# Міністерство освіти і науки України

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"

# Факультет інформатики та обчислювальної техніки Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 3 з дисципліни

«Основи програмування»

Варіант 13

Виконав студент Калашніков Андрій Євгенович

(шифр, прізвище, ім'я, по батькові)

Перевірив

( прізвище, ім'я, по батькові)

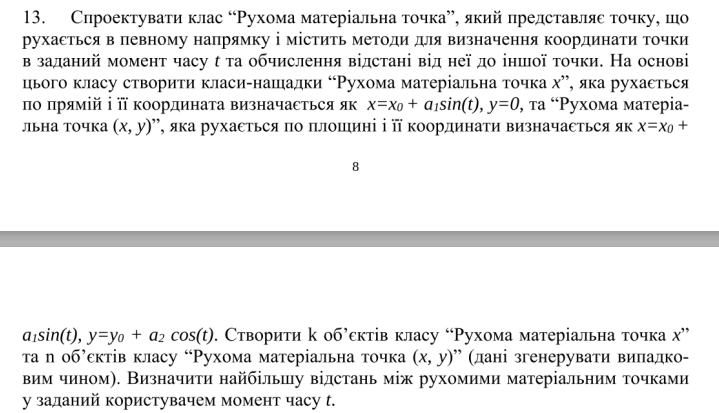
Київ 2021

**Лабораторна робота №4**

# **УСПАДКУВАННЯ ТА ПОЛІМОРФІЗМ**

**Мета –** вивчити механізми створення і використання класів та об’єктів.

**Індивідуальне завдання:**



**Постановка задачі**

Розробимо клас рухома матеріальна точка. На його основі створимо класи-нащадки «Рухома точка Х» і »Рухома точка ХУ». Створимо k,n об’єктів цих класів і знайдемо максимально відстань між двома матерільними рухомими точками.

**Код на С++ :**

**Header.h :**

#pragma once

#include <iostream>

#include <cmath>

#include<ctime>

#include <vector>

using namespace std;

class point {

protected:

double x = 0, y = 0;

int a1 = rand() % 5 + 1, a2 = rand() % 5 + 1, x0 = rand() % 5 + 1, y0 = rand() % 5 + 1;

public:

double getx() { return x; }

double gety() { return y; }

virtual void setCoordsT(int t) = 0;

double distance(point& obj);

void print();

};

class pointX :public point {

public:

void setCoordsT(int t);

};

class pointXY :public point {

public:

void setCoordsT(int t);

};

void max\_distance();

**Header.cpp**

#include "Header.h"

double point::distance(point& obj) {

double distance;

distance = sqrt(pow((obj.x - x), 2) + pow((obj.y - y), 2));

return distance;

}

void point::print() {

cout << endl << "Coordinates of the point are (" << x << ";" << y << ")";

}

void pointX::setCoordsT(int t) {

x = x0 + a1 \* sin(t);

y = 0;

}

void pointXY::setCoordsT(int t) {

x = x0 + a1 \* sin(t);

y = y0 + a2 \* cos(t);

}

void max\_distance() {

int k, n;

double x1, y1, x2, y2;

cout << "Input k: "; cin >> k;

cout << "Input n: "; cin >> n;

vector<pointX>v1(k);

vector<pointXY>v2(n);

int t;

double max\_dist = 0;

double dist = 0;

cout << "Enter t="; cin >> t; cout << endl;

point\* p;

for (int i = 0; i < k; i++) {

p = &v1[i];

p->setCoordsT(t); p->print();

}

for (int i = 0; i < n; i++) {

p = &v2[i];

p->setCoordsT(t); p->print();

}

for (int i = 0; i < k; i++) {

p = &v1[i];

for (int j = 0; j < k - 1; j++) {

dist = p->distance(v1[j + 1]);

if (dist > max\_dist) {

max\_dist = dist;

x1 = p->getx();

y1 = p->gety();

x2 = v1[j + 1].getx();

y2 = v1[j + 1].gety();

