מבוא לתכנות מערכות

הרצאה 4

מערכים, מחרוזות בשפת C

מבוסס על הרצאות של אוניברסיטת חיפה, ומבוא למדעי המחשב, אורט בראודה, תכנה ותעו"ן

מערכים – מוטיבציה

בהינתן סדרת ציונים של סטודנטים בקורס, ברצוננו למנות ולהדפיס את הציונים שהם מעל ממוצע הציונים בקורס.

הקושי:

- על מנת לחשב את הממוצע יש לעבור מעבר ראשון על כל הציונים.
- רק לאחר חישוב הממוצע ניתן לומר אילו ציונים הם מעל הממוצע. לשם כך, יש צורך במעבר נוסף על רשימת הציונים.
- כדי לעבור יותר מפעם אחת על קבוצת נתונים יש לשמור את כל הנתונים בזיכרון עם קליטתם.

הפתרון: שימוש ב**מערכים**

מערכים - arrays

- מערך הוא סידרה רציפה בזכרון של תאים מאותו הטיפוס.
- איך נגדיר מערך? <type> <name>[size];
- למשל: int grades[5];
- הפקודה int grades[5];

תגרום להקצאה רציפה של 5 תאים מטיפוס int. למשל, אם sizeof(int)=2, אזי המערך grades יתפוס 10 בתים רצופים בזיכרון.

address	4200 4201	4202 4203	4204 4205	4206 4207	4208 4209
contents					
index	0	1	2	3	4

- שימו לב - מספור תאי המערך מתחיל מ – 0 !!!

הגדרת מערך

<type> <name>[size];

type : טיפוס של כל איבר במערך

name: שם המערך

size: מספר התאים במערך; גודל המערך חייב להיות ערך קבוע וידוע בזמן קומפילציה (אפשר להשתמש ב- #define או בערך מספרי).

התאים במערך ממוספרים מ-0 עד size-1.

מספר (מקום) התא נקרא אינדקס (או מציין).

מערך – אפשר שיהיה מכל סוג חוקי בשפה

```
int    arr_1[5];                /* array of integers */
float  vex_2[10];               /* array of float */
double v_d[100];                /* array of double */
char   s2[20];                  /* array of char */
int * p_x[30]                   /* arr. of pointers to int */
etc.        ...
```

מערכים – אתחול

```
int days[]={31,28,31,30,31,30,31,31,30,31,30,31}
```

- אתחול של מערך נעשה ע"י כתיבת רשימת איבריו בתוך סוגריים מסולסלים.
- כאשר גודל המערך מושמט, הקומפיילר יקצה מספר תאים לפי גודל סדרת המאתחלים.
- כאשר גודל המערך מצוין, ונתונים יותר מאתחלים מגודל המערך, תתקבל טעות בזמן קומפילציה.
- כאשר גודל המערך מצוין, ונתונים פחות מאתחלים מגודל המערך, יאותחלו אברי המערך הראשונים לפי הרשימה ושאר האיברים יאותחלו לאפסים.
- ניתן לאתחל את המערך כולו באפסים באופן הבא:

```
int counters [6] = {0};
```

הגדרה ואתחול

/* מערך של 3 ממשיים, מאותחל */

```
float arr[3] = {1.2, 10.16, 20.19};
```

√

/*מערך של 4 תווים, רק שניים מאותחלים, היתר מקבלים 0 */

```
char v2[4] = {'k', 't'};
```

