

קורס תכנות

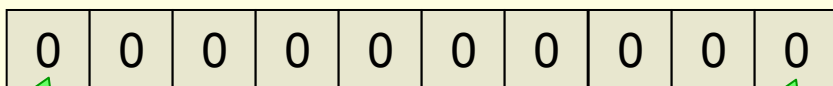
מערכים (המשך)

מערכים

• הגדרה:

- אוסף של משתנים מאותו סוג
- לכולם יש שם משתנה משותף ואינדקס
- בעזרתם ניתן לגשת למשתנה מסוים באוסף

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



arr[0]

arr[9]

```
#include <stdio.h>
```

```
int main()
```

```
{  
    int i=0, input[30]={0};
```

הגדרת מערך בגודל 30

```
    printf("Enter the 30 integers:\n");
```

```
    for (i=0; i < 30; i++)
```

```
        scanf("%d", &input[i]);
```

כתיבה לתוך התא ה- i במערך

```
    for (i=29; i >= 0; i--)
```

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

קריאה מתוך התא ה- i במערך

```
    return 0;
```

```
}
```

אתחול מערכים

```
type Array_Name[ Size ] = { value1, value2, ... };
```

- אתחול מערכים בשורת הגדרה:

- ניתן לאתחל מערכים בצורה זו אך ורק בשורת ההגדרה
- הערכים מוצבים בתאים לפי הסדר החל מהתא הראשון
- אם אין מספיק ערכים, שאר התאים מאותחלים לערך 0

```
int arr[5] = {0,1,2,3,4};
```

```
int arr[5] = {0,1,2};
```

```
int arr[5] = {0};
```

אתחול מערכים

אחרי הגדרת המערך לא ניתן להתייחס למערך כיחידה אחת, אפשר להתייחס רק לכל אחד מהתאים שלו בנפרד.

- אתחול מערכים:

- לאחר ההגדרה אי אפשר לאתחל את כל המערך כיחידה אחת.
- יש לגשת לכל תא בנפרד ולתת לו ערך.
- אי אפשר להשוות בין מערכים.

```
int arr[10];  
int i = 0;  
for ( i=0; i < 10; ++i )  
{  
    arr[i] = i;  
}
```

חריגה מגבולות המערך

- מותר לגשת לכל תא במערך.
- אסור לגשת לתאים שנמצאים מחוץ למערך.
- גישה לתא שנמצא מחוץ למערך תגרום לבעיות:
 - בסבירות גבוהה התוכנית תקרוס.
 - בסבירות יותר גבוהה היא לא תעבוד כראוי.

```
int arr[10];  
int i = 0;  
for ( i=0; i <= 10; ++i )  
{  
    arr[i] = i;  
}
```

arr[10] לא שייך למערך! ←

העברת מערך לפונקציה

- אפשר להעביר מערכים כפרמטר לפונקציה
- **גודל המערך לא מועבר!**
- **באחריותנו** לדאוג שהפונקציה לא תחרוג ממנו (נעביר גם את גודלו)
- **בשונה ממשתנים אחרים:**
 - כשמעבירים מערך לפונקציה הוא **אינו מועתק**.
 - לפונקציה מועבר **המערך המקורי**.
 - שינוי של ערכי המערך בתוך הפונקציה **ישנה את המערך המקורי!!**

העברת מערך לפונקציה - דוגמא

```
void init_array( int array[], int size, int value )
{
    int i = 0;
    for ( i=0; i<size; i++ )
        array[i] = value;
}
```

- הפונקציה הבאה מקבלת:
 - מערך של מספרים שלמים.
 - גודל המערך.
 - ערך לאיתחול המערך.
- הפקודה `init_array(digits, 10, 5)`:
 - תאתחל את 10 התאים הראשונים במערך `digits` לערך 5.
 - המערך המקורי ישתנה

