

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**Лабораторна робота №** **3**

з дисципліни “Основи програмування”

тема “Масиви даних”

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | Виконала  студент I курсу | | групи КП-01 | | Пецеля Артем Володимирович | | (*прізвище, ім’я, по батькові*) |   варіант №12 |  | Перевірив  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладач  Гадиняк Руслан Анатолійович  (*прізвище, ім’я, по батькові*) |

Київ 2020

**Мета роботи**

1. Навчитися працювати зі масивами різних типів даних та розмірності.
2. Застосувати на практиці різні види циклічних конструкцій при роботі з одновимірними та багатовимірними масивами даних.

**Постановка завдання**

**Частина 1. Одновимірні масиви**

Дано масив цілих чисел, де кожне число - висота землі.

a) Створити новий масив цілих чисел і записати в нього зсунуті висоти відносно найменшого значення так, щоби воно стало нульовим;

c) Користувач може задати число - висоту рівня води (від 0 до максимальної висоти включно). Створити одновимірний масив цілих чисел, що міститиме висоти стовпців води у кожній з секцій.

d) Нехай 1 клітинка - 1 м3 об'єму і 1 м2 площі поверхні. Знайти і вивести об'єм найбільшої гори (сукупність сусідніх стовпців землі) над рівнем води.

e) Реалізувати функцію, що

* виведе обриси землі (вид збоку) вписану в прямокутник від мінімальної (0) до максимальної висоти
* висоту кожного рівня
* покаже рівень води із заповненням порожнини під рівнем будь-яким спеціальним символом (або кольором фону).

**Частина 2. Двовимірні масиви**

Дано двовимірний масив значень 0 та 1 де 0 - вода, 1 - земля (вид зверху на групу островів).

a) Знайти об'єм найбільшого резервуару води.

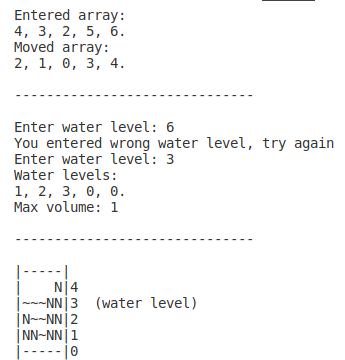
c) Реалізувати функцію, що виводить у консоль вид на острови і воду зверху.

