

מצביע כפול, מערך מצביעים, מצביע לפונקציה, מבוא ל-ADT

מצביע לפונקציה.

נסתכל בהגדרה הבאה :

```
void (*Func)(int , int )
```

Func – מצביע לפונקציה מסוג void שמקבלת שני פרמטרים מסוג int. קריאה לפונקציה דרך מצביע מתבצעת בצורה הבאה :
Func (a, b) או (*Func)(a, b)

נראה בהמשך את השימוש במצביע לפונקציה בכתיבת פונקציות כלליות. פונקציה כללית היא סוג של template שיודעת לעבוד עם כל טיפוס נתונים שבעולם בלי לעבור שינוי כלשהו. למשל, פונקציה שיודעת למיין מערך אבל מערך מכל טיפוס נתונים – מערך של מספרים שלמים, מערך של מספרים ממשיים וכו.

(ADT - abstract data type) הוא מבנה נתונים מופשט שמגדיר ממשק והוא חסר מימוש, ויכולים להיות מבני נתונים שונים שמממשים את הממשק שהוא מציע. בעזרת שימוש במצביע כללי ובמצביעים לפונקציות, נוכל ללמוד כללים למימוש תכנית ADT. נוכל לכתוב תכנית שיודעת כל פעם להתייחס לסוג אחר של נתונים.

כללים (לא פורמאליים) לכתיבת תכנית לפי כללי ADT

ב- ADT נחלק את כל הפונקציות ל**כלליות** ו**ספציפיות**.

פונקציה כללית – היא פונקציה שלא משתנת (לא ב-prototype ולא בתוכן) כאשר עוברים מטיפוס נתונים אחד לאחר. בפונקציה כללית אסור לבצע casting עבור הנתונים. פונקציה כללית צריכה להכיל כמה שיותר קוד (רוב הקוד נכנס לפונקציות כלליות).

פונקציה ספציפית היא פונקציה שמתאימה לטיפוס נתונים אחד בלבד.

בפונקציה ספציפית, בדרך כלל, ישנו casting לאותו טיפוס נתונים.

פונקציה ספציפית צריכה להכיל כמה שפחות שורות קוד (רוב הקוד נכנס לכללית).

פונקציה כללית אחת יכולה לקרוא לפונקציה כללית אחרת בשם שלה.

פונקציה כללית חייבת לקרוא לפונקציה ספציפית אך ורק בעזרת מצביע לפונקציה.

פונקציה ראשית main היא תמיד פונקציה ספציפית – ב-main תמיד ידוע על איזה סוג נתונים מדובר !

מצביע לפונקציה - דוגמא 1

```
void young(int);
void old(int);

int main()
{
    void (*fp)(int);    /*pointer to function – local variable*/
    int age;
    printf("How old are you? ");
    scanf("%d", &age);
    fp = (age > 50) ? old : young;
    fp(age);           /*call to relevant function using pointer*/
    return 0;
}

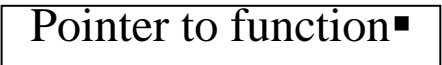
void young(int n) {
    printf("Being only %d, you sure are young.\n", n);
}

void old(int m) {
    printf("Being already %d, you sure are old.\n", m);
}
```

מצביע לפונקציה - דוגמא 2

```
void young( int );
void old( int );
void greeting( void (*)(int), int );
```

```
int main() {
    int age;
    printf("How old are you? ");
    scanf("%d", &age);
    if (age > 50)
        greeting( old, age);    /*send function as argument*/
    else
        greeting( young, age);
    return 0;
}
```



```
void greeting(void (*fp)(int), int k) {
    fp(k);    /*call to function by pointer*/
}
```

```
void young(int n) {
    printf("Being only %d, you sure are young.\n", n);
}
```

```
void old(int m) {
    printf("Being already %d, you sure are old.\n", m);
}
```

מצביע לפונקציה וכתובת פונקציות כלליות

- דוגמא 3

```
#include <stdio.h>

void use_int( void *);
void use_float( void *);
void greeting( void (*)(void *), void *);

int main()
{
    char ans;
    int i_age = 22;
    float f_age = 22.0;
    printf("Use int (i) or float (f)? ");
    scanf("%c", &ans);
    if (ans == 'i')
        greeting( use_int, &i_age);
    else
        greeting( use_float, &f_age);

    return 0;
}
```

דוגמא 3 - המשך

Pointer to function ■

```
/*General function*/  
void greeting(void (*fp)(void *), void *q) {  
    fp(q); /*call to function by pointer*/  
}
```

פונקציה כללית

```
/*Specific function – for integers only*/  
void use_int( void *r) {  
    int a;  
    a = * (int *) r;  
    printf(“ As an integer, you are %d years old.\n”, a);  
}
```

פונקציות ספציפיות

```
/*Specific function – for floats only*/  
void use_float( void *s) {  
    float b;  
    b = *(float *) s;  
    printf(“ As a float, you are %f years old.\n”, b);  
}
```

שאלה חשובה למחשבה: האם בפונקציה כללית יכול להיות משתנה מסוג `int` או שהכל חייב להיות `void*`?