מערך כפרמטר

```
int main()
```

```
{
```

```
    int numbers[10];
```

הגדרת מערך בגודל 10

```
    for (i = 0; i < size; i++)  
        printf("%d ", numbers[i]);
```

מה יודפס פה?

```
    init_array(numbers, 10, 100);
```

קריאה לפונקציה אתחול

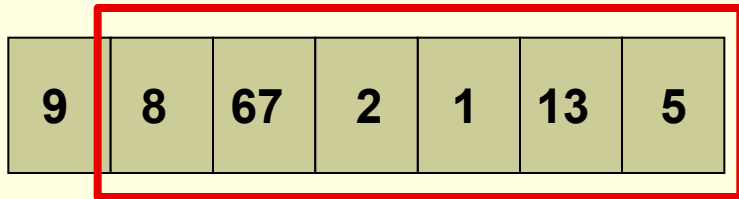
```
    for (i = 0; i < size; i++)  
        printf("%d ", numbers[i]);
```

ומה יודפס פה?

```
}
```

רקורסיה במערך

- באופן כללי נוח להפעיל רקורסיה על מערכים
- אפשר להתייחס לתת-מערך כמערך קצר יותר ולהפעיל עליו פונקציה רקורסיבית.



- לדוגמא – אם נרצה למצוא את האיבר המקסימלי במערך
- נמצא את האיבר המקסימלי במערך שמתחיל באיבר השני
- נשווה אותו לאיבר הראשון ונחזיר את הגדול מביניהם
- מה הוא בסיס הרקורסיה?
- במערך בגודל 1 – נחזיר את הערך היחיד בו.

רקורסיה במערך – מציאת ערך מקסימלי

```
int maxNum(int arr[], int startIndex, int endIndex)
{
    int max;
    if (startIndex == endIndex)
        return arr[startIndex];
    max = maxNum(arr, startIndex+1, endIndex);
    if (arr[startIndex] > max)
        return arr[startIndex];
    return max;
}

int main()
{
    int arr[] = {9,8,67,2,113,5};
    printf ("The maximal number is: %d\n", maxNum(arr,0,5));
}
```

מערכים רב-מימדיים

```
type Array_Name[ Size1 ][ Size2 ] ... [ SizeN ];
```

- משמשים לייצוג מטריצות או מבנים ממימדים גבוהים.
- למשל לוח של איקס-עיגול.
- הגישה לערכים מתבצעת בעזרת שני אינדקסים.

```
char ticTacToe[3][3];  
...  
if ( ticTacToe[i][j] == ' ' )  
{  
    ticTacToe[i][j] = 'X';  
}  
...
```


מערכים רב-מימדיים - אתחול

- ניתן לאתחל מערך רב מימדי בהגדרה:

```
char ticTacToe[3][3] = {{'X','O','X'},{'O','O','X'},{'X','O','O'}};
```

- אתחול נפוץ יותר הוא בעזרת לולאות מקוננות:

```
void init_array( int array[][SIZE], int size, int value )  
{  
    int i=0, j=0;  
    for ( i=0; i < size; ++i )  
        for ( j=0; j < SIZE; ++j )  
            array[i][j] = value;  
}
```



בהעברת מערך רב מימדי
לפונקציה חייבים לכתוב את
גדלי המימד השני והלאה

כפל מטריצות

- נכתוב תכנית שמחשבת כפל של 2 מטריצות a, b בגודל 3x3.
- את התוצאה נשמור במטריצה c.
- נוסחא לכפל מטריצות:

$$c[i][j] = \sum_{k=0}^{SIZE-1} a[i][k] \cdot b[k][j]$$

```
#define MATRIX_SIZE 3
```

$$c[i][j] = \sum_{k=0}^{SIZE-1} a[i][k] \cdot b[k][j]$$

```
int main(void)
```

```
{
```

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

```
    int b[MATRIX_SIZE] = { {1,0,0}, {0,2,0}, {0,0,3} };
```

```
    int c[MATRIX_SIZE][MATRIX_SIZE];
```

