Лабораторна робота № 1

ОБРОБКА МАСИВІВ У C#

Мета роботи:

- вивчити протокол оголошення та ініціалізації одновимірних і багатовимірних масивів у мові C#;
- закріпити на практиці використання операторів розгалуження і циклу мови C# для обробки масивів.

Теоретичні відомості

1. Оголошення, ініціалізація та використання одновимірних масивів

Масив – це сукупність послідовно розташованих комірок пам’яті, які можуть містити об’єкти однакового типу.

В мові C# масиви є об’єктами класу System.Array, що дозволяє програмісту використовувати широкий спектр властивостей і методів для роботи з масивами.

Для оголошення одновимірного масиву в C# використовується наступний синтаксис:

```
тип даних[] назва масиву;
```

Приклади:

```
int[] b;
double[] Weight;
```

На відміну від C++, де масиви можуть створюватись як статично так і динамічно, у C# всі масиви є динамічними, тому, у наведених прикладах

ідентифікатори “b” та “Weight” фактично є *посиланнями* на майбутні масиви. Спроба використати таке посилання до його ініціалізації адресою масиву призводить до помилки компіляції.

Масиви у C# створюються за допомогою оператора **new**, синтаксис якого для одновимірних масивів аналогічний синтаксису відповідного оператора мови C++:

```
назва масиву = new тип даних[кількість елементів];
```

Після створення масиву його елементи автоматично ініціалізуються нульовими значеннями, наприклад, 0, ‘\0’, null залежно від типу елементів.

Ініціалізацію елементів масиву можна виконати безпосередньо в момент його створення. При цьому вказувати кількість елементів не обов’язково: компілятор визначить її за кількістю вказаних ініціалізаторів:

```
назва масиву = new тип даних[] {зн.1, зн.2, ..., зн.N};
```

При використанні ніціалізаторів в момент оголошення дозволяється взагалі не використовувати оператор new для створення масиву.

Приклади:

```
float[] s = {0.1f, 0.2f, 0.4f, 0.8f};
int[] myArray = new int[100];
char[] a = new char[3] {‘i’, ‘ї’, ‘є’};
double[] x;
x = new double[] {0.1, 0.2, 0.3};
```

Для обробки масивів, як правило, використовуються цикли з лічильником-індексом, реалізовані на базі оператора for. При спробі звернутись

до елемента масиву за індексом, що виходить за допустимі межі, середовище виконання генерує виключення **IndexOutOfRangeException** (“Індекс знаходиться поза межами масиву”).

Якщо необхідно послідовно отримати доступ до *значення* кожного елемента масиву доцільно використати оператор foreach.

Приклади:

```
int[] myArray = new int[100];
Random RndGen = new Random ();
for (int i = 0; i < myArray.Length; i++)
{
    myArray[i] = RndGen.Next (10);
}
...
double[] x;
x = new double[] {0.1, 0.2, 0.3};
foreach (double Val in x)
{
    System.Write (Val + " ");
}
```

В таблицях 1 і 2 перелічені декілька важливих властивостей і методів класу Array, доступних при роботі з масивами.

Таблиця 1 – Властивості класу Array

Назва властивості	**Опис**
Length	Кількість елементів у масиві
Rank	Розмірність масиву

Таблиця 2 – Методи класу Array

Назва властивості	**Опис**
BinarySearch	Пошук заданого значення в одновимірному масиві (здійснюється лише після сортування)
Clear	Заповнення масиву нульовими значеннями
IndexOf	Пошук першого елемента в масиві із заданим значенням
LastIndexOf	Пошук останнього елемента в масиві із заданим значенням
Reverse	Інверсія порядку елементів масиву
Sort	Сортування масиву за зростанням

2. Оголошення, ініціалізація та використання багатовимірних масивів

Багатовимірні масиви у C# існують двох видів: *прямокутні*, в яких кількість елементів за кожною розмірністю є однаковою, та "*зубчасті*", розміри яких за різними розмірностями можуть бути різними. На рисунку 1 представлені приклади прямокутної та "зубчастої" матриці.

