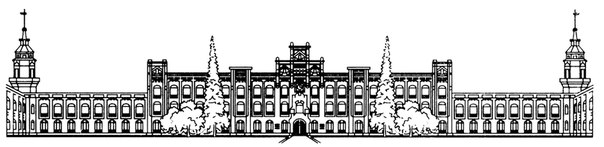
МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

Національний технічний університет України

"Київський політехнічний інститут імені Ігоря Сікорського"



Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

Програмування. Частина 2. Основи програмування

Практична робота №4

Тема: Масиви

Варіант №11

Виконав: Коваль Богдан

студент групи ІС-31

Перевірив:

Майер І.С.

Київ-2024

**Тема:** Управління потоком виконання

**Хід роботи:**

1. Повторити теоретичні відомості

2. Виконати три завдання з таблиці 2 відповідно до свого варіанту у таблиці 1.

- Кожне завдання має бути реалізовано як окремий клас.

- Кожен клас має складатись щонайменше з двох методів:

- public static void Main(string[] agrs) - точка входу. Містить код, що кілька разів

знаходить результат завдання при різних значеннях аргументів та параметрів. Для перевірки мають бути присутні як дозволені так і заборонені комбінації аргументів та

параметрів.

- Метод, що реалізує задане завдання. Метод має перевіряти аргументи і параметри

функції та у разі їх помилковості аварійно закінчувати свою роботу шляхом

викидання стандартного виключення ArgumentOutOfRangeException (дивись

приклад). В жодному разі цей метод не повинен напряму взаємодіяти з

користувачем через консоль або інший UI (ніколи не змішуйте бізнес-логіку та

користувацький інтерфейс).

- Клас може містити інші допоміжні методи.

**3. Відповісти на контрольні питання**

****

****

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Labs\_4

{

internal class Task\_1

{

static void Main(string[] args)

{

double[] numbers\_1 = { 2, 1, 3, 6, 8 };

PrintResult(numbers\_1);

double[] numbers\_2 = {5, 1, 3, 7, 8};

PrintResult(numbers\_2);

double[] numbers\_3 = {2, 1, 4, 4, 3};

PrintResult(numbers\_3);

}

public static double sumMinMax(double[] numbers)

{

double elemMin = numbers[0];

double elemMax = numbers[0];

foreach (double value in numbers) {

elemMin = (elemMin < value) ? elemMin : value;

elemMax = (elemMax > value) ? elemMax : value;

}

return (elemMin + elemMax);

}

static void PrintResult(double[] Array)

{

Console.WriteLine("\nArray");

foreach (double value in Array) Console.Write($"{value} ");

Console.WriteLine("\nResult");

try

{

Console.WriteLine(sumMinMax(Array));

}

catch (IndexOutOfRangeException e)

