

פרק 9 – המחלקה מחרוזת (string)

בתוכניות שכתבנו עד כה השתמשנו בטיפוסים שונים המוגדרים בשפת C#: שלם, ממשי, תווי ובוליאני. יכולנו להגדיר משתנים מטיפוסים אלו ולבצע עליהם פעולות שונות (קלט, פלט, חישובים וכו'). עם זאת מרבית התוכניות שכתבנו התייחסו גם למחרוזות, סדרות של תווים. עד עתה השתמשנו במחרוזות כאשר רצינו להדפיס הודעות למשתמש ותחמנו אותן בגרשיים, למשל בהוראה:

```
Console.WriteLine("Enter two numbers:");
```

ההודעה "Enter two numbers:" היא מחרוזת.

הנה דוגמאות נוספות למחרוזות בשפת C#: "Hello, How Are You?", "453", "J", "" (האחרונה היא מחרוזת ריקה, שאינה מכילה אף תו).

בנוסף להדפסה נרצה לעתים להפעיל על מחרוזות פעולות נוספות. בפרק זה נראה כיצד ניתן לקלוט מחרוזות מהמשתמש, ולעבד אותן. אם עד עתה קלטנו מספרים או תווים בודדים בלבד, עכשיו נוכל לבקש מהמשתמש להקיש את שמו, את כתובתו וכדומה ולבצע פעולות שונות על המחרוזות הנקלטות.

9.1 היכרות ראשונית עם המחלקה string

בשפת C# מוגדרת מחלקה בשם `string` שבאמצעותה אפשר ליצור מחרוזות. משתנה מטיפוס מחרוזת אינו משתנה פשוט כמו `int` או כמו `char` אלא מופע (`unstance`) של המחלקה `string`.

הצהרה על משתנה מטיפוס מחרוזת מתבצעת כך:

```
string s1;
```

אמנם הצהרה על עצמים דומה להצהרה על משתנים מטיפוסים רגילים. אבל בשונה ממשתנים מטיפוסים רגילים, עלינו ליצור עצמים. יצירת עצם כוללת הקצאת שטח בזיכרון עבור העצם ואת אתחולו. עבור משתנים רגילים, הקצאת מקום בזיכרון נעשית אוטומטית עם ההצהרה עליהם ואילו עבור עצמים הקצאת הזיכרון והאתחול מתבצעים באמצעות הפעולה `new`. מכיוון שהשימוש במחרוזות נפוץ כל כך, שפת C# מאפשרת ליצור ולאתחל מחרוזות בצורה ישירה ללא שימוש בפעולה `new`, למשל כך:

```
s1 = "";
```

פעולה זו יוצרת עצם בשם `s1` שמפנה לשטח בזיכרון המכיל מחרוזת מחרוזת ריקה "". אם נרצה לאתחל את העצם בערך כלשהו, נוכל לעשות זאת בציון הערך הרצוי בזמן היצירה, למשל כך:

```
s1 = "abc";
```

כעת העצם `s1` מפנה לשטח בזיכרון המכיל את המחרוזת "abc".

בדומה למשתנים פשוטים, אפשר לאחד את ההצהרה ואת האתחול כך:

```
string s1 = "hello";
```

```
string s2 = "good morning";
```

בהמשך הפרק נכיר פעולות שונות שאפשר לבצע על עצמים מטיפוס `string`.

בשפת C# מחרוזת היא עצם של המחלקה `string`.

```
הצהרה על מחרוזות מתבצעת באופן דומה להצהרה על משתנים רגילים, למשל:  
string s1;  
כדי שניתן יהיה להשתמש בעצם מהמחלקה מחרוזת, יש ליצור את העצם. יצירת העצם  
כוללת הקצאת שטח בזיכרון ואתחול העצם.  
יצירת העצם נעשית באמצעות השמה פשוטה של סדרת תווים בגרשיים:  
string s2 = "good morning";
```

קצ'ה 1

מטרת הבעיה ופתרונה: הצגת קלט של מחרוזות.

```
פתחו וישמו אלגוריתם שיקבל כקלט את שמכם ויציג: Your name is ומיד אחר כך את השם  
שנקלט.
```

בחירת משתנים

? באיזה טיפוס נשתמש כדי לשמור את השם שנקלט?

כיוון שעלינו לקלוט מילה נשתמש במחלקה מחרוזת. אם כך רשימת המשתנים תכלול את העצם
name מהמחלקה מחרוזת.

האלגוריתם:

1. קלוט מהקלט name-2
2. הציג כקלט "Your name is:"
3. הציג כקלט name

יישום האלגוריתם

עד כה ראינו כיצד לקלוט ערכים מטיפוסים פשוטים – שלמים, ממשיים ותווים. קליטת מחרוזת
נעשית בצורה דומה. את הוראה 1 נוכל לממש במשפטים:

```
name = Console.ReadLine();
```

בניגוד להוראות קלט אחרות שהשתמשנו בהן, הפעם לא נשתמש בפעולה נוספת (למשל
ב-`int.Parse`) כדי לפרש את הקלט ולתרגמו לערך מטיפוס מסוים, אלא נקלוט את שורת
המחרוזת שיקליד המשתמש ונשמור אותה כמו שהיא.

הפעולה `ReadLine` קולטת את כל התווים עד סוף השורה ומחזירה עצם מסוג מחרוזת. למשל
אם המשתמש הקליד:

```
my name is Moshe
```

העצם name יכיל את המחרוזת: "my name is Moshe"

התוכנית המלאה

```
/*  
קלט: שם, הנקלט כמחרוזת  
פלט: השם, מלווה בהודעה מקדימה  
*/  
using System;
```

```

public class YourName
{
    public static void Main ()
    {
        string name; // הצהרה על עצם מהמחלקה מחרוזת
        // הוראת קלט + הקצאת מקום בזיכרון
        Console.WriteLine("Enter your name: ");
        name = Console.ReadLine();
        // פלט
        Console.WriteLine("your name is: {0}", name);
    } // Main
} // class YourName

```

שימו ♥: כמו פעולות קלט של ערכים מטיפוסים פשוטים, גם פעולת קלט למחרוזת היא למעשה פעולה המחזירה את הערך הנקלט. הפעולה קולטת מחרוזת מהקלט, ויוצרת עצם חדש מהמחלקה מחרוזת. פעולת הקלט מקצה עבורו מקום בזיכרון ושומרת בו את מחרוזת הקלט. הפעולה `Console.ReadLine()`, מחזירה הפניה למחרוזת החדשה שהוקצתה. בעת ביצוע ההוראה:

```
name = Console.ReadLine();
```

אנו מבצעים השמה של המחרוזת הנקלטת במשתנה `name`. בעקבות ההשמה מִפְנֵה `name` אל אותו שטח זיכרון שהוקצה על ידי פעולת הקלט.

סוף פתרון קציה 1

ניתן לקלוט מחרוזת באמצעות הפעולה `Console.ReadLine`. פעולת הקלט מקצה שטח בזיכרון עבור המחרוזת החדשה.

9.2 תכונות של עצם ותכונת האורך של מחרוזת

לכל מחרוזת יש אורך. למשל אורכה של המחרוזת "abc" הוא 3, ואורכה של המחרוזת הריקה "" הוא 0. ניתן לומר כי אורך הוא אחד המאפיינים למחרוזת מסוימת, ולכן הוא מוגדר כ**תכונה** של המחרוזת. וכך עבור עצמים מהמחלקה מחרוזת מוגדרת התכונה `Length` השומרת לכל מחרוזת את אורכה.

אם `s1` הוא עצם מהמחלקה מחרוזת, אז ערכו של הביטוי `s1.Length` הוא אורכה של המחרוזת `s1`.

שימו ♥: את התכונה `Length` ניתן לקרוא בלבד. לא ניתן לשנות את ערכה בהשמה. ניסיון לשנות את ערך התכונה יגרום לשגיאת הידור.

קציה 2

מטרת הבעיה ופתרונה: הדגמת השימוש בתכונה של מחרוזת. הכרת התכונה `Length` ואופן הגישה לתו בתוך מחרוזת.