√

/* חסר גודל מערך – אך יש אתחול, תקין */

```
int U_vec[ ] = {1,10,20,30,40};
```

√

הגדרה ואתחול – איך לא

/*סתירה בין גודל ואתחול!! */

```
int arr[3]={1,10,20,30,40};
```

X

/* אין גודל או אתחול!! */

```
int Vector[];
```

X

/* גודל 0 , שלילי, או לא מספר שלם */

```
int V0[0], V1[3.5];
```

X

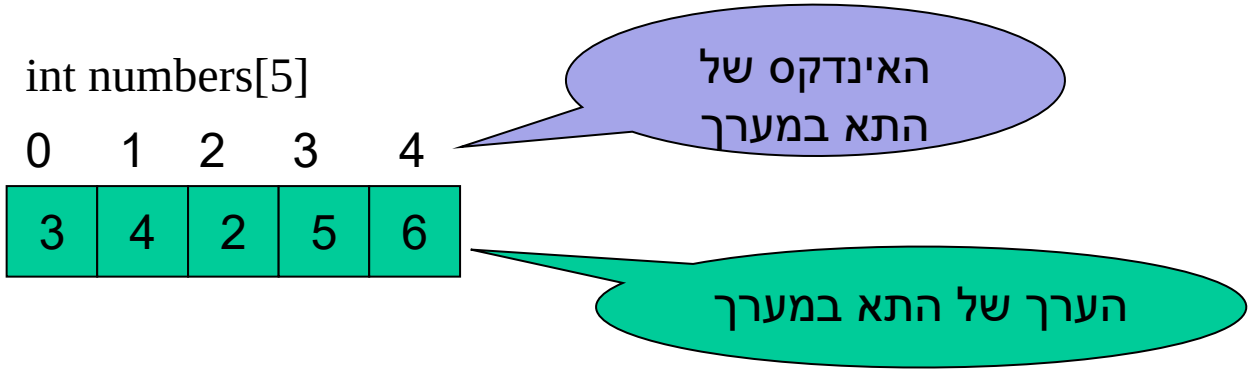
/* גודל המוגדר על סמך משתנה */

```
int n=5, Vx[n];
```

X

איך ניגשים לאיבר במערך?

- אופרטור []
- אם נרצה לגשת לאיבר מסוים במערך, חייבים להשתמש באופרטור זה.
- למשל:



- על מנת לגשת לתא השני במערך נקרא ל numbers[1]
- הערך שנקבל יהיה 4

גישה לאיברים במערך

נניח שהוגדר המערך הבא: int grades[20]

grades[0] מתייחס לתא ה**ראשון**. למשל, בהשמה: grades[0]=5

grades[1] מתייחס לתא **השני**. למשל, במשפט התנאי:

```
if (grades[1]<55) printf("fail\n");
```

grades[7] מתייחס לתא **השמיני**.

האינדקס (מציין) של האיבר הראשון במערך בשפת C הוא תמיד 0.

אינדקס של איבר במערך יכול להיות <u>ביטוי</u>, שערכו מספר טבעי או 0: קבוע, משתנה או ביטוי מורכב, אפילו כזה שערכו ידוע רק בזמן ריצה.

למשל: grades [len - i + 1] = 0

פעולות על מערכים

<u>הערה חשובה:</u>

שפת C אינה כוללת פעולות פרימיטיביות על מערכים. השוואה, העתקה, קליטה והדפסה של איברי מערך צריכה להיעשות ע"י שימוש בלולאה, איבר אחר איבר.

דוגמא:

```
for (i=0; i<NUM_GRADES; i++)
{
    scanf("%d", &(grades[i]));
}
```

מערכים - דוגמא

```
/* This program reads NUM_GRADES grades  and prints the average grade.  Then, it lists the above-average grades */
#include <stdio.h>
#define NUM_GRADES 5
int main()
{
	int grades[NUM_GRADES], sum=0, i;
	float average=0;
	printf ("Enter %d numbers", NUM_GRADES);
	for (i=0; i<NUM_GRADES; i++)
		scanf("%d", &(grades[i]));
	for (i=0; i<NUM_GRADES; i++)
		sum = sum+grades[i];
	average = (float)sum/NUM_GRADES;
	printf("The following grades are above average:\n");
	for (i=0; i<NUM_GRADES; i++)
		if (grades[i]>average)
			printf("%d\n", grades[i]);
	return 0;
}
```

הגדרת מערך בשם grades המכיל NUM_GRADES תאים, כל אחד מטיפוס int.

מערכים - דוגמא

```
/* This program reads NUM_GRADES grades  and prints the average grade.  Then, it lists the above-average grades */
#include <stdio.h>
#define NUM_GRADES 5
int main()
{
    int grades[NUM_GRADES], sum = 0, i;
    float average=0;
    printf ("Enter %d numbers", NUM_GRADES);
    for (i=0; i<NUM_GRADES; i++)
        scanf("%d", &(grades[i]));
    for (i=0; i<NUM_GRADES; i++)
        sum = sum+grades[i];
    average = (float)sum/NUM_GRADES;
    printf("The following grades are above average:\n");
    for (i=0; i<NUM_GRADES; i++)
        if (grades[i]>average)
            printf("%d\n", grades[i]);
    return 0;
}
```

קליטת נתונים לתוך המערך.
בכל איטרציה נקלט מספר אחד
לתוך התא ה- i (המשתנה i
משמש כמונה "הרץ" על
האינדקסים של המערך).