{

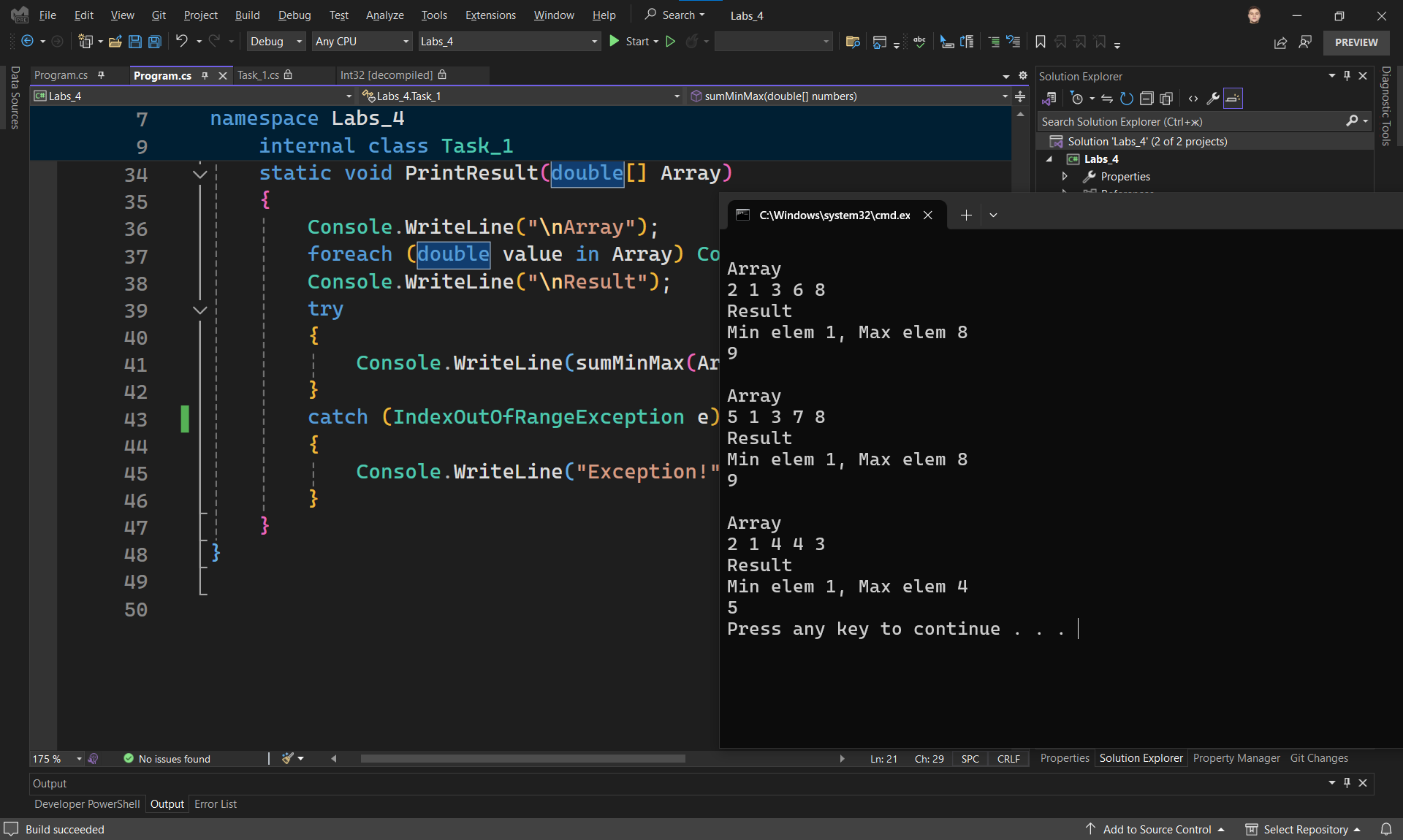
Console.WriteLine("Exception!", e.Message);

}

}

}

}





using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Task\_2

{

internal class Task\_2

{

static void Main(string[] args)

{

double[] numbers = { 4, 3, 1, -8 };

double[] numbers2 = { 5, -8, 7, 2 };

double[] numbers3 = { 3, 1, 4, 9, 7 };

PrintResult(numbers);

PrintResult(numbers2);

PrintResult(numbers3);

}

public static double AvergMinMax(double[] numbers)

{

double tempNum;

for (int i = 0; i < numbers.Length - 1; i++)

{

for (int j = i; j < numbers.Length; j++)

{

if (numbers[i] > numbers[j])

{

tempNum = numbers[i];

numbers[i] = numbers[j];

numbers[j] = tempNum;

}

}

}

return ((numbers[numbers.Length - 1] + numbers[2]) / 2);

}

static void PrintResult(double[] Array)

{

try

{

Console.WriteLine(AvergMinMax(Array));

}

catch (IndexOutOfRangeException ex)

