

תבניות אלגוריתמיות לפעולות רקורסיביות

תבנית צבירה

תבנית צבירה לסדרת ערכים:

סכום (סדרת ערכים)

1. אם מתקיים תנאי הסיום

החזר ערך

2. אחרת -

החזר את ערך הקצה + **סכום** (סדרת הערכים ללא ערך הקצה)

דוגמא:

```
/* פעולה המחזירה את סכום הספרות שבמספר שלם לא שלילי num */
סכום-ספרות (num)
```

1. אם $num < 10$ החזר num או אם $num == 0$ החזר 0

2. החזר את ספרת האחדות של num + **סכום-ספרות** (num ללא ספרת האחדות)

```
public static int SumOfDigits (int num)
```

```
{
```

```
    if (num < 10) return num;          //--- if (num == 0) return 0;
```

```
    return num % 10 + SumOfDigits (num / 10);
```

```
}
```

תבנית מניה

תבנית מניה למספר הערכים בסדרת ערכים

1. אם מתקיים תנאי הסיום

החזר ערך

2. אחרת -

1 (בגלל ערך הקצה) + **סכום** (סדרת הערכים ללא ערך הקצה)

דוגמא:

```
/* פעולה המחזירה את מספר הספרות שבמספר שלם לא שלילי num */
מספר-ספרות (num)
```

1. אם $num < 10$ החזר 1 או אם $num == 0$ החזר 0

2. החזר 1 + **מספר-ספרות** (num ללא ספרת האחדות)

```
public static int NumOfDigits (int num)
```

```
{
```

```
    if (num < 10) return 1;          //--- if (num == 0) return 0;
```

```
    return 1 + NumOfDigits (num / 10);
```

```
}
```

I **3בירה אותנית**

דרך א':

סכום-מותנה (סדרת ערכים)

1. אם מתקיים תנאי הסיום
החזר ערך
2. אחרת -
 - 2.1 אם ערך הקצה מקיים את תנאי החיפוש
החזר: ערך הקצה + **סכום-מותנה** (הערכים ללא ערך הקצה)
 - 2.2 אחרת - החזר: **סכום-מותנה** (סדרת הערכים ללא ערך הקצה)

/* פעולה המחזירה את סכום הספרות הגדולות מ-5 שבמספר שלם לא שלילי num */

סכום-ספרות-גדול-מ-5 (num)

1. אם $num == 0$ החזר 0
2. אם $num \% 10 > 5$
החזר: $num \% 10$ + **סכום-ספרות-גדול-מ-5** (num/10)
3. החזר: **סכום-ספרות-גדול-מ-5** (num/10)

דרך ב': (שימוש במשתנה עזר)

סכום-מותנה (סדרת ערכים)

1. אם מתקיים תנאי הסיום
החזר ערך
2. אחרת -
 - 2.1 אם ערך הקצה מקיים את תנאי החיפוש
ערך הקצה $z \leftarrow$
 - 2.2 אחרת
 $z \leftarrow 0$
 - 2.3 החזר: z + **סכום-מותנה** (הערכים ללא ערך הקצה)

/* פעולה המחזירה את סכום הספרות הגדולות מ-5 שבמספר שלם לא שלילי num */

סכום-ספרות-גדול-מ-5 (num)

1. אם $num == 0$ החזר 0
2. אם $num \% 10 > 5$ $z \leftarrow num \% 10$
אחרת - $z \leftarrow 0$
3. החזר: z + **סכום-ספרות-גדול-מ-5** (num/10)

II מנייה מותנית

דרך א':

מנייה-מותנית (סדרת ערכים)

1. אם מתקיים תנאי הסיום
החזר ערך
2. אחרת -
 - 2.1 אם ערך הקצה מקיים את תנאי החיפוש
החזר: $+ 1$ מנייה-מותנית (הערכים ללא ערך הקצה)
 - 2.2 אחרת - החזר: מנייה-מותנית (סדרת הערכים ללא ערך הקצה)

/* פעולה המחזירה את מספר הספרות הזוגיות שבמספר שלם לא שלילי num */

מספר-ספרות-זוגיות (num)

1. אם $num == 0$ החזר 0
2. אם $num \% 2 == 0$
החזר: $+ 1$ מספר-ספרות-זוגיות (num/10)
3. החזר: מספר-ספרות-זוגיות (num/10)