פתחו וישמו אלגוריתם אשר יקבל כקלט מחרוזת. אם אורך המחרוזת הוא זוגי יוצג כפלט התו הראשון במחרוזת ואם אורך המחרוזת הוא אי-זוגי יוצג התו האחרון במחרוזת. למשל, עבור הקלט `shalom` יוצג התו: 's'. ועבור הקלט `dog` יוצג התו: 'g'.

שימו ♥: בשפת C# מיקום התווים במחרוזת מתחיל ב-0. כלומר, התו הראשון במחרוזת נמצא במקום 0, התו השני במחרוזת הוא במקום 1, השלישי במקום 2 וכן הלאה. **?** אם התו הראשון נמצא במקום 0, באיזה מקום נמצא התו האחרון?
 נתבונן למשל במחרוזת "abcd" שאורכה 4. התו האחרון בה הוא 'd' והוא נמצא במקום 3.

מיקום התווים במחרוזת מתחיל ב-0.
 בכל מחרוזת, אם נסמן את אורך המחרוזת ב-len אז התו האחרון במחרוזת נמצא במקום len-1.

בחירת משתנים

נשתמש במשתנים הבאים:
 str – עצם מהמחלקה מחרוזת
 letter – האות המבוקשת, מטיפוס char
 len – לשמירת אורך המחרוזת, מטיפוס שלם

האלגוריתם

1. קלוט אגורא str-2
2. השם אג אורכה של האגורא str len-2
3. אס len צאי
- 3.1 השם letter-2 אג הגו הנמצא במקום 0 str-2
4. אגרא
- 4.1 השם letter-2 אג הגו הנמצא במקום len-1 str-2
5. הצג כפוט אג ערכו של letter

יישום האלגוריתם

? כדי לבדוק את אורכה של המחרוזת נשתמש בתכונה Length, כיצד נדע מהו התו שנמצא במקום מסוים במחרוזת?

פנייה לתו מסוים במחרוזת נעשית בסוגריים מרובעים []. למשל, כדי לפנות אל התו שנמצא במקום 0 במחרוזת str נכתוב str[0]. התו שנמצא במקום האחרון במחרוזת str הוא str[str.Length-1].

שימו ♥: את התווים שבמחרוזת ניתן לקרוא בלבד. לא ניתן לשנות את ערכם בהשמה. ניסיון לשנות את ערך התו יגרום לשגיאת הידור.

שימו ♥: פנייה לתו שאינו בתחום המחרוזת יגרום לחריגה במהלך ריצת התכנית.

התוכנית המלאה

```
/*
קלט: מחרוזת
פלט: התו הראשון או האחרון במחרוזת, בהתאם לאורכה (זוגי או אי-זוגי)
*/
using System;
public class IsEven
{
    public static void Main()
```

```

{
    string str;
    int len;
    char letter;
    Console.Write("Enter a string: ");
    str = Console.ReadLine();
    len = str.Length;
    if (len % 2 == 0)
        letter = str[0];
    else
        letter = str[len - 1];
    Console.WriteLine("The letter is: {0}", letter);
} // Main
} // class IsEven

```

סוף פתרון בעיה 2

נסכם את המושג המרכזי שהוצג בפתרון בעיה 2:

במחלקה מוגדרות **תכונות** שהן משתנים המוגדרים עבור עצמים מהמחלקה.
 כדי לגשת לערך התכונה כותבים כך:

שם התכונה.שם העצם

התכונה Length מוגדרת לעצמים מהמחלקה מחרוזת. התכונה שומרת את אורך המחרוזת. ערכה של התכונה אינו ניתן לשינוי.

שאלה 9.1

פתחו וישמו אלגוריתם שמקבל כקלט רשימה של 100 מחרוזות. האלגוריתם מחשב בכמה מחרוזות מתוך ה-100 התו הראשון שווה לתו האחרון, ומציג את הערך שחישב.

שאלה 9.2

פתחו אלגוריתם שיקבל כקלט מחרוזת ותו. פלט האלגוריתם יהיה מספר ההופעות של התו הנתון בתוך המחרוזת הנתונה. ישמו את האלגוריתם בשפת C#.

הדרכה: מאחר שניתן לברר את אורך המחרוזת הנתונה ניתן להשתמש בהוראה לביצוע-חוזר שמספר הסיבובים בה ידוע מראש (לולאת for).

9.3 ביצוע פעולות על מחרוזות

כדי לבצע פעולות על עצמים יש לרשום את שם העצם, אחריו את סימן הנקודה ולאחר מכן את שם הפעולה. למשל, כדי להגריל מספר הפעלו את הפעולה Next על עצם מסוג Random כך:

```

Random rnd = new Random();
int num;
num = rnd.Next(100);

```

במקרה זה הפעולה Next מחזירה מספר אקראי בין 0 ל-99 הנשמר במשתנה num.

באופן דומה אפשר להפעיל פעולות שונות על עצמים מסוג מחרוזת. אחת הפעולות המוגדרות עבור מחרוזות היא הפעולה CompareTo המשווה מחרוזת למחרוזת נוספת. אם s1 הוא עצם

מהמחלקה מחרוזת, ו-s2 הוא עצם נוסף מהמחלקה מחרוזת, אז הביטוי `s1.CompareTo(s2)` מפעיל את הפעולה `CompareTo` המשווה את `s1` למחרוזת נוספת (`s2`) שהתקבלה כפרמטר.

הפעולה `CompareTo` מחזירה את תוצאת השוואה כערך שלם באופן הבא:

- ◆ אם המחרוזות שוות יוחזר הערך 0.
- ◆ אם המחרוזת `s1` שמופעלת עליה הפעולה, קודמת למחרוזת `s2` שהתקבלה כפרמטר בסדר מילוני, יוחזר מספר שלם שלילי.
- ◆ אם המחרוזת `s1` שמופעלת עליה הפעולה, מופיעה אחרי המחרוזת `s2` שהתקבלה כפרמטר בסדר מילוני, יוחזר מספר שלם חיובי.

הסוגריים שאחרי שם הפעולה נועדו להעברת ערכים (פרמטרים) שהפעולה נזקקת להם. במקרה של הפעולה `CompareTo` מועבר פרמטר מסוג מחרוזת. בהמשך הפרק נכיר פעולות נוספות של המחלקה מחרוזת.

כיצד 3

מטרת הבעיה ופתרונה: הדגמת השימוש בפעולה להשוואת מחרוזת במחלקה `string`, ושילוב מחרוזות בתבנית מנייה.

בסיום בחינות הקבלה למקהלת הזמר העירונית יוצא הבוחן הראשי וקורא את רשימת המועמדים שהתקבלו על פי סדר הא"ב. מועמד אשר שומע את שמו יודע כי התקבל ללהקה. מועמד ששמו עדיין לא נקרא, אשר שומע שם אחר שמופיע אחריו בסדר הא"ב, יודע כי לא התקבל ללהקה.

פתחו אלגוריתם אשר הקלט שלו הוא שם מועמד, ולאחר מכן רשימת שמות שקורא הבוחן על פי סדר הא"ב. פלט האלגוריתם הוא אם המועמד התקבל, ומספר השמות שהיה עליו לשמוע לפני שהגיע למסקנה אם התקבל או לא. ישמו את האלגוריתם בשפת `C#`.

פירוק הבעיה לתת-משימות

למעשה האלגוריתם צריך להשוות מחרוזות ובנוסף לבצע מנייה. אלגוריתם זה דומה לאלגוריתמים שפיתחנו בפרקים קודמים. אם כך, נפרק את הבעיה לתת-משימות הבאות:

1. קליטת שם המועמד
2. קליטת שמות המועמדים שהתקבלו והשוואתם לשם המועמד
3. מניית מספר השמות שנקלטו
4. הצגה אם המועמד התקבל והצגת מספר השמות שהיה עליו לשמוע

תת-משימה 3 תבוצע בדומה למשימות מנייה אחרות, ההבדל הוא כמובן בטיפוס הערכים הנמנים – מחרוזת במקרה זה – ובאופן ביצוע הפעולות עליהם. בשלב יישום האלגוריתם נראה כיצד לעשות זאת.

