

# מערכים



קרון כליף

# ביחידה זו נלמד:

---

- מהו מערך
- כיצד מערך נראה בזיכרון
- גישה לאיברי המערך
- אתחול מערך
- הפונקציה sizeof
- חריגה מגבולות המערך
- השמת מערכים
- מערך דו-מימדי
- מערך רב-מימדי
- typedef למערך

# מוטיבציה

אם נרצה לכתוב תוכנית הקוראת 20 מספרים ומציגה את הממוצע שלהם, נצטרך להגדיר 20 משתנים:

```
void main()
{
    int num1, num2...num20, sum=0;
    printf("Please insert 20 numbers: ");
    scanf("%d %d...%d", &num1, &num2...&num20);
    sum = num1 + num2 + ... + num20;
    printf("The average is %f\n", sum/20.0);
}
```

התוכנית מסורבלת ומייגע לכתוב אותה, בייחוד כי יתכן גם שנרצה ממוצע של 100 מספרים, או אפילו יותר...

# הגדרה

□ מערך הוא אוסף של משתנים מאותו הסוג עם שם אחד, ובעלי תפקיד זהה

■ דוגמא: מערך של 20 מספרים

```
int numbers[20];
```

■ דוגמא: מערך של 10 תווים

```
char letters[10];
```

■ ובאופן כללי: `<var_name>[SIZE]> <type>;`

□ איברי המערך נשמרים ברצף בזיכרון

□ גודל המערך צריך להיות ידוע בזמן קומפילציה, כדי שהמחשב ידע כמה תאים להקצות למערך, ולכן גודלו יהיה קבוע או מספר

□ גודל המערך בזיכרון: `SIZE* <גודל הטיפוס>`

# דוגמא למערך בזיכרון

```
void main()
{
    int    x = 4;
    int    arr[3];
    char   ch = 'a';
}
```

int: x	<b>4</b>	<b>1000</b>
int[: arr	???	<b>1004</b>
	???	<b>1008</b>
	???	<b>1012</b>
char: ch	'a'	<b>1016</b>
		<b>1017</b>

ערכם של איברי המערך הוא זבל, כמו כל משתנה שלא  
אותחל

גודל המערך בזיכרון:  
 $\text{SIZE} * \text{sizeof}(\text{type})$   
ובדוגמא זו:  $\text{sizeof}(\text{int}) = 3 * 4 = 12$

# גישה לאיברי המערך

- כדי שניתן יהיה להתייחס לכל איבר בנפרד ניתנו להם אינדקסים
- האיבר הראשון בעל אינדקס 0, השני בעל אינדקס 1 והאחרון עם אינדקס SIZE-1
- דוגמא:

```
int arr[3];  
arr[0] = 1;
```

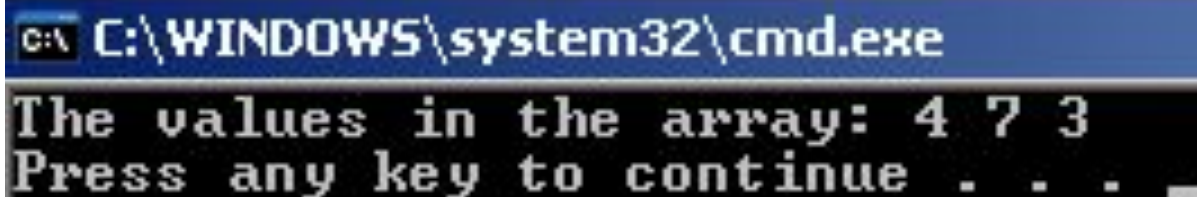
- הפניה לאיבר מסוים במערך היא ע"י [ ] כאשר בתוך הסוגריים יהיה האינדקס של האיבר אליו נרצה לגשת
- פניה לאיבר במערך היא כפניה למשתנה מטיפוס המערך
- מתייחסים ל- arr[0] כמו שמתייחסים ל- int

# גישה לאיברי המערך - דוגמא

```
void main()
```

```
{  
    int arr[3];  
    arr[0] = 4;  
    arr[1] = 7;  
    arr[2] = 3;  
    printf("The values in the array: %d %d %d\n",  
        arr[0], arr[1], arr[2]);  
}
```

int[:arr	4	1000
	7	1004
	3	1008



C:\WINDOWS\system32\cmd.exe  
The values in the array: 4 7 3  
Press any key to continue . . .

arr + \*sizeof(int)0 □ 1000 פירוש תפנה לתא בזיכרון שנמצא בכתובת: 1000  
arr + \*sizeof(int)1 □ 1004 פירוש תפנה לתא בזיכרון שנמצא בכתובת: 1004  
arr + \*sizeof(int)2 □ 1008 פירוש תפנה לתא בזיכרון שנמצא בכתובת: 1008

# גישה לאיברי המערך - לולאות

□ מאחר ועבודה עם איברי המערך היא עבודה זזה על כל האיברים, יותר חסכוני וקל להשתמש בלולאות

```
#define SIZE 5  
void main()  
{
```

```
    int arr[SIZE], i;
```

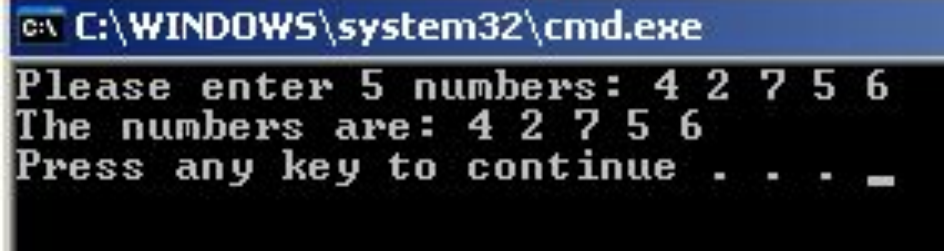
```
    printf("Please enter %d numbers: ", SIZE);
```

```
    for (i=0 ; i < SIZE ; i++)  
        scanf("%d", &arr[i]);
```

```
    printf("The numbers are: ");
```

```
    for (i=0 ; i < SIZE ; i++)  
        printf("%d ", arr[i]);  
    printf("\n");
```

```
}
```



```
C:\WINDOWS\system32\cmd.exe  
Please enter 5 numbers: 4 2 7 5 6  
The numbers are: 4 2 7 5 6  
Press any key to continue . . .
```

נשים לב שבעבודה עם מערכים ולולאות i יתחיל מ-0

האינדקס שאיתו פונים לאיבר במערך יכול להיות:

- מספר שלם (כמו בדוגמא הקודמת)
- משתנה (כמו בדוגמא זו)
- ערך של ביטוי למשל: `arr[i+2]`

ניתוח זמן הריצה של תוכנית זו

$$O(\text{main}) = O(1) + O(\text{SIZE}) + O(\text{SIZE}) = O(1) + 2 * O(\text{SIZE}) = O(\text{SIZE})$$



## גישה לאיברי המערך – לולאות (2)

הפעם נרצה להדפיס את איברי המערך מהסוף להתחלה..

