

קורס תכנות

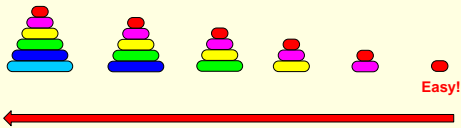
שיעור שישי: מערכים

1

בשיעור הקודם למדנו על רקורסיה

- פתרנו את בעיית מגדלי הנוי בעזרת **רקורסיה**.
- כלומר בעזרת פונקציה שקוראת לעצמה.
- רקורסיה מאפשרת לנו לפתור בעיה "גדולה" בעזרת פתרון של בעיות "קטנות" המרכיבות אותה.
- בכל **קריאה רקורסיבית** אנחנו "מקטינים" את הבעיה ולבסוף מגיעים למקרה קצה שאותו קל לפתור באופן ישיר.

פתרון בעיה "גדולה" בעזרת פתרון בעיות "קטנות"



3

פונקציה רקורסיבית

- פונקציה רקורסיבית מורכבת משלושה חלקים:
 1. הקטנת הבעיה
 2. פתרון הבעיה המקורית בעזרת פתרון לבעיה המוקטנת
 3. מקרה בסיס שאותו אנחנו יודעים לפתור (בסיס הרקורסיה)

4

חישוב נוסחאות רקורסיביות בשפת C

עצרת

$$\begin{cases} n! = n \cdot (n-1)! \\ 1! = 1 \end{cases}$$

```
int factorial (int n) /* n > 0 */
{
    if (n==1)
        return 1;
    return (n * factorial(n-1));
}
```

שאלה

- נרצה לקלוט N מספרים שלמים (N ידוע), ולאחר מכן להדפיס אותם בסדר הפוך מהסדר שקלטנו אותם (+ אופציה לפעולות נוספות על הקלט). כיצד נוכל לעשות זאת?
- לדוגמא, עבור N=3 והקלט
0 5 4
4 5 0 הפלט צריך להיות:
- פתרון אפשרי בעזרת N משתנים

```
int main()
{
    int a1, a2, a3;
    scanf("%d%d%d", &a1, &a2, &a3);
    printf("%d %d %d\n", a3, a2, a1);
    return 0;
}
```

6

גישה לתאים במערך

- ניתן לגשת לתא במערך ע"י כתיבת שם המערך ומספרו הסידורי של האיבר בסוגריים מרובעים.
- מספור התאים מתחיל מ-0.
- תא מסוים במערך שקול למשתנה רגיל מאותו הסוג.

13

מעבר על כל התאים במערך

- נוכל לעבור על כל תאי המערך בעזרת מעבר על כל האינדקסים של המערך

```
int i;
for (i = 0; i < 30; i++)
{
    grades[i] = 100;
}
```

- ממוצע הציונים בקורס

```
int i, sum = 0, number_of_student;
double average;
...
for (i = 0; i < number_of_students; i++)
    sum += grades[i];
average = sum / number_of_students;
```

14

פתרון לשאלה

- כיכד נפתור כעת את השאלה מתחילת השיעור

```
int main()
{
    int numbers[30];
    int i;

    for (i = 0; i < 30; i++)
        scanf("%d", &numbers[i]);

    for (i = 29; i >= 0; i--)
        printf("%d ", numbers[i]);
    printf("\n");

    return 0;
}
```

הגדרת 30 משתנים מטיפוס int

קליטת 30 ערכים (תאים 0-29)

כתיבת 30 ערכים (תאים 29-0)

15

איתחול מערכים

- הקוד הבא מאתחל מערך המכיל את חמשת המספרים האי-זוגיים הראשונים (1, 3, 5, 7, 9)

```
int main()
{
    int odds[5];
    int i;

    for (i = 0; i < 5; i++)
        odds[i] = i * 2 + 1;

    ...
    return 0;
}
```

16

איתחול מערכים

- עבור מערכים קטנים שערכי האיברים שלהם ידועים מראש קיימת צורת איתחול בשורת ההגדרה

```
int main()
{
    int odds[] = {1, 3, 5, 7, 9};
    ...
    return 0;
}
```