סכימת אברי מערך.

הדפסת התאים הרלוונטיים מן
המערך.

הקשרים שבין מערכים ומצביעים

שם של מערך הוא אוטומטית מצביע לתא הראשון במערך.

מה מתרחש כאשר אנחנו מגדירים מערך int a[10]; ?

בזמן ההגדרה הוקצה מקום ל-10 תאים מסוג int וגם הוקצה מקום עבור המצביע a שמקבל איתחול אוטומטי לכתובת של התא הראשון.

הטיפוס של משתנה a הוא int *.

<u>לדוגמה:</u>

```
int a[10], *p;
p=a;    (or   p=&a[0];)
```

$$a[i] \equiv *(a + i) \equiv *(p + i) \equiv p[i]$$

אריתמטיקה של מצביעים

int *p;

p+k : שווה לכתובת בזיכרון של k מקומות אחסון של שלמים אחרי הכתובת של p.
(k נקרא offset)

לדוגמה: אם p=2000 , k=3, int תופס 4 בתים בזיכרון:

p+k = 2000 + 3 *sizeof(int) = 2000+3*4 = 2012.

מהי המשמעות של: p++ ?

מקדם את p לתא הבא במערך.

int a[20], *p=&a[3], *q=&a[10];

האם הביטוי q-p אפשרי, ואם כן – למה שווה?

אפשרי: ההפרש בין המצביעים הוא 7.

מה ההבדלים בין הגדרת מערך ומצביע מבחינת הקצאת מקום בזיכרון וטיפוס?

```
int a[3];

int *p;
```

האם המשתנה a מאותחל ? כן, לאחר הקצאת מקום למערך בגודל 3, a מכיל את הכתובת של האיבר הראשון במערך, לכן a הוא מטיפוס int* (מצביע).

האם המשתנה p מאותחל ? לא. הוקצה מקום ל- p (שהרי הוא משתנה), אבל הוא (עדיין) לא מכיל כתובת של משתנה בתוכנית.

שאלה: האם שני המשתנים a ו-p בעלי אותו הטיפוס או טיפוסים שונים?

מצביעים ומערכים - המשך

ar+1 שקול ל-&ar[1]

ar+i שקול ל-&ar[i]

ar[i] שקול ל-*(ar+i)

- כמו לכל משתנה, גם למערך לא ניתן לשנות את כתובתו.
- אם הגדרנו למשל,

```
int arr[5]= {1, 2, 3, 4, 5}, *p=arr;
```

- אפשר לבצע:

```
(arr[2])++;  /* ≡ arr[2]=arr[2]+1; */
```

- ואפשר לבצע:

```
p++;
```

- **אבל אי אפשר לשנות את שם המקורי של המערך:**

```
arr++;
```

העברת מערך כפרמטר לפונקציה

אם פונקציה מקבלת מערך כפרמטר,כותרת הפונקציה היא:

```
void  func (int arr[]);
```

הקריאה לפונקציה תתבצע כך(**רק שם המערך מועבר**):

```
func(arr);
```

<u>הערה</u>: כאשר מעבירים מערך לפונקציה מה שמועבר הוא, למעשה, כתובת של תחילת המערך.

לכן, פונקציה מקבלת גישה למערך המקורי ולא יוצרת העתק של המערך!

אם הפונקציה מבצעת שינוי בערכי המערך, השינוי יישמר גם ביציאה מן הפונקציה(!)

חשוב! כיוון שפונקציה מקבלת רק מצביע לתחילת המערך, עליה לקבל בנפרד גם את גודל המערך!

העברת מערך כפרמטר לפונקציה

```
void change (int a[],int);

int main()
{
	int i, arr[10]={1,2,3,4,5,6,7,8,9,10};
	for (i=0;i<10;i++)
		printf("%d ",arr[i]);
	printf("\n");
	change(arr,10);
	for (i=0;i<10;i++)
		printf("%d ",arr[i]);
	return 0;
}
```

```
void change (int a[], int size)
{
	int i;
	for (i=0; i<size; i++)
		a[i]=i*i;
}
```

שתי דרכים לכותרת הפונקציה change משקף הקודם:

void change (**int a[], int size**)

void change (**int *a, int size**)

בשתי הדרכים, a הוא מבציע. בשתי הדרכים, הפנייה לאיברי מערך מתבצעת באותה הצורה.

