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 № 1

з дисципліни «Алгоритми та методи обчислення»

на тему «Поняття алгоритму. Задавання алгоритмів у вигляді блок-схем»

ВИКОНАВ:

Студент І курсу ФІОТ

групи ІО-91

Діденко Владислав Віталійович

Варіант – 9

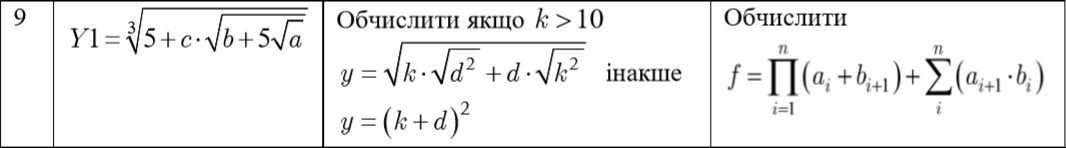
ПЕРЕВІРИВ:

ст.вик. Порєв В. М.

Київ – 2021

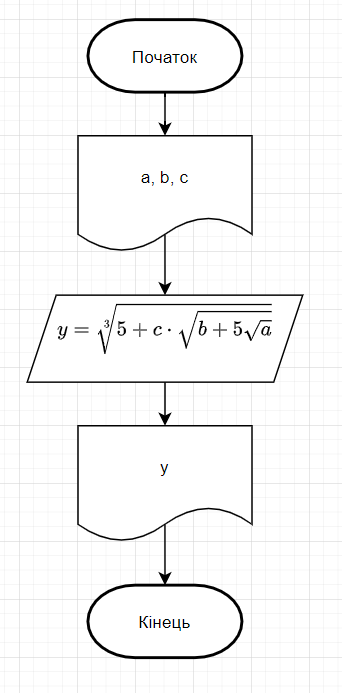
**Мета:** Навчитися створювати блок-схеми лінійного алгоритму; розгалуженого алгоритму та циклічного алгоритму за допомогою редактора блок-схем afce або іншого довільного редактора.

**Завдання:** Відповідно до варіанту завдання розробити блок-схеми обчислення виразів для лінійного алгоритму, алгоритму, що розгалужується та циклічного алгоритму. У відповідності до блок-схеми створити програму обчислення виразу алгоритмічною мовою, узгодженою з викладачем.

