2. Algorithmization

To strengthen your skills on this topic, solve the following tasks.

ONE SIZE ARRAYS

1. Array A [N] contains natural numbers. Find the sum of those elements that are multiples of a given K.

2. In a series of real numbers a1, a2 ... an replace all elements greater than Z with this number. Count the number of replacements.

3. Find out how many negative, positive and zero elements are in a given array N.

4. Given real numbers a1, a2, ..., an. Swap the largest and smallest numbers.

5. Given real numbers a1, a2, ..., an. Print only those numbers for which ai > i.

6. Given a sequence of N real numbers. Find the sum of numbers whose ordinal index are prime numbers.

7. Given a real numbers a_1, a_2 ... a_n. Find max (a_1 + a_2n, a_2 + a_2n − 1, ... a_n + a_n + 1).

8. Given an integer sequence a1, a2, ..., an. Create a new sequence without integers that are equal to min (a1, a2, ..., an).

9. Find the most repeated number in an array. If there are several such numbers, then find the smallest.

10. Delete every second element from the array and fill the vacated elements with zeros. Note: do not use an additional array.

MATRIX

1. Display all odd columns in the matrix with the first element greater than the last.

2. Print the diagonal of the given 2D square matrix.

3. Print k-row and p-column of the given 2D matrix.

4. Create a square matrix according to the example, where every second row is reversed (n is even).

$$\begin{pmatrix} 1 & 2 & 3 & \cdots & n \\ n & n-1 & n-2 & \cdots & 1 \\ 1 & 2 & 3 & \cdots & n \\ n & n-1 & n-2 & \cdots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ n & n-1 & n-2 & \cdots & 1 \end{pmatrix}$$

5. Create a square matrix according to an example (n is even).

$$\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 & 1 & 1 \\ 2 & 2 & 2 & \cdots & 2 & 2 & 0 \\ 3 & 3 & 3 & \cdots & 3 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ n-1 & n-1 & 0 & \cdots & 0 & 0 & 0 \\ n & 0 & 0 & \cdots & 0 & 0 & 0 \end{pmatrix}$$

6. Create a square matrix according to an example (n is even).

$$\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 & 1 & 1 \\ 0 & 1 & 1 & \cdots & 1 & 1 & 0 \\ 0 & 0 & 1 & \cdots & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 1 & 1 & \cdots & 1 & 1 & 0 \\ 1 & 1 & 1 & \cdots & 1 & 1 & 1 \end{pmatrix}$$

7. The program creates a matrix according to the function and counts the number of positive numbers in it.

$$A[I,J] = \sin\left(\frac{I^2 - J^2}{N}\right)$$

8. Swap two columns in a numeric matrix, namely swap all the elements from the first column to the corresponding positions to the second one and vice versa. Column numbers are entered by the user from the keyboard.

9. The program finds the sum of the elements in each column. Also it finds the column with the highest sum of elements.

10. Find the positive elements of the main diagonal of a square matrix.

11. The program fills a 10x20 matrix with random numbers from 0 to 15. Then it displays the matrix and the index of the rows in which the number 5 occurs 3 or more times.

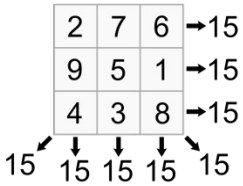
12. Sort the matrix in ascending and descending order row-wise.

13. Sort Matrix in ascending and descending order column-wise.

14. The program creates a random m x n matrix that consists only of zeros and ones. In each column the number of "1" is equal to the column index.

15. Find the maximum elements of the matrix and replace all odd elements with it.

16. Write a program that takes a number from the user and creates a magic square. A magic square is one where the sum of each row, column, and diagonal is the same. See the example below:



SORTING

1. The program concatenates two one-dimensional arrays into one, including a second array between k and k + 1 elements of the first, without using an additional array.

2. Given two arrays: a1 <= a2 ... <= an and b1 <= b2 ... bm. Create a new ascending array of them. Note: do not use an additional array.

3. Write the **selection sort algorithm.** Selection sort is basically a selection of an element position from the start with the rest of the elements. Elements are compared and exchanged depending on the condition and then the selection position is shifted to the next position till it reaches the end.

4. Write an **exchange sorting algorithm**. Count the number of replacement.

5. **Binary Insertion sort.** Binary insertion sort uses the binary search to find the right position to insert an element at a specific index at every iteration. First, the location where the element needs to be inserted is found. Then, the elements are shifted to the next right location. Now, the specific element is placed in the position.

6. **Shell sorting** in ascending order. Shell sort is a sorting technique that is similar to insertion sort, wherein elements are sorted which are at far ends (either end) of the array. This way, the interval size between the next and the second to last element reduces. This happens for all the elements in the array until the interval distance is reduced to 0.

7. Two ascending sequences are given. The program shows the places where you can insert the elements of the second sequence into the first, so that the new sequence is also ascending (taking into account the shift after insertion).

8. Create a program that finds a common denominator of the given fractions and orders them in ascending order.

DECOMPOSITION

1. Write methods for finding the greatest common divisor (GCD) and the least common multiple (LCM) of two natural numbers.

2. Write methods for finding the greatest common divisor (GCD) of more than two (or array) numbers.

3. Find the area of a regular hexagon with side a using the triangle area method.

4. Find, between which points n the greatest distance. Tip: the points coordinates arranged in an array.

5. Find the second largest number in the array.

6. Write a method that checks if the given three numbers are co-prime.

7. Find the sum of the factorials of all odd numbers from 1 to 9.

8. In array D find the following sums: D[l] + D[2] + D[3]; D[3] + D[4] + D[5]; D[4] + D[5] + D[6] etc. Tip. Find the sum of three consecutive array elements with numbers from k to m.

9. Find an area of a quadrilateral with one right angle.

10. Decompose the entered number N into elements and display them as an array.

11. Compare and find which of the two numbers has more digits.

12. The program creates an array in which the sum of the digits of each element is equal to the number K, but not more than N.

13. The program finds all twin prime numbers on the line segment [n, 2n], where n > 2

14. Find the Armstrong numbers from 1 to k. A positive number is called the Armstrong number if it is equal to the sum of cubes of its digits.

15. Find natural n-digit numbers whose digits form a strictly increasing sequence (for example 1234, 5789). To solve the problem, use decomposition.

16. Find the sum of n-digit numbers containing only odd digits. Also find how many even digits are in the found sum. To solve the problem, use decomposition.

17. From the given number subtract the sum of its digits. From the result subtract the sum of its digits, etc. How many such interactions should be performed to get zero?