**Міністерство освіти і науки України  
Національний технічний університет України  
«Київський політехнічний інститут імені Ігоря Сікорського»  
Факультет інформатики та обчислювальної техніки  
Кафедра обчислювальної техніки**

**Лабораторна робота №7**

з дисципліни  
«Інтелектуальні вбудовані системи»

на тему  
«Дослідження генетичного алгоритму»

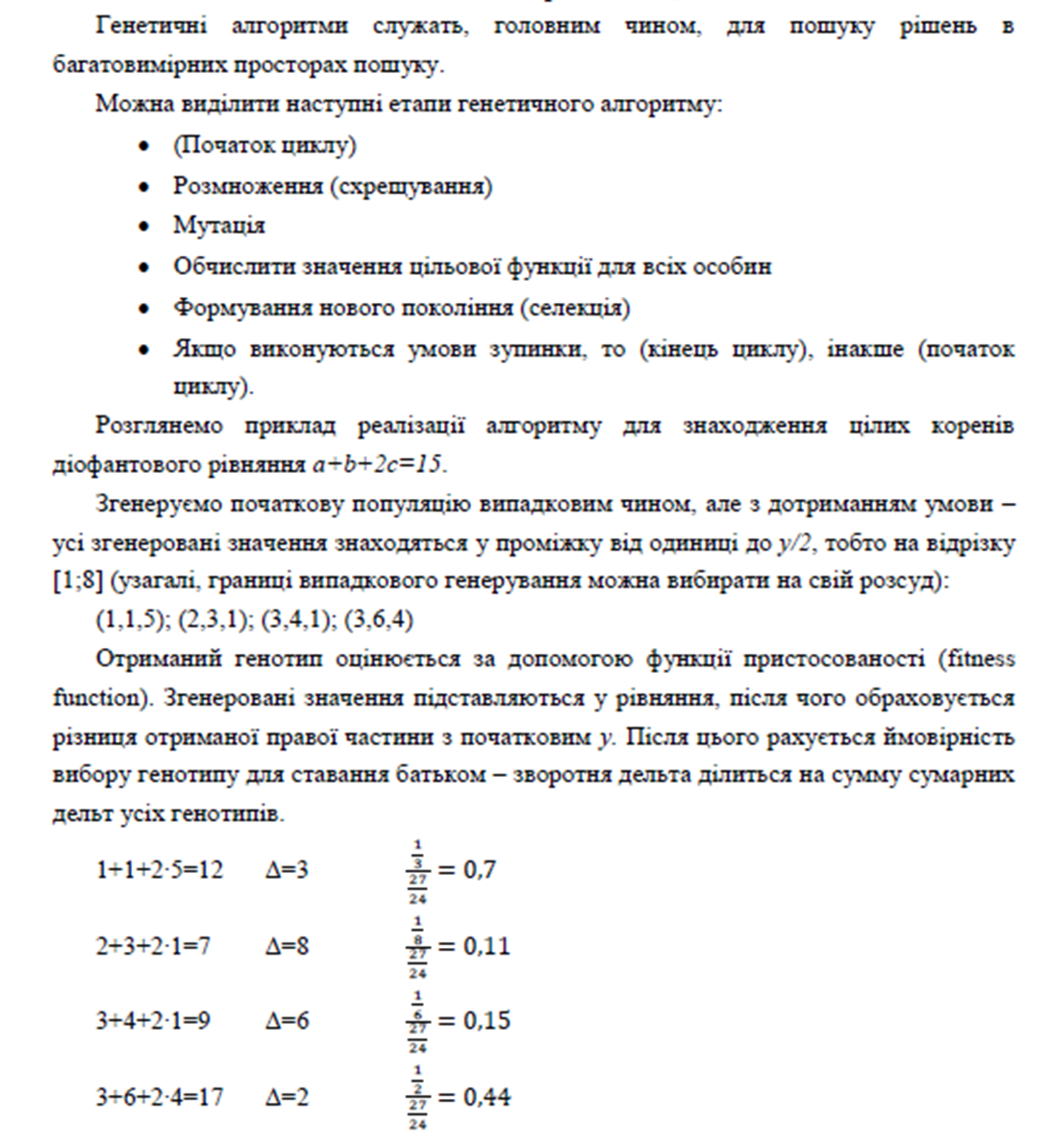
Виконала: Перевірив:

