פרק 5 : מבני נתונים סדרתיים

יעדים

הכרת המערך כאוסף לינארי של טיפוסים מאותו הסוג, עבודה עם מערכים חד ממדיים ודו ממדיים. הכרה ומימוש של אלגוריתמי חיפוש, מיון ומיזוג והבנת היעילות שלהם

תכנים

1. מערך של טיפוסי נתונים בסיסיים
2. מושגי יסוד בעבודה עם מערכים: מציין על ידי index , גישה על ידי x[i]
3. אורך המערך ובדיקת גבולות
4. הגדרה ואתחול של מערכים (ממערך חד-מימדי למערך דו-מימדי)
5. העמקה בנושא מחרוזות בשפת C כמערך של chars .
6. פעולות הספרייה של תווים ומחרוזות כגון: strcat , strcpy , strcmp , isdigit , isalpha , islower etc
7. אלגוריתמים למיון מערכים, דיון על יעילות מיון : BubbleSort ו- MaxSort

סיכום שעות ההוראה: 18 שעות עיוניות ו- 8 שעות התנסותיות. סה"כ 26 שעות.

מטרה : בפרק זה נלמד מהו מערך ומהי מחרוזת, איך מגדירים מערך/מחרוזת, איך מאתחלים אותם, כיצד ניגשים לאיבר כלשהו במערך/במחרוזת ונציג דוגמאות לעבודה עם מערך/מחרוזת.

5.1 מערך של טיפוסי נתונים בסיסיים

מערך - array - הוא רצף/סידרה של משתנים המסודרים אחד אחרי השני בזיכרון , כולם מאותו הטיפוס ולכולם אותו שם. הם נבדלים אחד מהשני במיקום שלהם במערך.

העבודה במערכים מאפשרת עבודה קלה, חסכון בזמן ויצירת ארגון וסדר בתוכנית כאשר אנו משתמשים במספר רב של משתנים שנועדו לאותה המטרה. **כל משתנה במערך נקרא איבר** או פריט . לכמות האיברים במערך קוראים גודל המערך ובאנגלית size .

תחביר להגדרת מערך:

< טיפוס איברי המערך > < שם המערך > [כמות האיברים במערך] ;

דוגמאות להגדרה של מערכים:

int arr[5]; // הגדרת מערך בשם arr מטיפוס שלם שיש בו 5 איברים / פריטים

שם המערך arr	**arr[0]**	**arr[1]**	**arr[2]**	**arr[3]**	**arr[4]**
אינדקס/מיקום	**0**	**1**	**2**	**3**	**4**

איור 1 : כיצד נראה מערך ?

יש לשים לב שהאיבר הראשון נקרא arr[0] (הווה אומר - arr במיקום אפס או arr באינדקס אפס) והאיבר האחרון הוא arr[4] . סה"כ יש 5 איברים .

במערך בן n איברים , האיבר ראשון במיקום/אינדקס 0 והאחרון הוא במיקום/אינדקס n-1 .

דוגמאות להגדרות נוספות :

הגדרת מערך מטיפוס תווי ששמו string ויש בו 8 איברים. מערך מטיפוס תווי נקרא מחרוזת // ;char string[8]

הגדרת מערך מטיפוס ממשי בשם f שיש בו 5 איברים // ;float f[5]

הגדרת מערך מטיפוס ממשי כפול בשם big_f שיש בו 10 איברים // ;double big_f[10]

ניתן גם לאתחל את האיברים במערך בזמן ההגדרה :

```
int arr[ 4] = { 10,-20,30,50};
```

ניתן היה גם לרשום את ההגדרה:

```
int arr[ ] = { 10,-20,30,50};
```

במקרה הזה, למרות שלא הגדרנו את כמות האיברים, הקומפיילר יפתח מערך של 4 איברים ויאתחל את האיברים בצורה הבאה :

שם המערך arr	**arr[0]**	**arr[1]**	**arr[2]**	**arr[3]**	**arr[4]**
אינדקס/מיקום	**10**	**20**	**30**	**40**	**50**

איור 2 : האיברים במערך arr

ההגדרה הבאה : int a[ ] ; איננה חוקית בשפת C כי הקומפיילר לא יודע בכמה איברים מדובר (ישנן שפות מתקדמות יותר שאין צורך להגדיר את כמות האיברים).

ההגדרה הבאה היא **כן** חוקית:

```
int a[4] = {4, ,8, };
```

במקרה הזה יוגדר מערך בן 4 איברים מטיפוס שלם בשם a . האיבר הראשון והשלישי יקבלו את הערכים 4 ו 8 בהתאמה. האיברים במקום ה 2 והרביעי לא אותחלו וערכם יהיה אקראי אם המערך אוטומטי או 0 אם המערך גלובאלי (יוסבר בהמשך ההבדל בין אוטומטי לגלובאלי).

היות והאיברים מסודרים אחד אחרי השני בזיכרון המחשב אז אם האיבר הראשון נמצא בכתובת עשרונית 2100 והמערך מטיפוס שלם (כל איבר "תופס" 4 כתובות בזיכרון) אז האיבר השני נמצא בכתובת 2104 והשלישי ב 2108 וכך הלאה.

במערך של איברים מטיפוס short (כל איבר "תופס" 2 בתים) שהאיבר הראשון שלו נמצא בכתובת 4010 , האיבר הבא יהיה בכתובת 4012 והבא אחריו בכתובת 4014 וכך הלאה.

אם המערך מטיפוס תווי (כל איבר "תופס" 1 ביית) והמערך מתחיל בכתובת 5000 אז האיבר השני בכתובת 5001 וכך הלאה.

5.2 מושגי יסוד בעבודה עם מערכים : מציין על ידי אינדקס , גישה על ידי x[i]

5.2.1 כיצד נוכל לשנות את ערכו של אחד האיברים במערך ?

נניח שהגדרנו מערך : int array1[4]={10,20,30,40}; ורוצים לשנות את האיבר הראשון ל 100 . מכאן שנרשום :

```
array1[0]=100;
```

אם נרצה לשנות את האיבר האחרון של המערך ל 95 נרשום :

```
array1[3]=95;
```

באופן כללי : אם נרצה לשנות את ערכו של האיבר i (איבר כלשהו מ 0 עד 2) במערך array1 ל 10 נרשום : **arr[i]=10;**

כלל חשוב : בזמן ההגדרה של מערך ניתן לאתחל את כל האיברים של המערך. לאחר שלב ההגדרה ניתן לשנות את ערכו של אחד האיברים בלבד !!

לדוגמה : נניח שהגדרנו מערך בצורה הבאה :

```
int array2[4] = {1,2,3,4};
```

אם נרצה לשנות את איברי המערך ונרשום array2[4]={4,3,2,1}; נקבל הודעת שגיאה ! אחרי שהגדרנו את המערך ניתן לשנות רק את ערכו של איבר אחד בלבד . כלומר במקרה שרוצים לשנות את האיברים יש לרשום :

```
array2[0]=4;
array2[1]=3;
array2[2]=2;
array2[3]=1;
```

תרגיל: מה מצב המערך בסיום קטע התוכנית הבא ?

```
int a[4] = {10,20,30,40}'
for(i=0 ; i<4 ; i++)
  a[i] = a[i]*2;
```

תשובה : כל איבר במערך מכפיל את ערכו פי 2 ונקבל : a[0]=20 a[1]=40 a[2]=60 a[3]=80 .

5.2.2 קלט אל מערך ופלט של מערך

קלט אל מערך

לא ניתן לקלוט אל מערך מספר נתונים בו זמנית אלא יש לבצע זאת עבור כל איבר בנפרד.

כאשר רוצים לקלוט מהמשתמש נתון אל איבר כלשהו במערך , לדוגמה אל האיבר השלישי במערך a בתרגיל שלמעלה, נרשום:

```
scanf("%d",&a[2]);
```

יש לזכור שהאיבר השלישי במערך הוא a[2] כי המערך מתחיל מ a[0] .

אם נרצה לקלוט בלולאה נתונים אל המערך הזה נרשום זאת בעזרת לולאה :

```
for(int i=0 ; i<4;i++)
  {
    printf("Please enter a number to a[%d] : " , i);
    scanf("%d",&a[i]);
  }
```

פלט של מערך

לא ניתן להדפיס את כל איברי המערך בו זמנית אלא מבצעים הדפסה עבור כל איבר בנפרד.