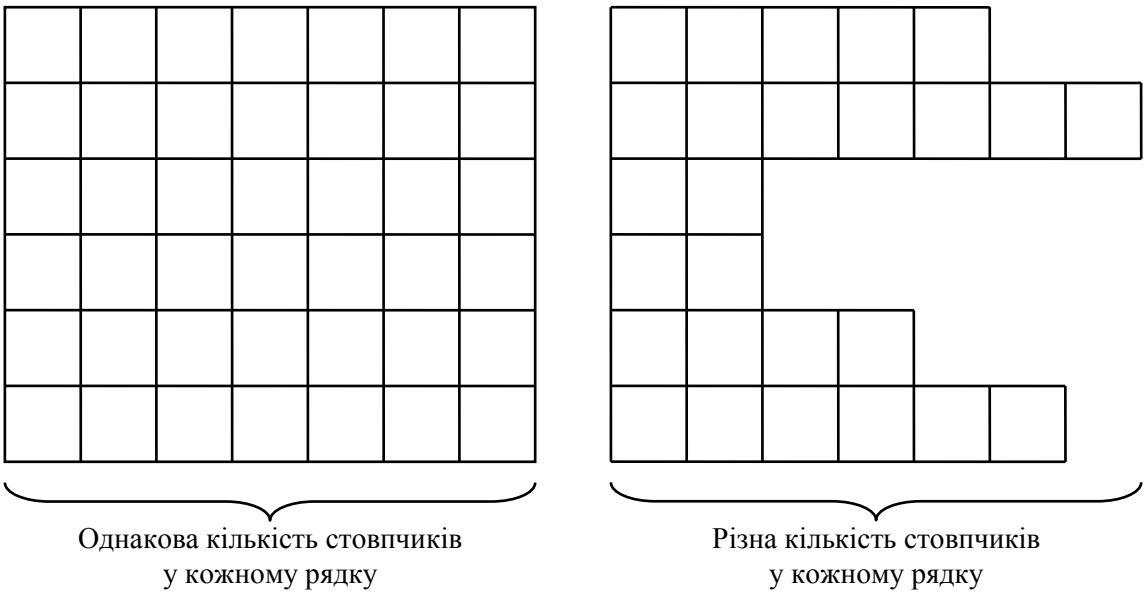

Рисунок 1 – Прямокутна та "зубчаста" матриці

Для оголошення багатовимірного прямокутного масиву в C# використовується наступний синтаксис:

```
тип даних[,,...,] назва масиву;
```

Кількість ком у квадратних дужках відповідає числу, на одиницю меншому від розмірності багатовимірного прямокутного масиву.

Приклади:

```
int[,] Matrix; // матриця (масив розмірністю 2)
double[,,] Weight; // трьохвимірний масив
```

Як одновимірні так і багатовимірні прямокутні масиви створюються за допомогою оператора new, але його синтаксис є простішим за синтаксис відповідного оператора C++:

```
назва масиву = new тип даних[p.1, p.2, ..., p.N];
```

Ініціалізацію елементів багатовимірного прямокутного масиву також можна виконати безпосередньо в момент його створення: кожна група елементів, що належать одному виміру, береться у фігурні дужки.

Приклади:

```
// матриця з двох рядків та трьох стовпчиків
float[,] Base = {{1.0f, 0.0f, 0.0f}, {0.0f, -1.0f, 0.0f}};
// трьохвимірний масив – "паралелепіпед" 3×2×2
char[,,] a = new char[3, 2, 2]     {{{'a', 'b'}, {'c', 'd'}},
                                    {{'o', 'u'}, {'f', 'g'}},
                                    {{'k', 'l'}, {'e', 'i'}}};
```

Для обробки багатовимірних масивів, як правило, використовуються вкладені цикли, реалізовані на базі оператора for з незалежними лічильниками-індексами для кожного виміру. Слід пам’ятати, що при здійсненні індексації елементів масиву індекси розділяються комами.

Приклад:

```
int[,] Matrix = new int[5, 8];
Random RndGen = new Random ();
for (int i = 0; i < Matrix.GetLength (0); i++)
{
    for (int j = 0; j < Matrix.GetLength (1); j++)
    {
         Matrix[i, j] = RndGen.Next (10);
    }
}
```

Метод GetLength для багатовимірних масивів повертає розмір масиву за вказаним виміром.

Прямокутний багатовимірний масив у C# є *неподільним* в тому сенсі, що елементи його окремо взятого виміру не можна адресувати як незалежний масив. Наприклад, на відміну від C++ (у випадку динамічного виділення пам’яті) рядки прямокутної матриці в C# не можна адресувати як окремі масиви.

Для оголошення багатовимірного “зубчастого” масиву в C# використовується наступний синтаксис:

```
тип даних[][]...[] назва масиву;
```

Кількість пар квадратних дужок відповідає числу, на одиницю меншому від розмірності багатовимірного “зубчастого” масиву.