- גודל המערך נקבע לפי מספר הערכים

17

אתחול מערכים בשורת ההגדרה

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

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

18

שימוש במערכים בתכנית

- לא ניתן לבצע פעולות על כל אברי המערך בו זמנית
- למעט באיתחול שבו ביצענו השמה לכל התאים במערך
- פעולות על המערך יבוצעו איבר איבר
- איפוס המערך


```
for (i = 0; i < 100; i++)
    array[i] = 0;
```
- העתקת מערך אחד לשני


```
array1 = array2
```

 לא ע"י `array1 = array2`
- כך גם פעולות השוואה, קלט/פלט וכדומה

19

שימו לב!

- חריגה מגבולות המערך היא טעות נפוצה כתוצאה:
- התכנית תקרוס (בסבירות גבוהה)
- לא תעבוד כראוי עם בעיות לא ברורות

```
double array[10];
for (i = 0; i <= 10; i++)
    array[i] = 0;
```

array[10] אינו איבר במערך

20

הגדרת גודל המערך

- גודל המערך נקבע בעת הקומפילציה לפיכך הוא חייב להיות ערך ידוע בזמן הקומפילציה ולא משתנה



```
double array[10];
int grades[3 + 5];
char text[256]
```



```
double array[x];
int grades[y];
char text[10 * i]
```

21

בחזרה לפתרון לשאלה

- מהם כל ה-30 המופיעים בתכנית? האם הם קשורים? האם הם קשורים למספר 29?

```
#include <stdio.h>

int main()
{
    int numbers[30];
    int i;
    for (i = 0; i < 30; i++)
        scanf("%d", &numbers[i]);
    for (i = 29; i >= 0; i--)
        printf("%d ", numbers[i]);
    printf("\n");
    return 0;
}
```

22

#define

- ניתן להגדיר קבועים סימבוליים במקום מספרים


```
#define SYMBOLIC_NAME value
```
- ההגדרות תופענה בתחילת הקובץ לאחר ה `#include`

```
#define ARRAY_SIZE 30
```

```
#define TRUE 1
#define FALSE 0
```

23

בחזרה לפתרון לשאלה

- מהם כל ה-30 המופיעים בתכנית? האם הם קשורים? האם הם קשורים למספר 29?

```
#include <stdio.h>

#define ARRAY_SIZE 30

int main()
{
    int numbers[ARRAY_SIZE];
    int i;
    for (i = 0; i < ARRAY_SIZE; i++)
        scanf("%d", &numbers[i]);
    for (i = ARRAY_SIZE - 1; i >= 0; i--)
        printf("%d ", numbers[i]);
}
```

כדי לטפל ב-100 מספרים כמה שינויים נצטרך לבצע בתכנית ללא `define`? וכמה עם שימוש ב `define`?

24

#define

```
#define ARRAY_SIZE 10
```

- שורת ה-define גורמת לקומפילר להחליף (בשלב העיבוד המוקדם) כל מופע בתוכנית של ARRAY_SIZE ב-10.
- רק אח"כ מתבצע התרגום לשפת מכונה.
- שימו לב: ההחלפות האלה לא מתבצעות בתוך "מחרוזת":

```
printf("Please enter ARRAY_SIZE numbers:\n");
```

```
Please enter ARRAY_SIZE numbers:
```

```
printf("Please enter %d numbers:\n", ARRAY_SIZE);
```

```
Please enter 10 numbers:
```

25

קונבנציות

מענה והלאה...