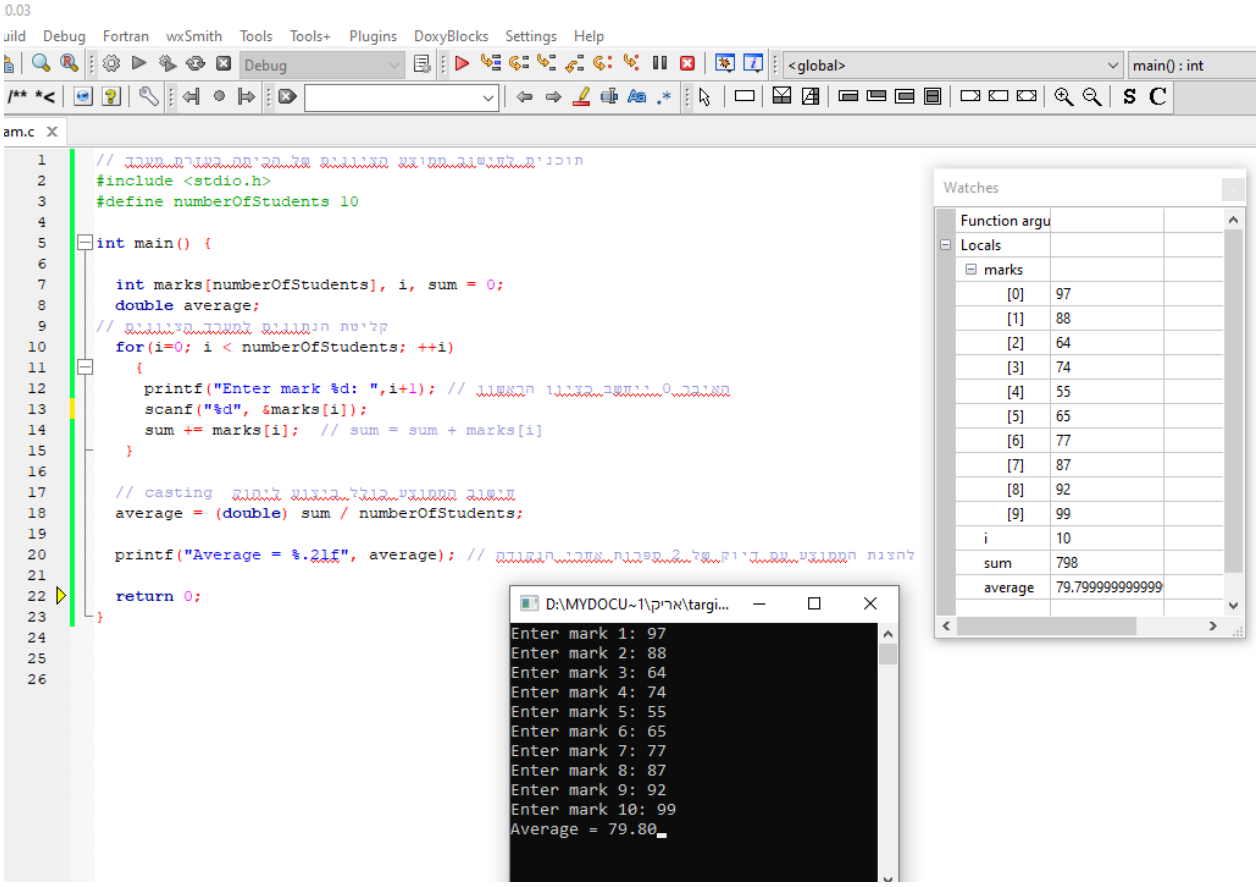
דוגמה : נדפיס למסך את האיבר השני של המערך a .

```
printf("The second item in a array is %d",a[1]);
```

אם נרצה להדפיס את כל איברי המערך נעשה זאת בעזרת לולאה:

```
for(int i=0 ; i<4;i++)
  {
    printf("a[%d]  = %d \t" , i, a[i]);
  }
```

דוגמה : חישוב ממוצע ציונים :



איור 3 : תוכנית לחישוב ממוצע ציונים בעזרת מערך.

בתחילת התוכנית קלטנו בעזרת לולאת for 10 ציונים מהמשתמש אחד אחרי השני והכנסנו אותם למערך marks . כל ציון שהוכנס הוספנו למשתנה sum . במשתנה sum יש את סכום כל הציונים. בשורה 18 חישבנו את ממוצע הציונים תוך ביצוע casting . התוצאה שקיבלנו בטבלת ה watches היא 79.7999999 . הדפסנו את התוצאה למסך וקיבלנו 79.80 כי ביקשנו הדפסה עם דיוק של 2 ספרות וזהו המספר הכי קרוב לתוצאה של הממוצע ב watches .

5.3 אורך המערך ובדיקת גבולות

נניח שהגדרנו מערך של 10 איברים int array[10]; . ניתן לגשת לכל איבר במערך מ array[0] ועד array[9] . מה קורה אם ננסה לגשת אל איבר שלא קיים במערך ? לדוגמה נרשום array[12]=90; . האיבר הזה לא קיים במערך והדבר עלול לגרום לפעולות לא הגיוניות. ייתכן שנקבל שגיאה וייתכן שהתוכנית תפעל בכל זאת כראוי. הכלל הוא שאין לגשת לאיבר שאיננו קיים במערך . במילים אחרות:

חריגה מגבולות מערך היא באחריות המשתמש !!

5.4 מערך רב ממדי - MULTIDIMENSIONAL ARRAY

עד עכשיו דיברנו על מערך שבו יש ממד אחד. מערך רב ממדי הוא מערך שבו יש יותר מממד אחד. לדוגמה נגדיר מערך דו ממדי :

```
int x[3][4] ;
```

הוא מערך שבו יש 3 שורות וארבע עמודות. הערך בסוגריים הראשונים הוא כמות השורות והערך בסוגריים המרובעים השני הוא כמות העמודות. מערך כזה נראה כך:

	עמודה 1	עמודה 2	עמודה 3	עמודה 4
שורה 1	x[0][0]	x[0][1]	x[0][2]	x[0][3]
שורה 2	x[1][0]	x[1][1]	x[1][2]	x[1][3]
שורה 3	x[2][0]	x[2][1]	x[2][2]	x[2][3]

איור 4 : מערך דו ממדי של 3 שורות ו 4 עמודות

האינדקס הראשון הוא מספר השורה והשני הוא מספר העמודה.

האיבר השמאלי למעלה הוא האיבר בשורה 0 והעמודה שלו היא 0 .

האיבר בשורה הראשונה מימין הוא בשורה 0 בעמודה 3 .

האיבר הימני ביותר למטה הוא בשורה 2 בעמודה 3 .

בצורה כללית נגדיר מערך דו ממדי כך :

< שם המערך > [כמות השורות][כמות העמודות];

אתחול מערך דו ממדי:

```
int a[3][4] = {1,2,3,4,5,6,7,8,9,10,11,12};
```

הסדר של האתחול הוא קודם ממלאים את השורות ואחר כך את העמודות. המערך ייראה כך :

a[3][4]	**עמודה 0**	**עמודה 1**	**עמודה 2**	**עמודה 3**
שורה 0	1	2	3	4
שורה 1	5	6	7	8
שורה 2	9	10	11	12

איור 5 : אתחול מערך דו ממדי

דוגמאות לאתחול מערך דו ממדי . נניח שנרצה להגדיר מערך של איברים מטיפוס שלם בשם z . נוכל לאתחל את האיברים שלו בזמן ההגדרה ב 4 הצורות הבאות:

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

או בדרך נוספת:

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

לא חייבים להגדיר את כמות השורות אם מאתחלים את כל האיברים. חייבים את כמות העמודות.

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

או בדרך נוספת:

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

מערך עם 3 ממדים

הגדרה של מערך עם 3 ממדים תיראה כך :

```
double a[2][4][3];
```

הוא מערך שיכול לתאר 2 שורות של אורך , 4 עמודות של רוחב ו 3 של גובה (או עומק). במערך כזה יש 24 איברים שונים.

דוגמה להגדרת מערך 3 ממדי עם אתחול המערך:

```
int a3d[2][3][4] = {{{1,2,3,4},{5,6,7,8},{4,3,2,1},{9,8,7,6},{-1,-2,-3,-4},{10,20,30,40}}};
```

גם ההגדרה הבאה חוקית :

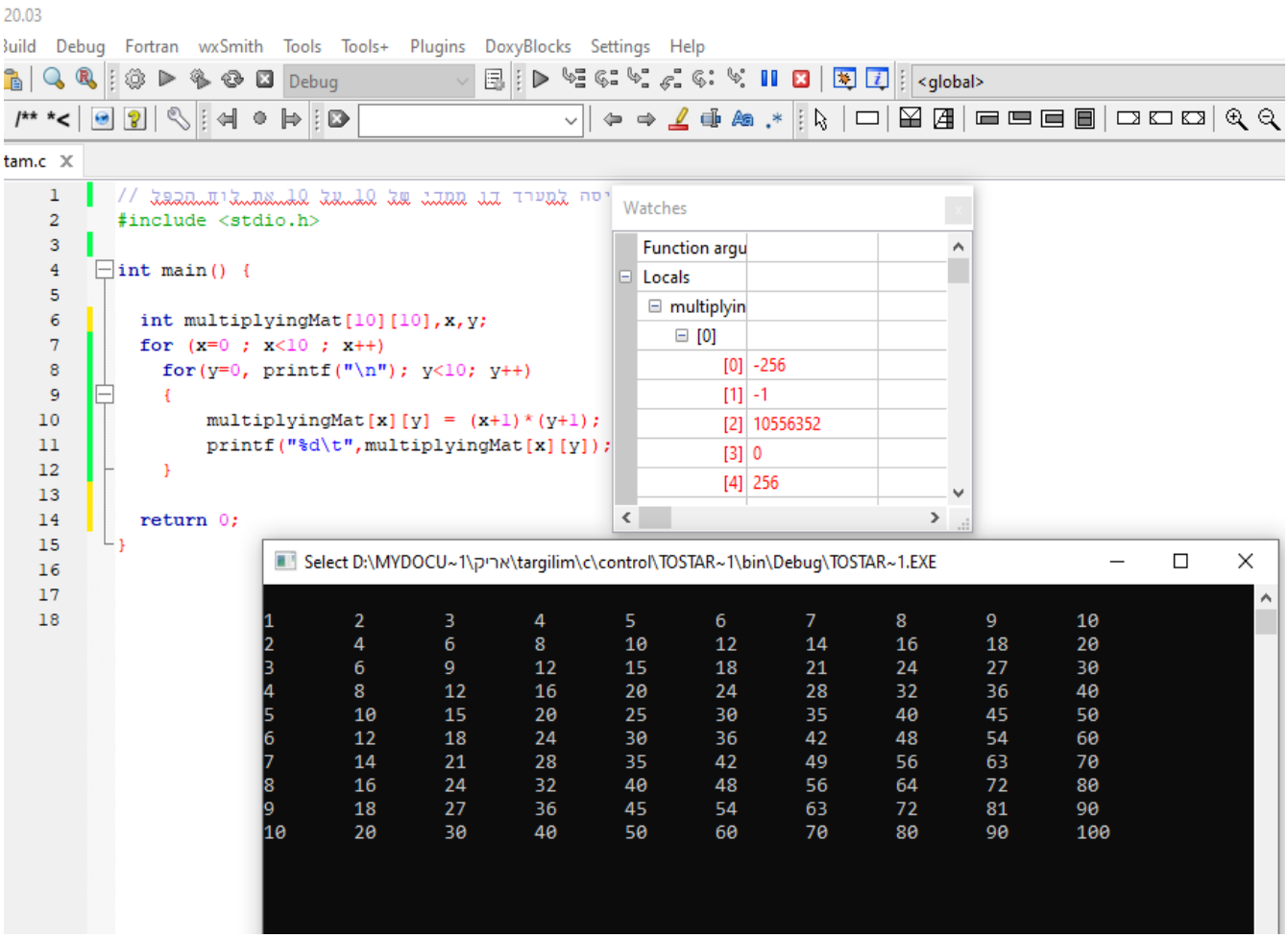
```
int a2d[2][3][4] = {1,2,3,4,5,6,7,8,4,3,2,1,9,8,7,6,-1,-2,-3,-4,10,20,30,40};
```