בחירת משתנים

name – שם המועמד, עצם מהמחלקה מחרוזת
str – השם התורן מהקלט, עצם מהמחלקה מחרוזת
counter – למניית מספר המחרוזות, מטיפוס שלם

האלגוריתם

1. אגף אג counter 1-2
2. קאוט אג/אג 2-name
3. קאוט אג/אג 2-str
4. כו ע/ו name > str (כסדר מלוי) כצע:
 - 4.1. העצא אג ערכו ל counter 1-2
 - 4.2. קאוט אג/אג 2-str
5. אס name = str
 - 5.1. העצא כפוט "המועמד <name> העקבא"
6. אגרא
- 6.1. העצא כפוט "המועמד <name> לא העקבא"
7. העצא אג ערכו ל counter

יישום האלגוריתם

? כיצד נבדוק אם שם המועמד גדול על-פי סדר מילוני מהמחרוזת הנקלטת?

כאמור, הפעולה CompareTo בודקת את היחס המילוני שבין שתי מחרוזות. אם שם המועמד name גדול מהמחרוזת הנקלטת str, הפעולה name.CompareTo(str) תחזיר ערך שלם חיובי, ואילו אם שם המועמד קטן מהמחרוזת הנקלטת, פעולה זו תחזיר ערך שלם שלילי. אם המחרוזות שוות הפעולה תחזיר 0.

? כיצד נבדוק אם המחרוזת האחרונה שנקלטה שווה לשם המועמד?

אפשר להשתמש שוב בפעולה CompareTo ולבדוק אם היא מחזירה את הערך 0. לחילופין, אפשר לבדוק שוויון בין שתי מחרוזות באמצעות הפעולה Equals הבודקת שוויון של המחרוזות שעליה היא מופעלת למחרוזת המתקבלת כפרמטר. למשל, הביטוי name.Equals(str) יחזיר true אם המחרוזת str שווה למחרוזת name, אחרת יחזיר false. בנוסף לכך, בשפת C# אפשר לבדוק שוויון בין שתי מחרוזות כפי שבודקים שוויון בין שני משתנים מאותו הסוג: על ידי פעולת השוואה == או != (לא שווה). אם כך, את הביטוי הבוליאני שבהוראה 5 אפשר ליישם כך:

```
name == str
```

שימו ♥: את המונה נאתחל ב-1 כיוון שמחוץ ללולאה אנו קולטים את השם הראשון שנקרא.

התוכנית המלאה

```
/*
קלט: שם מועמד ורשימה ממוינת של שמות
פלט: האם המועמד התקבל ומספר השמות שהיה עליו לשמוע
*/
using System;
public class NameFinder
{
    public static void Main ()
    {
        // הגדרת משתנים ואתחולם
        string name;
        string str;
        int counter = 1;
```

```

// קלט שם מבוקש
Console.WriteLine("Enter the candidate name: ");
name = Console.ReadLine();
// לולאת זקיף
Console.WriteLine("Enter first winner name: ");
str = Console.ReadLine();
// כל עוד לא עברנו את שם המועמד
while (name.CompareTo(str) > 0)
{
    counter++;
    Console.WriteLine("Enter next winner name: ");
    str = Console.ReadLine();
} // while
if (name == str)
    Console.WriteLine("{0} is accepted", name);
else
    Console.WriteLine("{0} is not accepted", name);
Console.WriteLine("{0} Names", counter);
} // Main
} // class NameFinder

```

שימו ♥ : לולאת ה-**while** ממשיכה כל עוד מחרוזות הקלט קטנות משם המועמד. קליטת מחרוזת שווה לשם המועמד או גדולה ממנה גורמת ליציאה מהלולאה. לכן, אחרי הלולאה עלינו לבדוק את סיבת היציאה מהלולאה: האם המחרוזת האחרונה שנקלטה שווה לשם המועמד או גדולה ממנה.

סוף פתרון בעיה 3

בסוף הפרק תוכלו למצוא טבלה המפרטת את הפעולות השכיחות המוגדרות ב-C# עבור עצמים מהמחלקה `string`.

העמודה הימנית בטבלת הפעולות מתארת את שם הפעולה ואת רשימת הפרמטרים שהיא מצפה לקבל (בתוך סוגריים). כפי שניתן לראות, יש פעולות בטבלה כגון `ToLower` אשר אינן מצפות לקבל ערך כלשהו, ולכן הסוגריים ריקים. לעומתן, הפעולה `Equals` מצפה לקבל פרמטר אחד שהוא עצם מהמחלקה מחרוזת, והפעולה `IndexOf` מצפה לקבל עצם מהמחלקה מחרוזת או לקבל תו.

העמודה השלישית מתארת את טיפוס הערך המוחזר כתוצאה מהפעלת הפעולה. בדומה לפעולות המתמטיות שלמדנו בפרק 4, גם כאן יש לשים לב לטיפוס הערך המוחזר ולוודא התאמה בינו ובין הפעולות שנבצע עליו. כלומר כאשר נשים בתוך משתנה ערך שמוחזר מפעולה, עלינו לוודא התאמה בין טיפוס הערך המוחזר לטיפוס המשתנה שההשמה תבצע בו.

מחלקות מגדירות אוסף של פעולות שאפשר להפעיל על עצמים מהמחלקה.

הפעלת פעולה נכתבת כך (משמאל לימין):

(פרמטרים) שם הפעולה. שם העצם

יש לבדוק היטב בתיאור הפעולה אילו ערכים היא מצפה לקבל ומה טיפוס הערך שהיא מחזירה.

שאלה 9.3

פתחו אלגוריתם המקבל כקלט רשימת מחרוזות המסתיימת במחרוזת "***". פלט האלגוריתם יהיה אורך המחרוזת הארוכה ביותר ברשימה. ישמו את האלגוריתם בשפת C#.

שאלה 9.4

פתחו אלגוריתם שיקבל כקלט שתי מחרוזות ויציג אותן לפי סדר הא"ב. ישמו את האלגוריתם בשפת C#.

שאלה 9.5

פתחו אלגוריתם שהקלט שלו הוא שלוש מחרוזות, והפלט שלו הוא שלוש המחרוזות לפי סדר מילוני. ישמו את האלגוריתם בשפת C#. למשל עבור הקלט: apple today good הפלט המתאים הוא: apple good today.

שרשור מחרוזות

פעולה נוספת ושימושית מאוד על מחרוזות היא פעולת השרשור. פעולה זו מקבלת שתי מחרוזות ויוצרת מחרוזת חדשה, המורכבת מהמחרוזות הראשונה ואחריה מוצמדת המחרוזת השנייה. הפעולה אינה משנה את המחרוזות המקוריות.

בשפת C# מתבצע שרשור באמצעות סימן הפעולה +. כלומר, אם s1 ו-s2 הן מחרוזות, אז כדי לשרשר אותן זו לזו נכתוב את הביטוי s1+s2. למשל, אם ב-s1 נמצאת המחרוזת "he" וב-s2 נמצאת המחרוזת "llo", אז s1+s2 היא המחרוזת החדשה "hello".

שימו ♥ : פעולת השרשור אינה פעולה של המחלקה מחרוזות ולכן אופן הפעלתה שונה מזה של פעולות כמו Equals, ToUpper או כמו פעולות אחרות השייכות למחלקה. היא אינה מופעלת על עצם מהמחלקה, ולכן אין שימוש בסימן הנקודה. אופן הפעלת פעולת השרשור על מחרוזות דומה לאופן הפעלת פעולות מתמטיות על טיפוסים רגילים בשפה.

ניתן לשרשר למחרוזות ערכים מטיפוסים שונים. למשל הביטוי:

```
"The number is " + number
```

הוא ביטוי חוקי. אם, למשל, ערכו של המשתנה number הוא 3, אז ערך הביטוי הוא המחרוזת הנוצרת משרשור של המחרוזת "3" למחרוזת "The number is: ". המחרוזת החדשה הנוצרת עקב השרשור אם כך היא: "The number is: 3".