**Текст коду програми**

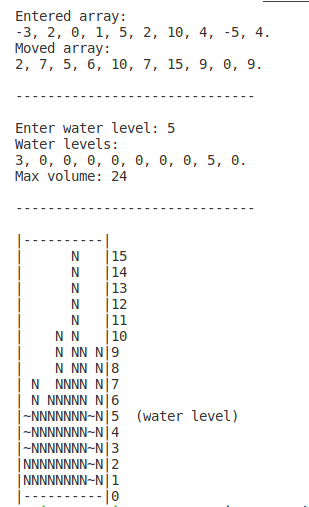
|  |
| --- |
| **Program.cs** |
| using System;  using static System.Console;  using System.Threading;  namespace lab3\_remake  {  class Program  {  static void Main(string[] args)  {  Write("Choose part: ");  string choise = ReadLine();  Clear();  if(choise == "1")  {  int lenth = 3;  bool checker1;  //check if lenth entered correct without closing program  do {  Write("Enter lenth of your array: ");  lenth = int.Parse(ReadLine());  if (lenth < 3)  {  WriteLine("You enter incorrect lenth, try again");  checker1 = false;  }  else  {  checker1 = true;  }  } while (checker1 == false);    int[] input\_array = ReadArray(lenth);  Clear();  WriteLine("Entered array:");  WriteArray(input\_array);  int[] moved\_array = MakeMovedArray(input\_array);  WriteLine("Moved array:");  WriteArray(moved\_array);    WriteLine();  WriteLine("------------------------------");  WriteLine();  int moved\_max = FindMax(moved\_array);  int water\_level = 0;  //check if water level was entered correct without stopping program  bool checker2;  do {  Write("Enter water level: ");  water\_level = int.Parse(ReadLine());  if (water\_level > 0 && water\_level < moved\_max)  {  checker2 = true;  }  else  {  WriteLine("You entered wrong water level, try again");  checker2 = false;  }  } while(checker2 == false);    int[] water\_array = MakeWaterArray(moved\_array, water\_level);  WriteLine("Water levels:");  WriteArray(water\_array);    int max\_volume = FindMaxVolume(moved\_array, water\_array, water\_level);  WriteLine("Max volume: " + max\_volume);  WriteLine();  WriteLine("------------------------------");  WriteLine();  int[,] graphics = MakeGraphicsArray(moved\_array, water\_level);  DrawDiagram(graphics, water\_level);  //Write2dArray(graphics);  }  else if (choise == "2")  {  Write("If you want to generate matrix from example, type '1', if you want to generate random matrix type '2': ");  string choise2 = ReadLine();  int[,] input;  if (choise2 == "1")  {  input = new int[10, 9]  {  {0, 1, 0, 1, 1, 0, 0, 0, 1},  {1, 1, 0, 0, 1, 1, 0, 1, 1},  {0, 1, 0, 1, 1, 1, 0, 1, 0},  {1, 0, 1, 1, 0, 0, 1, 0, 1},  {0, 0, 0, 1, 1, 1, 0, 1, 1},  {0, 0, 1, 0, 1, 0, 1, 1, 0},  {0, 1, 0, 0, 0, 0, 1, 1, 0},  {1, 0, 1, 1, 0, 0, 0, 0, 1},  {1, 0, 1, 1, 1, 1, 1, 0, 1},  {1, 1, 1, 1, 1, 1, 1, 1, 1}  };  }  else if (choise2 == "2")  {  input = Random2dArray();  }  else  {  WriteLine("You entered incorrect number, rerun your app");  return;  }  Clear();  WriteLine("Input matrix: ");  WriteLine();  Write2dArray(input);  WriteLine();  WriteLine();  DrawDiagram(input);  input = InvertArray(input);  input = MakeNumMatrix(input);    int[] counters = FillCounters(input);  int changes = 0;  do {  input = MakeChanges(input, out changes);  counters = FillCounters(input);  } while (changes != 0);  WriteLine();  WriteLine("The biggest volume of water reservoir: " + FindMax(counters));  }  else  {  WriteLine("You entered wrong part, rerun app.");  }  }  static int[] ReadArray(int lenth)  {  int[] input = new int[lenth];  for (int i = 0; i < lenth; i++)  {  Write("{0} element: ", i + 1);  input[i] = int.Parse(ReadLine());  }  return input;  }  static void WriteArray(int[] array)  {  for (int i = 0; i < array.Length; i++)  {  Write(array[i]);  if (i != array.Length -1)  {  Write(", ");  }  else  {  Write(".");  }  }  WriteLine();  }  static void WriteDoubleArray(double[] array)  {  for (int i = 0; i < array.Length; i++)  {  Write(array[i]);  if (i != array.Length -1)  {  Write(", ");  }  else  {  Write(".");  }  }  WriteLine();  }  static void Write2dArray(int[,] array)  {  for (int i = 0; i < array.GetLength(0); i++)  {  for (int j = 0; j < array.GetLength(1); j++)  {  Write(array[i,j] + " ");  }  WriteLine();  }  }  static int FindMax(int[] array)  {  int max = int.MinValue;  for (int i = 0; i < array.Length; i++)  {  if (max < array[i])  {  max = array[i];  }  }  return max;  }  static int FindMin(int[] array)  {  int min = int.MaxValue;  for (int i = 0; i < array.Length; i++)  {  if (min > array[i])  {  min = array[i];  }  }  return min;  }  static int FindMinNonZero(int[] array)  {  int min = int.MaxValue;  bool check = false;  for (int i = 0; i < array.Length; i++)  {  if (min > array[i] && array[i] != 0)  {  min = array[i];  check = true;  }  }  if (check)  {  return min;  }  else  {  return 0;  }  }  static int FindMax(int[,] array)  {  int height = array.GetLength(0);  int lenth = array.GetLength(1);  int max = int.MinValue;  for (int i = 0; i < height; i++)  {  for (int j = 0; j < lenth; j++)  {  if (array[i,j] > max)  {  max = array[i,j];  }  }  }  return max;  }  static int[] MakeMovedArray(int[] array)  {  int input\_min = FindMin(array);  int[] moved\_array = new int[array.Length];  for (int i = 0; i < array.Length; i++)  {  moved\_array[i] = array[i] - input\_min;  }  return moved\_array;  }  static int[] MakeWaterArray(int[] array, int water\_level)  {  int[] water\_array = new int[array.Length];  for (int i = 0; i < array.Length; i++)  {  if (array[i] < water\_level)  {  water\_array[i] = water\_level - array[i];  }  else  {  water\_array[i] = 0;  }  }  return water\_array;  }  static int FindMaxVolume(int[] moved\_array, int[] water\_array, int water\_level)  {  int max\_volume = int.MinValue;  int volume = 0;  for(int i = 0; i < water\_array.Length; i++)  {  if (water\_array[i] == 0)  {  volume += moved\_array[i] - water\_level;  }  else if (max\_volume < volume)  {  max\_volume = volume;  volume = 0;  }  }  if (max\_volume < volume)  {  max\_volume = volume;  }  return max\_volume;  }  static int[,] MakeGraphicsArray(int[] moved\_array, int water\_level)  {  int height = FindMax(moved\_array);  int width = moved\_array.Length;  int[,] graphics = new int[height, width];  for (int j = 0; j < width; j++)  {  for (int i = 0; i < height; i++)  {  if (i >= height - moved\_array[j])  {  graphics[i,j] = 1;  }  else if (i >= height - water\_level)  {  graphics[i,j] = 2;  }  }  }  return graphics;  }  static void DrawDiagram(int[,] graphics, int water\_level)  {  int height = graphics.GetLength(0);  int width = graphics.GetLength(1);  int top = CursorTop;  //rectangle  for (int i = 1; i <= width; i++)  {  SetCursorPosition(i, top);  Write("-");  SetCursorPosition(i, top + height + 1);  Write("-");  Thread.Sleep(100);  }  for (int j = 0; j <= height + 1; j++)  {  SetCursorPosition(0, top + j);  Write("|");  SetCursorPosition(width + 1, top + j);  Write("|");  if (height - j != -1)  {  SetCursorPosition(width + 2, top + j + 1);  Write(height - j);  }  if (height - j == water\_level)  {  SetCursorPosition(width + 5, top + j + 1);  Write("(water level)");  }  Thread.Sleep(100);  }  for (int j = 0; j < width; j++)  {  for (int i = 0; i < height; i++)  {  SetCursorPosition(j + 1, top + i + 1);  if (graphics[i,j] == 1)  {  Write("N");  }  else if (graphics[i,j] == 2)  {  Write("~");  }  Thread.Sleep(100);  }  }  WriteLine();  WriteLine();  }  static void DrawDiagram(int[,] graphics)  {  int height = graphics.GetLength(0);  int width = graphics.GetLength(1);  int top = CursorTop;  //rectangle  for (int i = 1; i <= width; i++)  {  SetCursorPosition(i, top);  Write("-");  SetCursorPosition(i, top + height + 1);  Write("-");  Thread.Sleep(100);  }  for (int j = 0; j <= height + 1; j++)  {  SetCursorPosition(0, top + j);  Write("|");  SetCursorPosition(width + 1, top + j);  Write("|");  Thread.Sleep(100);  }  for (int j = 0; j < width; j++)  {  for (int i = 0; i < height; i++)  {  SetCursorPosition(j + 1, top + i + 1);  if (graphics[i,j] == 1)  {  Write("N");  }  else  {  Write("~");  }  Thread.Sleep(100);  }  }  WriteLine();  WriteLine();  }  static int[,] Random2dArray()  {  Random rand = new Random();  int lenth = rand.Next(5, 21);  int height = rand.Next(5, 21);  int[,] array = new int[height, lenth];  for (int i = 0; i < height; i++)  {  for (int j = 0; j < lenth; j++)  {  array[i,j] = rand.Next(0, 2);  }  }  return array;  }  static int[,] InvertArray(int[,] array)  {  int height = array.GetLength(0);  int lenth = array.GetLength(1);  for (int i = 0; i < height; i++)  {  for (int j = 0; j < lenth; j++)  {  if (array[i,j] == 0)  {  array[i,j] = 1;  }  else  {  array[i,j] = 0;  }  }  }  return array;  }  static int[,] MakeNumMatrix(int[,] array)  {  int height = array.GetLength(0);  int lenth = array.GetLength(1);  int counter = 1;  for (int i = 0; i < height; i++)  {  for (int j = 0; j < lenth; j++)  {  if (array[i,j] == 1)  {  array[i,j] = counter;  counter++;  }  }  }  return array;  }  static int[] FillCounters(int[,] array)  {  int height = array.GetLength(0);  int lenth = array.GetLength(1);  int[] counters = new int[FindMax(array) + 1];  for (int i = 0; i < height; i++)  {  for (int j = 0; j < lenth; j++)  {  if (array[i,j] != 0)  {  counters[array[i,j]] += 1;  }  }  }  return counters;  }  static int[,] MakeChanges(int[,] array, out int changes)  {  int height = array.GetLength(0);  int lenth = array.GetLength(1);  changes = 0;  for (int i = 0; i < height; i++)  {  for (int j = 0; j < lenth; j++)  {  if (array[i,j] != 0)  {  //upper border  if (i - 1 == -1)  {  //left upper corner  if (j - 1 == -1)  {  int down = array[i + 1, j];  int right = array[i, j + 1];    int[] dimentions = new int[] {down, right};  int min = FindMinNonZero(dimentions);  if (min != 0 && min < array[i,j])  {  array[i,j] = min;  changes++;  }  }  //right upper corner  else if (j + 1 == lenth)  {  int down = array[i + 1, j];  int left = array[i, j - 1];  int[] dimentions = new int[] {down, left};  int min = FindMinNonZero(dimentions);  if (min != 0 && min < array[i,j])  {  array[i,j] = min;  changes++;  }  }  //non-corner upper border  else  {  int down = array[i + 1, j];  int right = array[i, j + 1];  int left = array[i, j - 1];  int[] dimentions = new int[] {down, left, right};  int min = FindMinNonZero(dimentions);  if (min != 0 && min < array[i,j])  {  array[i,j] = min;  changes++;  }  }  }  //lower border  else if (i + 1 == height)  {  //left lower corner  if (j - 1 == -1)  {  int up = array[i - 1, j];  int right = array[i, j + 1];  int[] dimentions = new int[] {up, right};  int min = FindMinNonZero(dimentions);  if (min != 0 && min < array[i,j])  {  array[i,j] = min;  changes++;  }  }  //right lower corner  else if (j + 1 == lenth)  {  int up = array[i - 1, j];  int left = array[i, j - 1];  int[] dimentions = new int[] {up, left};  int min = FindMinNonZero(dimentions);  if (min != 0 && min < array[i,j])  {  array[i,j] = min;  changes++;  }  }  //non-corner lower border  else  {  int up = array[i - 1, j];  int left = array[i, j - 1];  int right = array[i, j + 1];  int[] dimentions = new int[] {up, left, right};  int min = FindMinNonZero(dimentions);  if (min != 0 && min < array[i,j])  {  array[i,j] = min;  changes++;  }  }  }  //non-corner left border  else if (j - 1 == -1)  {  int up = array[i - 1, j];  int right = array[i, j + 1];  int down = array[i + 1, j];  int[] dimentions = new int[] {up, down, right};  int min = FindMinNonZero(dimentions);  if (min != 0 && min < array[i,j])  {  array[i,j] = min;  changes++;  }  }  //non-corner right border  else if (j + 1 == lenth)  {  int up = array[i - 1, j];  int left = array[i, j - 1];  int down = array[i + 1, j];  int[] dimentions = new int[] {up, down, left};  int min = FindMinNonZero(dimentions);  if (min != 0 && min < array[i,j])  {  array[i,j] = min;  changes++;  }  }  //not at border  else  {  int up = array[i - 1, j];  int left = array[i, j - 1];  int down = array[i + 1, j];  int right = array[i, j + 1];  int[] dimentions = new int[] {up, down, left, right};  int min = FindMinNonZero(dimentions);  if (min != 0 && min < array[i,j])  {  array[i,j] = min;  changes++;  }  }  }  }  }  return array;  }  }  } |