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

```
    /* Compute the product c = a * b */
```

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

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

```
        {
```

```
            c[i][j] = 0;
```

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

```
                c[i][j] += a[i][k] * b[k][j];
```

```
        }
```

```
    /* Print c */
```

```
    ...
```

```
    return 0;
```

```
}
```

קורס תכנות

שיעור שביעי: מחרוזות

מחרוזות

- מהי מחרוזת?
 - רצף של תווים, למשל: "hello world"
- כיצד נייצג מחרוזת במחשב?
 - char מייצג תו בודד
 - מחרוזת נייצג בעזרת מערך של תווים
- מה אורך המחרוזת? האם תופסת את כל המערך או רק את חלקו?
 - אפשרות א': בנוסף לגודל המערך נשמור גם את אורך המחרוזת
 - אפשרות ב': תו מיוחד יציין את סוף המחרוזת.

מחרוזות ב - C

```
char str[20] = "hello world";
```

- מחרוזת:
 - מערך של תווים המייצגים מחרוזת.
 - למשל מילה או משפט.
- בדרך כלל **אנחנו** לא נתייחס אל מחרוזות כאל מערכים:
 - נתייחס למחרוזת כולה בבת אחת.
 - נתייחס רק לחלק המערך שהוא בעל משמעות כמחרוזת.
- **בשפת C** קיימות פקודות וספריות שמקלות את העבודה עם מחרוזות

מחרוזות

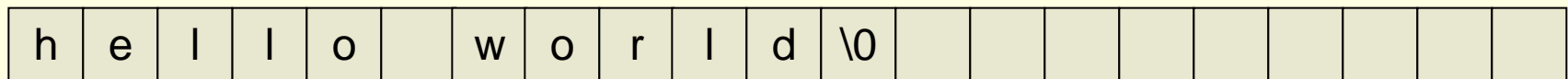
```
char str[20] = "hello world";
```

- ההבדל בין מחרוזות ומערכים:
- מחרוזת היא רצף של תווים, נשתמש במערך כדי לממש מחרוזת
- מחרוזת מסתיימת בתו ' \0 ' (ערכו 0 בטבלת ה-ascii).
- כלומר לאחר תווי המחרוזת יופיע התו ' \0 '.
- אם התו לא מופיע אז זהו מערך של תווים ולא מחרוזת!
- כל הפעולות והפקודות המיוחדות למחרוזות **מסתמכות** על הימצאותו של התו ' \0 ' בסוף המחרוזת.

מחרוזות - אתחול

```
char str[20] = "hello world";
```

- אתחול:
 - ניתן לאתחל מחרוזת על ידי מחרוזת קבועה.
- מחרוזת קבועה:
 - " ... " – גרשיים כפולים משמשים לייצוג מחרוזת קבועה.
 - בסוף המחרוזת מופיע הסימן '\0'.



↑
str[0]

↑
סוף המחרוזת

↑
str[19]

" "

גרשיים

- גרש אחת מתייחסת לתו
 - '5', 'd', 'A'
- גרשיים מתייחסות למחרוזת
 - "Hello", "I am a string", "54".
 - מתנהג לפי המוסכמות ומוסיף '\0' לסוף כל מחרוזת
 - לכן "A" ו- 'A' אינם זהים.
 - "A" זהה ל- 'A' שלאחריו '\0'
- בפעולות על מחרוזות יש להקפיד על שימוש בגרשיים.