שימו ♥ : מאחר שבמשתנה number יש ערך מטיפוס שלם יש להמיר אותו למחרוזת. ואכן טרם השרשור התבצעה פעולת המרה של הערך המספרי למחרוזת המתאימה לו באופן אוטומטי. כך קורה תמיד כאשר נשרשר ערך מטיפוס שאינו מחרוזת למחרוזת.

הפעולה + משמשת לשרשור מחרוזות באופן הבא: מחרוזת1 + מחרוזת2. פעולת השרשור יוצרת מחרוזת חדשה ומקצה עבורה מקום. היא אינה משפיעה על המחרוזות המקוריות. כאשר משרשרים למחרוזת ערך שאינו מחרוזת, הוא מומר תחילה למחרוזת ואז מתבצעת פעולת השרשור.

שאלה 9.6

פתחו אלגוריתם שיקבל כקלט 100 מחרוזות. עבור כל זוג מחרוזות תוצג כפלט מחרוזת שהיא שרשור של שתי המחרוזות עם הסימן '@' ביניהן (כלומר בסך הכול תוצגנה 50 מחרוזות). ישמו את האלגוריתם בשפת C#.

ומה בפנים? פעולות המסתכלות אל תוך הקנקן

המחלקה מחרוזות מגדירה פעולות שונות המאפשרות להסתכל פנימה לתוך המחרוזות ולהתייחס לתווים המרכיבים אותה. פעולות כאלו מסייעות מאוד בפתרון בעיות שונות, כפי שמדגימה הבעיה הבאה:

קצ'ה 4

מטרת הבעיה ופתרונה: הדגמת השימוש בפעולות המחלצות מידע מתוך מחרוזת.

פתחו אלגוריתם שהקלט שלו הוא מחרוזת המהווה משפט באנגלית, והפלט הוא האות האחרונה של המילה הראשונה במשפט ומספר המילים במשפט. למשל, עבור הקלט: Welcome to Israel and have a nice day! הפלט המתאים הוא 8 e. ישמו את האלגוריתם בשפת C#. אפשר להניח שהמשפט מכיל לפחות מילה אחת.

ניתוח הבעיה בעזרת דוגמאות

נתבונן במשפט שניתן כדוגמה: Welcome to Israel and have a nice day!.

? כיצד אפשר לדעת כמה מילים יש במשפט הנתון?

כיוון שלפני כל מילה פרט למילה הראשונה יש רווח נוכל למנות את מספר הרווחים במחרוזת ולהוסיף 1. למשל, במשפט Welcome to Israel and have a nice day! יש 7 רווחים, ולכן 8 מילים. (זאת בהנחה שבין מילה למילה יש רווח אחד בלבד!)

? מהי האות האחרונה במילה הראשונה?

במקרה שהמשפט מכיל מילה אחת בלבד, האות האחרונה במילה הראשונה היא האות האחרונה במחרוזת. בכל שאר המקרים, האות האחרונה במילה הראשונה היא האות שנמצאת מיד לפני הרווח הראשון.

פירוק הבעיה לתת-משימות

לשם פתרון הבעיה נפתור את התת-משימות הבאות:

1. קליטת המחרוזת
2. מניית מספר הרווחים במחרוזת שנקלטה
3. חישוב מספר המילים במחרוזת
4. מציאת התו האחרון של המילה הראשונה במחרוזת
5. הצגה של מספר המילים במחרוזת ושל התו שנמצא בתת-משימה 4

בחירת משתנים

sentence – מחרוזת לשמירת המשפט שנקלט
numOfSpaces – שלם, מונה את מספר הרווחים במשפט
numOfWords – שלם, שומר את מספר המילים במשפט
placeOfLastLetter – שלם, שומר את מיקומו של התו האחרון במילה הראשונה

יישום האלגוריתם

כדי למנות את מספר הרווחים במחרוזת, נעבור על כל תו במחרוזת ונבדוק אם הוא שווה לרווח. לצורך בדיקת התו במיקום ה-*i* נשתמש בסימון [i] המחזיר את התו במקום המבוקש:

```
if (sentence[i]==' ')
```

שימו ♥: הפנייה לתו במחרוזת מחזירה ערך מטיפוס **char** ולא מטיפוס מחרוזת, ולכן יש להשוות את הערך המוחזר לתו רווח ' ' (ולא למחרוזת המכילה את התו רווח " ").

כדי למצוא את התו האחרון של המילה הראשונה נבדוק את מספר המילים במחרוזת. אם המחרוזת מורכבת ממילה אחת בלבד נחלץ את התו במקום האחרון, אחרת נחלץ את התו שנמצא לפני הרווח הראשון.

כדי למצוא את מיקומו של הרווח הראשון אפשר להשתמש בפעולה `IndexOf` המקבלת תו ומחזירה מספר שלם המייצג את מקומו במחרוזת:

```
placeOfLastLetter = sentence.IndexOf(' ') - 1;
```

טבלת הפעולות בסוף הפרק מתארת גרסה נוספת של פעולה זו המקבלת מחרוזת ומחזירה את מיקומה בתוך המחרוזת שעליה מופעלת הפעולה.

שימו ♥: יש תמיד לדאוג להתאמה בין טיפוסים הערכים שאנו משתמשים בהם ובין אלו המתוארים בכותרת הפעולה.

התוכנית המלאה

```
/*  
קלט: משפט באנגלית  
פלט: האות האחרונה במילה הראשונה ומספר המילים במשפט  
*/  
using System;  
public class HowManyWords  
{  
    public static void Main ()  
    {  
        string sentence;  
        int placeOfLastLetter;  
        int numOfSpaces = 0;  
        int numOfWords;  
        Console.WriteLine("Enter a sentence, with exactly one " +  
            " space between words: ");  
        sentence = Console.ReadLine();  
        // מניית הרווחים במשפט  
        for (int i = 0 ; i < sentence.Length ; i++)  
            if (sentence[i] == ' ')  
                numOfSpaces++;  
    }  
}
```

```

numOfWords = numOfSpaces + 1;
// מציאת מקומו של הרווח הראשון
if (numOfWords == 1)
    placeOfLastLetter = sentence.Length - 1;
else
    placeOfLastLetter = sentence.IndexOf(' ') - 1;
Console.WriteLine("The last letter of the first word is: {0}"
    , sentence[placeOfLastLetter] );

// כמה מילים בחשפט
Console.WriteLine("There are {0} words", numOfWords);
} // Main
} // class HowManyWords

```

סוף פתרון בעיה 4

שאלה 9.7

פתחו וישמו אלגוריתם שהקלט שלו הוא תו ומחרוזת והפלט הוא הודעה אם התו נמצא במחרוזת או לא.

שאלה 9.8 (מתוך בגרות 1995)

פתחו וישמו אלגוריתם הקולט מחרוזת. האלגוריתם מציג כפלט כל תו במחרוזת פעמיים, פרט לתו ' * ' (כוכבית). גם אם התו כלול במחרוזת הוא לא מוצג כלל. הפלט מוצג בשורה אחת. למשל עבור הקלט: $AB*3B*?$ הפלט יהיה $AABB33BB??$.

שאלה 9.9

פתחו וישמו אלגוריתם הקולט מחרוזת. האלגוריתם מציג כפלט את המחרוזת ללא אותיות זהות צמודות. למשל עבור הקלט: apple הפלט יהיה: apple, עבור הקלט: Yellow balloon הפלט יהיה: Yelow balon ועבור הקלט: abbba הפלט יהיה: aba.

שאלה 9.10

פתחו וישמו אלגוריתם שיקבל כקלט מילה באנגלית ויציג אותה פעמיים, פעם באותיות גדולות ופעם באותיות קטנות. למשל עבור הקלט Memory הפלט יהיה: MEMORY ומיד אחר-כך memory. חפשו פעולות מתאימות בטבלת הפעולות המופיעה בסוף הפרק.