וגם ההגדרה הבאה היא חוקית:

```
int a2d[ ][3][4] = {1,2,3,4,5,6,7,8,4,3,2,1,9,8,7,6,-1,-2,-3,-4,10,20,30,40};
```

באופן תאורטי ניתן להגדיר מערך רב ממדי עם כמה ממדים שנרצה. קשה למוח האדם לתאר מבנה של מערך יותר מ 3 ממדים (זה נראה לנו כמו אורך רוחב ועומק). אנחנו נסתפק בספר זה במערך של לא יותר מ 3 ממדים.

דוגמה עם מערך דו ממדי של מערך שבו נשמור את לוח הכפל מ 1 עד 10:



איור 6 : תוכנית המכניסה את לוח הכפל למערך של 10 על 10 ומדפיסה את המערך

5.9 מחרוזות STRINGS

מחרוזת היא מערך מטיפוס תווי - char .

הגדרת מחרוזת דומה להגדרת מערך.

הגדרה :

char < שם המחרוזת > [כמות האיברים במחרוזת] ;

דוגמה : הגדרה של מחרוזת בת 10 אברים : char string1[10];

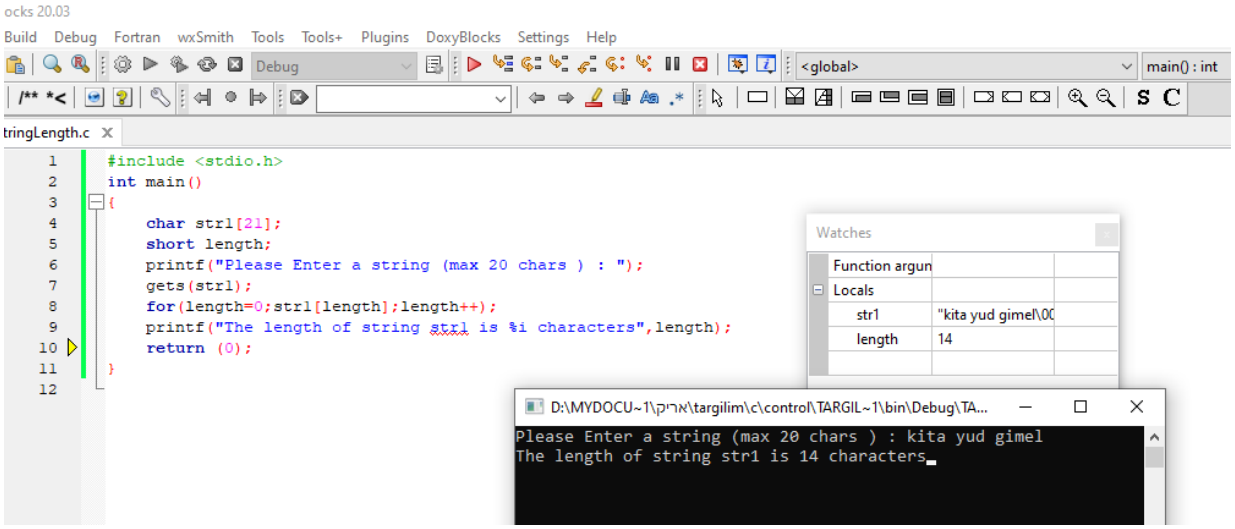
כאשר מגדירים מחרוזת הקומפיילר מכניס למחרוזת את התו \0 (NULL) . לכן יש לדאוג לאיבר נוסף עבור תו זה.

דוגמה: char str2[ ] = "Hello" ;

במקרה הזה הקומפיילר ייפתח מערך – מחרוזת – של 6 איברים (האיבר השישי הוא ה NULL).

כאשר הקומפיילר מגדיר מערך או מחרוזת הוא פותח גם מצביע שהשם שלו הוא שם המערך או המחרוזת, ומכניס אליו את הכתובת של האיבר הראשון. מכאן **שם של מערך או מחרוזת הוא הכתובת של האיבר הראשון של המערך או המחרוזת.**

דוגמה: נקלוט מהמשתמש מחרוזת ונמצא את כמות התווים של המחרוזת (אורך מחרוזת).



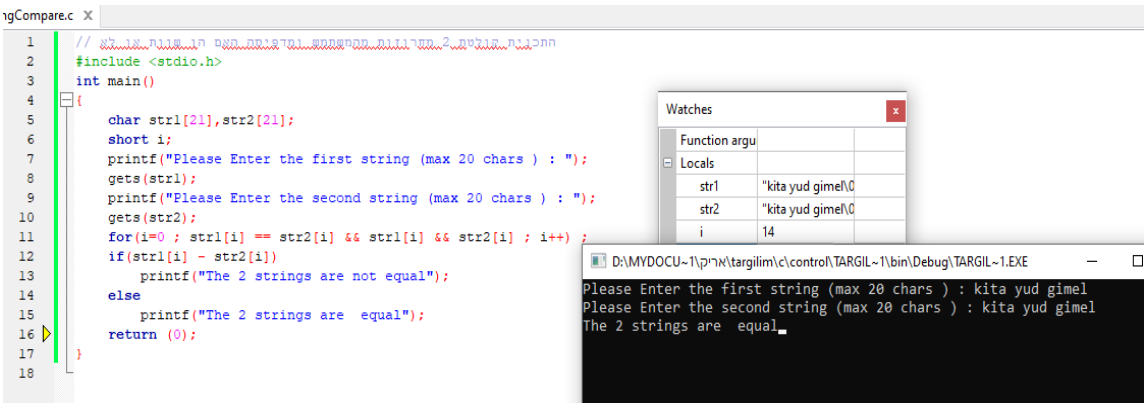
איור 7 : מציאת כמות התווים של מחרוזת.

בהנחה שהמשתמש הקיש את המחרוזת "kita yud gimel" '.

כדאי לשים לב שהתנאי של לולאת ה for בשורה 8 הוא str1[length] שאומר str1[length]==TRUE . בסוף לולאת ה for יש נקודה פסיק כלומר הלולאה לא כוללת משפטי גוף נוספים אלא שאם התנאי מתקיים מבצעים רק את המשפט length++ . הלולאה עוברת על כל התווים בלולאה כי לכל תו יש ערך ASCII שלו שהוא TRUE . הלולאה מסתיימת כאשר מגיעים ל NULL (שהוא '/0' = 00000000) ואז הלולאה היא FALSE והיא מסתיימת.

סה"כ הוכנסו 14 תווים על ידי המשתמש .

דוגמה : נקלוט 2 מחרוזות מהמשתמש ונבדוק האם 2 המחרוזות שוות .