`strcmp(input, "a")`

מחרוזת כפרמטר לפונקציה

- אין דרך מיוחדת להגדיר שהפרמטר לפונקציה זו מחרוזת ולא מערך
- הטיפוס שיועבר לפונקציה הוא המערך שמכיל אותה
- אז איך יודעים מתי נתייחס לפרמטר כמערך ומתי כמחרוזת?
 - זה נגזר מההגדרה לא פורמאלית של הפונקציה – מה היא אמורה לעשות
- כאשר נקבל מחרוזת כקלט לפונקציה נניח שהתו '0' נמצא במערך ומציין את סוף המחרוזת

דוגמא – פונקציה למציאת אורך מחרוזת

```
int my_strlen(const char str[])  
{  
    int counter = 0;  
  
    while (str[counter] != '\0')  
        counter++;  
  
    return counter;  
}
```

הפונקציה מקבלת מחרוזת כקלט

נספור את מספר התווים המופיעים
לפני תו סיום המחרוזת

העברת מערכים קבועים לפונקציה

- אפשר להגדיר מערכי קלט בפונקציה כקבועים (const).
- כך נוכל להבטיח שלא נשנה בטעות את ערך המשתנה
- נבטיח למשתמש שלא נשנה את המערך
- לא ניתן להעביר מערך שהוגדר כקבוע (const) לפונקציה שבה הוא לא קבוע ועלול להשתנות.

```
void init_array(const int array[], int size, int value)
{
    int i;

    for (i = 0; i < size; i++)
        array[i] = value;
}
```

error C3892: 'array' : you cannot assign to a variable that is const

קלט של מחרוזות בעזרת scanf

- קריאת תו אחד כל פעם, עד ש:
 - נגמר המקום
 - קראנו תו שמסמן לנו להפסיק (למשל סוף השורה)

```
#include <stdio.h>

#define MAX_STRING_LEN 200

int main() {
    char tav, sentence[MAX_STRING_LEN + 1];
    int i = 0;
    do {
        scanf("%c", &tav);
        if (tav != '\n') {
            sentence[i] = tav;
            i++;
        }
    } while ((tav != '\n') && (i < MAX_STRING_LEN));
    sentence[i] = '\0';
    ...
    return 0;
}
```

נרצה לקלוט מחרוזת בעלת
200 תווים לכל היותר ...
לכן נצטרך מערך בגודל 201

ע"י הוספת '\0' בסוף אנו הופכים
את מערך התווים למחרוזת

קלט של מחרוזות בעזרת getchar

- קריאת תו אחד כל פעם, עד ש:
 - נגמר המקום
 - קראנו תו שמסמן לנו להפסיק (למשל סוף השורה)

```
int main()
{
    char tav, sentence[MAX_STRING_LEN + 1];
    int i = 0;

    for (tav = getchar();
        tav != '\n' && i < MAX_STRING_LEN;
        tav = getchar())
    {
        sentence[i] = tav;
        i++;
    }

    sentence[i] = '\0';
    ...
    return 0;
}
```

קלט/פלט של מחרוזת שלמה

- אפשר לקלוט מחרוזות שלמה ב-`scanf` באמצעות הקידוד `%s` למשל:

```
char answer[100];  
scanf("%s", answer);
```

- הקלט יהיה רק עד רווח או ירידת-שורה (כלומר זו דרך לקלוט מילים)
- התו `'\0'` מוכנס אוטומטית ע"י `scanf` אחרי תווי המחרוזת שנקלטה
- שימו לב שלא רושמים את הסימן & ב-`scanf` של מחרוזת שלמה.

- שימו לב:
 - באחריות המשתמש לדאוג שיש במערך מספיק מקום (אחרת התכנית תעוף)
 - אפשר להגביל את מספר התווים שנקראים:

```
char answer[100];  
scanf("%99s", answer);
```

- דרך אחרת להגבלת מספר התווים היא לקלוט תו-תו בלולאה

פלט של מחרוזת שלמה

- נשתמש ב printf להדפסת המחרוזת

```
printf("%s", answer);
```

- יודפסו התווים מתחילת המערך ועד לתו '0\'

קלט/פלט של מחרוזת שלמה

gets(str) •

- פונקציה הקולטת מחרוזת שלמה (עד לתו '\n').
- לא מאפשרת הגבלת אורך הקלט.

puts(str) •

- מדפיסה מחרוזת + '\n'.