9.4 הוראת השמה במחרוזות

עד כה ראינו כיצד לבצע פעולות קלט ופלט של מחרוזות, פעולת שרשור ופעולות שונות שמחזירות מידע על המחרוזת. בעיבוד של משתנים רגילים, אחת מהפעולות השימושיות ביותר היא הוראת השמה. בסעיף זה נדון בהשמה עבור מחרוזות, כלומר בהשמה של מחרוזת במשתנה שטיפוסו מחרוזת.

למעשה, כפי שכבר אמרנו, גם בעת קליטת מחרוזת אנו מבצעים השמה של המחרוזת הנקלטת, בעצם מסוג מחרוזת. למשל, בהוראה:

```
str = Console.ReadLine();
```

הפעולה `Console.ReadLine`, מחזירה הפניה לשטח הזיכרון החדש שהוקצה עבור מחרוזת הקלט. בעקבות ההשמה, המשתנה `str` מפנה אל אותו שטח זיכרון.

בסעיף זה נראה דוגמאות נוספות להשמה של מחרוזות.

הצ'יה 5

מטרת הבעיה הבאה: הצגת השמת מחרוזות למחרוזת והדגמה נוספת של השימוש בפעולות של המחלקה `string`.

נאמר שכתובת דואר אלקטרוני היא חוקית אם היא מקיימת את התנאים הבאים:
מתחילה באות אנגלית, מורכבת מרצף לא ריק של תווים ואחריו מגיע התו '@', אחריו שוב רצף לא ריק של תווים, אחריו התו '.' ולאחריו עוד רצף לא ריק של תווים. כתובת ישראלית מסתיימת במחרוזת ".il".

בקביעת חוקיות של כתובת דואר אין משמעות להבדל בין אותיות גדולות לקטנות.

למשל, המחרוזת `d@ccv.hhh` היא כתובת דואר אלקטרוני חוקית, וכך גם `t@ii.il`, שהיא כתובת ישראלית. לעומתן, המחרוזות הבאות אינן כתובות חוקיות: `ח@ח.ע.ע` (לא מתחילה באות אנגלית), `tr@.il` (רצף התווים שבין התו '@' לבין התו '.' הוא ריק), `tt@jjj` (לא מכילה את התו '.') ו-`sda.asd@sad` (התו '.' מופיע לפני התו '@').

פתחו אלגוריתם המקבל כקלט מחרוזת. ניתן להניח כי במחרוזת המתקבלת התו '.' לא מופיע יותר מפעם אחת וכך גם התו '@'. האלגוריתם מציג כפלט את המחרוזת שנקלטה, בצירוף הודעה המבהירה אם המחרוזת מהווה כתובת דואר אלקטרוני חוקית, ואם כן, אם זוהי כתובת ישראלית.

ישמו את האלגוריתם בשפת התכנות `C#`.

ניתוח הבעיה בעזרת דוגמאות

הכתובת `tr@xxx.il` היא חוקית וישראלית. גם הכתובת `tr@xxx.il` היא חוקית וישראלית. אין משמעות לשימוש באותיות קטנות או גדולות – ההבדל ביניהן אינו משפיע על קביעת חוקיות המחרוזת וגם לא על סיווגה ככתובת ישראלית. אבל הפלט אינו לגמרי זהה בשני המקרים: אמנם בשני המקרים נציג הודעה כי המחרוזת חוקית וישראלית, אך במקרה הראשון נציג את המחרוזת `tr@xxx.il` ובמקרה השני את המחרוזת `tr@xxx.il`.

כיוון שבשביל לבדוק את חוקיות המחרוזת וכדי לקבוע אם היא ישראלית או לא אין משמעות להבדל בין אותיות גדולות לקטנות, נוכל לפשט את הבדיקה אם ראשית נמיר את המחרוזת למחרוזת זהה המורכבת מאותיות גדולות או קטנות בלבד, ורק אז נבדוק. נחליט כי המחרוזת האחידה תהיה כולה אותיות קטנות. את מחרוזת הקלט המקורית נשמור כמו שהיא, כדי שנוכל להציגה כפלט.

קל לבדוק אם התו הראשון הוא אות אנגלית. ברגע שבדיקה זו הצליחה, כבר ברור שהמילה אינה מתחילה בתו '@', כלומר שהרצף שלפני התו '@' אינו ריק.

פירוק הבעיה לתת-משימות

1. קליטת מחרוזת
2. יצירת מחרוזת זהה לזו שנקלטה ובה כל האותיות הן אותיות קטנות בלבד
3. בדיקה שהתו הראשון הוא אות אנגלית
4. בדיקה שהתווים '@' ו-'.' מופיעים במחרוזת
5. בדיקה שהתו '.' מופיע אחרי התו '@', אך לא מיד אחריו ולא בסוף המחרוזת

- 6. עבור מחרזת חוקית: בדיקה אם היא ישראלית
- 7. הצגה של המחרזת ושל הודעה מלווה מתאימה

בחירת משתנים

בשאלה מתוארים כמה תנאים שמחרוזת חוקית צריכה לעמוד בהם. מספיק שאחד מהם לא מתקיים כדי לקבוע שהמחרוזת אינה חוקית. נוכל להיעזר במשתנה בוליאני: כל עוד לא מצאנו בעיה בכתובת הדואר האלקטרוני ערכו של המשתנה יהיה true. כאשר נמצא שגיאה באחד התנאים נציב בו false. ההודעה לפלט תוצג לפי ערכו של המשתנה.

בנוסף נזדקק לשתי מחרוזות: האחת לשמירת מחרוזת הקלט המקורית, והשנייה לשמירת המחרוזת האחידה.

כדי לבדוק את חוקיות המחרוזת נשתמש בשני משתנים שלמים שישמרו את מיקום התווים '@' ו-'@'.

לבסוף, נקדיש משתנה גם לאורך של המחרוזת שנקלוט. כך נוכל לחשב את האורך פעם אחת, ולהשתמש בערך המשתנה בכל הפעמים שנזדקק לאורך המחרוזת.

- isLegal – משתנה בוליאני ("דגלי"), שיעיד אם הכתובת היא חוקית או לא
- str – מחרוזת לשמירת הכתובת הנקלטת
- lowerStr – מחרוזת זהה למחרוזת הקלט ובה כל האותיות הן אותיות קטנות
- atPlace – מספר שלם, לשמירת מיקומו של התו '@'
- dotPlace – מספר שלם, לשמירת מיקומו של התו '.'
- len – אורך מחרוזת הקלט

האלגוריתם

1. אגף את ערך isLegal ב-true
2. קלוט כתובת ב-str
3. צור מחרוזת חדשה, זהה ל-str ובה כל האותיות החדולות מולפות בקטנות והשם אותה ב-lowerStr
4. אס האת הראשונה היא לא את אנאית
 - 4.1. שנה את ערך isLegal ל-false
5. אס הגו '.' או הגו '@' אינן מופיעים במחרוזת
 - 5.1. שנה את ערך isLegal ל-false
6. אגרת
 - 6.1. אס הגו '.' מופיע לפני הגו '@' או מיז אגרו או בסוף המחרוזת
 - 6.1.1. שנה את ערך isLegal ל-false
 7. אס ערכו של isLegal שווה ל-true
 - 7.1. הצג כפלט הודעה כי הכתובת חוקית
 - 7.2. אס שושג הגוויס האגרוניס הס המחרוזת ".il"
 - 7.2.1. הצג הודעה כי הכתובת ישראלית
 - 7.3. אגרת
 - 7.3.1. הצג הודעה כי הכתובת אינה ישראלית
8. אגרת
 - 8.1. הצג כפלט הודעה כי הכתובת אינה חוקית

יישום האלגוריתם

עלינו לבדוק אם התווים '.' ו-'@' מופיעים במחרוזת str, ואם כן – אם מיקומם תקין. לשם כך נשתמש בפעולה str.IndexOf המקבלת תו, ומחזירה את מיקומו במחרוזת str.