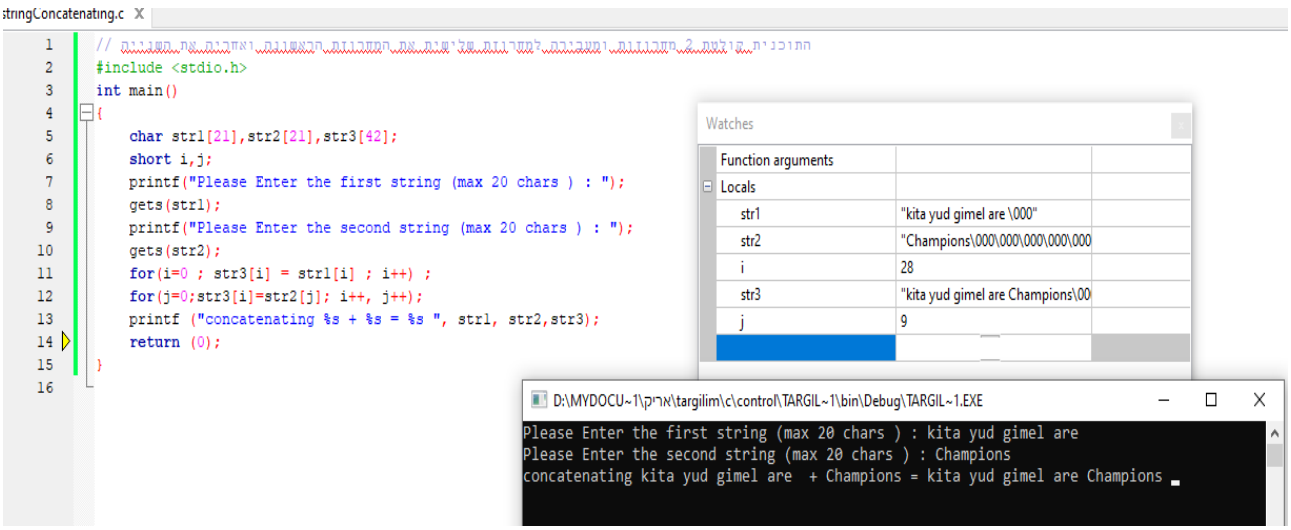


איור 8 : השוואה בין 2 מחרוזות

התנאי בלולאת ה for שבשורה 11 מתבצעת כל עוד האיבר במיקום i במחרוזת הראשונה שווה לאיבר i במחרוזת השנייה וגם (פעולת AND &&) str1[i] == TRUE וגם str2[i] == TRUE כלומר לא הסתיימה המחרוזת הראשונה וגם לא הסתיימה המחרוזת השנייה .

כאשר מסתיימת לולאת ה for בודקים בשורה 12 את ההפרש בין str1[i] – str2[i] . אם ההפרש הוא TRUE , כלומר מספר שאיננו 0 זה אומר שבמקום כלשהו התגלה שההפרש איננו 0 ולכן המחרוזות אינן שוות. אם ההפרש 0 אז עוברים לשורה 15 ומדפיסים שהמחרוזות שוות.

דוגמה : נקלוט 2 מחרוזות מהמשתמש ונעביר אל מחרוזת שלישית את המחרוזת הראשונה ולאחריה נוסיף את המחרוזת השנייה .



איור 9 : התוכנית קולטת 2 מחרוזות ומעבירה למחרוזת שלישית את המחרוזת הראשונה ואחריה את השנייה.

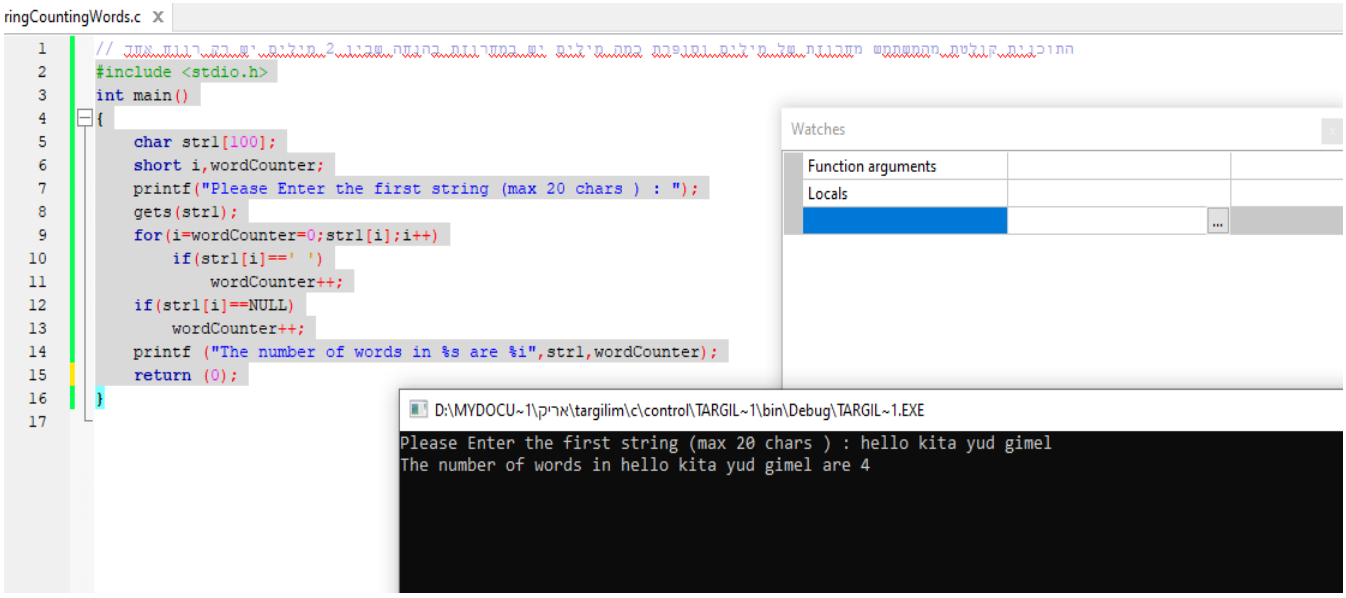
בלולאת ה for בשורה 11 העתקנו את מחרוזת str1 אל str3 . התנאי str3[i]=str1[i] ולא עם == (שווה שווה) אומר : א. להעביר את str1[i] אל str3[i] ב. לבדוק האם str3==TRUE . במקרה הזה כל המחרוזת הראשונה הועתקה אל השנייה. לולאת ה for בשורה 12 מעתיקה את המחרוזת השנייה בסיום הראשונה.

דוגמה : מה עושה התוכנית הבאה ? הנח שבין כל 2 מילים יש רק תו רווח - space - אחד והמחרוזת שהוכנסה על ידי המשתמש היא : hello kita yud gimel .

```
#include <stdio.h>
int main()
{
   char str1[100];
   short i,wordCounter;
   printf("Please Enter the first string (max 20 chars ) : ");
   gets(str1);
```

```
for(i=wordCounter=0;str1[i];i++)
   if(str1[i]==' ')
      wordCounter++;
if(str1[i]==NULL)
   wordCounter++;
printf ("The number of words in %s are %i",str1,wordCounter);
return (0);
```

הפתרון :



איור 10 : תוכנית הסופרת את כמות המילים במחרוזת.

בספריה הסטנדרטית של שפת C יש קובץ כותר הנקרא string.h . בקובץ כותר זה יש הצהרות על פונקציות כמו :

א. **strlen** – הפונקציה מקבלת מחרוזת ומחזירה את אורך המחרוזת.

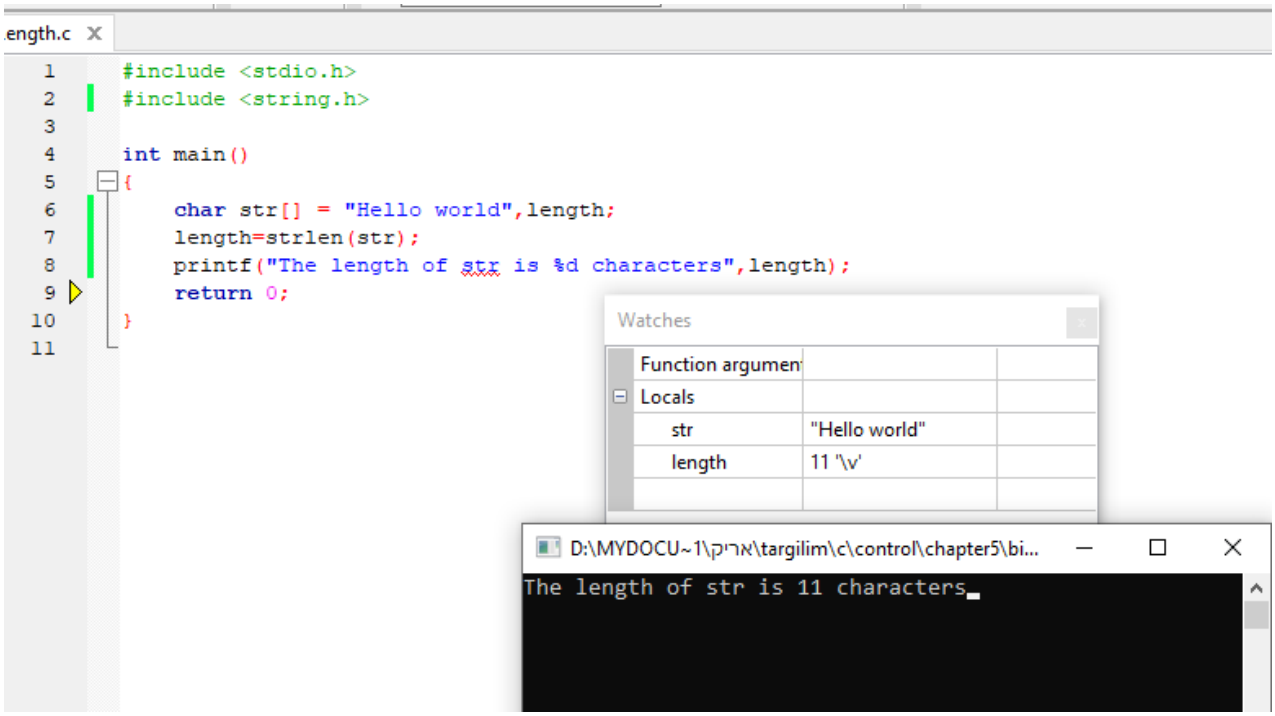
ב. **strcpy** - הפונקציה מקבלת 2 מחרוזות ומעתיקה את המחרוזת השנייה אל הראשונה.