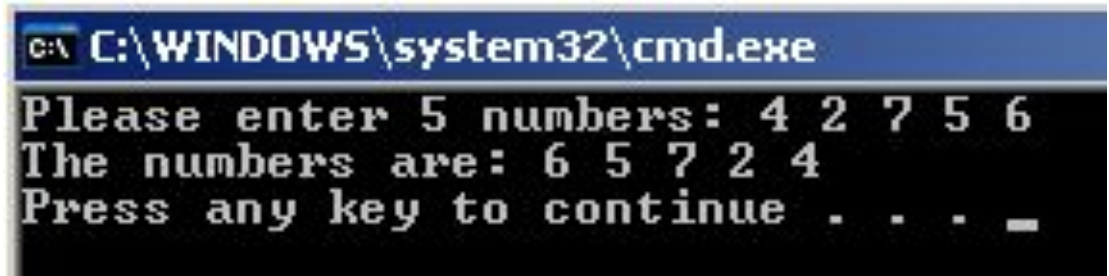
```
#define SIZE 5  
void main()  
{
```

```
    int arr[SIZE], i;
```

```
    printf("Please enter %d numbers: ", SIZE);  
    for (i=0 ; i < SIZE ; i++)  
        scanf("%d", &arr[i]);
```

```
    printf("The numbers are: ");  
    for (i=SIZE-1 ; i >= 0 ; i--)  
        printf("%d ", arr[i]);  
    printf("\n");
```

```
}
```



```
C:\WINDOWS\system32\cmd.exe  
Please enter 5 numbers: 4 2 7 5 6  
The numbers are: 6 5 7 2 4  
Press any key to continue . . . _
```

# גישה לאיברי המערך – לולאות - דוגמא

תוכנית המוצאת את הערך המקסימלי שהוכנס:

```
#define SIZE 4
void main()
{
    int arr[SIZE], i, max;

    printf("Enter %d numbers: ", SIZE);
    for (i=0 ; i < SIZE ; i++)
        scanf("%d", &arr[i]);

    max = arr[0];
    for ( i=1 ; i < SIZE ; i++ )
    {
        if (arr[i] > max)
            max = arr[i];
    }
    printf("The max is %d\n", max);
}
```

int[:arr	13	1000
	11	1004
	17	1008
	15	1012
int: i	4	1016
int: max	17	1020

```
Enter 4 numbers: 13 11 17 15
The max is 17
Press any key to continue . .
```

ניתוח זמן הריצה של תוכנית זו:

$$O(\text{main}) = O(1) + O(\text{SIZE}) + O(\text{SIZE}) = O(1) + 2*O(\text{SIZE}) = O(\text{SIZE})$$

# מציאת האינדקס המכיל את הערך המקסימלי

```
#define SIZE 4
void main()
{
    int arr[SIZE], maxIndex, i;

    printf("Enter %d numbers: ", SIZE);
    for (i=0 ; i < SIZE ; i++)
        scanf("%d", &arr[i]);

    maxIndex = 0;
    for ( i=1 ; i < SIZE ; i++ )
    {
        if (arr[i] > arr[maxIndex])
            maxIndex = i;
    }
    printf("The max is at index %d and its value is  %d\n",
        maxIndex, arr[maxIndex]);
}
```

int[:arr	13	1000
	11	1004
	17	1008
	15	1012
int: i	4	1016
int: maxIndex	2	1020

```
Enter 4 numbers: 13 11 17 15
The max is at index 2 and its value is 17
Press any key to continue . . . _
```

# אתחול מערך

- ❑ כאשר מגדירים מערך ערכי איבריו הוא זבל
- ❑ ניתן לאתחל את איברי המערך באחת מהדרכים הבאות:

```
int arr1[3] = {5, 3, 1}; //arr1[0]=5, arr1[1]=3, arr1[2]=1
```

```
int arr2[] = {5, 3, 1}; //arr2[0]=5, arr2[1]=3, arr2[2]=1  
                        and the size of the array is 3!
```

```
int arr3[3] = {5}; //arr3[0]=5, arr3[1]=0, arr3[2]=0
```

```
int arr4[3] = {0}; //arr4[0]=0, arr4[1]=0, arr4[2]=0
```

- ❑ נשים לב כי רק בעת האיתחול ניתן לתת ערך לכמה איברים יחד! כל נתינת ערך בהמשך הינה השמה, ולא איתחול, ולכן יבוצע על כל איבר בנפרד.

# אתחול מערך: הגדרת הגודל והערכים

עבור המערכים הבאים:

```
int numbers[3] = {5, 3, 1};  
char letters[3] = {'m', 'A', 'k'};
```

הזכרון יראה כך:

int[]: numbers	5	1000
	3	1004
	1	1008
char[]: letters	'm'	1012
	'A'	1013
	'k'	1014

# אתחול מערך: הגדרת הערכים בלבד

עבור המערך הבא:

```
double numbers[] = {5, 3.2, 1.1};
```

הזכרון יראה כך:

double[]: numbers	5.0	1000
	3.2	1008
	1.1	1016

נשים לב שאין צורך בהגדרת גודל המערך, הקומפיילר יודע זאת לבד לפי מספר הערכים שאותחלו

# אתחול מערך: הגדרת גודל וחלק מהערכים

□ כאשר נגדיר מערך באופן הבא:

```
int numbers[3] = {5};
```

□ הזכרון יראה כך:

int[: numbers	5	1000
	0	1004
	0	1008

□ כאשר מאתחלים את איברי המערך באופן חלקי, שאר האיברים מקבלים ערך 0 (בניגוד לזבל שהיה אם לא היינו מאתחלים כלל)

# אתחול מערך: איפוס כל איברי המערך

□ כאשר נגדיר מערך באופן הבא:

```
int numbers[3] = {0};
```

□ הזכרון יראה כך:

int[:numbers	0	1000
	0	1004
	0	1008

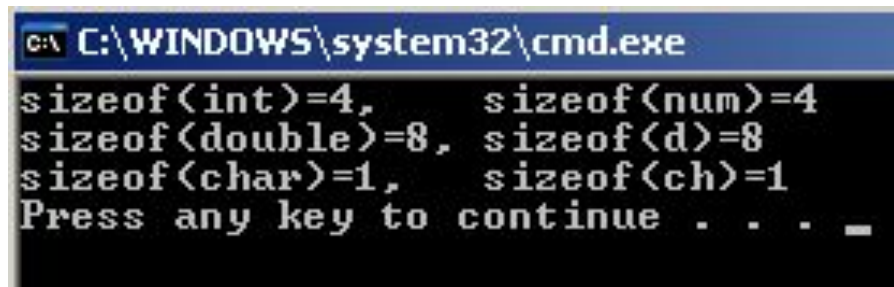
□ זהו מקרה פרטי של צורת האתחול הקודמת



# הפונקציה sizeof

sizeof היא פונקציה המקבלת משתנה או טיפוס ומחזירה את מספר הבתים שהוא תופס בזיכרון

```
void main()
{
    int    num;
    double d;
    char   ch;
```



A screenshot of a Windows command prompt window. The title bar reads 'C:\WINDOWS\system32\cmd.exe'. The command prompt shows the output of the sizeof operator for various types and variables: sizeof(int)=4, sizeof(num)=4, sizeof(double)=8, sizeof(d)=8, sizeof(char)=1, and sizeof(ch)=1. It ends with the prompt 'Press any key to continue . . . \_'.

```
printf("sizeof(int)=%d,\t sizeof(num)=%d\n",
        sizeof(int), sizeof(num));
printf("sizeof(double)=%d,\t sizeof(d)=%d\n",
        sizeof(double), sizeof(d));
printf("sizeof(char)=%d,\t sizeof(ch)=%d\n",
        sizeof(char), sizeof(ch));
}
```

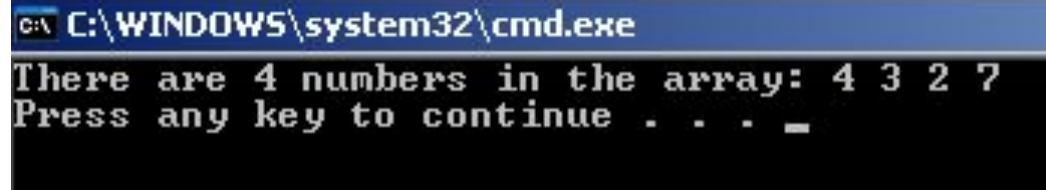
# חישוב גודל המערך

- גודל המערך בזיכרון הוא  $\text{SIZE} * \text{sizeof}(\text{type})$
- יהיו מקרים בהם נרצה לדעת בזמן ריצה כמה איברים יש במערך, ולא תמיד הגודל מוגדר לנו, למשל:

```
void main()
```

```
{
```

```
    int i, arr[] = {4, 3, 2, 7};
```