{

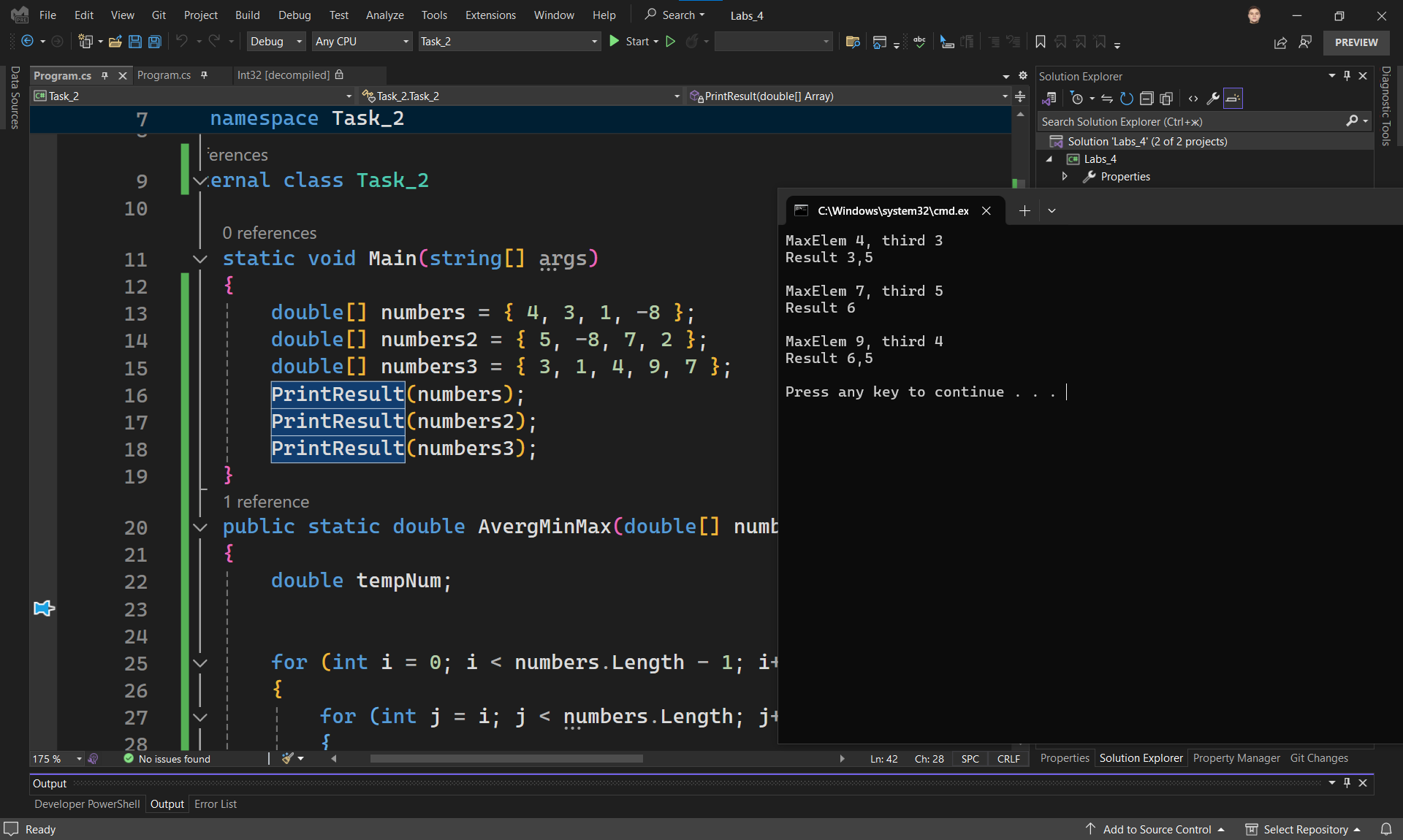
Console.WriteLine(ex.ToString());

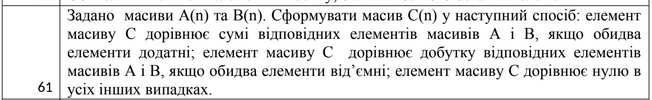
}

}

}

}

****

****

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.InteropServices;

using System.Text;

using System.Threading.Tasks;

namespace Task\_3

{

internal class Task\_3

{

static void Main(string[] args)

{

int[] number\_1 = { 3, 4, 2, 1, 7 };

int[] number\_2 = { 6, 2, 1, 8, 9, 5 };

int[] numbers\_3 = { 3, 5, 8, 7, 2 };

PrintResult(number\_1);

PrintResult(number\_2);

PrintResult(numbers\_3);

}

public static int[] ResultCal(int[] array)

{

int[] resultArray = new int[array.Length];

for (int i = 0; i < array.Length; i++)