- שמות משתנים ושמות פונקציות יכתבו באותיות קטנות size, array, power, num1, is_digit
- Defines יכתבו רק באותיות גדולות ARRAY_SIZE, TRUE, FALSE

26

מערכים - דוגמה נוספת

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

int main()
{
    int i, num, digits[DIGITS_NUM]={0};

    printf("Enter a positive integer:\n");
    scanf("%d", &num);
    while (num!=0)
    {
        digits[num%DIGITS_NUM]++;
        num=num/DIGITS_NUM;
    }
    for (i=0; i<DIGITS_NUM; i++)
        printf("Digit %d appeared %d times\n", i, digits[i]);

    return 0;
}
```

התוכנית קולטת מספר שלם חיובי ומדפיסה כמה פעמים הופיעה בו כל ספרה

תא i במערך יכיל את מספר ההופעות של הספרה i במספר num

27

העברת מערך לפונקציה

- אפשר להעביר לפונקציה מערכים
- גודל המערך לא מועבר!
 - באחריותנו לדאוג שהפונקציה לא תחרוג ממנו
 - בד"כ נרצה להעביר את גודלו כפרמטר נוסף
- בשונה ממשתנים אחרים:
 - כשמעבירים מערך לפונקציה הוא אינו מועתק
 - לפונקציה מועבר המערך המקורי
 - שינוי של ערכי המערך בתוך הפונקציה ישנו את המערך המקורי!!
- מה שמועבר לפונקציה הוא כתובת המערך
 - כלומר המיקום של המערך המקורי בזיכרון

28

העברת מערך לפונקציה - דוגמה

- נדגים פונקציה שמקבלת מערך של מספרים שלמים ואת גודלו, ומחזירה את סכום המספרים במערך.
- הקריאה לפונקציה:

```
sum = array_sum(array, size);
```
- כותרת הפונקציה:

```
int array_sum(int array[], int size);
```
- אין צורך לציין את גודל המערך בתוך ה-[] בכותרת הפונקציה

29

מערכים ופונקציות

- נכתוב פונקציה המקבלת מערך של מספרים שלמים ומחזירה את סכום ערכי המערך

```
int array_sum(int array[], int size)
{
    int i, sum;

    for (i = 0; i < size; i++)
        sum += array[i];

    return sum;
}
```

- הפונקציה מקבלת את המערך וגודלו, ציון גודל המערך בין הסוגריים המרובעים לא יעזור (הסבר בהמשך הקורס)

30

קריאה לפונקציה שמקבלת מערך

```
#include <stdio.h>
#define ARRAY_SIZE 10
int main()
{
    int numbers[ARRAY_SIZE];
    int i, sum;

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

    sum = array_sum(numbers, ARRAY_SIZE);
    printf("The sum is %d\n", sum);

    return 0;
}
```

העברת מערך לפונקציה ע"י שימוש בשם המערך

31

פונקציה שממלאת מערך

```
void array_fill(int array[], int size, int val)
{
    int i;

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

- בתוך פונקציה ניתן לשנות את הערכים במערך המקורי!
- בניגוד למשתנים רגילים...

32

שינוי ערכי המערך

```
int main()
{
    int i, numbers[ARRAY_SIZE];

    array_fill(numbers, ARRAY_SIZE, 10);

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

    return 0;
}
```

- הפלט: 10 10 10 10 10 10 10 10 10 10 10

33

דוגמא – פונקציה שהופכת מערך

- נניח שרוצים לכתוב פונקציה שמקבלת מערך והופכת אותו כולומר, בהינתן המערך:

5	13	1	2	67	8	9
---	----	---	---	----	---	---

- הפונקציה תשנה אותו למערך:

9	8	67	2	1	13	5
---	---	----	---	---	----	---

34

פונקציה שהופכת מערך - שלבים

- מה נרצה לעשות?
 - להחליף בין התא הראשון לתא האחרון
 - אח"כ להחליף בין התא השני לתא לפני אחרון
 - וכו'
- מימוש:
 - לולאה מתחילת המערך ועד אמצעו
 - בכל איטרציה, להחליף את התא הנוכחי עם המתאים לו מסוף המערך

5	13	1	2	67	8	9
---	----	---	---	----	---	---

9	8	67	2	1	13	5
---	---	----	---	---	----	---

35

פונקציה שהופכת מערך - מימוש

```
void reverse(int array[], int size)
{
    int i, temp;

    for (i = 0; i < size / 2; i++)
    {
        temp = array[i];
        array[i] = array[size - 1 - i];
        array[size - 1 - i] = temp;
    }
}
```

