

## עיבוד מחרוזות

מהו עיבוד מחרוזות? זהו התהליך של קריאת טקסטים וטיפול בהם בכל מיני רמות של דיוק, למשל הפרדה של משפט למילים או הפרדה של מילה לאותיות, אפשרות להחלפת תו בתו אחר (עבור תירגום או הצפנה).

כל מעבד תמלילים, כמו וורד למשל, מבוסס על עיבוד מחרוזות.

תחום היישומים מהחשובים בעולם המחשבים הוא עיבוד של מחרוזות. מספיק לראות למשל בחיפוש פשוט בגוגל, אם ניטעה במילה, מנוע החיפוש ינסה לתקן זאת - זהו עיבוד מחרוזות.

כמובן שיש כאן תיחכום לוגי של בדיקת איות של מילים, אולם כל המנגנון מבוסס על קריאת הטקסט, למשל כמילים בודדות, השוואה למילון או לחיפושים אחרים, כלומר חשוב לדעת כיצד להפריד מילים ממשפטים, להיות מסוגלים לעבור תו תו על מילה וכד'.

בעבר, כשמחשבים רק נולדו, רוב היישומים היו מדעיים ומתמטיים, היום עיבוד המחרוזות עולה לאין שיעור על המתמטי.

מספיק לראות למשל מה שניקרא בסיס נתונים טיפוסי (מעין טבלה שמכילה נתונים) - הטבלה יכולה לייצג נתוני תלמיד. לתלמיד יהיו מאפיינים כגון: שם, מין, תאריך לידה, מגמה, תחביב וציון ממוצע. מכל אלה רק ציון הוא ערך נומרי - כל השאר מחרוזות.

מחרוזת ב- C# היא למעשה עצם וראינו מספר מתודות (פעולות) ב- C# שמיועדות לעבד מחרוזות.

כפי שכבר למדנו בעבר, כדי להפעיל פעולה או מתודה, אנחנו צריכים עצם כלשהו ואז על ידי נקודה יכולים להפעיל את המתודה.

למשל, ראינו את המתודה שניקראת: Substring שמיועדת 'לגזור' חלק ממחרוזת (תת-מחרוזת) ולהחזיר ערך זה למקום שבא היא ניקראה.

נניח שיש לנו הגדרה כזו:

```
string str1="Thursday is a great day";
```

וכעת נירצה לגזור רק את שם היום בשבוע (Thursday) למחרוזת ניפרדת:

```
string str2=str1.Substring(0,8);
```

המספרים בתוך הסוגרים הם הפרמטרים של המתודה. הראשון אומר מאיזה מיקום להתחיל לגזור, כאשר אפס הוא ההתחלה משמאל (כמו במערכים) והפרמטר השני הוא אורך הגזירה. בדוגמה נתנו (0,8), כלומר התחל מאפס וקח 8 תווים ולכן במחרוזת str2, יהיה הערך:

"Thursday"

נסו מספרים שונים בזמנכם החופשי. נסו לראות מה יקרה עם ניתן אורך גדול מידי או מיקום התחלתי לא קיים, כמו למשל: (100,10) או (10,50)

המתודה הבאה שעליה דיברנו ניקראת: IndexOf שבתרגום חופשי אומרת: המיקום של...

למתודה זו יכולים להיות פרמטר אחד או שניים. הפרמטר הראשון (או היחיד) הוא תת מחרוזת. מה שהיא מחזירה זה את מיקום תת-המחרוזת בתוך המחרוזת שעליה היא ניקראה. לדוגמה:

```
string str1 = "serendipity";
int i_index = str1.IndexOf("i");
```

הערך שיקבל המשתנה: `i_index` הוא: 6 שהוא המיקום של ה-`i` הראשון (משמל) במחרוזת: "serendipity"

אם נירצה שהחיפוש של תת-המחרוזת יתחיל לא מהתחלת המחרוזת, אפשר לתת פרמטר נוסף ל-`Substring`, שהוא המיקום (משמאל) ממנו להתחיל לבדוק. בדוגמה הקודמת אם היינו כותבים:

```
i_index = str1.IndexOf("i", 7);
```

כעת נקבל את הערך 8 במשתנה: `i_index`. הסיבה היא שהמחשבמתחיל לבדוק מהתו `p` (שהוא השביעי משמאל כשופרים מאפס), ואז הוא מוצא את ה-`i` השני, זה שאחריו יש `t`.

