



Programming

Pointers

נדגים היום בעזרת מצביעים

- העברת משתנים לפונקציה שמשנה אותם
 - פונקציה שמקבלת מצביעים
- לסמן תא בזיכרון
 - פונקציה שמחזירה מצביע
- מערך של מערכים
 - למשל אוסף שמות: כל תא במערך הוא מחרוזת.
- לרוץ עם מצביע
 - אריתמטיקה של מצביעים.

תזכורת

```
void wishful_swap(double a, double b)
{
    double tmp;
    tmp=a;
    a=b;
    b=tmp;
}

void effective_swap (double* a, double* b)
{
    double tmp;
    tmp=*a;
    *a=*b;
    *b=tmp;
}
```

תזכורת

```
int main()
{
    double x,y;
    x = 3.14 ; y = 2.71;
    printf("x = %3.2lf , y = %3.2lf\n",x,y);
    wishful_swap(x,y);
    printf("x = %3.2lf , y = %3.2lf\n",x,y);
    effective_swap(&x,&y);
    printf("x = %3.2lf , y = %3.2lf\n",x,y);
    return 0;
}
```

x = 3.14 , y = 2.71

x = 3.14 , y = 2.71

x = 2.71 , y = 3.14

חימום – המרת קואורדינטות

```
void polar2rect(double r, double t, double* x, double* y)
{
    *x = r*cos(t);
    *y = r*sin(t);
}
```

כתבו main (שמשתמש בפונקציה זו) והדפיסו את
הקואורדינטות הקרטזיות של:

- א) מרחק 2 מטר מהראשית בכיוון רדיאן וחצי (מציר X החיובי).
- ב) מרחק 3.6 מטר מהראשית בכיוון עשירית רדיאן.

נ.ב. בשביל טריגונומטריה צריך `#include <math.h>`

Solution

```
int main()
{
    double x, y;

    polar2rect(2.0, 1.5, &x, &y);
    printf("(%.21f, %.21f)\n", x, y);
    polar2rect(3.6, 0.1, &x, &y);
    printf("(%.21f, %.21f)\n", x, y);
    return 0;
}
```

(0.14, 1.99)
(3.58, 0.36)

חימום

כתבו פונקציה בעלת המפרט הבא:

```
void cond_swap(int* i1, int *i2);
```

כאשר $i1$ ו- $i2$ הם מצביעים לאיברים שלמים.
הפונקציה תבדוק אם התא המוצבע ע"י $i1$ מכיל
ערך גדול מהתא המוצבע ע"י $i2$. אם כן, היא
תחליף בין תכני התאים.

Solution

```
void cond_swap(int* i1, int *i2)
{
    int tmp;

    if (*i1 > *i2)
    {
        tmp = *i1;
        *i1 = *i2;
        *i2 = tmp;
    }
}
```


לסמן תא: האיבר הגדול במטריצה

להלן פונקציה שמקבלת מטריצה ומחזירה מצביע לאחד התאים - לזה בעל האיבר הגדול ביותר.

```
double *largest_element_in_matrix(double mat[N][N])
{
    int i,j;
    double *result = &(mat[0][0]);

    for ( i = 0 ; i<N ; ++i ){
        for ( j = 0 ; j<N ; ++j ){
            if ( *result < mat[i][j])
                result = &(mat[i][j]);
        }
    }
    return result;
}
```

ניתן לשנות את התא המסומן

```
int main()
{
    double nice[N][N];
    double *peak = NULL;

    read_matrix(nice);
    printf("\n--Original matrix--\n");
    print_matrix(nice);
    peak = largest_element_in_matrix(nice);
    *peak = 3.1416;
    printf("--Modified matrix--\n");
    print_matrix(nice);
    return 0;
}
```

```
--Original matrix--
1.76  3.20  0.12
4.10  10.70 84.76
0.42   8.30  11.62
--Modified matrix--
1.76  3.20  0.12
4.10  10.70  3.14
0.42   8.30  11.62
```

דוגמא שימושית

נדגים קוד של פונקציה בעלת המפרט הבא:

```
int split_by_pointers(int numbers[],int* pointers[],int size)
```

הפונקציה מקבלת מערך של שלמים, מערך של מצביעים לשלמים ואת גודל שני המערכים `size`.