C:\WINDOWS\system32\cmd.exe

There are 4 numbers in the array: 4 3 2 7  
Press any key to continue . . . \_

```
    int size = sizeof(arr) / sizeof(int);
```

```
    printf("There are %d numbers in the array: ", size);
```

```
    for (i=0 ; i < size ; i++)
```

```
        printf("%d ", arr[i]);
```

```
    printf("\n");
```

```
}
```

## חישוב גודל המערך (2)

ניתן לחשב את כמות האיברים במערך גם באופן הבא:

```
void main()
{
    int arr[] = {4, 3, 2, 7};
    sizeof(arr[0])
    int size = sizeof(arr) / sizeof(int);
    ...
}
```

דרך זו עדיפה, שכן אם נשנה את טיפוס איברי המערך לא נצטרך לתקן את השורה המחשבת את ה-size

# דוגמא: הדפסת היסטוגרמה של ערכי המערך

```
void main()
{
    int arr[] = {4, 3, 2, 7}, i, j;
    int size = sizeof(arr) / sizeof(arr[0]);

    printf("There are %d numbers in the array: ", size);
    for ( i=0 ; i < size ; i++ ) {
        printf("%d: ", arr[i]);
        for ( j=0 ; j < arr[i] ; j++ )
            printf("*");
        printf("\n");
    }
    printf("\n");
}
```

4: \*\*\*\*

3: \*\*\*

2: \*\*

7: \*\*\*\*\*

int[:arr	4	1000
	3	1004
	2	1008
	7	1012
int: size	4	1016
int: i	4	1020
int: j		1024

```
There are 4 numbers in the array:
4: ****
3: ***
2: **
7: *****
```

# הגדרה: מערך סימטרי (פלינדרום)

□ מערך שאם נקרא את ערכיו משמאל לימין או ההפך  
נקבל אותו הדבר



# והנה דוגמאות נוספות



# דוגמא: האם איברי המערך סימטריים

```
#define SIZE 5
void main()
{
    int arr[SIZE], i, left, right, isSymetric=1;

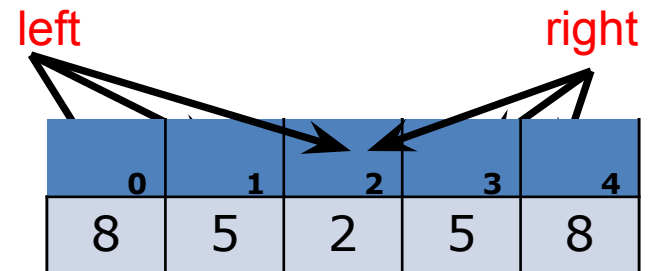
    printf("Enter %d numbers: ", SIZE);
    for (i=0 ; i < SIZE ; i++)
        scanf("%d", &arr[i]);

    for (left=0, right=SIZE-1; left < right && isSymetric ; left++, right--)
    {
        if (arr[left] != arr[right])
            isSymetric = 0;
    }

    if (isSymetric)
        printf("The array is symetric\n");
    else
        printf("The array is NOT symetric\n");
}
```

```
C:\WINDOWS\system32\cmd.exe
Enter 5 numbers: 8 5 2 5 8
The array is symetric
Press any key to continue . . .

C:\WINDOWS\system32\cmd.exe
Enter 5 numbers: 8 5 4 7 8
The array is NOT symetric
Press any key to continue . . .
```



isSymetric = 0



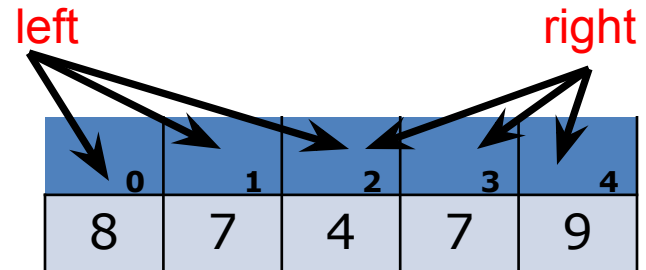
# שגיאה נפוצה

```
#define SIZE 5
void main()
{
    int arr[SIZE], i, left, right, isSymetric=1;

    printf("Enter %d numbers: ", SIZE);
    for (i=0 ; i < SIZE ; i++)
        scanf("%d", &arr[i]);

    for (left=0, right=SIZE-1; left < right && isSymetric ; left++, right--)
    {
        if (arr[left] != arr[right])
            isSymetric = 0;
        else
            isSymetric = 1;
    }

    if (isSymetric)
        printf("The array is symetric\n");
    else
        printf("The array is NOT symetric\n");
}
```



**isSymetric = 0**



# דוגמא: היום המועדף בשבוע - פלט

```
C:\WINDOWS\system32\cmd.exe
Insert the day you like most <1-7>, -1 to EXIT: 2
Insert the day you like most <1-7>, -1 to EXIT: 6
Insert the day you like most <1-7>, -1 to EXIT: 7
Insert the day you like most <1-7>, -1 to EXIT: 4
Insert the day you like most <1-7>, -1 to EXIT: 7
Insert the day you like most <1-7>, -1 to EXIT: 2
Insert the day you like most <1-7>, -1 to EXIT: 5
Insert the day you like most <1-7>, -1 to EXIT: 3
Insert the day you like most <1-7>, -1 to EXIT: 7
Insert the day you like most <1-7>, -1 to EXIT: -1
Each day and number of persons who liked it most:
1: 0
2: 2
3: 1
4: 1
5: 1
6: 1
7: 3
The favorite day is: 7
Press any key to continue . . . _
```

□ בהמשך נראה מה היה קורה אם המשתמש היה מכניס  
ערך שאינו בין 1 ל-7!

# דוגמא: היום המועדף בשבוע

```
#define EXIT -1
void main()
{
    int i, daysFrequency[7] = {0}, day, maxDayIndex;

    do
    {
        printf("Insert the day you like most (1-7), %d to EXIT: ", EXIT);
        scanf("%d",&day);
        if (day != EXIT)
            daysFrequency[day-1]++;
    } while (day != EXIT)

    printf("Each day and number of persons who liked it most:\n");
    for(i=1 ; i<=7 ; i++)
        printf("%d: %d\n", i, daysFrequency[i-1]);

    maxDayIndex = 0;
    for (i=1 ; i < 7 ; i++)
    {
        if (daysFrequency[i] > daysFrequency[maxDayIndex])
            maxDayIndex = i;
    }
    printf("The favorite day is: %d\n", maxDayIndex+1);
}
```

int: i	??	100 0
int[: daysFreq	0	100 4
	0	100 8
	0	101 2
	2	101 6
	0	102 0
	0	102 4
	1	102 8
int: day	-1	103 2
int: maxDayIndex	0	103 6

# חריגה מגבולות המערך

❑ כדי לגשת לאחד מאיברי מערך בגודל N ניגש עם אינדקסים  $N-1 \dots 0$

❑ כאשר ננסה לפנות לאינדקס שאינו בגבולות המערך אנו למעשה מנסים לגשת בתא בזיכרון שאיננו יודעים מה יש בו ומה השפעתו, וזוהי גישה לא חוקית לשטח בזיכרון

```
int arr[3];  
arr[5] = 7;
```

int[:arr	???	1000
	???	1004
	???	1008
		1012
		1016
	7	1020

arr[5] פירושו תפנה לתא בזיכרון שנמצא בכתובת: arr +

\*sizeof(int)5

כלומר לכתובת 1020\*1000+5 sizeof(int)

## חריגה מגבולות המערך (2)

- ❑ אחריות המתכנת לפנות לתא שבגבולות המערך
- ❑ הקומפיילר אינו מתריע על ניסיון פניה לתא שאינו בגבולות המערך (יתכן שבזמן קומפילציה אינו ידוע מהו התא אליו מנסים לגשת)

```
int arr[4];  
int i;  
scanf("%d", &i);  
arr[i] = 7;
```

i יכול להיות בין 0-3 ואז הכל בסדר, או לחילופין מספר שלילי או גדול מ-3 ואז אנו חורגים מגבולות המערך...