- שקולה ל: `printf("%s\n", str);`

מה אורך המחרוזת שקראנו?

- נשתמש בפונקציה my_strlen שכתבנו

```
int main()
{
    char sentence[MAX_STRING_LEN + 1];
    int i = 0, len;

    scanf("%s", sentence);
    len = my_strlen(sentence);
    printf("The length of the string \"%s\" is %d\n",
          sentence, len);

    return 0;
}
```

דוגמא: תוכנית שמרווחת מילה

```
#include <stdio.h>

#define WORD_SIZE 30

int main()
{
    char word[WORD_SIZE + 1],
        spaced_word[WORD_SIZE * 2 + 1];
    int i;

    scanf("%29s", word);

    for (i = 0; word[i] != '\0'; i++)
    {
        spaced_word[i * 2] = word[i];
        spaced_word[i * 2 + 1] = ' ';
    }
    spaced_word[i * 2] = '\0';
    printf("The word after spacing: %s\n", spaced_word);

    return 0;
}
```

| | | | | | |
|---|---|---|---|---|----|
| h | e | l | l | o | \0 |
|---|---|---|---|---|----|



| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|----|
| h | “ | e | “ | l | “ | l | “ | o | “ | \0 |
|---|---|---|---|---|---|---|---|---|---|----|

דוגמא: העתקה של מחרוזת

```
#include <stdio.h>

#define WORD_SIZE 20

void string_copy(const char source[], char destination[])
{
    int i;

    for (i = 0; source[i] != '\0'; i++)
        destination[i] = source[i];
    destination[i] = '\0';
}

int main()
{
    char text[WORD_SIZE + 1], copy[WORD_SIZE + 1];
    scanf("%s", text);
    string_copy(text, copy);
    printf("%s\n", copy);
    return 0;
}
```

פעולות על מחרוזות

- מאחר ומחרוזות ממומשות בעזרת מערכים לא נוכל לבצע פעולות על מחרוזת שלמה, כמו השוואה או השמה.
- נבצע פעולות כאלו בעזרת הפונקציות המוגדרות בספרייה - `string.h`.
 - השוואת מחרוזות `strcmp`
 - העתקת מחרוזות `strcpy`
 - שירשור מחרוזות `strcat`
 - ועוד ...

הספריה string.h – פונקציות לדוגמה

```
int strlen( const char str[] );
```

- מציאת אורך של מחרוזת:

- השוואה לקסיקוגרפית בין מחרוזות:

```
int strcmp( const char str1[], const char str1[] );
```

- מחזירה **0** אם שתי המחרוזות זהות.
- **מספר חיובי** אם הראשונה קודמת לשנייה.
- **מספר שלילי** אם השנייה קודמת לראשונה.

- העתקת מחרוזת למחרוזת אחרת:

```
char* strcpy( char target[], const char source[] );
```

- שרשור מחרוזת למחרוזת אחרת:

```
char* strcat( char target[], const char source[] );
```

דוגמה למימוש strcmp

```
int my_strcmp(const char str1[], const char str2[])
{
    int i=0;

    while ( (str1[i] == str2[i]) && (str1[i] != '\0') )
        i++;

    return (str2[i] - str1[i]);
}
```



כל עוד אותו התו ולא
הסתיימו המחרוזות

חישוב אורך מחרוזת ברקורסיה

- הגדרה רקורסיבית של אורך מחרוזת:
 - אם התו הראשון הוא `'\0'` ← אורך המחרוזת `str` הוא: `0`
 - אחרת –
- אורך מחרוזת `str` הוא: `1 + (אורך המחרוזת str החל מהתו השני)`

```
int strlen_rec(char str[], int index)
{
    if (str[index] == '\0')
        return 0;
    return 1 + strlen_rec(str, index + 1);
}
```