בסיום הפונקציה מערך המצביעים `pointers` צריך להכיל מצביעים לכל המספרים השליליים שבמערך השלמים `numbers`.

הערך המוחזר הוא אורך המערך (כלומר מספר השליליים שנמצאו). הניחו כי יש במערך שאתם מקבלים מספיק מקום לכתיבה.

שימו לב, המשתמש יגש לשליליים על ידי `pointers` בלי להכיר כלל את `numbers` !

Solution

```
int split_by_pointers(int numbers[],
                      int* pointers[], int size)
{
    int i, n=0;

    for (i=0; i<size; i++) {
        if (numbers[i]<0)
            pointers[n++] = &numbers[i];
    }
    return n;
}
```

Solution – equivalent

```
int split_by_pointers(int numbers[],
                      int* pointers[], int size)
{
    int n=0;
    int *p, *last = numbers+size;

    for ( p=numbers ; p<last ; p++){
        if (*p<0)
            pointers[n++]=p;
    }
    return n;
}
```

תרגיל כיתה (שימוש בפונקציה)

כתבו תכנית המקבלת מערך של שלמים בגודל 10 ומחליפה כל מספר שלילי בערכו המוחלט, תוך שימוש בפונקציה מהסעיף הקודם.

לדוגמא:

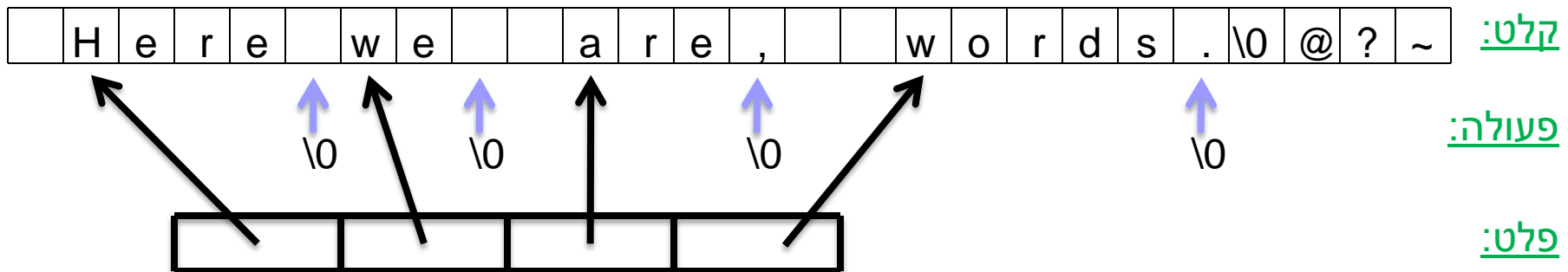
-20 11 6 5 4 3 7 -1 3 1

ישונה ל:

20 11 6 5 4 3 7 1 3 1

דוגמא מורכבת - פירוק משפט למילים

- קלט: משפט (מחרוזת עם רווחים וסימני פיסוק).
- פלט: מערך של מחרוזות ללא רווחים (מילים).
- הרעיון: להצביע לתוך המערך המקורי.
- צריך להכניס '\0' בכמה מקומות.