- ❑ יתכנו מקרים בהם פנינו לתא חשוב, והתוכנית תעוף
- ❑ יש קומפיילרים שתמיד יעיפו אותנו כאשר נפנה לתא שאינו בגבולות המערך, ויש כאלו שלא, ואז הבעיות עלולות להגיע אח"כ...

# השמת מערכים

□ כדי לבצע השמה בין משתנים מאותו הסוג אנו משתמשים באופרטור =

```
;int x, y=5  
;x = y
```

□ עבור מערכים לא ניתן לבצע זאת:

```
int arr1[]={1,2,3}, arr2[3];  
arr2 = arr1;
```

□ השמה בין מערכים תבוצע בעזרת לולאה, בה נעתיק איבר-איבר

# השמת מערכים - דוגמא

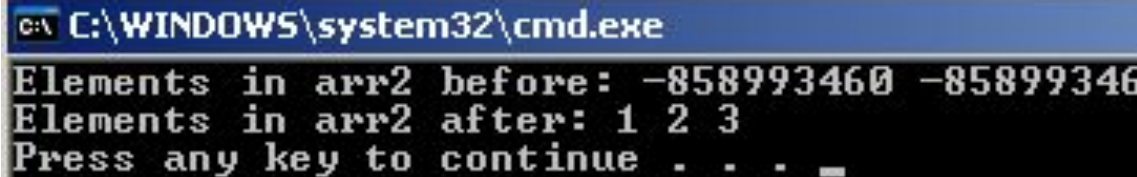
```
void main()
{
    int arr1[] = {1,2,3}, arr2[3], i;

    // arr2 = arr1; // DOESN'T COMPILE!!

    printf("Elements in arr2 before: ");
    for (i=0 ; i < 3 ; i++)
        printf("%d ", arr2[i]);

    for (i=0 ; i < 3 ; i++)
        arr2[i] = arr1[i];

    printf("\nElements in arr2 after: ");
    for (i=0 ; i < 3 ; i++)
        printf("%d ", arr2[i]);
    printf("\n");
}
```



```
C:\WINDOWS\system32\cmd.exe
Elements in arr2 before: -858993460 -858993460
Elements in arr2 after: 1 2 3
Press any key to continue . . . _
```

# דוגמא: קבל מספר והחזר מהי הספרה המופיעה הכי הרבה פעמים

□ למשל, עבור המספר 12,327,293 תוצג הספרה 2 כי היא מופיעה הכי הרבה פעמים במספר (3 פעמים)

□ אופציה 1: לספור באמצעות לולאה כמה פעמים בכל המספר מופיעה הספרה 1, כנ"ל עבור הספרה 2 וכו'

■  $O(n) \cdot 10$ , כאשר  $n$  הוא מספר הספרות במספר

■  $O(n) = O(n) \cdot 10$

□ אופציה 2: מיון דליים

■  $O(n)$ , כאשר  $n$  הוא מספר הספרות במספר

```
void main()
```

```
{
```

```
    int num, i; O(n)  
    int maxDigit=0, maxCount=0;
```

```
    printf("Enter number :");
```

```
    scanf("%d", &num);
```

```
    for (i = 0; i < 10; i++)
```

```
    {
```

```
        int temp = num;
```

```
        int count = 0;
```

```
        while (temp > 0)
```

```
        {
```

```
            if (temp % 10 == i)
```

```
                count++;
```

```
            temp /= 10;
```

```
        }
```

```
        if (count > maxCount)
```

```
        {
```

```
            maxCount = count;
```

```
            maxDigit = i;
```

```
        }
```

```
    }
```

```
    printf("The digit that appears most  
is %d\n", maxDigit);
```

```
void main()
```

```
{
```

```
    int num, i;
```

```
    int counters[10] = {0};
```

```
    int max;
```

```
    printf("Enter number :");
```

```
    scanf("%d", &num);
```

```
    while (num > 0) O(n)
```

```
    {
```

```
        counters[num % 10]++;
```

```
        num /= 10;
```

```
    }
```

```
    max = 0;
```

```
    for (i = 1; i < 10; i++)
```

```
        if (counters[i] > counters[max])
```

```
            max = i;
```

```
    printf("The digit that appears  
most is %d\n", max);
```

```
}
```

הקוד



# דוגמא

□ הגדר מערך של 10 תווים, קלוט לתוכו נתונים מהמקלדת, והדפס למסך את התווים שהוקלדו ללא חזרות, וכן את כמות התווים השונים

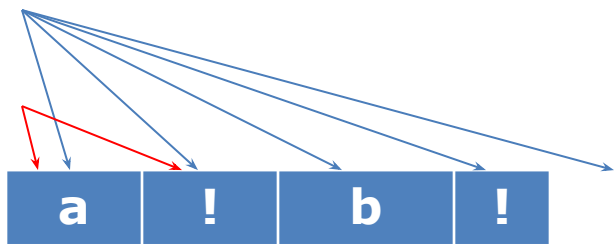
□ **דוגמא:** אם המשתמש הכניס את התווים הבאים:

a 0 ? T T 0 \$ a T x

□ אזי התכנית תודיע שיש 6 תווים שונים ותדפיס למסך:

a 0 ? T \$ x

```
Please enter 10 chars: a0?TT0$aTx
The chars are:
a0?T$x
There were 6 different letters
Press any key to continue . . .
```



a ! b

counter = 3

# אסטרטגיית הפתרון

fHasAppear = true

הרעיון: עבור כל תו נבדוק אם הוא כבר הופיע בתווים שלפניו.  
 במידה ולא נגדיל את ה- counter ונדפיס את התו  
המשתנים בהם נשתמש:

- **counter**: תפקידו לספור את מספר התווים השונים
- **fHasAppear**: דגל שתפקידו לדעת האם תו כבר הופיע בתווים שלפניו

## האלגוריתם:

- אפס counter
- עבור כל תו במערך:
  - אפס דגל
  - עבור כל אחד מהתווים שלפניו והתו לא הופיע בתווים שלפניו (הדגל לא סומן):
    - אם התו הנבדק והתו הנוכחי זהים:
      - סמן את הדגל (כדי להפסיק את לולאת בדיקת התווים שלפניו)

# קוד הפתרון

```
#define SIZE 10
```

```
void main()
```

```
{
```

```
    int count=0;
```

```
    int fHasAppear, i, j;
```

```
    char str[SIZE];
```

```
    printf("Please enter %d chars: ", SIZE);
```

```
    for (i=0 ; i < SIZE ; i++)
```

```
        scanf("%c", &str[i]);
```

קליטת הנתונים

```
// print each char only once
```

```
printf("The chars are:\n");
```

```
for (i=0 ; i < SIZE ; i++)
```