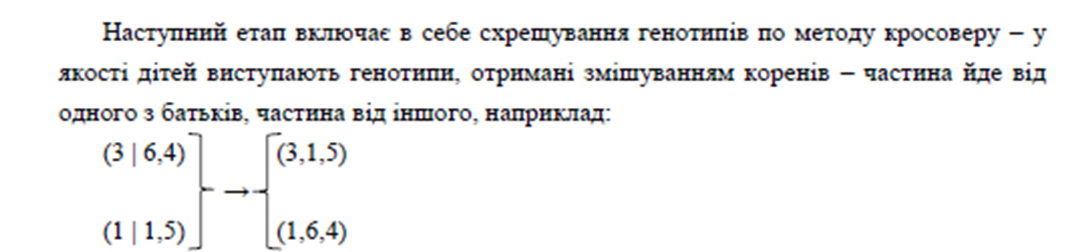
студентка групи ІП-84 ас. Регіда П. Г.

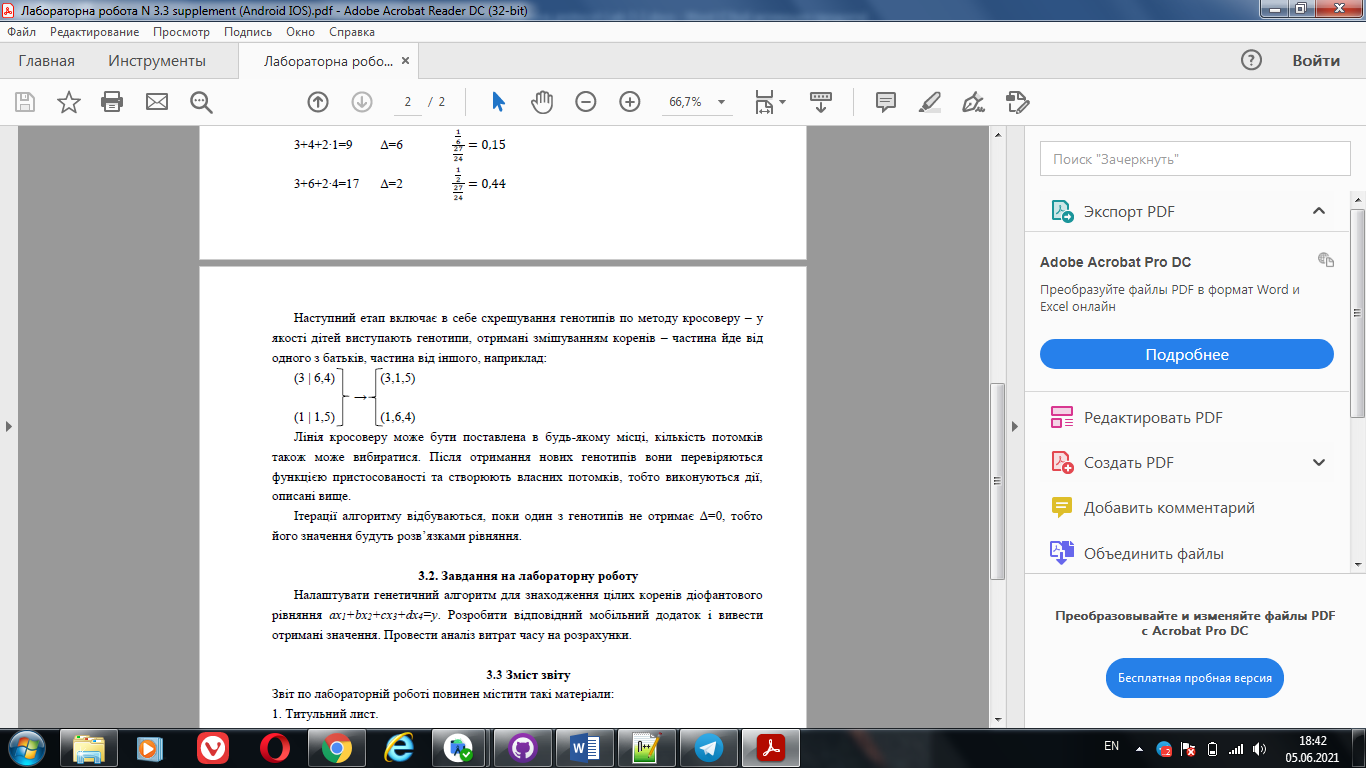
Коломієць Євгенія Валеріївна   
номер залікової книжки: 8412