{

if (array[i] < 0)

resultArray[i] = array[i] \* (-1);

else if (array[i] > 0)

resultArray[i] = array[i] - 3;

else

resultArray[i] = -2;

}

return resultArray;

}

static void PrintResult(int[] numbers)

{

int[] resultArray;

try

{

resultArray = ResultCal(numbers);

Console.WriteLine("\nData: ");

foreach (int elem in numbers)

{

Console.Write($"{elem} ");

}

Console.WriteLine("\nResult:");

foreach (int elem in resultArray)

{

Console.Write($"{elem} ");

}

Console.WriteLine();

}

catch(Exception ex)

{

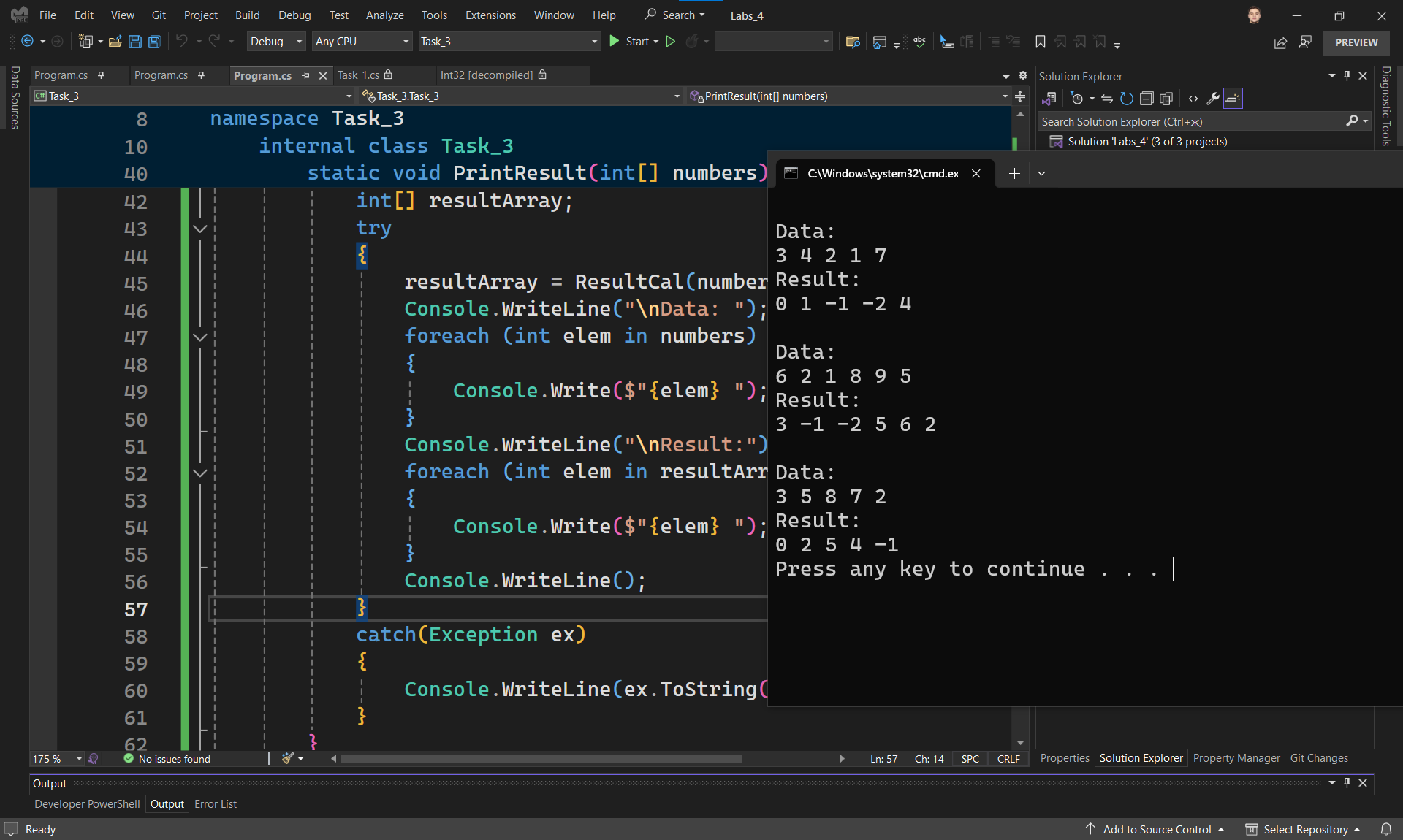
Console.WriteLine(ex.ToString());