עבור כל תו

```
{
```

```
    fHasAppear = 0;
```

איפוס הדגל עבור התו הנוכחי

```
    for (j=0 ; j < i && !fHasAppear; j++)
```

מעבר על כל התווים שלפניו, וגם כל

```
{
```

עוד התו לא נמצא

```
    if (str[i] == str[j])
```

אם התו זהה לאחד התווים

```
        fHasAppear = 1;
```

שלפניו, נדליק את הדגל

```
}
```

```
    if (!fHasAppear)
```

```
{
```

אם התו לא הופיע, נדפיס

```
        printf("%c", str[i]);
```

אותו ונגדיל את ה-counter

```
        count++;
```

```
    }
```

```
{
```

```
printf("\nThere were %d different letters\n", count);
```

```
}
```

## מערך דו-מימדי - מוטיבציה

□ כדי לשמור ציונים של 30 סטודנטים בכיתה נגדיר מערך בגודל 30:

```
int grades[30];
```

□ אם יש לנו 3 כיתות שעבורן נרצה לשמור ציונים של 30 סטודנטים בכיתה נצטרך להגדיר 3 מערכים:

```
int grades1[30], grades2[30], grades3[30];
```

□ אבל 3 הכיתות האלו הן גם אוסף של משתנים מאותו הסוג – מערך של מספרים בגודל 30

□ לכן נרצה להגדיר מערך שיש בו 3 איברים, וכל איבר הוא מערך בגודל 30:

```
int grades[3][30];
```

## מערך דו-מימדי

□ בהגדרת מערך חד-מימדי מגדירים את כמות התאים בו (מספר העמודות):

int arr[4];

arr[0]	arr[1]	arr[2]	arr[3]
--------	--------	--------	--------

□ בהגדרת מערך דו-מימדי נגדיר את כמות התאים בו ע"י ציון מספר השורות ומספר העמודות:

int arr[2][4];

arr [0][0]	arr [0][1]	arr [0][2]	arr [0][3]
arr [1][0]	arr [1][1]	arr [1][2]	arr [1][3]

□ מערך דו-מימדי הוא למעשה מטריצה, או ניתן להסתכל עליו כמערך של מערכים

# הגדרת מערך דו-מימדי

□ כדי להגדיר מערך חד-מימדי הגדרנו למשל:

```
double numbers[4];
```

■ ובאופן כללי:

```
;type <name>[SIZE]
```

□ כדי להגדיר מערך דו-מימדי נגדיר למשל:

```
double numbers[2][4];
```

■ ובאופן כללי:

```
;type <name>[ROWS][COLS]
```

# מערך דו-מימדי - פניה לאיבר

```
int arr[2][4];
```

arr [0][0]	arr [0][1]	arr [0][2]	arr [0][3]
arr [1][0]	arr [1][1]	arr [1][2]	arr [1][3]

□ כדי לפנות לאיבר במערך דו-מימדי צריך לציין את מספר השורה ואת מספר העמודה של האיבר אשר איתו אנו רוצים לעבוד

■ למשל, כדי לשנות את ערכו של האיבר בשורה השנייה בעמודה השלישית:

```
arr[1][2] = 5;
```

■ למשל, כדי לשנות את ערכו של האיבר בשורה הראשונה בעמודה הראשונה:

```
arr[0][0] = 5;
```

# מערך דו-מימדי – דוגמא: קליטת ציונים לכמה כיתות והדפסתם - פלט

---

C:\WINDOWS\system32\cmd.exe

Please enter grades for students in 3 classes:

Please enter grades for 5 students in class #1: 100 90 98 80 60

Please enter grades for 5 students in class #2: 87 98 60 30 100

Please enter grades for 5 students in class #3: 89 74 69 90 77

The grades in all classes:

Class #1: 100 90 98 80 60

Class #2: 87 98 60 30 100

Class #3: 89 74 69 90 77

Press any key to continue . . . \_



# מערך דו-מימדי – דוגמא: קליטת ציונים לכמה כיתות והדפסתם

```
#define NUM_CLASSES 3
#define STUDENTS_IN_CLASS 5
void main()
{
    int grades[NUM_CLASSES][STUDENTS_IN_CLASS];
    int i, j;

    printf("Please enter grades for students in %d classes:\n", NUM_CLASSES);
    for (i=0 ; i < NUM_CLASSES ; i++)
    {
        printf("Please enter grades for %d students in class #%d:",
            STUDENTS_IN_CLASS, i+1);
        for (j=0 ; j < STUDENTS_IN_CLASS ; j++)
            scanf("%d", &grades[i][j]);
    }

    printf("The grades in all classes:\n");
    for (i=0 ; i < NUM_CLASSES ; i++)
    {
        printf("Class #%d: ", i+1);
        for (j=0 ; j < STUDENTS_IN_CLASS ; j++)
            printf("%d ", grades[i][j]);
        printf("\n");
    }
}
```

# מערך דו-מימדי – דוגמא: קליטת ציונים לכמה כיתות והדפסת הממוצע – פלט

---

C:\WINDOWS\system32\cmd.exe

```
Please enter grades for students in 3 classes:
Please enter grades for 5 students in class #1: 100 90 98 80 60
Please enter grades for 5 students in class #2: 87 98 60 30 100
Please enter grades for 5 students in class #3: 89 74 69 90 77
The average for each classe:
Average for class #1: 85.599998
Average for class #2: 75.000000
Average for class #3: 79.800003
Press any key to continue . . . _
```

# מערך דו-מימדי – דוגמא: קליטת ציונים לכמה כיתות והדפסת הממוצע

```
#define NUM_CLASSES 3
#define STUDENTS_IN_CLASS 5
void main()
{
    int grades[NUM_CLASSES][STUDENTS_IN_CLASS];
    float average[NUM_CLASSES];
    int i, j, sum;

    printf("Please enter grades for students in %d classes:\n", NUM_CLASSES);
    for (i=0 ; i < NUM_CLASSES ; i++)
    {
        printf("Please enter grades for %d students in class #%d: ",
            STUDENTS_IN_CLASS, i+1);
        for (j=0, sum=0 ; j < STUDENTS_IN_CLASS ; j++)
        {
            scanf("%d", &grades[i][j]);
            sum += grades[i][j];
        }
        average[i] = (float)sum/STUDENTS_IN_CLASS;
    }

    printf("The average for each classe:\n");
    for (i=0 ; i < NUM_CLASSES ; i++)
        printf("Average for class #%d: %f\n", i+1, average[i]);
}
```

# מערך דו-מימדי – ייצוגו בזיכרון

כמו מערך חד-מימדי, גם מערך דו-מימדי נשמר בזיכרון ברצף, כאשר איברי השורה הראשונה נשמרים קודם, ומיד אח"כ איברי השורה השניה וכו'

`int arr[2][4];`

arr [0][0]	arr [0][1]	arr [0][2]	arr [0][3]
arr [1][0]	arr [1][1]	arr [1][2]	arr [1][3]

ההתאמה בין המקום במערך  
הדו-מימדי למערך החד-  
מימדי היא  
$$\text{COLUMNS} * i + j$$

int[2][4]: arr[0][0] □	???	1000
arr[0][1] □	???	1004
arr[0][2] □	???	1008
arr[0][3] □	???	1012
arr[1][0] □	???	1016
arr[1][1] □	???	1020
arr[1][2] □	???	1024
arr[1][3] □	???	1028

# מערך דו-מימדי – ייצוגו בזיכרון

□ כמו מערך חד-מימדי, גם מערך דו-מימדי נשמר בזיכרון ברצף, כאשר איברי השורה הראשונה נשמרים קודם, ומיד אח"כ איברי השורה השניה וכו'

int arr[2][4];

arr [0][0]	arr [0][1]	arr [0][2]	arr [0][3]
arr [1][0]	arr [1][1]	arr [1][2]	arr [1][3]