חיפוש לינארי - דוגמא

יש לכתוב פונקציה המקבלת מערך, את גודלו וערך כלשהו, בודקת אם הערך נמצא במערך, ומחזירה 1 או 0 בהתאם.

```
int search(int ar[], int n, int num)
{
    int i;
    for (i=0; i<n; i++)
        if (ar[i]==num)
            return 1;
    return 0;
}
```

מחרוזת (string) - מהי?

מחרוזת היא מבנה נתונים המורכב מ**סדרה של תווים**.

שימוש במחרוזות נועד לעיבוד מידע טקסטואלי: חיפוש מילה בטקסט, ספירת מילים, החלפת מילה במילה...

הטיפול במחרוזת דומה במידה רבה לטיפול במערכים של תווים, אך ישנן פעולות שהן מיוחדות למחרוזות.

מחרוזת - רצף של תווים המסתיים בתו מיוחד המסומן כ- '**0**\' והוא בעל ערך ASCII 0.

לדוגמה: מילה בת 9 אותיות תאוחסן ב- 10 תווים: 9 אותיות ולאחריהן התו '0\'.

כל פקודות קלט עבור מחרוזות מוסיפות את התו הזה בסיום בעצמן.
יש לזכור להשאיר מקום אחד ריק לצורך זה.

ייצוג ואתחול של מחרוזת ב- C

אפשרויות שקולות לאתחול משתנה מסוג מחרוזת :

```
char str[] = "Hello!";
char str[10] = "Hello!";
char str[] = {'H', 'e', 'l', 'l', 'o', '!', ' \0' };
```

תמונת הזיכרון עבור המחרוזת:

	'H'	'e'	'l'	'l'	'o'	'!'	'\0'

אורך המחרוזת הוא 6: מספר התווים עד (לא כולל) סימן סוף מחרוזת.

באתחול למחרוזת כולה (שורות 2 ו-3 בדוגמא) אין צורך להוסיף '\0' כי הוא נוסף אוטומטית.

אתחול מחרוזת אפשרי רק בשורת ההצהרה, ולא בזמן ריצת התוכנית.

תו, מחרוזת ומערך תווים - הבחנות

- נבחין בין תו 'a' לבין מחרוזת שמכילה תו בודד "a".
- מחרוזת של תו בודד תופסת שני מקומות: התו, וסימן סוף המחרוזת.
- מערך של תווים ללא התו '\0' איננו מחרוזת.

פעולת קלט על מחרוזת

ניתן לקרוא מילת קלט באמצעות scanf(), תוך שימוש בפורמט **%s**.

ההוראה מדלגת על רווחים בתחילת המחרוזת, וקוראת את התווים עד לרווח (או enter) הבא.

למשתנה לתוכו נכתבת המחרוזת מתווסף "אוטומטית" תו סוף המחרוזת '\0' .

שימו לב: **אין להשתמש ב- & לפני שם המחרוזת!**

```
char name[30];
printf ("Enter your name");
scanf ("%s", name);
printf("your name is: %s\n", name);
```

פונקציות ב- stdio.h לקליטה והדפסה של מחרוזות fgets(), puts()

char* fgets(char *string, int length, FILE * stream);

int puts(const char *s);

כאשר רוצים לקלוט שורה שלמה שכוללת רווחים, לא ניתן להשתמש ב-scanf, לכן משתמשים הפקודה נוספת- fgets.

הפקודה מסיימת את הקלט כאשר קוראת length-1 תוים או מגיעה ל-enter או מגיעה לסוף הקובץ (על האפשרות האחרונה נדון בהמשך).כמו שאר הפקודות שמתעסקות במחרוזות, היא מוסיפה '\0' בסוף המחרוזת. אם הפקודה עוצרת ב-enter, היא מכניסה אותו כתו לתוך המחרוזת לפני '\0'.

```
char str[5];
printf("Enter the string\n");
fgets(str, 5, stdin);
printf("\n The string is ");
puts(str);
```

stdin – מצביע לקובץ קלט הסטדנרטי - הכוונה למקלדת

דוגמאות לפקודות עבור מחרוזות מתוך <string.h>

unsigned int strlen(char *st)	מקבלת מחרוזת ומחזירה את אורכה בפועל,ללא '\0'
char *strcpy(char *dest, char *src)	מעתיקה את תוכן המחרוזת src למחרוזת dest,כולל '\0'. גם מחזירה את המחרוזת dest.
char *strcat(char *dest, char *src)	משרשרת את המחרוזת src לסוף המחרוזת dest. גם מחזירה את המחרוזת dest.
int strcmp(char *st1, char *st2)	השוואה מילונית (לקסיקוגרפית): מחזירה 0 אם שתי המחרוזות זהות, מספר חיובי אם הראשונה "גדולה" מהשנייה, ומספר שלילי אם הראשונה "קטנה" מהשנייה.
char* strstr(char *st1, char *st2)	מחזירה מצביע למופע הראשון של st2 בתוך st1,אם קיים, אחרת מחזירה NULL.
char* strchr(char *st1, int ch)	מחזירה מצביע למופע הראשון של ch בתוך st1,אם קיים, אחרת מחזירה NULL. (ch is unsigned char)