ג. **strcmp** – הפונקציה מקבלת 2 מחרוזות ומבצעת השוואה ביניהן. אם מוחזר 0 – 2 המחרוזת שוות. אם מוחזר מספר שונה מ 0 (חיובי או שלילי) זה ההפרש הלקסיקוגרפי במקום הראשון שיש חוסר שוויון בין המחרוזות , בין תו האסקי במחרוזת הראשונה ותו האסקי במחרוזת השנייה.

ד. **strcat** - הפונקציה מקבלת 2 מחרוזות ומשרשרת (מוסיפה) את המחרוזת השנייה בסיום הראשונה.

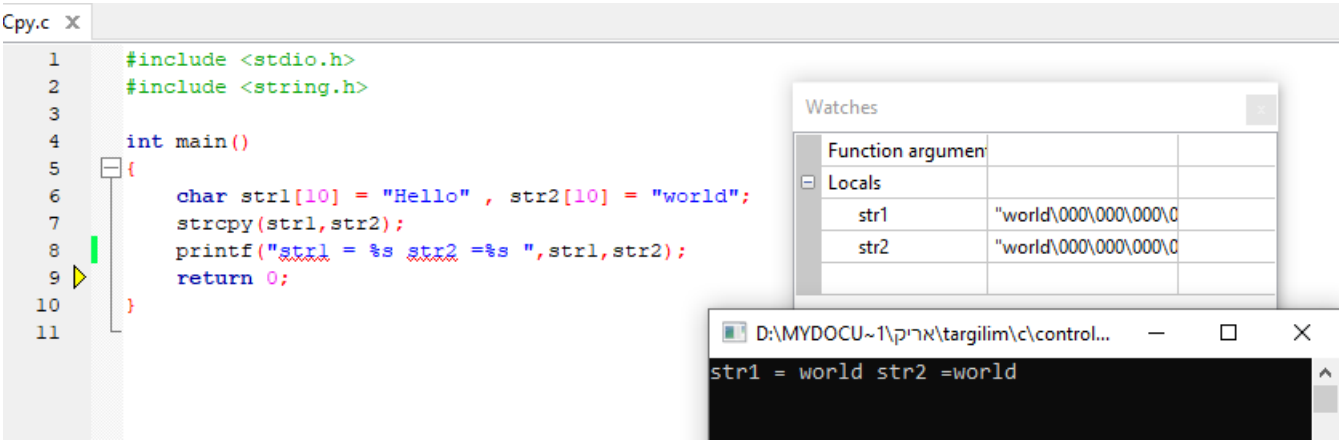
אנחנו רשאים להשתמש בפונקציות אלו . כמובן שנרשום בתחילת התוכנית : #include <string.h> .

לדוגמה : מציאת כמות התווים במחרוזת עם שימוש בפונקציה strlen :



איור 11 : תוכנית למציאת אורך מחרוזת המשתמשת בפונקציה strlen .

דוגמה להעתקת מחרוזת עם הפונקציה strcpy

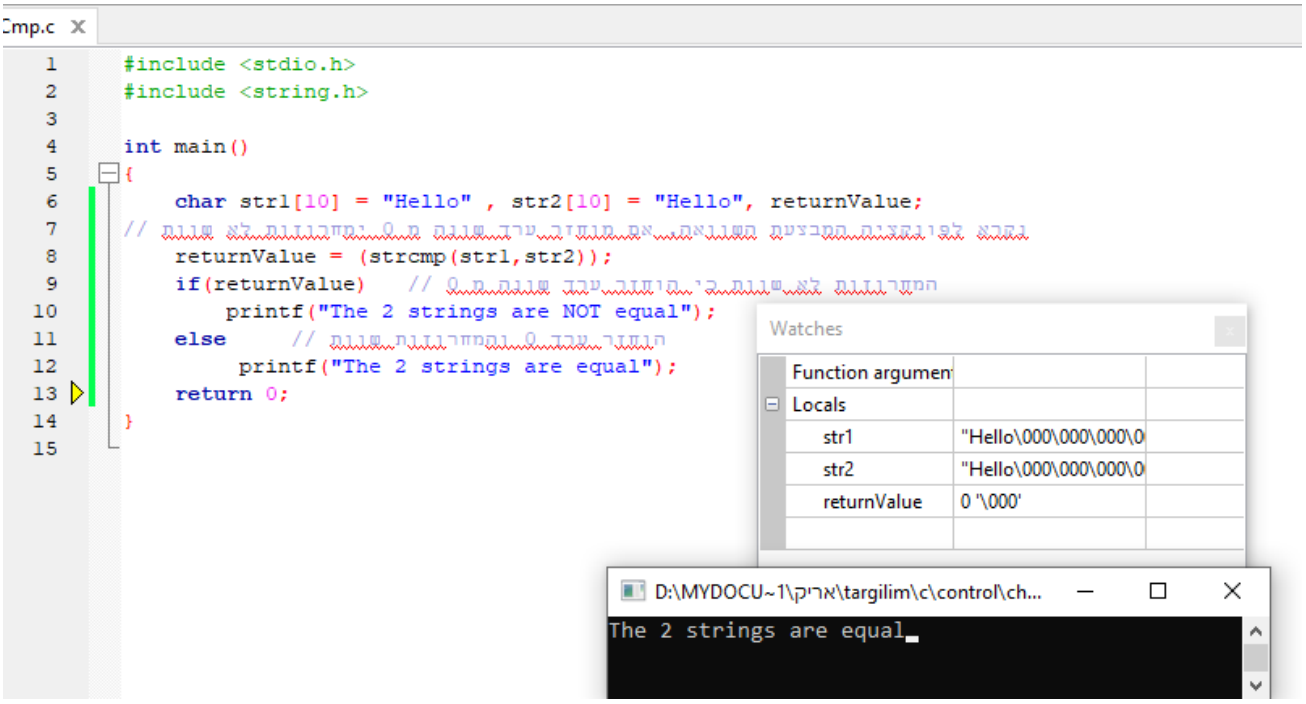


איור 12 : תוכנית להעתקת מחרוזת אחת לאחרת

כדאי לשים לב שמחרוזת 2 הועתקה אל מחרוזת אחת . התוכן הקודם של מחרוזת אחד "נמחק".

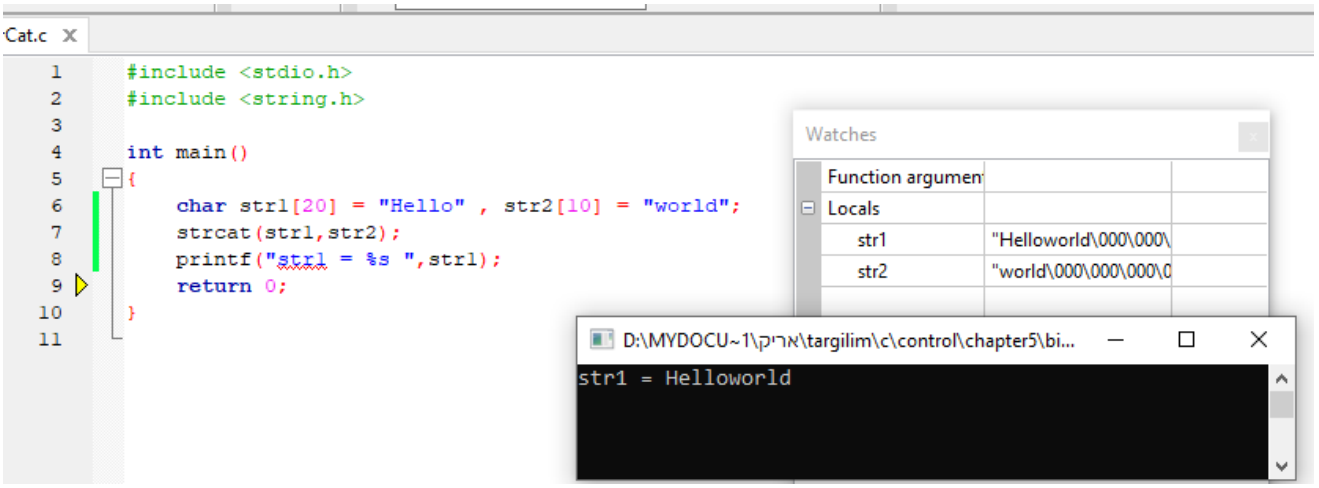
אם המקום שורה 7 היינו רושמים strcpy(str2,str1); היינו מקבלים str1=Hello str2=Hello .

דוגמה : תוכנית המשווה בין 2 מחרוזות :



איור 13 : תוכנית להשוואת 2 מחרוזות

דוגמה : תוכנית המשרשרת מחרוזת בסיום מחרוזת בעזרת הפונקציה strcat



איור 14 : תוכנית המשרשרת את המחרוזת השנייה בסיום הראשונה.

כדאי לשים לב שלמחרוזת השנייה הקצינו מספר רב יותר של בתים.