□ ההתאמה בין המקום במערך הדו-מימדי למערך החד-מימדי היא

$$\text{COLUMNS} * i + j$$

■ למשל: arr[0][3] נמצא במקום  $3 = 0 + 3 * 4$

int[2][4]: arr[0][0] □	???	1000
arr[0][1] □	???	1004
arr[0][2] □	???	1008
<b>arr[0][3] □</b>	<b>???</b>	<b>1012</b>
arr[1][0] □	???	1016
arr[1][1] □	???	1020
arr[1][2] □	???	1024
arr[1][3] □	???	1028

# מערך דו-מימדי – ייצוגו בזיכרון

כמו מערך חד-מימדי, גם מערך דו-מימדי נשמר בזיכרון ברצף, כאשר איברי השורה הראשונה נשמרים קודם, ומיד אח"כ איברי השורה השניה וכו'

int arr[2][4];

arr [0][0]	arr [0][1]	arr [0][2]	arr [0][3]
arr [1][0]	arr [1][1]	arr [1][2]	arr [1][3]

ההתאמה בין המקום במערך  
הדו-מימדי למערך החד-  
מימדי היא

$$\text{COLUMNS} * i + j$$

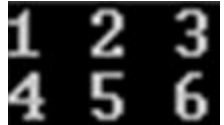
■ למשל: arr[0][3] נמצא  
במקום  $3 = 0 + 3 * 4$

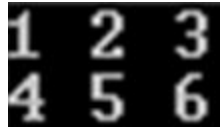
■ למשל: arr[1][2] נמצא  
במקום  $6 = 1 + 2 * 4$

int[2][4]: arr[0][0] □	???	1000
arr[0][1] □	???	1004
arr[0][2] □	???	1008
arr[0][3] □	???	1012
arr[1][0] □	???	1016
arr[1][1] □	???	1020
<b>arr[1][2] □</b>	<b>???</b>	<b>1024</b>
arr[1][3] □	???	1028

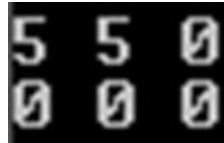
# מערך דו-מימדי - איתחול

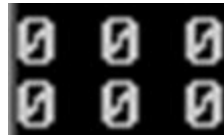
ניתן לאתחל מערך דו-מימדי באחת מהדרכים הבאות:

`int arr1[2][3] = { {1,2,3}, {4,5,6} };` 

`int arr2[][3] = { {1,2,3}, {4,5,6} };` 

ניתן לאתחל בלי ציון מספר השורות, אבל תמיד חייבים לציין את מספר

`int arr3[2][3] = { {5,5} };` 

`int arr4[2][3] = {0};` 

# מערך דו-מימדי – חישוב מספר השורות

```
#include <stdio.h>
#define NUM_OF_COLS 3
void main()
{
```

```
    int arr[][NUM_OF_COLS] = { {1,2,3}, {4,5,6} };
```

```
    int i, j;
```

```
    int numOfRows = sizeof(arr)/sizeof(int)/NUM_OF_COLS;
```

```
    for (i=0 ; i < numOfRows ; i++)
```

```
    {
```

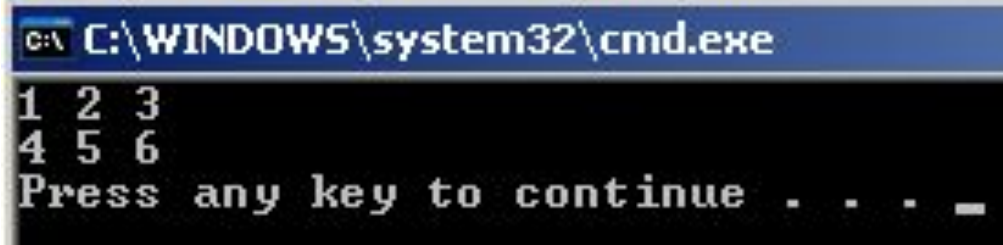
```
        for (j=0 ; j < NUM_OF_COLS ; j++)
```

```
            printf("%d ", arr[i][j]);
```

```
        printf("\n");
```

```
    }
```

```
}
```



```
C:\WINDOWS\system32\cmd.exe
1 2 3
4 5 6
Press any key to continue . . . _
```



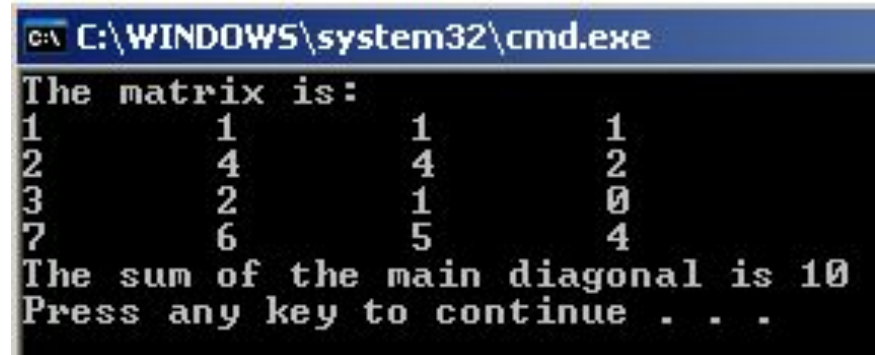
# מערך דו-מימדי – דוגמא: מציאת סכום איברי האלכסון של מטריצה ריבועית

```
#define SIZE 4
void main()
{
    int matrix[SIZE][SIZE] = { {1,1,1,1}, {2,4,4,2}, {3,2,1,0}, {7,6,5,4} };
    int i, j, sum=0;

    //printf("The matrix is:\n");
    //for (i=0 ; i < SIZE ; i++)
    //{
    //    for (j=0 ; j < SIZE ; j++)
    //        printf("%d\t", matrix[i][j]);
    //    printf("\n");
    //}

    // calc the sum of the main diagonal
    for (i=0 ; i < SIZE ; i++)
        for (j=0 ; j < SIZE ; j++)
            if (i == j)
                sum += matrix[i][j];
    printf("The sum of the main diagonal is %d\n", sum);
}
```

ניתוח זמן הריצה של תוכנית זו (ללא לולאת הפלט):  
 $O(\text{main}) = O(1) + \text{SIZE} * O(\text{SIZE}) = O(\text{SIZE}^2)$



```
C:\WINDOWS\system32\cmd.exe
The matrix is:
1      1      1      1
2      4      4      2
3      2      1      0
7      6      5      4
The sum of the main diagonal is 10
Press any key to continue . . .
```

# מערך דו-מימדי – דוגמא: מציאת סכום איברי האלכסון של מטריצה ריבועית (2)

```
#define SIZE 4
```

```
void main()
```

```
{
```

```
int matrix[SIZE][SIZE] = { {1,1,1,1}, {2,4,4,2}, {3,2,1,0}, {7,6,5,4} };
```

```
int i, j, sum=0;
```

```
//printf("The matrix is:\n");
```

```
//for (i=0 ; i < SIZE ; i++)
```

```
//{
```

```
//    for (j=0 ; j < SIZE ; j++)
```

```
//        printf("%d\t", matrix[i][j]);
```

```
//    printf("\n");
```

```
//}
```

```
// calc the sum of the main diagonal
```

```
for (i=0 ; i < SIZE ; i++)
```

```
    sum += matrix[i][i];
```

```
printf("The sum of the main diagonal is %d\n", sum);
```

```
}
```

ניתוח זמן הריצה של תוכנית זו (ללא לולאת הפלט):

$$O(\text{main}) = O(1) + O(\text{SIZE}) = O(\text{SIZE})$$

הדוגמאות פותרות בדרך שונה את אותה הבעיה 2 אבל בסיבוכיות שונה. לכן נעדיף פתרון זה אשר הסיבוכיות שלו קטנה בסדר גודל שלם מהפתרון הקודם.

```
C:\WINDOWS\system32\cmd.exe
The matrix is:
1      1      1      1
2      4      4      2
3      2      1      0
7      6      5      4
The sum of the main diagonal is 10
Press any key to continue . . .
```