דוגמאות לשימוש בפקודות

```
char str[15] = "Boker Tov!";
int length = strlen( str );
printf("%d\n", length);
```

- כדי להעתיק מחרוזת,יש להעתיק תו אחר תו. הפונקציה strcpy עושה זאת.

<u>לדוגמה:</u>

```
char source[] = "Shalom";
char dest[15];
strcpy( dest, source );
printf("%s\n%s\n", source, dest);
```

דוגמאות לשימוש בפקודות

```
char source[] = "olam";
char dest[30] = "Shalom ";
strcat( dest, source );
printf("%s\n%s\n", source, dest);
```

קוד זה ידפיס:

```
olam
Shalom olam
```

חישוב אורך מחרוזת
דוגמא למימוש strlen():

```
int  my_strlen(char *st)
{
     int count=0;

     while ( *st != '\0')
     {
        count++;
        st++;
     }
    return count;
}
```

מציאת המקום הראשון במחרוזת שמופיע בו תו שהוא ספרה (ואם אין כל ספרה במחרוזת יוחזר 1-)

פתרון בגישת מערך ואינדקסים

```
int find_first_digit(char st[])
{
    int i;
    for (i=0; st[i] != '\0'; i++)
    {
        if (isdigit (st[i]))            /* isdigit() in ctype.h library*/
                return i;
     }
    return -1;                          /* no digit was found */
}
```

חיפוש תו במחרוזת strchr()
(מחזירה מצביע לתו, אם נמצא, אחרת מחזירה NULL)

```
char *my_strchr (char *st,  char ch)
{
   char *p = st;
   while (*p)
   {
        if (*p == ch)
            return p;
        p++;
    }
    return NULL;
 }
```

מימוש פקודת strcat

```
char *my_strcat(char *st1, char *st2)
        //הנחה שיש מספיק מקום ב- st1 עבור המחרוזת המשורשרת
{
    char *q;
    char *p;
    p = strchr(st1, '\0');
    for (q=st2;  *q !='\0';   q++, p++)
       *p=*q;
    *p='\0';
    return st1;
}
```

דוגמא - הפונקציה מונה כמה פעמים מופיעה תת-המחרוזת pattern בתוך המחרוזת st.

```
int countstr (char *st, char *pattern)
{
    int count=0;
    char *p = st;
    p = strstr(p, pattern);
    while (p!=NULL)
    {
      count++;
      p++;
      p = strstr(p, pattern);
    }
    return count;
}
```

דוגמא נוספת למערך - היפוך סדר איברים במערך

יש לכתוב תכנית הקולטת מן המשתמש ערכים לתוך מערך באורך N והופכת את סדר איבריו. התוכנית תדפיס את המערך לפני ואחרי ההיפוך.

כתבו פונקציות עזר:

- פונקציה הקולטת את ערכי המערך.
- פונקציה המקבלת מערך והופכת את סדר איבריו.
- פונקציה המדפיסה מערך.

```
#define N 20
void  get_ar (int[],int);
void  print_ar(int[], int);
void  reverse_ar(int[], int);

int main()
{
    int ar[N];
    get_ar(ar, N);
    print_ar(ar, N);
    reverse_ar(ar, N);
    print_ar(ar, N);
    return 0;
}
void get_ar (int ar[], int n)
{
    int i;
    printf (“Enter %d numbers”, n);
    for (i=0; i<n; i++)
        scanf(%d”, &ar[i])
 }
```

```
void print_ar (int ar[], int n)
{
    int i;
    for (i=0; i<n; i++)
        printf("%d  ", ar[i]);
    printf("\n");
}

void reverse_ar(int ar[], int n)
{
     int i, temp;
     for (i=0; i<n/2; i++)
     {
         temp=ar[i];
         ar[i] = ar[N-i-1];
         ar[N-i-1] = temp;
     }
}
```