מה עושה התכנית הבאה?

```
#include <stdio.h>
#include <string.h>

int main()
{
    int mccain_votes = 0 , obama_votes=0, loopFlag = 1;
    char vote[10] = {0};

    while (loopFlag)
    {
        scanf("%s", vote);
        if (strcmp(vote, "McCain")==0)
            mccain_votes++;
        else if ( strcmp(vote, "Obama")==0 )
            obama_votes++;
        else if (strcmp(vote, "Stop")==0)
            loopFlag = 0;
        else printf("Wrong vote!\n")
    }

    printf("McCain: %d votes\nObama: %d votes\n", mccain_votes, obama_votes);

    return 0;
}
```

תרגום מחרוזת למספר

- פונקציות לתרגום מחרוזות למספרים
- בספרייה `stdlib.h`

```
int atoi( const char str[] );
```

- תרגום למספר שלם:

```
double atof( const char str[] );
```

- תרגום למספר ממשי:

- למשל:

```
int num = atoi("12345");
```

שאלה ממבחן

סמסטר א' תשס"ז

שאלה מס' 2 (25 נקודות)

סעיף א' (12 נקודות):

כתבו פונקציה בעלת המפרט (prototype) הבא:

```
void kth word (char words[], int k, char kth[]);
```

הניחו ש-words היא מחרוזת המכילה מילים באנגלית, כשבין כל שתי מילים סמוכות יש תו-רווח אחד, k הוא מספר שלם חיובי, ו-kth היא מחרוזת נוספת. בסיום הפונקציה, המחרוזת ב-kth צריכה להיות המילה ה-k-ית מבין המילים שהועברו ב-words (למשל אם k=1 אז תועתק ל-kth המילה הראשונה). אם יש ב-words פחות מ-k מילים, אז הפונקציה תשים ב-kth מחרוזת ריקה (כלומר תו '־' בלבד). הניחו שב-kth יש מספיק מקום לצורך ההעתקה.

שאלה ממבחן

סעיף א'

כתבו פונקציה בעלת המפרט (prototype) הבא:

```
void kth_word (char words[], int k, char kth[]);
```

- words היא מחרוזת המכילה מילים באנגלית
 - בין כל שתי מילים סמוכות יש תו-רווח אחד בדיוק
 - k הוא מספר שלם חיובי
 - kth היא מחרוזת נוספת
- בסיום הפונקציה, המחרוזת kth צריכה להיות המילה ה-k-ית מבין המילים שהועברו ב-words
- אם יש ב- words פחות מ-k מילים, הפונקציה תשים ב-kth מחרוזת ריקה (תו '\0')
- הניחו שב-kth יש מספיק מקום לצורך ההעתקה

כיצד נממש?

- עוברים על המחרוזת words
- תוך כדי מעבר על המחרוזת:
 - סופרים רווחים
 - עוצרים במקרה שקורה אחד מהשניים:
 - הגענו לסוף המחרוזת (כלומר ל-'0').
 - הגענו לתחילת המילה ה-k (כלומר מצאנו את הרווח ה-k-1).
- אם הגענו לסוף המחרוזת, מכניסים '0' לתא הראשון ב-kth.
- אם הגענו לרווח ה-(k-1):
 - לולאה עד תו הרווח הבא או עד '0'.
 - מעתיקים כל תו ב- words ל-kth.