Приклад:

```
int[][] Notched; // дві розмірності
```

Багатовимірні “зубчасті” масиви створюються поетапно, подібно до динамічних масивів у C++. Наприклад, при створенні “зубчастої” матриці спочатку створюється масив її рядків, а потім створюється кожен рядок як одновимірний масив.

Приклад:

```
Notched = new int[5][]; // п’ять рядків
for (int i = 0; i < Notched.Length; i++)
{
    // кількість стовпчиків змінюється
    // залежно від рядка: 2, 3, ..., 6
    Notched[i] = new int[i + 2];
}
```

Обробка багатовимірних “зубчастих” масивів виконується за допомогою вкладених циклів, але при цьому слід брати до уваги той факт, що властивість Length для “всього” масиву фактично дорівнює довжині “зовнішнього” масиву (наприклад, для “зубчастої” матриці Length дорівнює кількості рядків, а не загальній кількості її елементів).

Приклад:

```
Random RndGen = new Random ();
for (int i = 0; i < Notched.Length; i++)
{
```

```
        for (int j = 0; j < Notched[i].Length; j++)
        {
             Notched[i][j] = RndGen.Next (10);
        }
    }
```

Елементи окремо взятого виміру “зубчастого” масиву можна адресувати як незалежний масив.

Методичні вказівки

Завданням даної лабораторної роботи є створення, компіляція, відладка та виконання програми, в якій у консольному режимі реалізується заданий алгоритм обробки одно- і/або багатовимірного масиву. *Результатом* виконання лабораторної роботи має бути демонстрація коректної роботи програми на тестових прикладах.

Залежно від поставленої задачі необхідно визначити, масив якого типу: прямокутний чи “зубчастий” буде використовуватись для її розв’язання. Слід розуміти, що “зубчасті” масиви доцільно використовувати тоді, коли в процесі обробки багатовимірний масив зручно розглядати як сукупність масивів меншої розмірності. Наприклад, при розв’язанні задачі про сортування кожного рядка матриці, її рядки зручно розглядати як незалежні одновимірні масиви.

Для реалізації обробки масивів за заданим алгоритмом рекомендується широко використовувати готові рішення: вбудовані властивості і методи класу Array. Послідовний доступ до значень елементів масивів рекомендується здійснювати з використанням оператора foreach.

Демонстрація результатів роботи програми повинна бути поетапною. Необхідно передбачити вивід на екран у відформатованому вигляді всіх масивів, задіяних у розв’язанні задачі, в їх початковому стані, після здійснення кожного заданого перетворення та в кінцевому вигляді.

При виконанні лабораторної роботи слід, також, дотримуватись

методичних вказівок до лабораторної роботи № 1.

Приклад

Записати всі не нульові елементи прямокутної матриці A в одновимірний масив W. Створити матрицю B, кількість рядків якої дорівнює кількості елементів у масиві W, а кількість стовпчиків у кожному рядку задається значенням відповідного елемента масиву W. Заповнити матрицю B випадковими числами. Елементи рядків з парними індексами (0, 2, 4, ...) відсортувати за збільшенням, а в рядках з непарними індексами інвертувати порядок елементів.

