Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

ЛАБОРАТОРНА РОБОТА № 5

з дисципліни «Алгоритми та методи обчислення»

на тему «Розв’язання систем лінійних алгебраїчних рівнянь»

ВИКОНАВ:

cтудент ІІ курсу ФІОТ

групи ІО-91

Діденко Владислав Віталійович

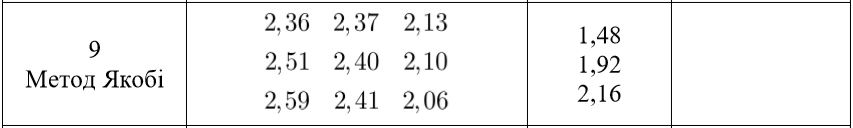
Варіант – 9

ПЕРЕВІРИВ:

ст.вик. Порєв В. М.

Київ – 2021

**Мета:** Вивчити алгоритми методів розв'язання систем лінійних алгебраїчних рівнянь на ЕОМ.

**Завдання:** Відповідно до варіанту завдання скласти схему алгоритму розв’язання систем лінійних алгебраїчних рівнянь зазначеним у варіанті методом. Відповідно до блок-схеми скласти програму розв'язання систем лінійних алгебраїчних рівнянь алгоритмічною мовою, узгодженою з викладачем. Розв’язати СЛАР на комп’ютері відповідно до варіанту. 