כדי ליצור ממחרוזת הקלט (השמורה ב-str) מחרוזת חדשה הזזה לה ובה כל האותיות הן קטנות, נשתמש בפעולה str.ToLower. כמו כל פעולה הפועלת על מחרוזות, גם הפעולה ToLower אינה משנה את המחרוזת שעליה היא מופעלת, אלא יוצרת מחרוזת חדשה. כמו כל פעולה שיוצרת מחרוזת חדשה, הפעולה ToLower מקצה מקום עבור המחרוזת החדשה.

אנו מעוניינים שהמשתנה lowerStr יפנה למחרוזת החדשה. לכן נשתמש במשפט השמה, ונשים את הערך המוחזר מהפעולה str.ToLower במשתנה lowerStr, כך:

```
lowerStr = str.ToLower();
```

בעקבות הוראה זאת העצם lowerStr מפנה אל שטח בזיכרון, ששמורה בו מחרוזת חדשה, הזזה למחרוזת הקלט ובה כל האותיות הן קטנות. במחרוזת המקורית השמורה ב-str לא חל כל שינוי.

לא היינו צריכים להקצות במפורש שטח זיכרון לעצם lowerStr, מפני שהפעולה ToLower הקצתה בעצמה שטח עבור המחרוזת. לאחר פעולת ההשמה העצם lowerStr מפנה לשטח זה.

התוכנית המלאה

```
/*
התוכנית מקבלת כקלט מחרוזת, מציגה אותה כפלט, ומציגה גם הודעה המבהירה
*/ אם זו מחרוזת דואר אלקטרוני חוקית, ואם כן, אם היא כתובת ישראלית
using System;
public class EmailAddress
{
    public static void Main ()
    {
        string str;           // מחרוזת הקלט
        string lowerStr;      // מחרוזת כמחרוזת הקלט, פרט לכך שאותיות
                               // ובה כל האותיות קטנות
        int atPlace;          // שומר מיקום התו '@'
        int dotPlace;         // שומר מיקום התו '.'
        bool isLegal = true;  // דגל חוקיות הכתובת
        int len;              // אורך המחרוזת שנקלטה
        // קלט מחרוזת ושמירת אורכה
        Console.WriteLine("Enter a valid e-mail address: ");
        str = Console.ReadLine();
        len = str.Length;
        // יצירת המחרוזת החדשה ושמירתה
        lowerStr = str.ToLower();
        // בדיקה שהתו הראשון הוא אות לועזית
        if (!(lowerStr[0] >= 'a' && lowerStr[0] <= 'z'))
            isLegal = false;
        // מציאת מקום התווים '.' ו-'@'
        atPlace = lowerStr.IndexOf('@');
        dotPlace = lowerStr.IndexOf('.');
        if ((dotPlace == -1) || (atPlace == -1)) // אחד משני התווים חסר
            isLegal = false;
        else
    }
```

```

if ((dotPlace < atPlace) || (dotPlace == atPlace + 1)
    || (dotPlace == len - 1))
    // הסדר בין התווים שגוי או שהם מופיעים ברצף
    // או שהנקודה מופיעה בסוף
    isLegal = false;
// הפלט
if (isLegal)
{
    Console.WriteLine("{0} is a legal Email address", str);
    // בדיקת שלושת התווים האחרונים במחרוזת, האם שווים ל-il.
    if (lowerStr.IndexOf(".il") == len - 3 )
        Console.WriteLine("Email address is Israeli");
    else
        Console.WriteLine("Email address is not Israeli");
}
else
    Console.WriteLine("{0} is not a valid Email address",
        str);

} // Main
} // EmailAddress

```

סוף פתרון בעיה 5

השמת מחרוזות נכתבת כהוראת השמה רגילה:

```
s1 = s2;
```

בעקבות ביצוע ההשמה, מפנה s1 אל אותו שטח זיכרון שאליו מפנה s2. לכן אין צורך לבצע קודם הקצאת זיכרון עבור s1.

שאלה 9.11

פתחו אלגוריתם המקבל כקלט רשימת מחרוזות המסתיימת במחרוזת "stop". פלט האלגוריתם יהיה המחרוזות הארוכה ביותר ברשימה. ישמו את האלגוריתם בשפת C#.

שאלה 9.12

פתחו אלגוריתם המקבל מחרוזת קלט ומציג אותה בסדר הפוך. למשל עבור הקלט university הפלט יהיה: ytisrevinu. אין צורך ליצור מחרוזת הפוכה אלא רק להציג את התווים שלה בסדר הפוך. ישמו את האלגוריתם בשפת C#.

בעיה 6

מטרת הבעיה ופתרונה: הדגמת בניית מחרוזות בשלבים.

פתחו אלגוריתם אשר הקלט שלו הוא מחרוזת. הפלט יהיה מחרוזת חדשה, הזהה למחרוזת המקורית, פרט לכך שמופיעה בה האות c בכל מקום שהופיעה האות b או B במחרוזת המקורית. למשל עבור המחרוזת "abcd" הפלט יהיה המחרוזת "accd".

שימו ♥: פרט לשינוי המתואר, המחרוזת החדשה צריכה להיות זהה למקורית. לא ניתן להפוך בה אותיות גדולות לקטנות או להפך.

פירוק הבעיה לתת-משימות

1. קליטת מחרוזת
2. בניית המחרוזת החדשה
3. הדפסת המחרוזת החדשה

את תת-משימה 2 נוכל לפרק באופן הבא:

- 2.1. יצירת עצם מסוג מחרוזת ואתחולו במחרוזת ריקה
- 2.2. בנייה הדרגתית של המחרוזת החדשה, תו אחר תו

את תת-משימה 2.2 נוכל ליישם בלולאת `for` ומספר הפעמים לביצועה נקבע לפי אורך המחרוזת.

בחירת משתנים

`str` – המחרוזת שמתקבלת כקלט
`newStr` – המחרוזת החדשה

יישום האלגוריתם

אתחול העצם `newStr` במחרוזת ריקה (תת-משימה 2.1) אפשר לבצע יחד עם הצהרת המחרוזת:

```
string newStr = "";
```

את תת-משימה 2.2 נוכל ליישם בלולאת `for`, שמספר הפעמים לביצועה נקבע לפי אורך המחרוזת. בכל פעם נבדוק את התו הבא במחרוזת בעזרת הסימנים `[]`. ונבנה את המחרוזת החדשה באמצעות פעולת השרשור, למשל כך:

```
newStr = newStr + 'c';
```

שימו ♥: פעולת השרשור היא פעולה שמחזירה מחרוזת. כמו כל פעולה שמחזירה מחרוזת, פעולת השרשור אינה משנה את המחרוזות שהיא פועלת עליהן, אלא יוצרת מחרוזת חדשה (כמובן אחרי שהקצתה מקום עבורה), המתקבלת משרשור המחרוזות המקוריות.

נניח למשל שב-`newStr` נמצאת המחרוזת "ad". מה קורה בעת ביצוע הוראת ההשמה שלעיל?

פעולת השרשור מקבלת שתי מחרוזות (במקרה זה את `newStr` ואת המחרוזת "c", אחרי המרת התו 'c' למחרוזת) ויוצרת תוך הקצאת מקום מתאים בזיכרון מחרוזת חדשה שמכילה את שרשור שתי המחרוזות, כלומר את המחרוזת "adc". היא מחזירה את המחרוזת החדשה.

בפעולת ההשמה המחרוזת החדשה מושמת ב-`newStr`. כלומר במקום ש-`newStr` יפנה אל המחרוזת המקורית ("ad"), הוא מפנה כעת אל השטח החדש בזיכרון. מה קרה למחרוזת המקורית? ניתן לומר שהיא הלכה לאיבוד, מכיוון שאין הפניות אליה. בכך פעולת השמה של מחרוזות אינה שונה מפעולת השמה רגילה של משתנים: גם כאשר אנו מבצעים השמה כגון $x = y$ למשתנים x ו- y שהם משתנים רגילים, אז ערכו המקורי של x אובד והערך החדש של y מחליף אותו.