פתרון חלק א'

```
void kth_word (char words[], int k, char kth[]) {
```

```
    int j=0, i=0, word_counter=1;
```

```
    while ((words[i]!='\0') && (word_counter < k)) {
```

```
        if (words[i]==' ')
            word_counter++;
        i++;
```

נמצא את האינדקס i בו
מתחילה המילה ה- k

```
    }
```

```
    if (words[i]=='\0')
```

```
        kth[0]='\0';
```

אם אורך המשפט קצר מ- k
מילים, נחזיר מחרוזת ריקה

```
    else {
```

```
        for (j=0; words[i]!='\0' && words[i]!=' '; i++,j++)
```

```
            kth[j]=words[i];
```

```
        kth[j]='\0';
```

אחרת, נעתיק את המילה ה- k למחרוזת kth

```
    }
```

```
}
```

שאלה ממבחן

סמסטר א' תשס"ז

סעיף ב' (13 נקודות):

כיתבו תוכנית הקולטת משפט באנגלית, באורך 300 תווים לכל היותר (בסיומו מוקלד Enter). לאחר מכן התוכנית קולטת מילה באנגלית, באורך 30 אותיות לכל היותר (שבסיומה מוקלד Enter). על התוכנית להדפיס את מספר הפעמים שהמילה הזו מופיעה כמילה במשפט שנקלט, בהנחה שהמשפט מכיל רק מילים באנגלית ובין כל שתי מילים סמוכות יש תו-רווח. לדוגמא:

- אם המשפט הוא "I am what I am" והמילה היא "am" אז התוכנית תדפיס 2
- אם המשפט הוא "I am what I am" והמילה היא "at" אז התוכנית תדפיס 0

הערות:

- ניתן להניח שאורך כל מילה במשפט הוא 30 אותיות לכל היותר.
- ניתן להניח שהקלט תקין, וגם שהמשתמש יודע איזה קלט להכניס (אין צורך להדפיס הודעות שיסבירו לו מה הקלט הדרוש).

שאלה ממבחן

סעיף ב'

- כיתבו תוכנית הקולטת:

1. משפט באנגלית, באורך 300 תווים לכל היותר, שבסיומו מוקלד Enter.
2. מילה באנגלית, באורך 30 אותיות לכל היותר, שבסיומה מוקלד Enter.

- על התוכנית להדפיס את מספר הפעמים שהמילה הזו מופיעה במשפט, בהנחה שהמשפט מכיל רק מילים באנגלית ובין כל שתי מילים סמוכות יש תו-רווח יחיד.

לדוגמא:

- אם המשפט הוא "I am what I am" והמילה היא "am" אז התוכנית תדפיס 2
- אם המשפט הוא "I am what I am" והמילה היא "at" אז התוכנית תדפיס 0

הערות:

- ניתן להניח שאורך כל מילה במשפט הוא 30 אותיות לכל היותר.
- ניתן להניח שהקלט תקין, וגם שהמשתמש יודע איזה קלט להכניס.

כיצד נממש?

- נקלוט משפט לתוך מחרוזת.
- נקלוט מילה לתוך מחרוזת נוספת.
- נשתמש בפונקציה מהסעיף הקודם:

```
void kth_word (char words[], int k, char kth[])
```

- נרוץ בלולאה על מיקום המילה במשפט:
 - נקרא לפונקציה `kth_word` עם הפרמטרים:
 - המשפט שקלטנו
 - מספר המילה הנוכחית
 - מחרוזת ריקה אליה תועתק המילה
 - אם המילה שחזרה היא מחרוזת ריקה, נסיים
 - אם לא, נשווה אותה למילה שנקלטה
 - אם המילים זהות, נקדם מונה.

פתרון חלק ב'

```
#include<string.h>
#define WORD_SIZE 30
#define SENTENCE_SIZE 300

int main()
{
    char sentence[SIZE+1], word[W_SIZE+1], kth[W_SIZE+1];
    int k=1, count=0;

    gets(sentence);
    gets(word);

    kth_word(sentence,k,kth);
    while(kth[0]!='\0')
    {
        if (strcmp(kth, word) == 0) count++;
        k++;
        kth_word(sentence,k,kth);
    }

    printf("The word appeared %d times in the sentence\n", count);
    return 0;
}
```

לולאה על המילים במשפט

לכל מילה במשפט – נבדוק
אם היא זהה למילה word

נדפיס את התוצאה

קורס תכנות - מחרוזות

שאלות נוספות?