Київ 2020

**Основні теоретичні відомості:**

****

****



**Завдання за варіантом:**

Варіант 12

**Лістинг програми**:

package com**.**example**.**lab\_33

**import** androidx**.**appcompat**.**app**.**AppCompatActivity

**import** android**.**os**.**Bundle

**import** android**.**widget**.**Button

**import** android**.**widget**.**EditText

**import** android**.**widget**.**TextView

**import** java**.**lang**.**Math**.**abs

**import** java**.**util**.**Random

**import** kotlin**.**math**.**absoluteValue

class MainActivity **:** AppCompatActivity**()** **{**

private fun gen\_alg**(**x1\_base**:** Double**,**

x2\_base**:** Double**,**

x3\_base**:** Double**,**

x4\_base**:** Double**,**

y\_base**:** Double**)** **:** MutableList**<**Double**>{**

var counter **=** 0

val populationZero**:** MutableList**<**MutableList**<**Double**>>** **=** mutableListOf**()**

val population**:** MutableList**<**MutableList**<**Double**>>** **=** mutableListOf**()**

val listOfFitnesses**:** MutableList**<**Double**>** **=** mutableListOf**()**

val populationOfChild**:** MutableList**<**MutableList**<**Double**>>** **=** mutableListOf**()**

var bestPopulation**:** MutableList**<**Double**>** **=** mutableListOf**()**

fun fitness**(**population**:** MutableList**<**Double**>):** Double **{**

val fitness**:** Double **=** y\_base **-**

population**[**0**]** **\*** x1\_base **-**

population**[**1**]** **\*** x2\_base **-**

population**[**2**]** **\*** x3\_base **-**

population**[**3**]** **\*** x4\_base

**return** fitness**.**absoluteValue

**}**

fun populationZeroFind**()** **{**

**for** **(**i in 0..3**)** **{**

populationZero**.**add**(**mutableListOf**())**

**for** **(**j in 0..3**)** **{**

populationZero**[**i**].**add**((**1..8**).**random**().**toDouble**())**

**}**

**}**

**}**

fun findFitnessOfPopulation**()** **{**

listOfFitnesses**.**clear**()**

**if** **(**population**.**isEmpty**())** **{**

populationZero**.**mapTo**(**population**)** **{** it **}**

**}**

**for** **(**i in 0..3**)** **{**

listOfFitnesses**.**add**(**fitness**(**population**[**i**]))**

**}**

**}**

fun findRoulette**()** **{**

populationOfChild**.**clear**()**

var roulette **=** 0.00

val roulettePercent**:** MutableList**<**Double**>** **=** mutableListOf**()**

val circleRoulette**:** MutableList**<**Double**>** **=** mutableListOf**()**

listOfFitnesses**.**forEach **{** roulette **+=** 1 **/** it **}**

**for** **(**i in 0..3**)** **{**

roulettePercent**.**add**(**1 **/** listOfFitnesses**[**i**]** **/** roulette**)**

**}**

**for** **(**i in 0..3**)** **{**

**if** **(**i **==** 0**)** **{**

circleRoulette**.**add**(**roulettePercent**[**i**])**

**}** **else** **{**

circleRoulette**.**add**(**circleRoulette**[**i **-** 1**]** **+** roulettePercent**[**i**])**