כאשר נשרשור למחרוזת תו נוסף, אותו תהליך יחזור על עצמו: הקצאת שטח למחרוזת חדשה שתהיה השרשור של התו החדש למחרוזת המקורית, והשמה הגורמת ל-`newStr` להפנות אל השטח בזיכרון של המחרוזת החדשה.

אם כך בבנייה הדרגתית של מחרוזת כפי שתואר לעיל, לא יהיה מדויק לומר שאותה מחרוזת גדלה כל פעם בתו נוסף. למעשה, נוצרת סדרה של מחרוזות: בכל פעם נוצרת מחרוזת חדשה, ארוכה בתו אחד, והיא תופסת את מקומה של המחרוזת הקודמת.

התוכנית המלאה

```
/*
קלט: מחרוזת
פלט: מחרוזת זהה למקורית, ובמקום כל 'B' או 'b' מופיע 'c'
*/
using System;
public class ReplaceCForB
{
    public static void Main ()
    {
        string str;           // מחרוזת הקלט
        string newStr = "";   // המחרוזת החדשה כעצם
                             // המאותחל במחרוזת ריקה/

        // קלט
        Console.Write("Enter a string: ");
        str = Console.ReadLine();
        // בניית המחרוזת החדשה
        for(int i = 0; i < str.Length; i++) // סריקת מחרוזת הקלט
                                           // תו אחר תו
        {
            if ((str[i] == 'b') || (str[i] == 'B'))
                newStr = newStr + 'c'; // 'c' ב-'b' או 'B' החלפת
            else
                newStr = newStr + str[i]; // העתקת התו המקורי
        } // for
        // פלט
        Console.WriteLine("The new string is: {0}", newStr);
    } // Main
} // class ReplaceCForB
```

סוף פתרון קציה 6

כאשר אנו נדרשים לבנות מחרוזת חדשה בשלבים, תו אחרי תו ניתן לעשות זאת באופן הבא: ליצור עצם שיהיה מחרוזת חדשה ולאתחל אותו במחרוזת ריקה (""); באמצעות הוראה לביצוע-חוזר, לבצע בכל פעם שרשור של התו החדש למחרוזת הקיימת, והחלפת המחרוזת הקיימת במחרוזת החדשה שהתקבלה מפעולת השרשור.

שאלה 9.13

פתחו וישמו אלגוריתם שהקלט שלו הוא מחרוזת המהווה משפט משובש: במקום כל רווח מופיע הסימן \$. האלגוריתם מייצר מחרוזת חדשה ובה המשפט התקני, ומציג אותו כפלט. ישמו את האלגוריתם בשפת C#.

שאלה 9.14

אורי ונעמי המציאו שפה מוצפנת. המשפטים הניתנים להצפנה כוללים מילים ורווחים בלבד, בהינתן שכל מילה נכתבת רק באותיות אנגליות קטנות. ההצפנה מתבצעת באופן הבא: כל רווח מוחלף בסימן '@', ולאחר כל אות מופיעה האות האנגלית הגדולה המתאימה לה. למשל, המשפט good morning מוצפן כך gGoOoOdD@mMoOrRnNiInNgG. פתחו אלגוריתם המקבל משפט כקלט ומציג את המשפט המוצפן המתאים לו כפלט. ישמו את האלגוריתם בשפת C#.

הדרכה: צרו מחרוזת חדשה המורכבת מאותיות גדולות, והשתמשו בה ובמחרוזת הקלט לצורך בניית המחרוזת המוצפנת.

שאלה 9.15

פתחו אלגוריתם שמקבל כקלט מחרוזת, מייצר מחרוזת חדשה ובה סדר התווים הפוך למחרוזת המקורית, ומציג את המחרוזת החדשה כפלט. בנוסף האלגוריתם בודק אם שתי המחרוזות (המקורית וההפוכה) שוות זו לזו. במילים אחרות האלגוריתם בודק אם המחרוזת המקורית היא פלינדרום. האלגוריתם מציג כפלט הודעה מתאימה לתוצאת הבדיקה.
למשל: עבור הקלט: dog יהיה הפלט: god – not a palindrome
עבור הקלט: aba יהיה הפלט: aba – a palindrome
ישמו את האלגוריתם בשפת C#.

שאלה 9.16

תלמידי הכיתה התעניינו לדעת למי יש סבתא בשם הארוך ביותר. פתחו אלגוריתם אשר יקבל כקלט את מספר התלמידים בכיתה, ולאחר מכן רשימה של שמות הסבתות של תלמידי הכיתה כמחרוזות. אורך הרשימה כמספר תלמידי הכיתה. פלט האלגוריתם יהיה השם הארוך ביותר. ישמו את האלגוריתם בשפת C#.

שאלה 9.17

כתובת אתר אינטרנט של חברה מסחרית בינלאומית בנויה בדרך כלל מ-3 חלקים המופרדים בנקודות:

www.שם החברה.com

פתחו אלגוריתם המקבל כקלט כתובת של אתר של חברה מסחרית בינלאומית, במבנה שתואר לעיל, ומציג כפלט את שם החברה בלבד. ישמו את האלגוריתם בשפת C#.
הדרכה: השתמשו בפעולה `Substring` שבטבלת הפעולות הנמצאת בסוף הפרק.

שאלה 9.18

כתובת אתר אינטרנט של חברה מסחרית שאינה בינלאומית בנויה בדרך כלל מ-3 חלקים המופרדים בנקודות:

www.סיומת המדינה.שם החברה.com

פתחו אלגוריתם המקבל כקלט רשימת כתובות של אתרי חברות מסחריות כאלה (כל אחת מהן במבנה שתואר לעיל). הרשימה תסתיים במחרוזת "end". האלגוריתם יציג כפלט עבור כל חברה את שמה ואת סיומת המדינה שלה. ישמו את האלגוריתם בשפת C#.

שאלה 9.19

יעל ועומרי המציאו משחק. כל אחד בתורו רושם משפט המתחיל במילה האחרונה של המשפט מהתור הקודם. המשפט הראשון במשחק יתחיל במילה "start". למשל:

start the game

game is one of the best ways to kill time

time is money

פתחו אלגוריתם המסייע למשחקים: האלגוריתם מקבל כקלט את מספר התורות המבוקש. בתחילת כל תור הוא מציג את המילה שצריך להתחיל בה המשפט הבא, ולאחר מכן הוא מקבל כקלט את המשפט החדש. המשחק מסתיים כאשר מספר התורות הסתיים, או כאשר המשפט שבחר אחד השחקנים מתחיל במילה שגויה. ישמו את האלגוריתם בשפת C#.

סיכום

בפרק זה הכרנו את המחלקה `string` ולמדנו כיצד לעבד מחרוזות.

בשפת C# מחרוזת אינה טיפוס פשוט, כמו `int` או `char`. הטיפוס החדש מוגדר בשפה כ**מחלקה** בשם `string`, ומחרוזות הן **עצמים** של מחלקה זו.

למעשה כבר הכרנו עצמים כאשר השתמשנו במחלקה `Random` ליצירת מספרים אקראיים. עצמים הם משתנים של הטיפוס החדש ואפשר להפעיל עליהם פעולות המוגדרות במחלקה. אופן השימוש בעצמים שונה מהשימוש במשתנים מהטיפוסים הפשוטים המוכרים לנו.

הצהרה על עצם ממחלקה מסוימת דומה להצהרה על משתנה מטיפוס סטנדרטי (כגון `int` או `char`). למשל:

```
string str;
```

עבור משתנים רגילים, הקצאת מקום בזיכרון נעשית אוטומטית עם ההצהרה עליהם ואילו עבור עצמים הקצאת הזיכרון והאתחול מתבצעים באמצעות הפעולה `new`. מכיוון שהשימוש במחרוזות נפוץ כל כך, שפת C# מאפשרת ליצור ולאתחל מחרוזת בצורה ישירה ללא שימוש בפעולה `new`.

למשל, ההוראה הבאה מקצה מקום עבור `str` ומאתחלת אותו כך שישמור את המחרוזת "ab":

```
str = "ab";
```

מיקום התווים במחרוזת מתחיל ב-0. בכל מחרוזת, אם נסמן את אורך המחרוזת ב-`len`, אז התו האחרון במחרוזת נמצא במיקום `len-1`, והתו הראשון נמצא במיקום ה-0.