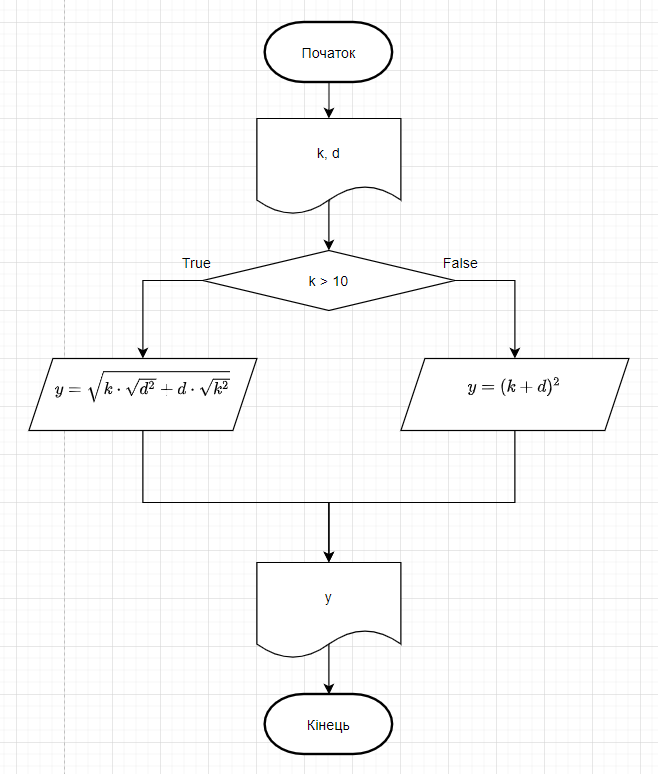


**Блок схеми:**

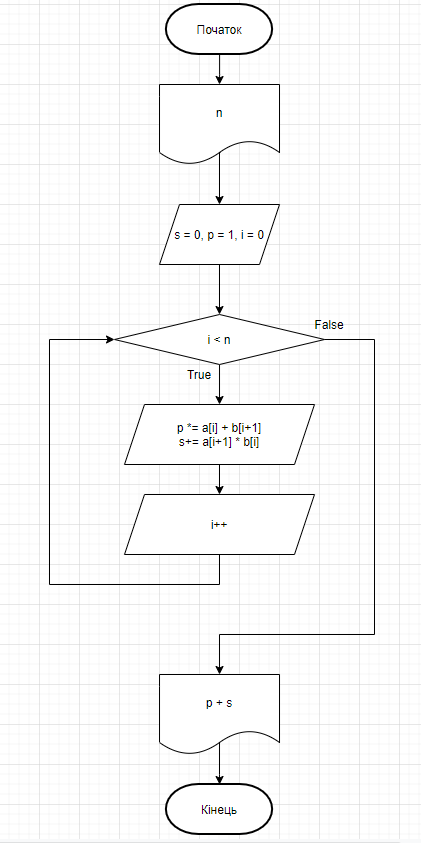
Лінійний алгоритм



Розгалужений алгоритм



Циклічний алгоритм



**Текст програми:**

LinearAlgorithm.cs

using UnityEngine;  
  
public class LinearAlgorithm {  
 private float a;  
 private float b;  
 private float c;  
  
 public LinearAlgorithm(float a, float b, float c) {  
 this.a = a;  
 this.b = b;  
 this.c = c;  
 }  
  
 public float FindResult() {  
 return Mathf.Pow(5 + c \* Mathf.Sqrt(b + 5 \* Mathf.Sqrt(a)), 1 / 3f);  
 }  
}

RamifiedAlgorithm.cs

using UnityEngine;  
  
public class RamifiedAlgorithm {  
 private float k;  
 private float d;  
  
 public RamifiedAlgorithm(float k, float d) {  
 this.k = k;  
 this.d = d;  
 }  
  
 public float FindResult() {  
 return k > 10  
 ? Mathf.Sqrt(k \* Mathf.Sqrt(Mathf.Pow(d, 2)) + d \* Mathf.Sqrt(Mathf.Pow(k, 2)))  
 : Mathf.Pow(k + d, 2);  
 }  
}

CyclicAlgorithm.cs

using System.Collections.Generic;  
using UnityEngine;  
  
public class CyclicAlgorithm {  
 private int n;  
 private float min;  
 private float max;  
 private List<float> a = new List<float>();  
 private List<float> b = new List<float>();  
  
 public static bool CheckVars(int n, float min, float max) {  
 return n >= 1 && !(min > max);  
 }  
  
 public CyclicAlgorithm(int n, float min, float max) {  
 this.n = n;  
 this.min = min;  
 this.max = max;  
 for (var i = 0; i <= n; i++) {  
 a.Add(Random.Range(min, max));  
 b.Add(Random.Range(min, max));  
 }  
 }  
  
 public float FindResult() {  
 var p = 1f;  
 var s = 0f;  
 for (var i = 0; i < n; i++) {  
 p \*= a[i] + b[i + 1];  
 s += a[i + 1] \* b[i];  
 }  
  
 return p + s;  
 }  
}

CanvasController.cs

using System;  
using System.Collections;  
using System.Collections.Generic;  
using System.Globalization;  
using UnityEngine;  
using UnityEngine.UI;  
  
public class **CanvasController** : MonoBehaviour {  
 public static CanvasController instance;  
 private const string inputErrorMessage = "Wrong input";  
 private const string resultErrorMessage = "Result is not a number (NaN)";  
 public GameObject **info**;  
 public GameObject **linear**;  
 public GameObject **ramified**;  
 public GameObject **cyclic**;  
 public GameObject **error**;  
 public InputField **a**;  
 public InputField **b**;  
 public InputField **c**;  
 public InputField **k**;  
 public InputField **d**;  
 public InputField **n**;  
 public InputField **min**;  
 public InputField **max**;  
 public Text **resultLinear**;  
 public Text **resultRamified**;  
 public Text **resultCyclic**;  
 private Text errorText;  
 private Coroutine errorCoroutine;  
   