**Приклади результатів**

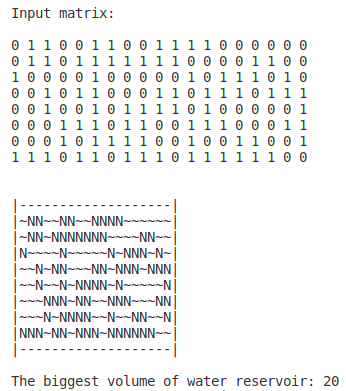
**Приклад 1. Частина 1.** Виконання першої частини з масивом, заданим користувачем. Після неправильного вводу рівня води користувачу пропонується ввести ще раз.



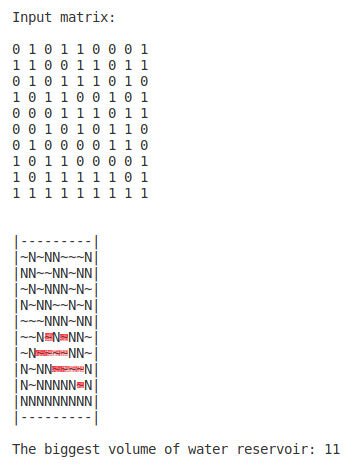
**Приклад 2. Частина 1.** Виконання першої частини з більшим масивом та більшими числами.



**Приклад 3. Частина 2.** Виконання другої частини з псевдовипадковою заданою матрицею.



**Приклад 4. Частина 2.** Виконання другої частини з використанням матриці з прикладу. На рисунку виділений об’єм найбільшого водного резервуару.



**Висновки**

Виконавши дану лабораторну роботу було проведено дії над одно- та двовимірними масивами. Було зроблено пошук різних елементів у середні масиву (максимальне, мінімальне і т.д.).

На практиці було застосовані різні циклічні конструкції, а саме for та foreach. Також було застосовано Length та GetLength() для визначення розмірності(ей) масивів.

Компіляція коду відбувалася за допомогою утиліти dotnet.