**Текст програми:**

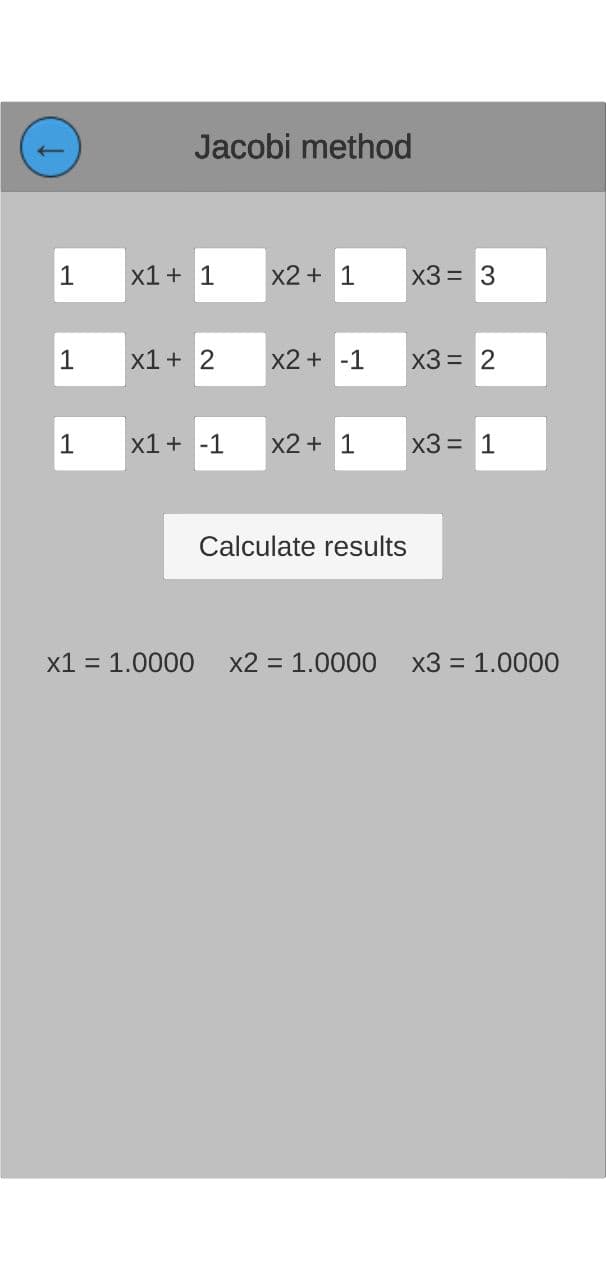
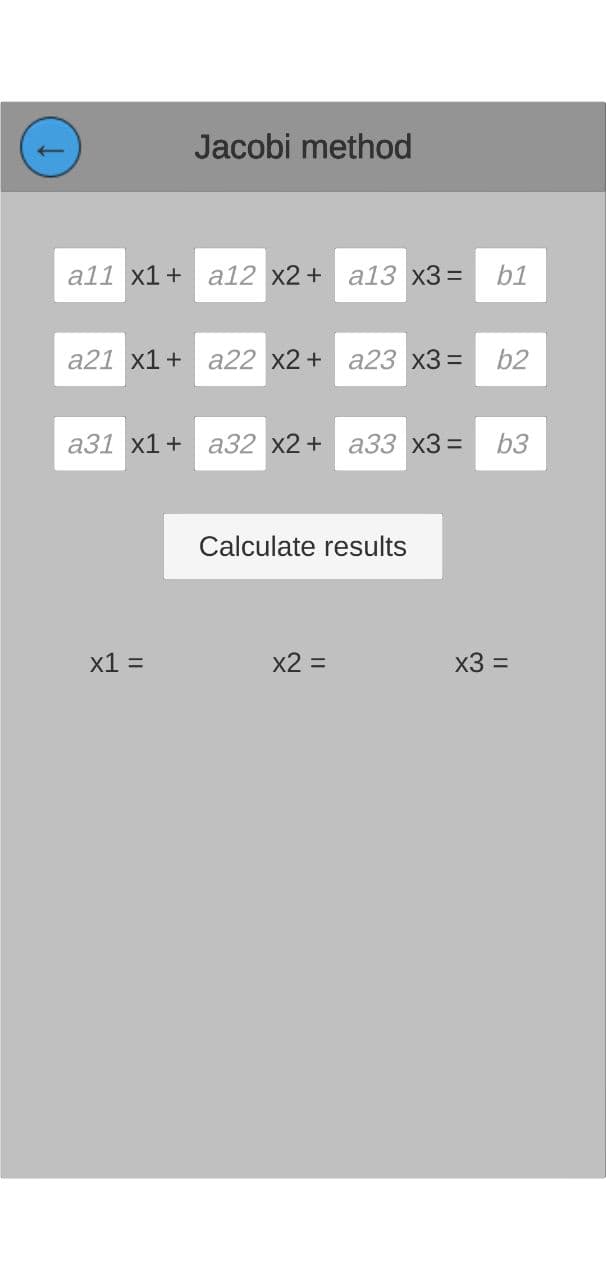
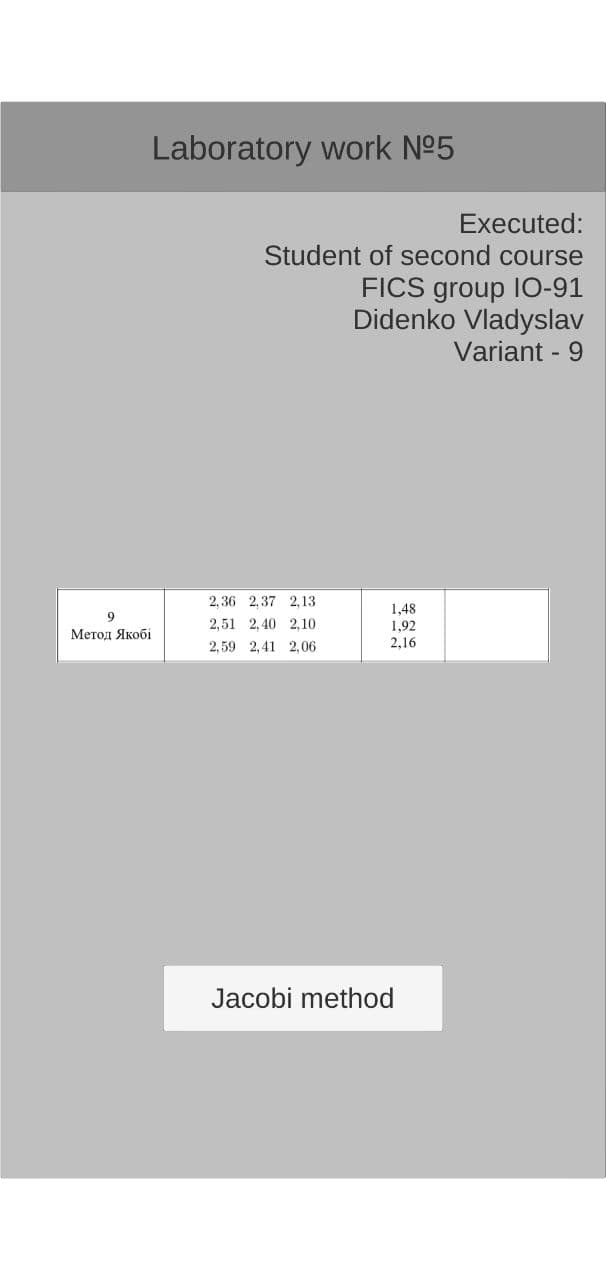
CanvasController.cs

using System;  
using System.Collections;  
using System.Collections.Generic;  
using System.Globalization;  
using System.Linq;  
using AwesomeCharts;  
using UnityEngine;  
using UnityEngine.UI;  
  
public class **CanvasController** : MonoBehaviour {  
 public static CanvasController instance { get; private set; }  
 private const float DEFAULT\_ACCURACY = 0.001f;  
 private const float ERROR\_MESSAGE\_TIME = 2f;  
 private const string INPUT\_ERROR\_MESSAGE = "Wrong input";  
  
 public GameObject **info**;  
 public GameObject **jacobiMethodGO**;  
  
 public GameObject **error**;  
 public Image **errorImage**;  
 public Text **errorText**;  
 private Coroutine errorCoroutine;  
  
 public InputField[] **inputMatrix**;  
 public InputField[] **inputAdditional**;  
 public Text[] textResults;  
  
 private double[,] matrix = new double[3, 3];  
 private double[] additional = new double[3];  
 private int numOfDimension;  
  