דרך ב':

מנייה-מותנית (סדרת ערכים)

1. אם מתקיים תנאי הסיום
החזר ערך
2. אחרת -
 - 2.1 אם ערך הקצה מקיים את תנאי החיפוש
 $z \leftarrow 1$
 - 2.2 אחרת
 $z \leftarrow 0$
 - 2.3 החזר: $+ z$ מנייה-מותנית (סדרת הערכים ללא ערך הקצה)

/* פעולה המחזירה את מספר הספרות הזוגיות שבמספר שלם לא שלילי num */

מספר-ספרות-זוגיות (num)

1. אם $num == 0$ החזר 0
2. אם $num \% 2 == 0$
 $z \leftarrow 1$
- אחרת
 $z \leftarrow 0$
3. החזר: $+ z$ מספר-ספרות-זוגיות (num/10)

III **3בירה במערכ**

סכום-איברי-המערך (arr, place)

1. אם מתקיים תנאי הקצה – החזר את איבר הקצה
2. החזר את האיבר שבקצה + סכום-איברי-המערך (המערך ללא האיבר שבקצה)

/* פעולה המחזירה את סכום איברי המערך */

<pre>public static int SumOfArr (int [] arr) { return SumOfArr (arr, 0); }</pre>	<pre>public static int SumOfArr (int [] arr) { return SumOfArr (arr, arr.length - 1); }</pre>
<pre>private static int SumOfArr (int [] arr, int place) { if (place == arr.length) return 0; return arr[place] + SumOfArr (arr, place + 1); }</pre>	<pre>private static int SumOfArr (int [] arr, int place) { if (place == 0) return arr[place]; return arr[place] + SumOfArr (arr, place - 1); }</pre>

שימו ♥ :

משפט הזימון מבקש לבדוק את סכום איברי המערך.

לצורך המימוש אנו זקוקים למקומו של האיבר שבקצה המערך (האיבר שנמצא במקום הראשון או האחרון), דרך המימוש מוסתרת מהמשתמש בפעולה.

ולכן הפעולה **סכום-המערך (arr)** הינה פעולה גלויה – *public* ואילו הפעולה **סכום-המערך (arr, place)** היא פעולה מוסתרת – *private*

הגדרת שתי הפעולות, במחלקת השירות המטפלת במערך, אפשרית מכיוון שלכל אחת מהן חתימה שונה (חתימה – כותרת הפעולה, כולל הפרמטרים).

מושגים :

Encapsulation – הסתרת מידע

Methods Overloading – העמסת פעולות

מנייה מותנית מאצרך

IV

מספר- האיברים (arr, place)

1. אם מתקיים תנאי הקצה – החזר את איבר הקצה
2. אם האיבר שבקצה מקיים את תנאי הבדיקה, החזר 1 + מספר- האיברים (המערך ללא איבר הקצה)
- החזר מספר- האיברים (המערך ללא איבר הקצה)

/* פעולה המחזירה את מספר איברי המערך האי-זוגיים */

<pre>public static int NumOfOddNumbers (int [] arr) { return NumOfOddNumbers (arr, 0); }</pre>	<pre>public static int NumOfOddNumbers (int [] arr) { return NumOfOddNumbers (arr, arr.length - 1); }</pre>
<pre>private static int NumOfOddNumbers (int [] arr, int place) { if (place == arr.length) return 0; if (arr[place] % 2 == 1) return 1 + NumOfOddNumbers(arr, place+1); return NumOfOddNumbers(arr, place+1); }</pre>	<pre>private static int NumOfOddNumbers (int [] arr, int place) { if (place < 0) return 0; if (arr[place] % 2 == 1) return 1 + NumOfOddNumbers(arr, place-1); return NumOfOddNumbers(arr, place-1); }</pre>

הערה:

במקרה של מנייה (או צבירה) מותנית, נעדיף לקבוע את תנאי הקצה אחרי האיבר הראשון (או האחרון) כדי לחסוך בבדיקה של איבר זה.

דוגמה: אם תנאי הקצה היה `if (place == 0)`

היה עלינו לבדוק את הזוגיות של האיבר במקום 0 כדי לקבוע איזה ערך יוחזר