פנייה אל תו בתוך מחרוזת נעשית בסוגריים מרובעים. למשל ערכו של הביטוי `str[k]` הוא התו הנמצא במקום `k` במחרוזת `str`. ניתן לקרוא את ערכו של תו במחרוזת, אך לא לשנות אותו.

לעצם יכולות להיות תכונות (נרחיב בנושא בפרק 11). למשל עבור עצם של המחלקה `string`, כלומר עבור מחרוזת, מוגדרת התכונה `Length` המייצגת את אורכה של המחרוזת. כדי לקרוא את ערכה של התכונה משתמשים ב**בסימון הנקודה**: למשל הביטוי `str.Length` מחזיר את מספר התווים במחרוזת `str`.

ערכה של התכונה `Length` אינו ניתן לשינוי.

במחלקה `string` מוגדרות **פעולות שאפשר לבצע על מחרוזות**, כלומר על עצמים של המחלקה.

לכל פעולה מוגדרים סוגי הערכים (הפרמטרים) שהיא מצפה לקבל, וכן מוגדר טיפוס הערך המוחזר ממנה. חשוב לוודא התאמה של טיפוס הפרמטרים המועברים לפעולה לאלה שהיא מצפה לקבל, ושל טיפוס הערך המוחזר ממנה לביטוי שהוא משולב בו. למשל, אם אנו מבצעים השמה של הערך המוחזר מפעולה במשתנה, יש לוודא התאמה של טיפוס המשתנה לטיפוס הערך המוחזר. יש פעולות שלא מצפות לקבל פרמטרים ופעולות שלא מחזירות ערך.

כדי לציין ביצוע פעולה של עצם מהמחלקה משתמשים ב**בסימון הנקודה**: ראשית נכתוב את שם העצם, אחריו נקודה, לאחר מכן את שם הפעולה לביצוע ומיד אחר כך נפרט בסוגריים את הערכים המועברים לפעולה. למשל:

```
if (str.Equals(otherStr))
    place = str.IndexOf('.');
```

עבור עצמים של המחלקה `string` מוגדרות פעולות רבות, המשמשות אותנו ביישום אלגוריתמים הקשורים למחרוזות. למחלקה זו פעולות רבות נוספות פרט לאלו שהוצגו בפרק. תוכלו למצוא הרחבה על כל אחת מהפעולות בקישור הזה:

http://msdn2.microsoft.com/en-us/library/system.string_members.aspx

ביחידה זו נשתמש רק בפעולות שהוצגו בפרק זה.

הפעולות שמחזירות מחרוזת, אינן משנות את המחרוזת שהופעלו עליה. הן יוצרות מחרוזת חדשה ומקצות שטח עבורה.

לביצוע **קלט של מחרוזת** השתמשנו בפעולה `Console.ReadLine`. גם פעולה זו מקצה שטח זיכרון עבור המחרוזת שנקלטה.

בעקבות ביצוע **השמה של מחרוזת** אל `s1 = s2`, מפנה `s1` אל אותו שטח זיכרון שאליו מפנה `s2`. לכן אין צורך לבצע קודם הקצאת זיכרון עבור `s1`.

בנוסף לפעולות המוגדרות במחלקה `string`, למדנו בפרק זה על **פעולת השרשור**. פעולה זו מקבלת שתי מחרוזות ויוצרת מחרוזת חדשה והיא השרשור של שתי המחרוזות. זוהי פעולה שימושית מאוד לצורך הדפסה. פעולת השרשור אינה פעולה של המחלקה `string` ולכן בכתובתה אין שימוש בסימון הנקודה. סימן פעולת השרשור הוא הסימן `+`.

בפרקים הבאים נלמד להגדיר מחלקות בעצמנו, ולא רק להשתמש במחלקות קיימות.

רשימת פעולות על מחרוזות

דוגמאות		טיפוס הערך המוחזר	תיאור הפעולה	הפעולה
הערך המוחזר	הפעולה			
3	<code>s1.IndexOf("pl")</code> כאשר ב- <code>s1</code> נמצאת המחרוזת "people"	שלם	פעולה המקבלת מחרוזת או תו, ומחפשת בתוך המחרוזת שעליה מופעלת הפעולה את המיקום הראשון שבו מופיעה המחרוזת או התו שהתקבלו. הפעולה תחזיר את המיקום. היא תחזיר את הערך -1, אם החיפוש נכשל.	<code>IndexOf(string s)</code>
2	<code>s1.IndexOf('o')</code> כאשר ב- <code>s1</code> נמצאת המחרוזת "people"			<code>IndexOf(char c)</code>

<code>false</code>	<code>s1.Equals(s2)</code> כאשר ב-s1 נמצאת המחרוזת "love" וב-s2 נמצאת המחרוזת "Love"	בוליאני	פעולה המקבלת מחרוזת, ומחזירה <code>true</code> אם המחרוזת שעליה הופעלה הפעולה והמחרוזת שהתקבלה שוות זו לזו בדיוק. אחרת, היא מחזירה <code>.false</code> פעולה זו זהה לפעולת השוואה <code>==</code>	<code>Equals(string s)</code>
<code>true</code>	<code>s1.Equals(s2)</code> כאשר ב-s1 נמצאת המחרוזת "love" וב-s2 נמצאת המחרוזת "love"			
מספר שלילי כלשהו	<code>s1.CompareTo(s2)</code> כאשר ב-s1 נמצאת המחרוזת "aa" וב-s2 נמצאת המחרוזת "ab"	שלם	פעולה המקבלת מחרוזת, ומשווה אותה למחרוזת שעליה הופעלה הפעולה. אם הן שוות זו לזו מוחזר הערך אפס. אם המחרוזת שעליה מופעלת הפעולה קודמת למחרוזת שהתקבלה בסדר מילוני , יוחזר מספר שלם שלילי. אם המחרוזת שעליה מופעלת הפעולה, מופיעה אחרי המחרוזת שהתקבלה בסדר מילוני, יוחזר מספר שלם חיובי.	<code>CompareTo(string s)</code>
המחרוזת החדשה "peace"	<code>s1.ToLower()</code> כאשר ב-s1 נמצאת המחרוזת "Peace"	מחרוזת	פעולה שיוצרת מחרוזת זהה למחרוזת שעליה היא מופעלת, ובה כל האותיות מוחלפות באותיות קטנות. הפעולה מחזירה את המחרוזת החדשה.	<code>ToLower()</code>
המחרוזת החדשה "PEACE"	<code>s1.ToUpper()</code> כאשר ב-s1 נמצאת המחרוזת "Peace"	מחרוזת	פעולה שיוצרת מחרוזת זהה למחרוזת שעליה היא מופעלת, ובה כל האותיות מוחלפות באותיות גדולות. הפעולה מחזירה את המחרוזת החדשה.	<code>ToUpper()</code>
המחרוזת החדשה "Bye"	<code>s1.Substring(4)</code> כאשר ב-s1 נמצאת המחרוזת "GoodBye"	מחרוזת	פעולה שיוצרת מחרוזת זהה לתת-מחרוזת המתחילה מהמקום ה-k של המחרוזת שעליה הפעולה מופעלת ועד סופה. הפעולה מחזירה את המחרוזת החדשה.	<code>Substring(int k)</code>

<p>המחרוזת החדשה "Bye Is"</p>	<p>s1.Substring(4, 6) כאשר ב-s1 נמצאת המחרוזת "GoodBye Israel"</p>	<p>מחרוזת</p>	<p>פעולה שיוצרת מחרוזת זהה לתת-מחרוזת המתחילה מהמקום ה-k של המחרוזת עליה היא מופעלת ואורכה הוא s. הפעולה מחזירה את המחרוזת החדשה. שימו לב שעל ערכו של k+s להיות קטן או שווה לאורך המחרוזת.</p>	<p>Substring(int k, int s)</p>
---------------------------------------	--	---------------	--	------------------------------------