}

}

}

}

****

**Контрольні питання:**

1. Змінна і масив - це обидва типи даних у програмуванні, але вони відрізняються в своїй природі. Змінна зберігає одне значення, тоді як масив - це структура даних, яка зберігає послідовність елементів одного типу.
2. Стек і купа - це дві різні структури даних для зберігання та організації даних в пам'яті комп'ютера. Стек - це структура даних типу "Last In, First Out" (LIFO), тобто останній елемент, який був доданий в стек, буде першим, що буде вилучено. Купа (також відома як "heap") - це динамічна область пам'яті, де зберігаються об'єкти, доступ до яких може бути отриманий в будь-якому порядку.
3. Змінна може бути розташована у стеку або у купі, залежно від того, як вона була створена і яка її область видимості. Масив також може бути розташований у стеку або у купі, в залежності від того, чи є він локальною змінною чи динамічно виділеним об'єктом.
4. Посилання на масив - це адреса в пам'яті, яка вказує на початок масиву. Він відрізняється від самого масиву тим, що це не сам масив, а лише посилання на нього. Посилання на масив може бути розташовано у стеку або у купі, так само, як і сам масив.
5. Індекси першої та останньої комірок у масиві зазвичай починаються з 0 і закінчуються на n-1, де n - це розмір масиву. Таким чином, для масиву з 10 комірок перший індекс буде 0, а останній - 9.
6. Якщо звернутися до комірки, яка не існує у масиві, це може призвести до виникнення помилки виконання програми, залежно від мови програмування. У багатьох мовах програмування це може викликати виняток або неочікувану поведінку програми.
7. Поведінка при створенні нового масиву без явної ініціалізації може відрізнятися залежно від мови програмування та середовища виконання. У деяких випадках усі комірки масиву будуть проініціалізовані спеціальними значеннями за замовчуванням (наприклад, 0 для числових типів даних, null для посилань тощо), в інших - вони можуть містити довільні значення, що знаходяться в цей момент у відповідній області пам'яті.
8. Номер першої комірки масиву визначається як 0, а номер останньої - на одиницю менше за загальну кількість комірок у масиві. Якщо відомо лише посилання на масив, можна використати це посилання для доступу до першої та останньої комірок шляхом використання індексів.
9. Розмір масиву можна змінити, створивши новий масив з бажаним розміром і копіюючи дані зі старого масиву у новий. Це означає виділення нової області пам'яті та копіювання елементів. У деяких мов програмування також може бути доступна функція зміни розміру масиву, яка автоматично виконує цей процес.
10. При копіюванні посилання на масив створюється нове посилання, яке вказує на ту саму область пам'яті, де зберігається масив. Жодних нових даних не створюється, а просто створюється додатковий спосіб доступу до існуючих даних.
11. При втраті посилання на масив (наприклад, коли воно виходить за межі області видимості або коли воно переприсвоюється на інший об'єкт) масив залишається в пам'яті, поки його область видимості існує. Якщо на масив більше немає посилань, він стає недосяжним для програми, і його пам'ять може бути відведена системі для подальшого використання.
12. Конструкція "for" використовується для ітерації по масиву, вказуючи явно індекси. Конструкція "for-each" (або "enhanced for loop" у Java) автоматично ітерується по всіх елементах масиву без необхідності вказувати індекси. Перевагою "for" є більша гнучкість у роботі з індексами, що може бути корисно у складних обчисленнях. "For-each" зазвичай більш зручний у випадках, коли потрібно просто переглянути або змінити кожен елемент масиву.
13. У масив типу "double[]" можна записати значення типу "int", але вони будуть автоматично приведені до типу "double". У масив типу "int[]" неможливо записати значення типу "double", оскільки це призведе до втрати точності.