  
 private void ShowErrorMessage(string message) {  
 if (errorCoroutine != null) {  
 StopCoroutine(errorCoroutine);  
 }  
  
 errorCoroutine = StartCoroutine(\_ShowErrorMessage(message));  
 }  
  
 private IEnumerator \_ShowErrorMessage(string message) {  
 error.SetActive(true);  
 errorText.text = message;  
 var time = 0f;  
 while (time < ERROR\_MESSAGE\_TIME) {  
 errorImage.color = Color.Lerp(Color.red, Color.clear, time / ERROR\_MESSAGE\_TIME);  
 time += Time.deltaTime;  
 yield return null;  
 }  
  
 error.SetActive(false);  
 errorCoroutine = null;  
 }  
  
 public void RaiseAndShowError(string message) {  
 ShowErrorMessage(message);  
 }  
  
 private void **Awake**() {  
 instance = this;  
 CultureInfo.CurrentCulture = CultureInfo.GetCultureInfo("en-US");  
 }  
  
 private void **Start**() {  
 SetDefault();  
 numOfDimension = (int) Mathf.Sqrt(inputMatrix.Length);  
 var test1 = new double[,] {{1, 1, 1}, {1, -1, 1}, {1, -1, -1}};  
 var test2 = new double[] {3, 1, -1};  
 JacobiMethod.instance.SetData(test1, test2, DEFAULT\_ACCURACY);  
 JacobiMethod.instance.CalculateMatrix();  
 }  
  
 public void **SetDefault**() {  
 info.SetActive(true);  
 jacobiMethodGO.SetActive(false);  
 }  
  
 private void OnClick(GameObject other) {  
 info.SetActive(false);  
 other.SetActive(true);  
 }  
  
 public void **OnJacobiMethod**() {  
 OnClick(jacobiMethodGO);  
 }  
  
 private bool TryReadInput() {  
 for (var i = 0; i < inputMatrix.Length; i++) {  
 var \_i = i / numOfDimension;  
 var \_j = i % numOfDimension;  
 if (!double.TryParse(inputMatrix[i].text, NumberStyles.Float, CultureInfo.InvariantCulture,  
 out matrix[\_i, \_j])) {  
 return false;  
 }  
 }  
 for (var i = 0; i < inputAdditional.Length; i++) {  
 if (!double.TryParse(inputAdditional[i].text, NumberStyles.Float, CultureInfo.InvariantCulture,  
 out additional[i])) {  
 return false;  
 }  
 }  
  
 return true;  
 }  
  
  
 public void **OnFindResults**() {  
 if (!TryReadInput()) {  
 RaiseAndShowError(INPUT\_ERROR\_MESSAGE);  
 return;  
 }  
  
 PrintResults(FindResults());  
 }  
  
 private double[] FindResults() {  
 JacobiMethod.instance.SetData(matrix, additional, DEFAULT\_ACCURACY);  
 return JacobiMethod.instance.CalculateMatrix();  
 }  
  
 private void PrintResults(double[] results) {  
 for (var i = 0; i < textResults.Length; i++) {  
 textResults[i].text = $"x{i+1} = {results[i]:f4}";  
 }  
 }  
}

JacobiMethod.cs

using System;  
using System.Threading;  
using UnityEngine;  
  
class JacobiMethod {  
 public static JacobiMethod instance { get; } = new JacobiMethod();  
 private double[,] matrix;  
 private double[] additional;  
  
 private double \_accuracy;  
  
 public double Accuracy {  
 get => \_accuracy;  
 set => \_accuracy = value <= 0.0 ? 0.1 : value;  
 }  
  
 private JacobiMethod() { }  
   
  
 public void SetData(double[,] Matrix, double[] FreeElements, double Accuracy) {  
 matrix = Matrix;  
 additional = FreeElements;  
 this.Accuracy = Accuracy;  
 }  
  
 public double[] CalculateMatrix() {  
 var n = additional.Length;  
 var x = new double[n];  
 for (var i = 0; i < n; i++) {  
 x[i] = 0;  
 }  
  
 for (var k = 0; k < n - 1; k++) {  
 for (var i = k + 1; i < n; i++) {  
 for (var j = k + 1; j < n; j++) {  
 matrix[i, j] = matrix[i, j] - matrix[k, j] \* (matrix[i, k] / matrix[k, k]);  
 }  
  
 additional[i] = additional[i] - additional[k] \* matrix[i, k] / matrix[k, k];  
 }  
 }  
  
 for (var k = n - 1; k >= 0; k--) {  
 double s = 0;  
 for (var j = k + 1; j < n; j++)  
 s += matrix[k, j] \* x[j];  
 x[k] = (additional[k] - s) / matrix[k, k];  
 }  
  
 return x;  
 }  
}

**Скріншоти виконання:**

****

**Висновок:** У ході виконання лабораторної роботи я закріпив знання з базових понять алгоритмів розв’язання систем лінійних алгебраїчних рівнянь, а також закодував алгоритм Якобі. Отримані результати виконання програми є вірними.