Розв'язання:

```
using System;
namespace MyLab02
{
    class MyProgram
    {
        static void Main(string[] args)
        {
            // кількість рядків, кількість стовпчиків і коефіцієнт заповнення матриці A
            int rows, cols, fill;
            // посилання на майбутню прямокутну матрицю A
            int[,] A;
            // посилання на одновимірний масив W та "зубчасту" матрицю B
            int[] W;
            int[][] B;

            Console.Clear();
            Console.WriteLine("Введіть кількість рядків і стовпчиків матриці:");
            rows = int.Parse(Console.ReadLine());
            cols = int.Parse(Console.ReadLine());
            // створення прямокутної матриці
            A = new int[rows, cols];

            Console.WriteLine("Введіть коефіцієнт заповнення матриці (0...100 %):");
            fill = int.Parse(Console.ReadLine());

            Random Rnd = new Random();
            // заповнення матриці випадковими числами із заданою щільністю
            int fillcount = fill * rows * cols / 100, pos = 0;
            while (pos < fillcount)
            {
                int rpos = Rnd.Next(A.GetLength(0));
                int cpos = Rnd.Next(A.GetLength(1));
                if (A[rpos, cpos] == 0)
                {
                    A[rpos, cpos] = Rnd.Next(2, 10);
                    ++pos;
```

```
    }
}
// вивід матриці на дисплей
Console.WriteLine("Вміст матриці A:");
for (int i = 0; i < A.GetLength(0); i++)
{
    for (int j = 0; j < A.GetLength(1); j++)
    {
        Console.Write("{0, 4}", A[i, j]);
    }
    Console.WriteLine();
}
Console.WriteLine("Натисніть довільну клавішу...");
// обчислення кількості не нульових елементів у матриці
pos = 0;
foreach (int v in A)
{
    if (v != 0) ++pos;
}
// створення масиву із обчисленою кількістю елементів
W = new int[pos];
// копіювання не нульових елементів з матриці у масив
pos = 0;
foreach (int v in A)
{
    if (v != 0) W[pos++] = v;
}
// вивід масиву на дисплей
Console.WriteLine("Вміст масиву W:");
foreach (int v in W)
{
    Console.Write("{0, 4}", v);
}
Console.WriteLine("\nНатисніть довільну клавішу...");
// створення "зубчастої" матриці
B = new int[W.Length][];
for (int i = 0; i < W.Length; i++)
{
    B[i] = new int[W[i]];
}
// заповнення матриці випадковими числами
for (int i = 0; i < B.Length; i++)
{
    for (int j = 0; j < B[i].Length; j++)
    {
        B[i][j] = Rnd.Next(100);
    }
}
// вивід матриці на дисплей
Console.WriteLine("Вміст матриці B:");
foreach (int[] u in B)
{
    foreach (int v in u)
    {
        Console.Write("{0, 4}", v);
    }
    Console.WriteLine();
}
Console.WriteLine("Натисніть довільну клавішу...");
// перетворення рядків матриці
for (int i = 0; i < B.Length; i++)
{
```

```
            if (i % 2 == 0) Array.Sort(B[i]);
            else Array.Reverse(B[i]);
        }
        // вивід матриці на дисплей
        Console.WriteLine("Вміст перетвореної матриці B:");
        foreach (int[] u in B)
        {
            foreach (int v in u)
            {
                Console.Write("{0, 4}", v);
            }
            Console.WriteLine();
        }
        Console.WriteLine("Натисніть довільну клавішу...");

        Console.ReadKey(true);
    }
  }
}
```

Результати роботи програми представлені на наступному рисунку.