  
 public void ShowMessage(string message) {  
 if (errorCoroutine != null) {  
 StopCoroutine(errorCoroutine);  
 }  
  
 errorCoroutine = StartCoroutine(\_ShowMessage(message));  
 }  
  
 private IEnumerator \_ShowMessage(string message) {  
 error.SetActive(true);  
 errorText.text = message;  
 yield return new WaitForSeconds(2);  
 error.SetActive(false);  
 errorCoroutine = null;  
 }  
  
 private void **Awake**() {  
 instance = this;  
 }  
  
 private void **Start**() {  
 SetDefault();  
 errorText = error.GetComponent<Text>();  
 }  
  
 private void Update() { }  
  
 public void **SetDefault**() {  
 info.SetActive(true);  
 linear.SetActive(false);  
 ramified.SetActive(false);  
 cyclic.SetActive(false);  
 }  
  
 private void OnClick(GameObject other) {  
 info.SetActive(false);  
 other.SetActive(true);  
 }  
  
 public void **OnLinear**() {  
 OnClick(linear);  
 }  
  
 public void **OnRamified**() {  
 OnClick(ramified);  
 }  
  
 public void **OnCyclic**() {  
 OnClick(cyclic);  
 }  
  
 public void **OnCalculateLinear**() {  
 if (!float.TryParse(a.text, NumberStyles.Float, CultureInfo.InvariantCulture, out var \_a) ||  
 !float.TryParse(b.text, NumberStyles.Float, CultureInfo.InvariantCulture, out var \_b) ||  
 !float.TryParse(c.text, NumberStyles.Float, CultureInfo.InvariantCulture, out var \_c)) {  
 ShowMessage(inputErrorMessage);  
 return;  
 }  
  
 var result = new LinearAlgorithm(\_a, \_b, \_c).FindResult();  
 if (float.IsNaN(result)) {  
 ShowMessage(resultErrorMessage);  
 resultLinear.text = "=";  
 return;  
 }  
  
 resultLinear.text =  
 $"= {result.ToString("G3", CultureInfo.InvariantCulture)}";  
 }  
  
 public void **OnCalculateRamified**() {  
 if (!float.TryParse(k.text, NumberStyles.Float, CultureInfo.InvariantCulture, out var \_k) ||  
 !float.TryParse(d.text, NumberStyles.Float, CultureInfo.InvariantCulture, out var \_d)) {  
 ShowMessage(inputErrorMessage);  
 return;  
 }  
  
 var result = new RamifiedAlgorithm(\_k, \_d).FindResult();  
 if (float.IsNaN(result)) {  
 ShowMessage(resultErrorMessage);  
 resultLinear.text = "y =";  
 return;  
 }  
  
 resultRamified.text =  
 $"y = {result.ToString("G3", CultureInfo.InvariantCulture)}";  
 }  
  
 public void **OnCalculateCyclic**() {  
 if (!int.TryParse(n.text, NumberStyles.Integer, CultureInfo.InvariantCulture, out var \_n) ||  
 !float.TryParse(min.text, NumberStyles.Float, CultureInfo.InvariantCulture, out var \_min) ||  
 !float.TryParse(max.text, NumberStyles.Float, CultureInfo.InvariantCulture, out var \_max) ||  
 !CyclicAlgorithm.CheckVars(\_n, \_min, \_max)) {  
 ShowMessage(inputErrorMessage);  
 return;  
 }  
  
 var result = new CyclicAlgorithm(\_n, \_min, \_max).FindResult();  
 if (float.IsNaN(result)) {  
 ShowMessage(resultErrorMessage);  
 resultLinear.text = "f =";  
 return;  
 }  
  
 resultCyclic.text =  
 $"f = {result.ToString("G3", CultureInfo.InvariantCulture)}";  
 }  
}

**Висновок:** У ході виконання лабораторної роботи я закріпив знання з базових понять алгоритмів, вивчив основні правила складання блок-схем алгоритмів і покращив навички роботи з draw.io. Отримані результати виконання програми є вірними.