תרגיל

דוגמא 1

כתוב פונקציה כללית למחיקת נתון ממערך.
התאים במערך צריכים להיות מסוג void* (מצביע כללי) כאשר כל תאים מצביעים לנתונים ספציפיים.

דוגמא:

עבור מערך מספרים שלמים 23 7 4 1, אחרי הוצאת 7
המערך נראה כך: 23 4 1.
עבור מערך מחרוזות cvk cvg bbg asd, אחרי הוצאת bbg
המערך נראה כך: cvk cvg asd

דרישות התכנית:

1.ה- prototype של הפונקציה:

```
int del_elemnt(int(*f1)(void*, void*), int(*f2)(void*),  
void *element, void **array)
```

כאשר f1 ו-f2 – מצביעים לפונקציות הבאות :

פונקציה לשיחרור נתון מסוג int :

```
int Free_Int( void* arg)  
{  
    free((int*) arg);  
}
```

תרגיל - המשך

פונקציה לשיחרור נתון מסוג char* :

```
int Free_String( void* arg)
{
    free((char*) arg);
}
```

פונקציה להשוואת שני מספרים מסוג int :

```
int Compare_Int( void* a , void* b)
{
    if (*(int*)a == *(int*)b)
        return 0;
    return 1; }
```

פונקציה להשוואת שתי מחרוזות:

```
int Compare_String( void* a , void* b)
{
    return strcmp( (char*)a, (char*)b);
}
```

void *element מגדיר את הנתון להוצאה,

void **array מגדיר את המערך.

2. הפונקציה del_elemnt מחזירה 1 אם הנתון נמצא והוצא,

או 0 - אם הנתון אינו איבר במערך.

3. אפשר להניח שקיים משתנה גלובלי CurrentCount (כמות הנתונים במערך), המתעדכן עם כל פעולת הכנסה/הוצאה מהמערך.

תרגיל - המשך

פתרון:

```
int CurrentCount;

int del_elemnt( int(*f1)(void *, void *), void(*f2)(void *),
void *element, void ** array)
{
    int i = 0;                /*Looking for an element*/
    while( i < CurrentCount && f1(array[i], element) !=0)
        i++;
    if (i == CurrentCount)    /*There is no such element*/
        return 0;
    f2(array[i]);             /*Using the special function for deleting*/
    while(i < CurrentCount-1 )
    {
        array[i] = array[i+1];    /*Shift left one place*/
        i++;
    }
    CurrentCount -= 1;
    return 1;
}
```

דוגמא נוספת

דוגמא 2

להלן מימוש של מיון MaxSort עבור מערך
מטיפוס int:

```
int index_of_max(int a[ ], int n)
{
    int i, i_max = 0;
    for(i = 1; i < n; i++)
        if(a[i] > a[i_max])
            i_max = i;
    return i_max;
}

void swap(int *a, int *b)
{
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
}
```

דוגמא נוספת - המשך

```
void max_sort(int a[], int n)
{
    int length;
    for(length = n ; length > 1; length--)
    {
        int i_max = index_of_max(a, length);
        swap(&a[length-1], &a[i_max]);
    }
}
```

יש לשנות את הפונקציות הנתונות כדי שהמיון יתבצע על מערך מצביעים מסוג void* .

דרישות התכנית:

שלושת הפונקציות אמורות להיות כלליות ז"א בלתי תלויות בטיפוס הנתונים (כמובן, למעט void*).

יש להוסיף פונקציה ספציפית אחת עבור טיפוס נתונים int. עליכם להחליט מה תפקידה ומה תבצע. יש להגדיר ולממש את הפונקציה.

שימו לב ! יש למיין את המערך מצביעים עצמו ולא את הנתונים המוצבעים ע"י המערך.

ה-prototype של הפונקציה swap אחרי השינוי אמור להיות:

void swap (void **a, void **b)

דוגמא נוספת - המשך

פתרון:

```
int Int_Comp(void* a, void* b)
{
    if(*(int*)a > *(int*)b)
        return 1;
    else
        return 0;
}
```

```
void swap( void **a, void **b )
{
    void* temp;
    temp=*a;
    *a=*b;
    *b=temp;
}
```

דוגמא נוספת - המשך

```
int index_of_max(void* a[], int n, int(*f)(void*,void*))
{
    int i, i_max = 0;
    for(i = 1; i < n; i++)
        if(f(a[i],a[i_max])==1)
            i_max = i;
    return i_max;
}
```

```
void max_sort(void* a[], int n, int(*f)(void*,void*))
{
    int length;
    for(length = n ; length > 1; length--)
    {
        int i_max = index_of_max(a, length,f);
        swap(&a[length-1], &a[i_max]);
    }
}
```