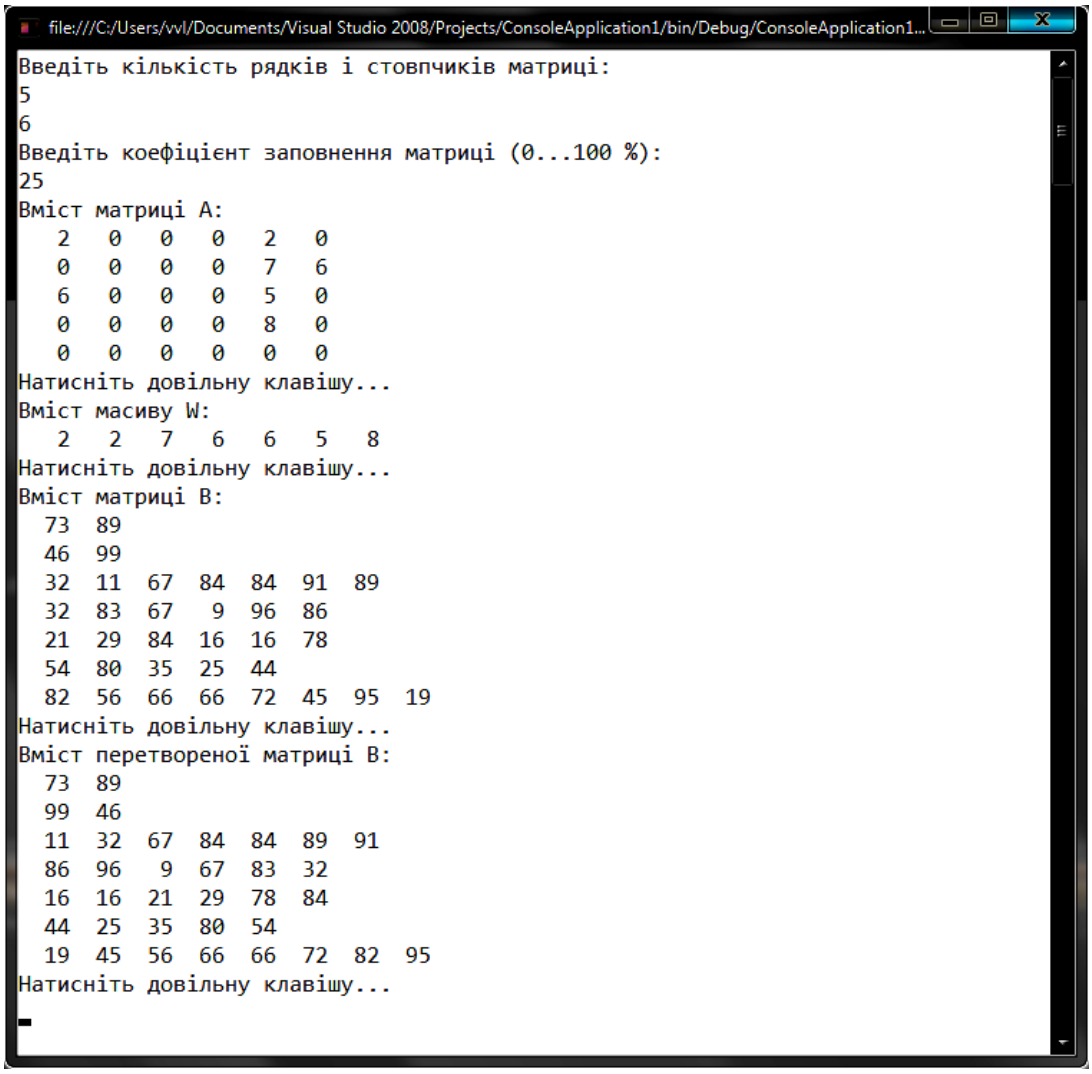

Рисунок 1 – Консольне вікно з результатами роботи програми на контрольному прикладі (кольори інвертовані)

Завдання

1) вивчіть теоретичні відомості та методичні вказівки до лабораторної роботи;

2) виберіть завдання для виконання згідно з варіантом;

3) спроектуйте алгоритм розв’язання завдання та створіть програму, яка його реалізує;

4) відкомпілюйте та відлагодіть програму;

5) дайте відповіді на контрольні запитання;

6) зробіть висновки.

Варіанти завдань

Варіант	Завдання
1	У кожному рядку матриці *A* знайти найменший елемент і поміняти його місцями з першим елементом рядка. Визначити рядок, в якому перший елемент є найбільшим серед перших елементів інших рядків, переписати його елементи в одновимірний масив *B* та інвертувати порядок їх розташування.
2	Елементи одновимірного масиву *S* містять кількості стовпчиків у відповідних за номерами рядках матриці *M*. Створити матрицю *M* і переписати до неї елементи матриці *E*, кількість стовпчиків якої є однаковою і дорівнює найбільшому значенню в *S*. З матриці *E* переписувати лише ті елементи, для яких існують відповідні позиції в матриці *M*. Заповнити нулями той рядок матриці *M*, що має найбільшу довжину.
3	Кожен рядок матриці *A* на першій і останній позиції містить індекси відповідно початку і кінця діапазону елементів рядка, які необхідно переписати у відповідний рядок матриці *B*. Створити матрицю *B* з необхідною кількістю стовпчиків у кожному рядку та переписати до неї вказані елементи з матриці *A*. Відсортувати кожен рядок матриці *A* за зростанням.
4	Одновимірний масив *Z* послідовно заповнити збільшеними на одиницю індексами останніх нулів у кожному рядку матриці *P* (якщо у деякому рядку матриці нулі відсутні, у масив записати число, що дорівнює кількості стовпчиків у матриці *P*). Створити матрицю *Q*, кількість стовпчиків якої у кожному рядку дорівнює відповідному значенню в *Z*, заповнити її випадковими числами та відсортувати кожен рядок за зменшенням.