# מערך דו-מימדי – דוגמא: הדפסת הסכום של כל עמודה

```
#define ROWS 3
#define COLS 4
void main()
{
    int matrix[ROWS][COLS] = { {1,1,1,1}, {2,4,4,2}, {7,6,5,4} };
    int i, j, sum;

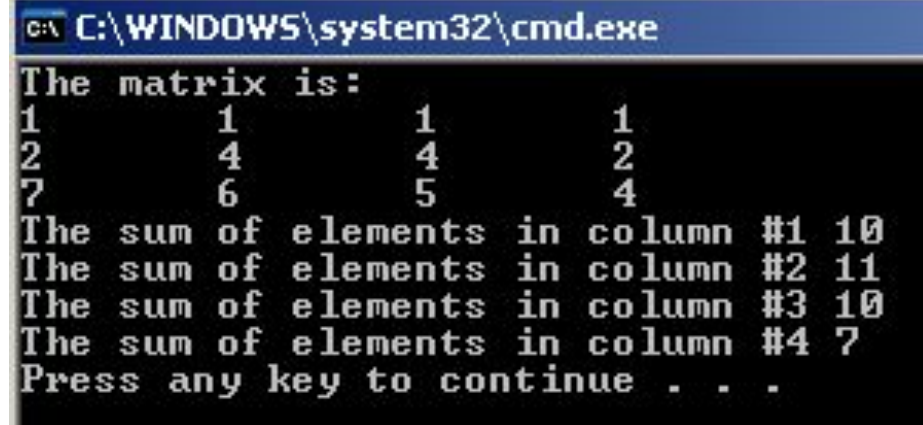
    printf("The matrix is:\n");
    for (i=0 ; i < ROWS ; i++)
    {
        for (j=0 ; j < COLS ; j++)
            printf("%d\t", matrix[i][j]);
        printf("\n");
    }

    for (i=0 ; i < COLS ; i++)
    {
        for (sum=0, j=0 ; j < ROWS ; j++)
            sum += matrix[j][i];
        printf("The sum of elements in column # %d %d\n", i+1, sum);
    }
}
```

ניתוח זמן הריצה של תוכנית זו:

$O(\text{main}) =$

$O(1) + \text{ROWS} * O(\text{COLS}) + \text{COLS} * O(\text{ROWS}) =$   
 $O(\text{ROWS} * \text{COLS})$



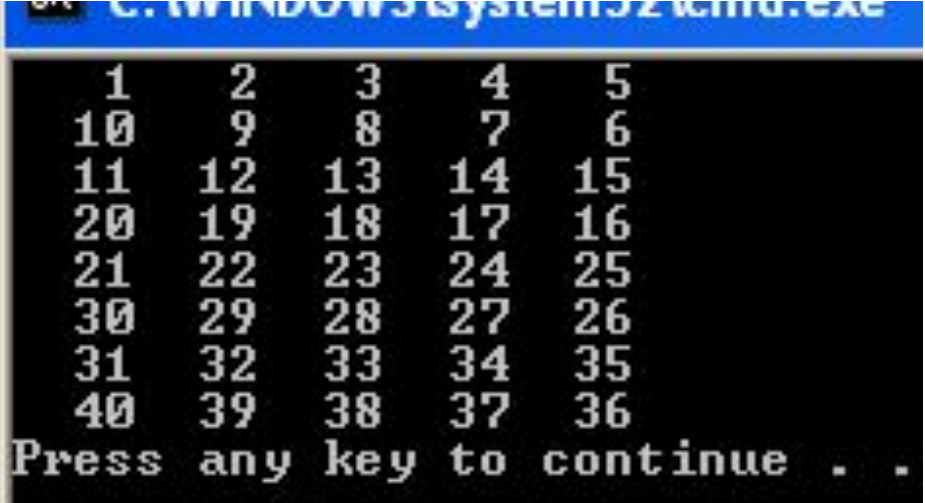
```
C:\WINDOWS\system32\cmd.exe
The matrix is:
1      1      1      1
2      4      4      2
7      6      5      4
The sum of elements in column #1 10
The sum of elements in column #2 11
The sum of elements in column #3 10
The sum of elements in column #4 7
Press any key to continue . . .
```

# מערך דו-מימדי – דוגמא: הכנסת מספרים

## למטריצה בצורת נחש

```
#define ROWS 8  
#define COLS 5
```

```
void main()  
{  
    int matrix[ROWS][COLS];  
    int i, j;  
    int value = 1;  
  
    for (i=0 ; i < ROWS ; i++)  
    {  
        if (i%2 == 0)  
            for (j=0 ; j < COLS ; j++)  
                matrix[i][j] = value++;  
        else  
            for (j=COLS-1 ; j >= 0 ; j--)  
                matrix[i][j] = value++;  
    }  
  
    // print matrix  
    for (i=0 ; i < ROWS ; i++)  
    {  
        for (j=0 ; j < COLS ; j++)  
            printf("%4d", matrix[i][j]);  
        printf("\n");  
    }  
}
```



```
C:\WINDOWS\system32\cmd.exe  
1    2    3    4    5  
10   9    8    7    6  
11   12   13   14   15  
20   19   18   17   16  
21   22   23   24   25  
30   29   28   27   26  
31   32   33   34   35  
40   39   38   37   36  
Press any key to continue . .
```

# מערך רב-מימדי

□ עד כה ראינו מערכים חד-מימדיים ומערכים דו-מימדיים

□ ניתן להרחיב את ההגדרה לכל מספר סופי של מימדים

■ למשל: מערך תלת-מימדי

```
int matrix[LENGTH][HEIGHT][DEPTH];
```

■ דוגמא לשימוש: נרצה לשמור ממוצע ציונים עבור 5 בתי-ספר,

כאשר בכל בית-ספר יש 10 כיתות, ובכל כיתה 30

סטודנטים:

```
double average[5][10][30];
```

□ במקרה זה נשתמש בלולאה, בתוך לולאה, בתוך לולאה..

# מערך רב-מימדי – דוגמאת נתוני בתי-הספר

```
#define SCHOOLS 3
#define CLASSES 2
#define STUDENTS 4
```

```
void main()
```

```
{
```

```
    float grades[SCHOOLS][CLASSES][STUDENTS] = {
        { {90, 100, 95, 88}, {87, 70, 90, 98} },
        { {88, 75, 80, 60}, {55, 87,
          90, 82} },
        { {60, 91, 40, 95}, {77, 66, 88, 99} }
    };
```

```
    int i, j, k;
```

```
    for (i=0 ; i < SCHOOLS ; i++)
```

```
    {
```

```
        printf("Classes in school #%d:\n", i+1);
```

```
        for (j=0 ; j < CLASSES ; j++)
```

```
        {
```

```
            printf(" Grades in class #%d: ", j+1);
```

```
            for (k=0 ; k < STUDENTS ; k++)
```

```
                printf("%.2f ", grades[i][j][k]);
```