נשתמש ב `strchr` (למדנו עליה)

■ מקבלת מחרוזת ותו. `char *strchr (char *x, char c);`

□ אם התו אינו נמצא במחרוזת מחזירה `NULL`

□ אחרת מחזירה מצביע למופע הראשון.

```
int main ()  
{  
    char s[10] = "abcd";  
    char *x;  
    x = strchr(s, 'b');  
    printf("strchr(\"abcd\", 'b') = %p [%s], next letter is %c\n", x, x, x[1]);  
    return 0;  
}
```

`strchr("abcd", 'b') = 003BFEEED [bcd], next letter is c`

פונקציה שמפרקת מחרוזת למילים

```
int decompose(char *s, char *seperators, char *words[])
{
    int n = 0;

    while (*s!='\0'){
        /*First, skip all spaces before next word.*/
        while(strchr(seperators,*s)) ++s;
        if(*s=='\0')
            return n;

        words[n++] = s; /*Not space, not null => a word starts.*/
        /* loop over the word itself (no null nor space)*/
        while((*s!='\0') && (strchr(seperators,*s)==NULL))
            ++s;
        if(*s=='\0')
            return n;

        *s = '\0'; /*For sure s points to the first space after word.*/
        ++s;
    }
    return n;
}
```

שימוש בה

```
int main (void)
{
    char s[100];
    char *words[20];
    int n,j;

    printf("\nEnter a sentence: ");
    gets(s);
    printf("\n\t===>> %s\n",s);
    n = dicompose(s,"\t ,;. ",words);
    for ( j = 0 ; j<n ; ++j )
        printf("words #%d is \"%s\" \n",j,words[j]);
    return 0;
}
```

Enter a sentence:

===>> Here we are, in the center; of town

words #0 is "Here"

words #1 is "we"

words #2 is "are"

words #3 is "in"

words #4 is "the"

words #5 is "center"

words #6 is "of"

words #7 is "town"

Another exercise (optional)

כתבו פונקציה בעלת המפרט (prototype) הבא:

```
int convert (int num[], int size, int *p_value);
```

נאמר שהמערך `num` שגודלו `size` מייצג מספר אם התא הראשון בו מכיל את ספרת האחדות, התא השני את ספרת העשרות וכן הלאה. הפונקציה מחשבת את הערך המספרי שמייצג המערך `num` ומחזירה אותו דרך המשתנה ש- `p_value` מצביע אליו.

על הפונקציה להחזיר 1 במקרה שהמערך מייצג מספר שמתחלק ב 3 (למשל 1 7 3 4 או 8 3 4), אחרת 0 (למשל 1 2 1 0 או 5 2).

Solution

```
int convert (int num[],int size, int *p_value)
{
    int power=1;
    int itr;

    if (num[size-1]<1)
        return 0;
    *p_value=0;
    for (itr=0;itr<size;itr++)
    {
        if (num[itr]>=0 && num[itr]<=9)
            (*p_value)+=num[itr]*power;
        else
            return 0;
        power*=10;
    }
    return 1;
}
```

Exercise (optional) – Part B

כתבו פונקציה בעלת המפרט (prototype) הבא:

```
int calc_sum (int nums[][MAX_LEN], int digits,int n);
```

(עבור MAX_LEN שלם חיובי כלשהו).

הפונקציה מקבלת מערך של n מערכים בשם `nums` והיא מחשבת (ומחזירה) את סכום המספרים שמערכים אלו מייצגים, תוך התעלמות מן המערכים בהם המספר אינו מתחלק בשלוש.

Solution

```
int calc_sum (const char nums[][MAX_LEN], int digits, int n){
    int i, number, valid, sum=0;

    for(i=0; i< n; i++){
        valid = convert (nums[i],digits, &number);
        if(valid)
            sum +=number;
    }
    return sum;
}
```

Solution to class exercise

```
#include <stdio.h>

#define SIZE 10

int main()
{
    int arr1[SIZE];
    int *arr2[SIZE];
    int k,n;
    printf("Enter numbers\n");
    for (k=0;k<SIZE;k++)
        scanf("%d",&arr1[k]);
    n=split_by_pointers(arr1,arr2,SIZE);
    itr=0;
    for (k=0;k<n;++k)
        (*arr2[k])*=-1;
    for (k=0;k<SIZE;k++)
        printf("%d",arr1[itr]);

    return 0;
}
```