Варіант	Завдання
5	В матриці C кількість стовпчиків у кожному рядку є випадковим натуральним числом з інтервалу [a; b], але загальна кількість елементів є квадратом натурального числа. Переписати всі елементи з матриці C в одновимірний масив F, відсортувати його за зростанням, після чого переписати його елементи у квадратну матрицю Q по рядках.
6	В матриці W, кількість рядків якої є парним числом, а кількість стовпчиків у кожному рядку є різною, відсортувати рядки по черзі за зростанням та за збуттям. З найбільших елементів кожного рядка матриці W сформувати одновимірний масив T, після чого інвертувати порядок елементів у ньому.
7	Одновимірний масив R заповнений числами, які описують структуру і вміст матриці наступним чином: кількість елементів першого рядка, перелік елементів першого рядка, кількість елементів другого рядка, перелік елементів другого рядка і т. д. Створити прямокутну матрицю H з масиву R, в якій кількість стовпчиків у кожному рядку є однаковою і дорівнює найбільшій вказаній кількості елементів у масиві R. Переписати до матриці H задані елементи з масиву R; не задані елементи залишити нульовими. Відсортувати масив R за зменшенням.
8	У прямокутній матриці B, що містить лише нулі та одиниці, обчислити кількість нулів у кожному рядку і записати отримані кількості в одновимірний масив P. Створити матрицю Z, кількість стовпчиків у кожному рядку якої визначається відповідним значенням з масиву P. В кожен рядок матриці Z записати підряд індекси нулів з відповідних рядків матриці B. Інвертувати порядок елементів у кожному рядку матриці Z.
9	Транспонувати прямокутну матрицю X, заповнену випадковими натуральними числами з інтервалу [a; b]. В отриманій матриці Y поміняти місцями перший рядок з останнім, другий – з передостаннім і т. д. аж до центру матриці. Обнулити ті рядки матриці Y, в який перший нуль зустрічається у першій половині рядка, а інші рядки відсортувати за зменшенням.
10	Перетворити матрицю A, кількість елементів у кожному рядку якої є різною, на одновимірний масив V наступним чином: для кожного елемента матриці в одновимірний масив записуються спочатку його індекси, а потім – значення. Інвертувати порядок елементів масиву V та відновити на його основі прямокутну матрицю B за зворотним принципом.
11	Переписати з кожного рядка матриці P у відповідні рядки матриці Q лише непарні елементи. Відсортувати рядки отриманої матриці Q за зростанням. Сформувати з рядків матриці Q одновимірний масив та визначити ті його елементи, значення яких співпадають із власним індексом.
12	Задані дві квадратні матриці однакового розміру – A і B. Переписати в одновимірний масив R ті елементи матриць A і B, що мають однакові індекси і значення. В отриманому масиві знайти перший (a) і останній (b) індекси заданого користувачем натурального числа s. Якщо число s зустрічається в масиві R не менше двох разів, замінити цим числом елемент матриці A з індексами (a mod N; b mod N), якщо ні – елемент матриці B з індексами (b mod N; a mod N). N – кількість рядків і стовпчиків у матрицях A і B.

Варіант	Завдання
13	Реалізувати додавання двох матриць: S_1, кількість елементів якої у кожному рядку є однаковою та S_2, яка в кожному рядку має різну кількість елементів. Додаються лише елементи на тих позиціях, які існують в кожній матриці, всі інші переходять до сумарної матриці без змін. В отриманій прямокутній сумарній матриці інвертувати порядок елементів кожного рядка.
14	Сформувати квадратну матрицю *C* з одновимірного масиву *X* за наступним правилом: для будь яких $i, j = 1 \ldots N$, де N – розмір масиву, якщо $X_i = X_j$, в елемент C_{ij} записати одиницю, а інакше – нуль. Відсортувати елементи кожного рядка *C* за зменшенням. Створити і заповнити випадковими числами матрицю *Y*, кількість рядків якої дорівнює кількості рядків матриці *X*, а кількість стовпчиків у кожному рядку дорівнює кількості одиниць у відповідному рядку матриці *X*; обчислити суму елементів матриці *Y*.
15	Задані два одновимірні масиви однакової довжини: *R* і *S*. Сформувати квадратну матрицю *A*, кожен елемент якої, що знаходиться в *i*-му рядку та *j*-му стовпчику, дорівнює сумі елементів масива *R* на позиції *i* та масива *S* на позиції *j*. Транспонувати матрицю *A* та інвертувати порядок елементів кожного її рядка, після чого поміняти місцями перший і останній рядок.

Контрольні запитання

1) дайте визначення масиву; перелічіть та поясніть декілька властивостей і методів класу Array;

2) сформулюйте правила створення, ініціалізації та використання одновимірних масивів;

3) поясніть різницю між прямокутними та “зубчастими” багатовимірними масивами;

4) сформулюйте правила створення, ініціалізації та використання прямокутних багатовимірних масивів;

5) сформулюйте правила створення, ініціалізації та використання “зубчастих” багатовимірних масивів;

6) назвіть виключення, яке відбувається при некоректній індексації масиву.