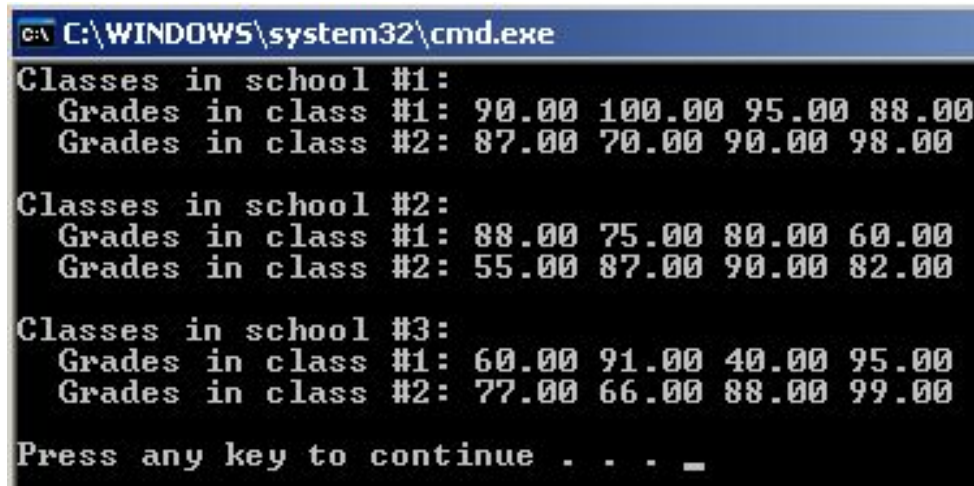
```
            printf("\n");
```

```
        }
```

```
        printf("\n");
```

```
    }
```

```
}
```



```
C:\WINDOWS\system32\cmd.exe
Classes in school #1:
  Grades in class #1: 90.00 100.00 95.00 88.00
  Grades in class #2: 87.00 70.00 90.00 98.00
Classes in school #2:
  Grades in class #1: 88.00 75.00 80.00 60.00
  Grades in class #2: 55.00 87.00 90.00 82.00
Classes in school #3:
  Grades in class #1: 60.00 91.00 40.00 95.00
  Grades in class #2: 77.00 66.00 88.00 99.00
Press any key to continue . . . _
```

# הגדרת typedef למערך

□ תזכורת: typedef הוא מתן שם חילופי לטיפוס

□ ניתן להגדיר typedef למערך באורך מסויים

■ במקום לכתוב `int arr[6]` ניתן להגדיר טיפוס חדש של מערך בעל 6 איברים מטיפוס `int`:

```
typedef int Array[6];
```

```
#define COLS 6

typedef int Array[COLS];

void main()
{
    Array arr;
}
```

ניתן לראות בדיבגר שאכן הוקצה מערך בגודל 6

Name	Value
arr	0x0012ff4c
[0]	-858993460
[1]	-858993460
[2]	-858993460
[3]	-858993460
[4]	-858993460
[5]	-858993460

# הגדרת typedef למערך דו-מימדי

ניתן להגדיר typedef גם למערך דו-מימדי

```
#define ROWS 2
#define COLS 3

typedef int Row[COLS];
typedef Row Matrix[ROWS];

void main()
{
    Matrix mat;
}
```

Name	Value
mat	0x0012ff4c
[0]	0x0012ff4c
[0]	-858993460
[1]	-858993460
[2]	-858993460
[1]	0x0012ff58
[0]	-858993460
[1]	-858993460
[2]	-858993460



# ביחידה זו למדנו:

---

- מהו מערך
- כיצד מערך נראה בזיכרון
- גישה לאיברי המערך
- אתחול מערך
- הפונקציה sizeof
- חריגה מגבולות המערך
- השמת מערכים
- מערך דו-מימדי
- מערך רב-מימדי
- typedef למערך

# תרגילי חימום:

1. הגדר מערך בגודל 10 של מספרים שלמים וקלוט לתוכו ערכים. הצג רק ערכים שהם זוגיים.
2. הגדר מערך בגודל 10 של תווים וקלוט לתוכו ערכים. הצג את האינדקסים שבתוכם ישנו תו שהוא אות גדולה.
3. הגדר מערך של מספרים בגודל 10 ושים בתוכו לפי הסדר ערכים שהם כפולות של 3: כלומר הערכים 0, 3, 6 וכו'.
4. הגדר מערך של מספרים שלמים וקלוט לתוכו ערכים. הגדל ב-1 את הערכים שנמצאים במיקומיים זוגיים (0, 2, 4 וכו').
5. הגדר מערך של מספרים שלמים בגודל 10 וקלוט לתוכו ערכים. הגדל ב-1 את הערכים שנמצאים במיקומיים זוגיים (0, 2, 4 וכו') ואח"כ הקטן ב-1 את הערכים שנמצאים במיקומים שהם כפולה של 3 (0, 3, 6 וכו').

# תרגילי חימום:

6. הגדר מערך של תווים וקלוט לתוכו ערכים, וכן קלוט תו נוסף. הצג כמה פעמים התו הנוסף שהתקבל מופיע במערך.
7. הגדר שני מערכים של מספרים שלמים בגודל 5 כל אחד. קלוט ערכים לתוך המערך הראשון ואז קלוט ערכים לתוך המערך השני. הצג את המיקומים אשר הערכים בהם בשני המערכים זהים.
8. הגדר 3 מערכים של מספרים שלמים בגודל 5 כל אחד. קלוט ערכים לתוך המערכים הראשון והשני. שים בכל איבר במערך השלישי את סכום האיברים במיקומים התואמים במערכים הראשון והשני.
9. הגדר מערך של 5 תווים וקלוט לתוכו ערכים. בדוק האם כל התווים שהוקלדו למערך זהים והציגו הודעה מתאימה.
10. הגדר מערך של 5 מספרים שלמים וקלוט לתוכו ערכים. בדוק האם ערך כל איבר גדול מערך האיבר שלפניו והצג בסוף הודעה

# תרגיל 1: הדפסת האינדקסים של הערך המינימלי

- כתוב תוכנית המגדירה מערך בגודל 10 של מספרים שלמים
- קלוט ערכים מהמשתמש, והדפס את האינדקסים של האיברים שערכם שווה לערך המינימלי במערך
- כמו כן יש להדפיס את מספר האינדקסים שבתוכם יש ערך השווה לערך המינימלי

0	1	2	3	4	5	6	7	8	9
6	14	12	14	12	5	14	6	5	5

■ דוגמא:

עבור המערך

- יש להדפיס 5 8 9 (כי 5 הוא המינימלי והוא נמצא באינדקסים אלו)

- וכן להדפיס את הערך 3, מאחר והערך המינימלי מופיע 3

פעמים.

# תרגיל 2: הדפסת האינדקסים של הערך המינימלי

□ כתבו תוכנית המגדירה מערך בגודל 10 של מספרים שלמים

□ קלוט ערכים מהמשתמש

□ הצג האם ערכי המערך מהווים סדרה חשבונית

□ דוגמאות:

0	1	2	3	4	5	6	7	8	9
6	8	10	12	14	16	18	20	22	24



0	1	2	3	4	5	6	7	8	9
6	8	10	12	14	17	18	20	22	24



# תרגיל 3: השמת כוכביות על אלכסוני המטריצה

- כתוב תוכנית והגדר בה מטריצה ריבועית של תווים בגודל SIZE
- יש לשים את התו '\*' על איברי האלכסון הראשי והמשני ורווח בשאר האיברים (לדמות את הצורה X)
- הדפס את המטריצה
- למשל, עבור SIZE=5 המטריצה תראה כך:

*				*
	*		*	
		*		
	*		*	
*				*

# תרגיל 4: העמודה בה מספר הופיע הכי הרבה פעמים

- כתוב תוכנית והגדר בה מטריצה בגודל ROWSxCOLS של מספרים
- קלוט לתוכה ערכים מהמקלדת
- קלוט מהמשתמש מספר והדפס את האינדקס של העמודה בה המספר שהוקלד מופיע הכי הרבה פעמים
  - אם התו כלל לא מופיע במטריצה יש לתת הודעה מתאימה
  - שימו לב: אין לעבור על המטריצה בהתחלה כדי לבדוק זאת!
- למשל, עבור המטריצה הבאה והמספר 3 יוצג 2 כי המספר 3 מופיע הכי הרבה פעמים בעמודה 2

1	9	3	1	8
0	2	5	7	2
6	3	3	4	3