5.10 מיון – SORT

מיון של מערך הוא שינוי סדר האיברים של המערך כך שהם יסודרו לפי סדר עולה (או יורד) מהערך הקטן אל הגדול (או ההיפך). או כאשר נרצה לסדר מחרוזת לפי סדר לקסיקוגרפי כמו מילון (לפי סדר האלף בית של ערכי האסקי) וכו'.
שיטת המיון מתבצעת על ידי פעולות השוואה והחלפת מיקום האיברים.

מתי נרצה להשתמש במיון ?

כאשר יש לנו מערך של ציוני תלמידים ונרצה לדרג אותם מהציון הנמוך אל הגבוה או כאשר יש לנו משכורות של עובדים ונרצה לסדר אותם מהמשכורת הנמוכה לגבוהה וכו' .

קיימות שיטות מיון רבות כמו מיון בועות – Bubble Sort . מיון מקסימום – Max Sort , מיון הכנסה – Insertion Sort , מיון מהיר – Quick Sort , מיון מיזוג – Merge Sort ועוד סוגי מיון נוספים. ההבדל בין המיונים הוא כמות פעולות ההשוואה וההחלפות שיש לבצע עד שמגיעים למרכך ממוין.

. אנחנו נזכיר את השיטות הבאות: **א. מיון בועות ב. מיון מקסימום - Max Sort**

5.10.1 מיון בועות – Bubble Sort

מיון בועות הוא מיון שבו מבצעים איטרציות (פעולות חוזרות של לולאה). נבצע מיון בועות בסדר עולה, כלומר מהערך הקטן אל הגדול. בכל איטרציה משווים בין כל שני ערכים סמוכים של המערך . אם הערך באינדקס x גדול מהערך באינדקס x-1 מבצעים החלפה בין הערכים כך שהערך באינדקס [x] יהיה הערך הגבוה יותר. המטרה בכל איטרציה היא "להציף" את הערך הגבוה שיש במערך במיקום האחרון במערך . בכל איטרציה מבצעים השוואה אחת פחות מהאיטרציה הקודמת עד אין יותר השוואות והמערך ממוין (במקרה הזה מהקטן אל הגדול) .אולי מכאן בא השם "מיון בועות" שכמו בועה במים היא צפה כלפי מעלה.

לדוגמה נביא מערך שבו יש 5 ערכים מאינדקס [0] ועד אינדקס [4] .

אינדקס	[0]	[1]	[2]	[3]	[4]
ערך	7	8	2	9	1

איטרציה הראשונה :

א. משווים בין שני הערכים הנמצאים באינדקס 0 ובאינדקס 1 . היות והערך באינדקס [1] גדול מזה של [0] לא עושים כל פעולה והמערך לא השתנה.

אינדקס	[0]	[1]	[2]	[3]	[4]
ערך	7	8	2	9	1

ב. מבצעים השוואה בין הערכים באינדקס [1] ו [2] . היות ו 8 גדול מ 2 מבצעים ביניהם החלפה ומקבלים :

אינדקס	[0]	[1]	[2]	[3]	[4]
ערך	7	2	8	9	1

ג. מבצעים השוואה בין הערכים באינדקס [2] ו [3] . 8 לא גדול מ 9 ולא מבצעים החלפה ונשאר המערך כמו מקודם :

אינדקס	[0]	[1]	[2]	[3]	[4]
ערך	7	2	8	9	1

ד. מבצעים השוואה בין הערכים באינדקס [3] ו [4] . היות ו 9 גדול מ 1 מבצעים החלפה ומקבלים :

אינדקס	[0]	[1]	[2]	[3]	[4]
ערך	7	2	8	1	9

נסכם את האיטרציה הראשונה :

- במערך בן 5 ערכים ביצענו 4 פעולות השוואה . ניתן לומר שבמערך בן n ערכים מבצעים n-1 פעולות השוואה.
- הערך הגדול במערך נמצא באינדקס האחרון של המערך.

איטרציה שנייה:

היות והערך הגדול נמצא באינדקס האחרון של המערך נבצע את האיטרציה עם 3 פעולות השוואה בלבד .

א. נשווה בין אינדקס [0] ואינדקס [1] . היות ו 7 גדול מ 2 נבצע החלפה בין הערכים ונקבל:

אינדקס	[0]	[1]	[2]	[3]	[4]
ערך	2	7	8	1	9

ב. נשווה בין אינדקס [1] ו [2] . היות ו 7 לא גדול מ 8 לא מבצעים כל החלפה.

ג. נשווה בין אינדקס [2] ואינדקס [3] . היות ו 8 גדול מ 1 נבצע החלפה ונקבל :

אינדקס	[0]	[1]	[2]	[3]	[4]
ערך	2	7	1	8	9

נסכם :

במערך של 5 הערכים ביצענו 3 השוואות (כמות הערכים פחות מספר האיטרציה 5-2=3) וכרגע באינדקס [3] יש את המספר השני הכי גדול. ניתן לומר שבמערך של n ערכים - באיטרציה השנייה עשינו n-2 השוואות .

איטרציה שלישית:

כרגע נעשה 2 פעולות השוואה בלבד.

א. נשווה את הערך של אינדקס [0] והערך של אינדקס [1] . היות ו 2 לא גדול מ 7 לא מבצעים כל פעולה.

ב. נשווה בין הערך של אינדקס [1] והערך של אינדקס [2] . היות ו 7 גדול מ 1 מבצעים החלפה ומקבלים:

אינדקס	[0]	[1]	[2]	[3]	[4]
ערך	2	1	7	8	9

נסכם:

במערך של 5 ערכים ביצענו 2 השוואות (כמות הערכים פחות האיטרציה 2= 3-5). באופן כללי ניתן לומר שבמערך של n ערכים מבצעים באיטרציה השלישית n-3 השוואות.

איטרציה רביעית:

כרגע נבצע השוואה בין הערך באינדקס [0] לערך באינדקס [1] . היות ו 2 גדול מ 1 נבצע החלפה ונקבל :

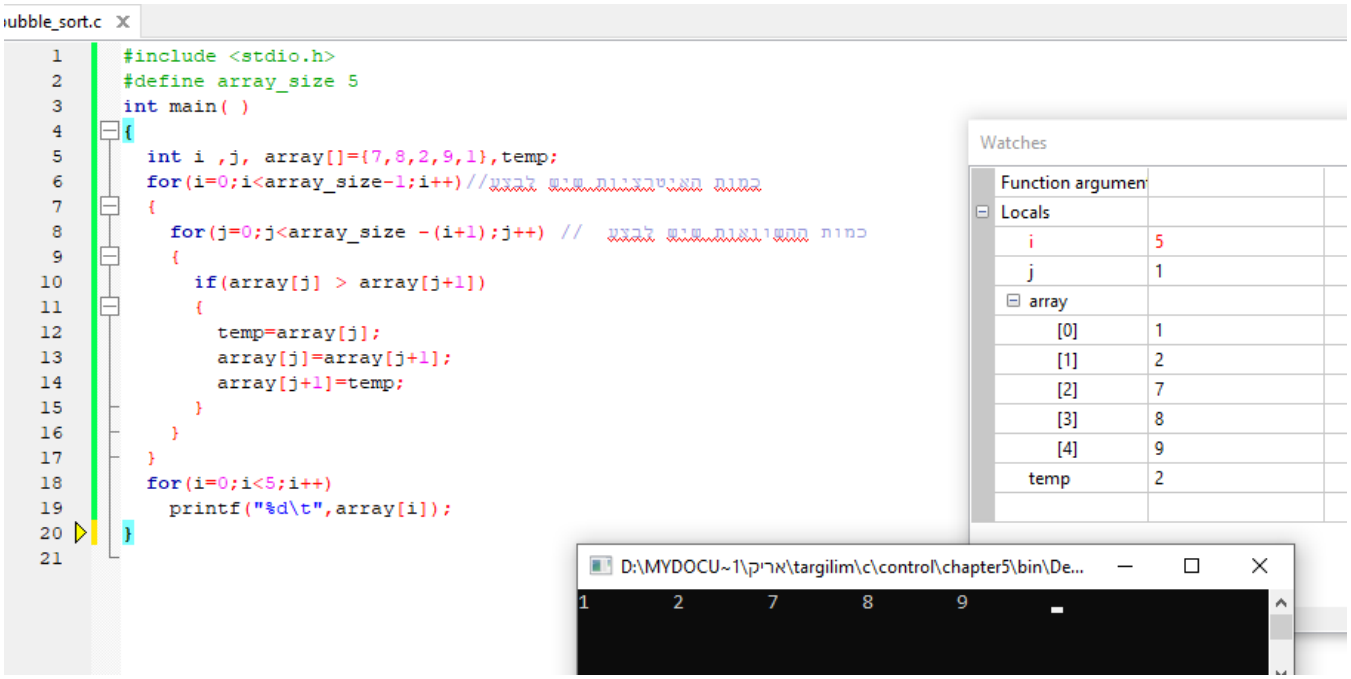
אינדקס	[0]	[1]	[2]	[3]	[4]
ערך	1	2	7	8	9

נסכם :

במערך של 5 ערכים ביצענו השוואה אחת (כמות הערכים פחות מספר האיטרציה 1=4-5) באופן כללי במערך של n ערכים ביצענו n-4 השוואות.