**}**

**}**

var i **=** 0

populationOfChild**.**clear**()**

**while** **(**i **<** 4**)** **{**

val piu**:** Double **=** **(**1..100**).**random**().**toDouble**()** **/** 100

var thisChild **=** 0

**for** **(**k in 0..3**)** **{**

**if** **(**piu **>=** circleRoulette**[**k**])** **{**

thisChild **=** k

**}**

**}**

populationOfChild**.**add**(**population**[**thisChild**])**

i**++**

**}**

**}**

fun crossingOver**()** **{**

counter**++**

population**.**clear**()**

**for** **(**p in 0..3**)** **{**

val c**:** MutableList**<**Double**>** **=** mutableListOf**()**

c**.**clear**()**

**for** **(**j in 0..3**)** **{**

**if** **(**p **%** 2 **==** 0**)** **{**

**if** **(**j **<** 2**)** **{**

c**.**add**(**populationOfChild**[**p**][**j**])**

**}** **else** c**.**add**(**populationOfChild**[**p **+** 1**][**j**])**

**}** **else**

**if** **(**j **<** 2**)** **{**

c**.**add**(**populationOfChild**[**p**][**j**])**

**}** **else** c**.**add**(**populationOfChild**[**p **-** 1**][**j**])**

**}**

population**.**add**(**c**)**

**}**

**}**

fun bestFitnessFind**():** Boolean **{**

findFitnessOfPopulation**()**

listOfFitnesses**.**forEach **{** **if** **(**it **==** 0.0**)** **return** **true** **}**

**return** **false**

**}**

fun life**()** **{**

var q **=** 0

**while** **(!**bestFitnessFind**()** **&&** q **<** 10**)** **{**

findFitnessOfPopulation**()**

findRoulette**()**

crossingOver**()**

q**++**

**}**

**}**

fun result**():** MutableList**<**Double**>** **{**

populationZeroFind**()**

life**()**

**while** **((!**listOfFitnesses**.**contains**(**0.0**))** **&&**

population**[**0**]** **==** populationOfChild**[**0**]** **&&**

population**[**1**]** **==** populationOfChild**[**1**]** **&&**

population**[**2**]** **==** populationOfChild**[**2**]** **&&**

population**[**3**]** **==** populationOfChild**[**3**]**

**)** **{**

populationZero**.**clear**()**

population**.**clear**()**

listOfFitnesses**.**clear**()**

populationOfChild**.**clear**()**

populationZeroFind**()**

life**()**

**}**

**for** **(**i in 0..3**)** **{**

**if** **(**listOfFitnesses**[**i**]** **==** 0.0**)** **{**

bestPopulation **=** population**[**i**]**

**}**

**}**

**return** bestPopulation

**}**

val answer **:** MutableList**<**Double**>** **=** result**();**

**if** **(**answer**.**isEmpty**())** **{**

**return** answer**;**

**}**

answer**.**add**(**counter**.**toDouble**())**

**return** answer

**}**

override fun onCreate**(**savedInstanceState**:** Bundle**?)** **{**

**super.**onCreate**(**savedInstanceState**)**

setContentView**(**R**.**layout**.**activity\_main**)**

val result**:** TextView **=** findViewById**(**R**.**id**.**show\_result**)**

val input\_a**:** EditText **=** findViewById**(**R**.**id**.**input\_data\_a**)**

val input\_b**:** EditText **=** findViewById**(**R**.**id**.**input\_data\_b**)**

val input\_c**:** EditText **=** findViewById**(**R**.**id**.**input\_data\_c**)**

val input\_d**:** EditText **=** findViewById**(**R**.**id**.**input\_data\_d**)**

val input\_y**:** EditText **=** findViewById**(**R**.**id**.**input\_data\_y**)**

val buttonCalculate**:** Button **=** findViewById**(**R**.**id**.**button\_calculate**)**

buttonCalculate**.**setOnClickListener **{**

**if** **(**input\_a**.**text**.**isEmpty**()){**

result**.**text **=** "There was no number entered!"

**}** **else** **{**

val a **=** input\_a**.**text**.**toString**().**toDouble**()**

val b **=** input\_b**.**text**.**toString**().**toDouble**()**

val c **=** input\_c**.**text**.**toString**().**toDouble**()**

val d **=** input\_d**.**text**.**toString**().**toDouble**()**

val y **=** input\_y**.**text**.**toString**().**toDouble**()**

val ans **=** gen\_alg**(**a**,** b**,** c**,** d**,** y**)**

**if** **(!**ans**.**isEmpty**())** **{**

ans**.**removeAt**(**4**)**

result**.**text **=** "X1 = ${ans[0].toInt()}, X2 = ${ans[1].toInt()}, X3 = ${ans[2].toInt()}, X4 = ${ans[3].toInt()}"