תת-המחרוזת שנימצאת בסוגריים יכולה גם להינתן כמשתנה ולא דוקא כקבוע, והיא יכולה להכיל מספר תוים לא רק אחד, לדוגמה:

```
string sub = "ren";
int p = str1.IndexOf(sub);
```

יחזיר לנו ב-`p` ערך של: 2 (המיקום של התחלת תת-המחרוזת "ren" בתוך המחרוזת: "serendipity")

ישנה גם מתודה שעובדת מהסוף להתחלה וניקראת: **LastIndexOf** ההסבר שלה קצת יותר מורכב. המחשב מתחיל לסרוק מהמיקום שאומר הפרמטר השני (המספר), שוב סופרים משמאל ומתחילים ב-0, אבל הפעם הוא בודק מאותו מיקום אחורה עד להתחלת המחרוזת, ואז מחזיר (אם מצא) את המיקום של התת-מחרוזת (שוב המיקום שמתחיל ב-0 משמאל).

הנה מספר דוגמאות:

```
string str2 = "abcdeabfg";
int k = str2.LastIndexOf("ab", 7);
```

`k` מקבל את הערך: 5. מדוע? הסריקה מתחילה מהתו: 'f' והולכת אחורה, כך שהוא מגלה את ה-`ab` השני (לא זה שבתחילת המחרוזת), ומחזיר את המיקום של התחלתו (כלומר של ה-`a`).

עוד דוגמה, כי זה קצת מסובך:

```
int k = str2.LastIndexOf("ab", 4);
```

הפעם `k` מקבל את הערך: 0. מתחילים לסרוק ממיקום 4 שזה מיקום התו: "e", ולכן הוא מוצא את ה-`ab` הראשון (משמאל), ועל כן מחזיר את המיקום של התחלתו שזה התו `a` משמאל שהוא 0.

כל הצורות השונות לקריאה של IndexOf יחזירו את הערך -1 אם תת-המחרוזת איננה נימצאת במחרוזת עליה פועלים:

```
int s = str2.IndexOf("ggg");
```

s יקבל את הערך -1

בנוסף, למדנו מתודה שיודעת להשוות בין מחרוזות, ניקראת: Equals. נדגים זאת:

```
string s1, s2;
s1="abc";
s2="abd";
if (s1.Equals(s2))
    Console.WriteLine("Strings are equal");
else
    Console.WriteLine("Strings are NOT equal");
```

מה שיודפס זה:

Strings are NOT equal

כלומר Equals מחזיר False אם המחרוזות אינו זהות, ומחזיר True אם הן כן זהות.

התרגיל שהתחלנו בכתה, היה תרגיל לא פשוט, אבל יכול להיות שימושי. היינו צריכים לקרוא משפט או סיפור למשתנה מחרוזתי ולנסות לזהות מצב שיש מילים כפולות שעוקבות זו אחר זו, ואם קיימות, להדפיס את המיקום של המילה השניה בצמד.

למשל אם המשפט היה:

the the fox went over over the fence

הפלט יהיה: (אם ספרתי נכון)

4 22

אני מניחים שיש רווח בודד בין מילים, כלומר מילה היא אוסף התווים שנימצא בין רווחים או בין ההתחלה והרווח הראשון או אחרי הרווח האחרון.

הנה שוב השלד של התוכנית (להשלים בבית):

```
string current_word, next_word;
int p; //our progress position
int l; //word's length
int s_index;
int pair_count = 0;
Console.WriteLine("Please enter a sentence or a story");
string text = Console.ReadLine();
space = " ";
s_index = text.IndexOf(space); //find the first space
while (s_index != -1)
{
}
if (pair_count == 0)
    Console.WriteLine("No doubles");
else
    Console.WriteLine("There were{ 0} doubled words",
        pair_count);
```