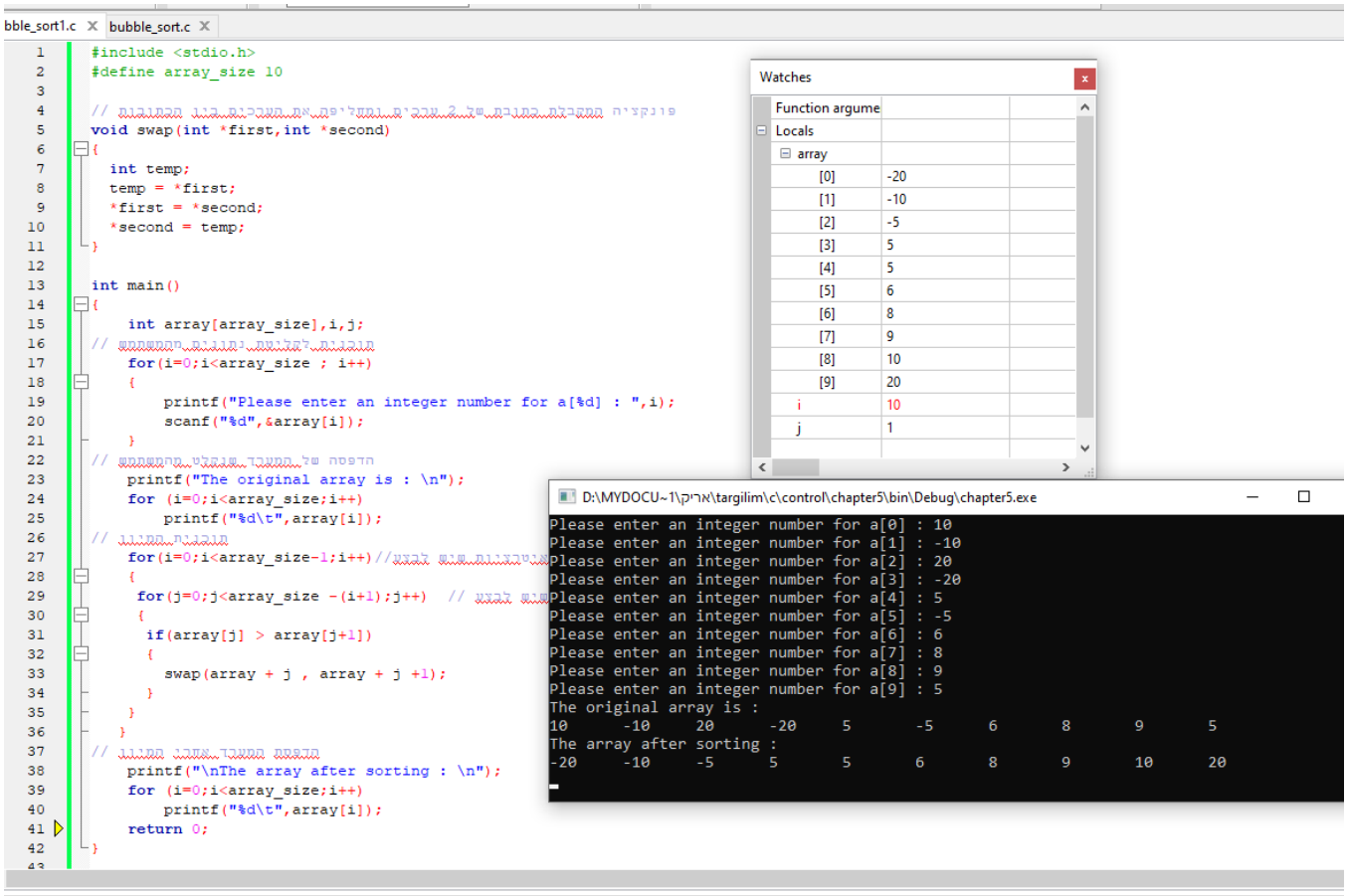
כרגע המערך ממוין . אין צורך באיטרציה חמישית כי אז כמות ההשוואות במערך של 5 ערכים תהיה 0 (0=5-5) ואין צורך לעשות השוואות.

התוכנית נראית כך :



איור 15 : תוכנית מיון בועות

נרשום תוכנית מיון כללית יותר שבה קולטים מהמשתמש 10 ערכים למערך ומבצעים עליו מיון מהקטן אל הגדול :



איור 16 : תוכנית מיון שבה קולטים מהמשתמש את הערכים למערך

את כמות האברים במערך ניתן לשנות בקלות על ידי שינוי הערך בשורה 2 של array_size .
כדי למיין את המערך מהערך הגדול אל הקטן נשנה בשורה 31 ובמקום לרשום if(array[j] > array[j+1]) נרשום if(array[j] < array[j+1]) .

לפעמים מקבלים מערך גדול שכבר ממוין או שלאחר כמה איטרציות המערך כבר ממוין . במקרה כזה אין טעם לעבור על כל האיטרציות כי המערך כבר ממוין. הפתרון הוא בדרך זו :

```
#include <stdio.h>
#define array_size 10

// פונקציה המקבלת כתובת של 2 ערכים ומחליפה את הערכים בין הכתובות
void swap(int *first,int *second)
{
  int temp;
  temp = *first;
  *first = *second;
  *second = temp;
}

int main()
{
   int array[array_size],i,j, flag;
// תוכנית לקליטת נתונים מהמשתמש
   for(i=0;i<array_size ; i++)
   {
      printf("Please enter an integer number for a[%d] : ",i);
      scanf("%d",&array[i]);
   }
// הדפסה של המערך שנקלט מהמשתמש
   printf("The original array is : \n");
   for (i=0;i<array_size;i++)
      printf("%d\t",array[i]);
// תוכנית המיון
   for(i=0,flag=1;i<array_size-1 && flag ; i++)      //כמות האיטרציות שיש לבצע
   {
```

```
    for(j=0,flag=0;j<array_size -(i+1);j++)          // כמות ההשוואות שיש לבצע
    {
     if(array[j] > array[j+1])
     {
       swap(array + j , array + j +1);
       flag=1;
     }
    }
   }
// הדפסת המערך אחרי המיון
   printf("\nThe array after sorting : \n");
   for (i=0;i<array_size;i++)
      printf("%d\t",array[i]);
   return 0;
}
```

הקטעים שצבועים בצבע צהוב הם התוספת לבדוק האם המערך ממוין. הוספנו משתנה בשם flag . בלולאת ה for הראשונה התחלנו אותו ב 1 ואילו בלולאת ה for השנייה התחלנו אותו ב 0 . אם בלולאת ה for השנייה לא קוראים לפונקציה swap המסקנה היא שלא בוצעה הפילו החלפה אחת והמערך ממוין. כאשר נסיים את ההשוואות ונחזור ללולאת ה for הראשונה ב flag=0 והתנאי לא מתקיים והסתיים המיון.

5.10.2 מיון Max - Max sort

גם כאן האלגוריתם המתבצע ב n-1 איטרציות.

גם כאן נמצא בכל איטרציה את האיבר הגדול ונעביר אותו למקום הנכון. העברה תתבצע על ידי החלפת המיקום של האיבר הגדול עם האיבר שבמיקום האחרון.

גם כאן כמות ההשוואות בכל איטרציה תלך ותקטן.

נניח שיש לנו את המערך הבא :

אינדקס	[0]	[1]	[2]	[3]	[4]
ערך	7	8	2	9	1

איטרציה ראשונה :

האיבר הגדול הוא 9 ונחליף את המיקום בין אינדקס [3] ו [4] ונקבל : (בין האינדקס בו המספר מקסימלי ואינדקס n-1) .

אינדקס	[0]	[1]	[2]	[3]	[4]
ערך	7	8	2	1	9

איטרציה שנייה:

עכשיו אחרי ששמנו את האיבר הגדול בסוף המערך נבדוק מי האיבר הבא עם הערך הגדול. האיבר הגדול הוא 8 . נחליף מיקום בין אינדקס [1] ו [3] ונקבל: (בין האינדקס בו המספר מקסימלי ואינדקס n-2).

אינדקס	[0]	[1]	[2]	[3]	[4]
ערך	7	1	2	8	9

איטרציה שלישית:

כעת 9 ו 8 יושבים בסוף המערך . נבדוק מיהו המספר הגדול : האיבר הגדול הוא 7 . נחליף מיקום [0] עם [2] ונקבל: (בין האינדקס בו המספר מקסימלי ואינדקס n-3).

אינדקס	[0]	[1]	[2]	[3]	[4]
ערך	2	1	7	8	9

איטרציה רביעית:

האיבר הגדול הוא 2 ונחליף בין המיקומים [0] ו [1] ונקבל : (בין האינדקס בו המספר מקסימלי ואינדקס n-4).

אינדקס	[0]	[1]	[2]	[3]	[4]
ערך	1	2	7	8	9

עכשיו המערך ממוין.

