, écrire une fonction

```
int compte(char *s, char c)
```

qui renvoie le nombre d'occurrences de `c` dans `s`.

Solution:

```

int compte(char *s, char c) {
    int n = 0;
    s = recherche(s, c);
    while (s != NULL) {
        n++;
        s++;
        s = recherche(s, c);
    }
    return n;
}

```

3 Arithmétique des pointeurs

Exercice 8 Écrivez deux versions d'une fonction `int my_strlen(char* s)` qui renvoie la longueur d'une chaîne de caractères. La première version doit utiliser la notation des tableaux, et la deuxième l'arithmétique des pointeurs.

Solution:

```

int my_strlen1(char* s){
    int i=0;
    while(1){
        if(s[i] == '\0')return i;
        i++;
    }
}

int my_strlen2(char* s){
    int i=0;
    while(1){
        if(*(s+i) == '\0')return i;
        i++;
    }
}

```

Exercice 9 Écrivez une fonction `void inverse(char* s, int n)` qui reçoit un pointeur `s` sur une chaîne de caractères de longueur `n`, et qui inverse la chaîne. Utilisez l'arithmétique des pointeurs.

Solution:

```
void inverse(char* s, int n){
    int i;
    for(i = 0; i < n / 2; i++){
        char c = *(s+i);
        *(s+i) = *(s+n-i-1);
        *(s+n-i-1) = c;
    }
}
```