36

דוגמא: מחיקת איבר במערך

- נרצה למחוק ערך (value) מהמערך
- נמצא את הערך במערך
- נזיז את הסיפא של המערך מקום אחד שמאלה
- שלבים:
 - פונקציה (find) שמוצאת ערך במערך ומחזירה את מיקומו
 - או -1 אם איננו נמצא
 - פונקציה (delete) שמוחקת ערך ומחזירה TRUE אם מחקה ו-FALSE אם לא
 - פונקציית main שמחברת את הכל

37

פונקציה למציאת ערך במערך

```
int find(int array[], int size, int value)
{
    int i;

    for (i = 0; i < size; i++)
    {
        if (array[i] == value)
            return i;
    }

    return -1;
}
```

38

פונקציה למחיקת איבר

```
#define TRUE 1
#define FALSE 0
```

```
int delete(int array[], int size, int value)
{
    int i;
    int position = find(array, size, value);

    if (position == -1)
        return FALSE;

    for (i = position; i < size - 1; i++)
        array[i] = array[i + 1];

    return TRUE;
}
```

39

התכנית השלמה

```
int main()
{
    int numbers[ARRAY_SIZE] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
    int val = 0;

    printf("Before delete: ");
    print_array(numbers, ARRAY_SIZE);

    printf("Choose value for deletion: ");
    scanf("%d", &val);

    if (delete(numbers, ARRAY_SIZE, val))
        printf("After delete: ");
        print_array(numbers, ARRAY_SIZE - 1);
    else {
        printf("couldn't find %d\n", val);
    }

    return 0;
}
```

לא לשכוח לשנות את גודל המערך

40

לא ניתן להחזיר מערך מפונקציה

Syntax Error

```
int[] copy10(int array[], int size)
{
    int i;
    int result[] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};

    for (i = 0; i < size && i < 10; i++)
        result[i] = array[i];

    return result;
}
```

- לא ניתן להחזיר מערך כערך מוחזר
- ככל משתנה, בעת סיום הפונקציה הוא מפסיק להתקיים

41

רקורסיה במערך

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

9	8	67	2	1	13	5
---	---	----	---	---	----	---

- 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.

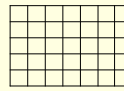
42

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 דו ממדיים



- משמשים לייצוג מטריצות או טבלאות
- שימוש בשני אינדקסים – שורות ועמודות
- הגדרה:

```
data_type array_name[row_size][column_size];
```

- מטריצה של int בגודל 10x20

```
int matrix[10][20]
```

- גישה לתא בעזרת שני אינדקסים

```
matrix[0][0] ← התא הראשון בשורה הראשונה
Matrix[9][19] ← התא האחרון בשורה האחרונה
```

44

איתחול מערך רב מימד

- בהגדרה

```
int my_matrix[3][2] = { {1, 0}, {0, 1}, {1, 1} };
int array2d[][3] = { {1, 0, 0}, {0, 1, 0}, {0, 0, 1} };

```

אין חובה לציין את הממד הראשון, אך חובה לציין את הממדים האחרים.

- איתחול בעזרת לולאות מקוננות

```
void init_array(int array[][100], int rows, int value )
{
    int row, column;
    for (row = 0; row < rows; i++)
        for (column = 0; column < 100; column++)
            array[row][column] = value;
}
```

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

לוח הכפל

```
#define SIZE 10
void init_table(int table[SIZE][SIZE]) {
    int i, j;
    for (i = 0; i < SIZE; i++)
        for (j = 0; j < SIZE; j++)
            table[i][j] = (i + 1) * (j + 1);
}

void print_table(int table[SIZE][SIZE]) {
    int i, j;
    for (i = 0; i < SIZE; i++) {
        for (j = 0; j < SIZE; j++)
            printf("%4d", table[i][j]);
        printf("\n");
    }
}

int main() {
    int table[SIZE][SIZE];
    init_table(table);
    print_table(table);
    return 0;
}
```

46

כפל מטריצות

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

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

47

```
#define MATRIX_SIZE 3
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;
}
```

48