התוכנית נראית כך :

x_sort.c | bubble_sort.c

```c
#include <stdio.h>
#define array_size 5
// פונקציה המקבלת כתובת של 2 שבירים ומשנה אותם כאשר הכתובות בנו הפוכות
void swap(int *first,int *second)
{
  int temp;
  temp = *first;
  *first = *second;
  *second = temp;
}
// פונקציה בודקת מהו האינדקס עם הערך הגדול ביותר
int max_index(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;
}
// הפונקציה מוצאת פונקציה המוצאת את הערך הגדול ביותר במערך ואחרי כך הוראת לפונקציה המתל'פת במקום בנו הערך הגדול ביותר ak ythcr cnheuo
//  n - number of iteration
void max_sort(int a[], int n)
{
int length;
for(length = n ; length > 1; length--)
    {
      int i_max = max_index(a, length);
      swap(&a[length-1], &a[i_max]);
    }
}
int main()
{
    int ar[array_size]={7,8,2,9,1},i=array_size;
    printf("The original array before sorting\n");
    for(i=0;i<array_size; i++)
        printf("%d\t", ar[i]);
    max_sort (ar,i);
    printf("\nThe array after sorting\n");
    for(i=0;i<array_size; i++)
        printf("%d\t", ar[i]);
    return 0 ;
```

Watches

Function argumen	
Locals	
ar	
[0]	1
[1]	2
[2]	7
[3]	8
[4]	9
i	5

D:\MYDOCU~1\אריק\targilim\c\control\chapter5...

```
The original array before sorting
7       8       2       9       1
The array after sorting
1       2       7       8       9
```

תרגילים במערכים 5.11.1

1. רשום תוכנית שתגדיר מערך בן 5 איברים מטיפוס שלם ותקלוט אליו נתונים.

2. הגדר מערך בן 10 איברים מטיפוס שלם ובעזרת לולאה הכנס אליו מספרים מ 1 ועד 10 בסדר עולה.

3. הגדר מערך בן 10 מספרים מטיפוס שלם ובעזרת לולאה הכנס אליו מספרים מ 10 ועד 1 בסדר יורד.

4. הגדר מערך נוסף מטיפוס שלם בן 10 איברים והעבר אליו את המערך שבשאלה 3 .

5. הגדר 2 מערכים בני 5 איברים מטיפוס ממשי, קלוט לשני המערכים איברים. לאחר מכן בצע החלפה בין התכנים של שני המערכים.

6. הגדר מערך בן 5 איברים מטיפוס ממשי והעבר אליו את סכום כל שני איברים מתאימים במערכים של השאלה הקודמת.

7. הגדר מערך בן 8 איברים שיקלוט משכורת של 8 עובדים וידפיס את הגבוהה ביותר ואת הנמוכה ביותר. במידה והמספר הגבוה/הנמוך מופיעים כמה פעמים - ציין זאת .

8. הגדר מערך שאיבריו הם 2,4,6 ומערך שאיבריו הם 3,5,7 ומערך שלישי אשר איבריו יהיו מכפלת כל שני איברים בהתאמה.

9. רשום תוכנית שתקלוט ציונים של 25 תלמידים ותדפיס א. את הממוצע ב. כמה תלמידים נכשלו(מתחת ל-55) ג. מהו הציון הגבוה ביותר וכמה תלמידים קבלו אותו

10. הגדר מערך מטיפוס שלם בן 10 איברים ואתחל אותו כרצונך. רשום תוכנית שתמיין את המערך מהקטן אל הגדול.

11. הגדר מערך דו ממדי של מספרים בגודל 4*4 . קלוט לתוכו מספרים. חשב והדפס : א. סכום כל שורה ב. סכום כל עמודה ג. סכום של כל אחד מהאלכסונים .

12. הגדר 2 מערכים דו ממדיים של מספרים שלמים בגודל 4*4 . לאחד מהם קלוט מספרים. רשום תוכנית שתמלא את המערך השני כשהיא הופכת בין השורות והעמודות . כלומר השורה הראשונה תהפוך להיות העמודה הראשונה ולהיפך וכך הלאה.

13. רשום תוכנית עם פונקציות. התוכנית הראשית תכיל : **א.** פונקציה לקליטת מערך בן 10 איברים. **ב.** פונקציה המדפיסה את התפריט הבא : 1. מציאת המספר הגדול וכמה פעמים הוא מופיע. 2. מציאת המספר הקטן וכמה פעמים הוא מופיע. 3. סכום של איברי המערך. 4. בדיקה האם מספר שהמשתמש הקיש נמצא ואם כן כמה פעמים. 5. מיון המערך. 6. נא הקש את בחירתך. **ג.** משפט switch הבודק מהו המקש שהקיש המשתמש ולפי המקש קוראים לפונקציה הממלאת את בקשת המשתמש.

14. ריבוע קסם הוא ריבוע שבו סכום איברי כל שורה שווה לסכום איברי כל עמודה שווה לסכום כל אלכסון. קלוט מערך מטיפוס שלם בגודל 3*3 והוצא הדפסה האם הוא ריבוע קסם.

15. כתוב תוכנית המגדירה מערך של 10 איברים מטיפוס שלם, קולטת לתוכו מספרים ויוצרת מערך חדש מהערך הקטן אל הגדול כאשר במערך החדש אין ערכים שמופיעים מספר פעמים. לדוגמה : המערך המקורי : 5, 6, 5,10,9 . המערך החדש ייראה : 5,6,9,10 .

16. רשום תוכנית בעזרת פונקציות. הגדר 3 מערכים . שניים בני 8 איברים והשלישי בן 16 איברים. התוכנית קולטת ל 2 המערכים הראשונים נתונים מהמשתמש. (בעזרת פונקציה אחת כדוגמת השאלה הקודמת). לאחר מכן קרא לפונקציה שתמיין כל אחד מהמערכים. בשלב האחרון התוכנית תמזג את שני המערכים אל המערך שלישי שיהיה ממוין מהקטן לגדול .

5.11.2 תרגילים במחרוזות:

1. קלוט למחרוזת אחת שם פרטי ולאחריה קלוט למחרוזת נוספת שם משפחה וחברם למחרוזת אחת.
2. קלוט מחרוזת והדפס הודעה מתאימה אם אורכה קטן גדול או שווה ל 8
3. קלוט 2 מחרוזות ופלוט מי מהן ראשונה ע"פ סדר אלפביתי ומי ארוכה יותר.
4. קלוט מחרוזת ותו והדפס את מספר מופעי התו במחרוזת (כמה פעמים מופיע התו במחרוזת).
5. קלוט מחרוזת והפוך בין התו השני והשלישי והדפס את המחרוזת לאחר השינוי (לדוגמה במקום bla נקבל bal).
6. קלוט מחרוזת והחלף בין כל שני תווים סמוכים. לדוגמא במקום abcde נקבל badce
7. קלוט מחרוזת והדפס אותה בסדר הפוך.
8. קלוט מחרוזת בת 20 תווים. בדוק והדפס כמה תווים "קטנים" , כמה תווים "גדולים", כמה ספרות, כמה תווים לבנים וכמה מילים במחרוזת שקלטת. (תווים קטנים – a b c וכו. תווים גדולים – A B C D וכו' . ספרות – '0' עד '9' . תווים לבנים – כל שאר התווים).
9. קלוט מחרוזת ולאחריה מחרוזת קטנה יותר. בדוק האם המחרוזת הקטנה מוכלת בתוך הגדולה ואם כן כמה פעמים .
10. הגדר 2 מערכים. האחד מטיפוס תווי (דו ממדי - name[10][20]) והשני מטיפוס ממשי age[10]. הכנס למערך name את השם הפרטי ושם המשפחה ולמערך age את גילו של כל אחד מהשמות שהכנסת. לאחר מכן מצא והדפס מיהו הצעיר מבין השמות ומה הגיל שלו ומיהו המבוגר ומהו גילו. במידה ויש יותר מאחד באותו הגיל (צעיר או מבוגר) הדפס גם אותם.
11. הגדר 2 מחרוזות. האחת בגודל 31 והשנייה בגודל 11 . קלוט אליהן 2 מחרוזות האחת בגודל של עד 20 תווים והשנייה בגודל של עד 10 תווים. הכנס את המחרוזת הקטנה לתוך הגדולה החל מאינדקס מסוים שאותו תקלוט מהמשתמש ובסיום ההכנסה ימשיכו התווים של המחרוזת הגדולה. לדוגמא קלטנו למחרוזת הראשונה abcde ולשנייה abc . האינדקס שנקלט הוא 3 . בסיום המחרוזת הגדולה תיראה : abcabcde .
12. הגדר מחרוזת של 31 תווים. קלוט מהמשתמש מחרוזת (מספר התווים שהמשתמש יקיש לא ידוע אך הוא לא יעלה על 30). חלק את המחרוזת הנקלטת ל 2 מחרוזות באורך שווה, בדוק והדפס האם 2 המחרוזות שוות. (הערה : אם המשתמש הקיש כמות לא זוגית של תווים – ברור שהמחרוזות אינן שוות).
13. הגדר מחרוזת של 46 תווים. קלוט מהמשתמש מחרוזת (שלא תעלה על 45 תווים). חלק את המחרוזת הנקלטת ל 3 מחרוזות שוות באורכן. בדוק והדפס האם 3 המחרוזות שוות או 2 מהן שוות. (הערה – אם כמות התווים שהמשתמש הקיש איננה מתחלקת ב 3 אז חלק אותן בצורה לא שווה. לדוגמא אם המשתמש הקיש 16 תווים אז גודל המחרוזות יהיה 6 6 ו 4).
14. קלוט מהמשתמש 3 מחרוזות של עד 10 תווים . הגדר מחרוזת רביעית ושרשר את המחרוזות אחת בהמשך השנייה אל המחרוזת הרביעית.
15. קלוט מהמשתמש 2 מחרוזות בדוק והדפס האם 2 המחרוזות שוות.

עבודה נעימה