**}** **else** **{**

result**.**text **=** "Couldn't find in range y/2. Try again"

**}**

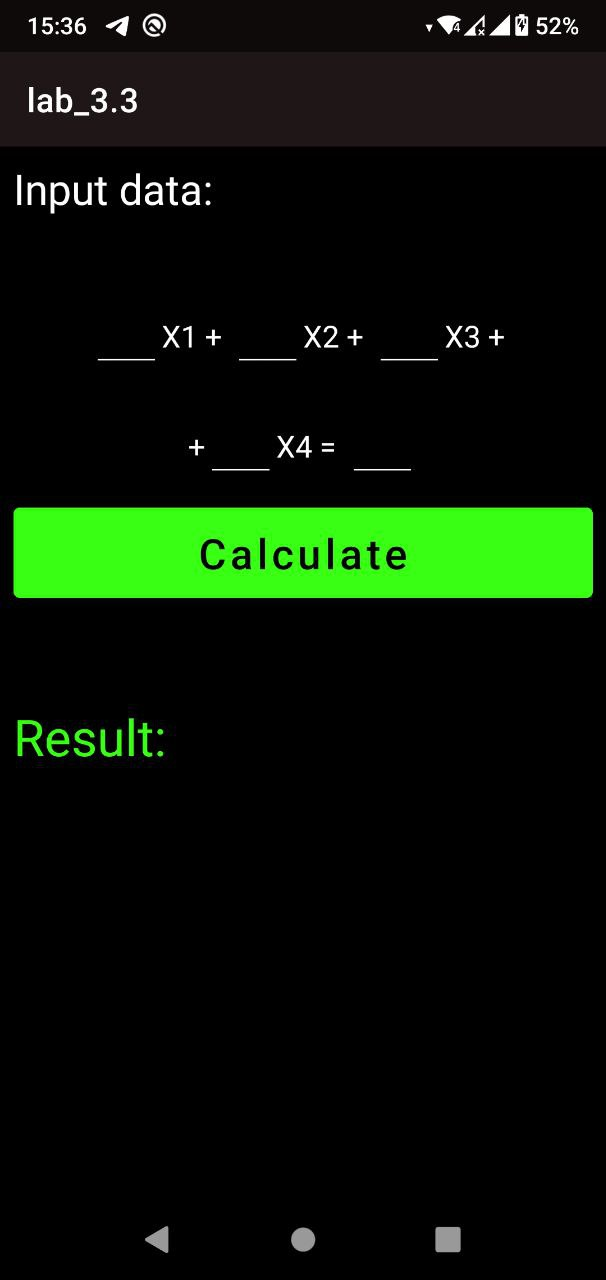
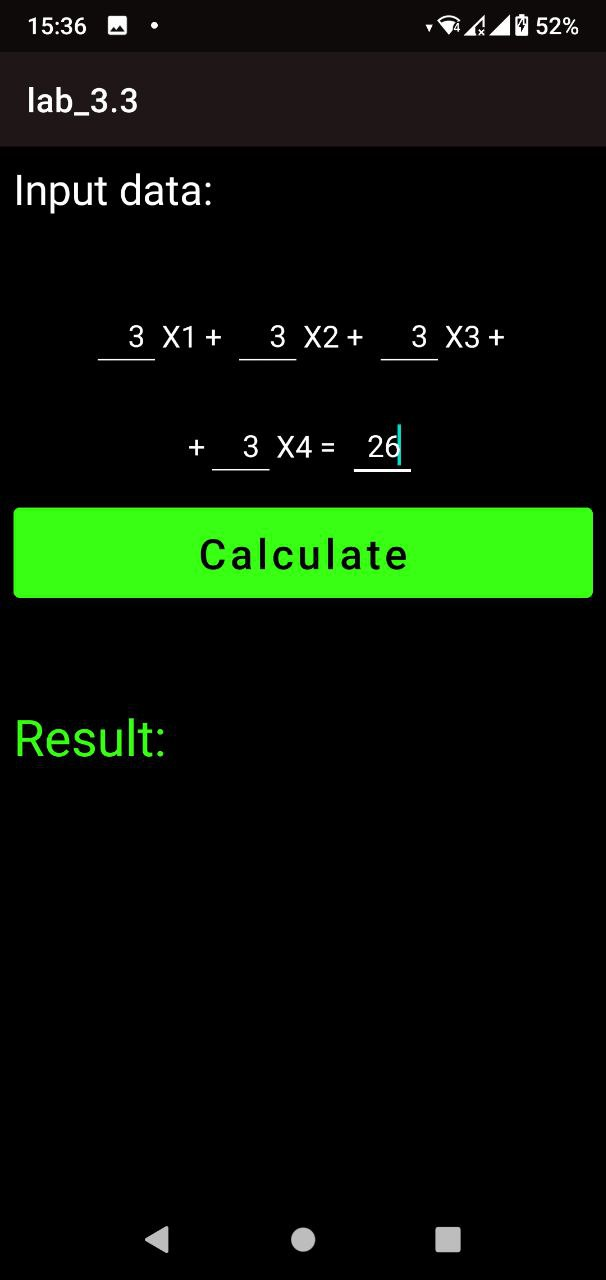
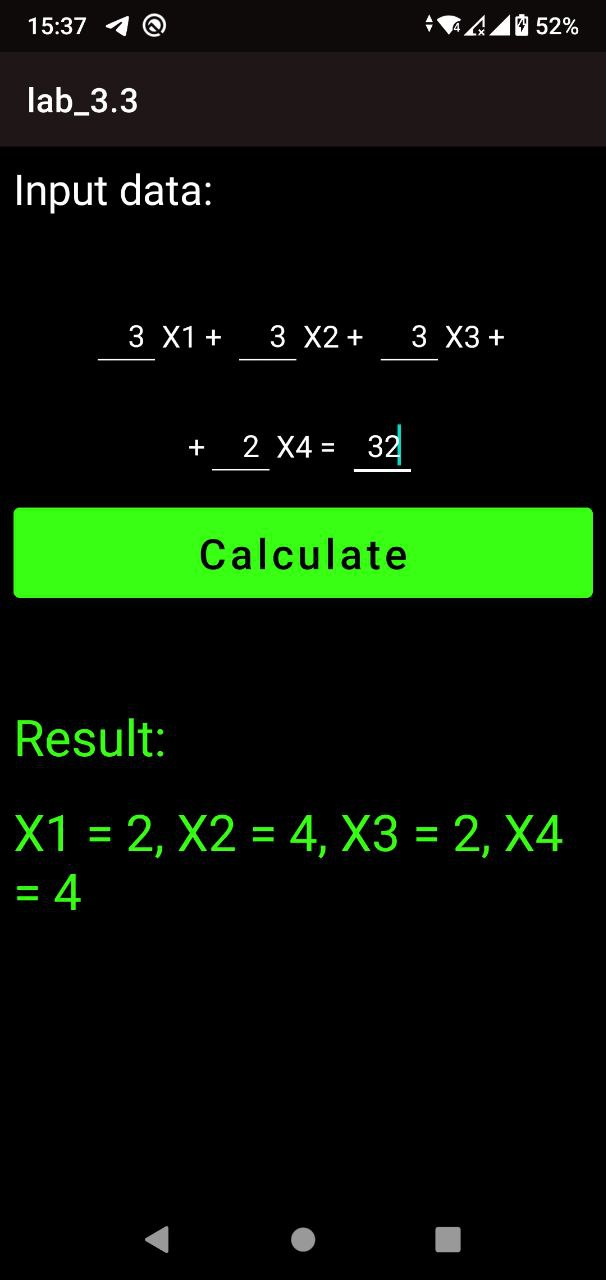
**}**

**}**

**}**

**}**

**Приклад роботи програми**:

**Висновки**:

Під час виконання лабораторної роботи я ознайомилась з принципами реалізації генетичного алгоритму, вивчення та дослідження особливостей даного алгоритму з використанням засобів моделювання і сучасних програмних оболонок.

Було розроблено реалізацію генетичноо алгоритму для знаходження цілих коренів діофантового рівняння. Було реалізовано користувацький інтерфейс з можливістю вводу даних за допомогою Android.