}

}

for (int j = 0; j < n; j++) {

dist = p->distance(v2[j]);

if (dist > max\_dist) {

max\_dist = dist;

x1 = p->getx();

y1 = p->gety();

x2 = v2[j].getx();

y2 = v2[j].gety();

}

}

}

for (int i = 0; i < n; i++) {

p = &v2[i];

for (int j = 0; j < n - 1; j++) {

dist = p->distance(v2[j + 1]);

if (dist > max\_dist) {

max\_dist = dist;

x1 = p->getx();

y1 = p->gety();

x2 = v2[j + 1].getx();

y2 = v2[j + 1].gety();

}

}

}

cout << endl << "Max distance is=" << max\_dist << " Between points with coords (" << x1 << ";" << y1 << ") and (" << x2 << ";" << y2 << ")";

}

**Source.cpp:**

#include "Header.h"

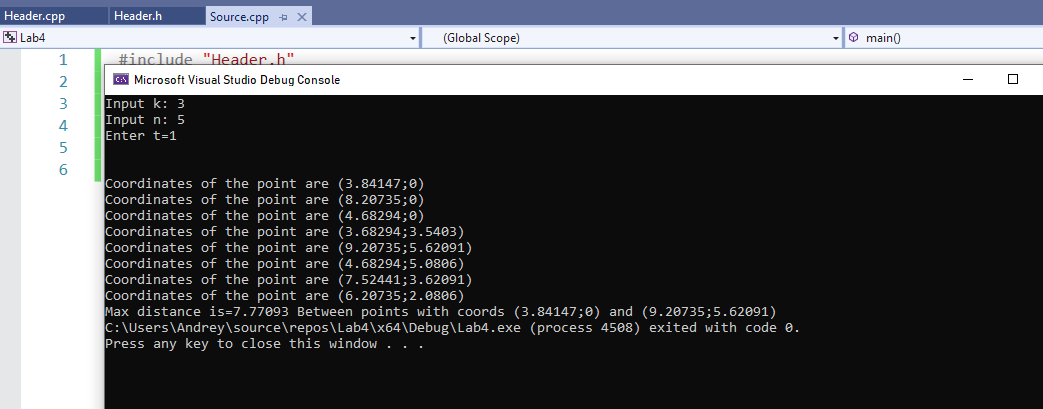
int main() {

srand(time(NULL));

max\_distance();

}

**Скріншот результатів програми на C++ :**

****

**Код на Python :**

**point.py:**

from abc import ABCMeta, abstractmethod, ABC

import random

import math

class point(metaclass=ABCMeta):

    def \_\_init\_\_(self, t):

        self.\_a1, self.\_a2, self.\_x0, self.\_y0 = random.randint(0, 5), random.randint(0, 5), \

                                                 random.randint(0, 5), random.randint(0, 5)

        self.\_t = t

        self.\_x, self.\_y = self.\_find\_x(), self.\_find\_y()

    def \_\_str\_\_(self):

        return f'(x;y) = ({self.\_x}; {self.\_y})'

    @abstractmethod

    def \_find\_x(self): pass

    @abstractmethod

    def \_find\_y(self): pass

    @abstractmethod

    def get\_x(self): pass

    @abstractmethod

    def get\_y(self): pass

    @abstractmethod

    def get\_distance(self, ox, oy): pass

class pointX(point, ABC):

    def \_\_init\_\_(self, t):

        super().\_\_init\_\_(t)

    def \_find\_x(self):

        return round(self.\_x0 + self.\_a1 \* math.sin(self.\_t), 4)

    def get\_x(self):

        return self.\_x

    def \_find\_y(self):

        return 0

    def get\_y(self):

        return self.\_y

    def get\_distance(self, ox, oy):

        return round(math.sqrt((self.\_x - ox) \*\* 2 + (self.\_y - oy) \*\* 2), 4)

class pointXY(point, ABC):

    def \_\_init\_\_(self, t):

        super().\_\_init\_\_(t)

    def \_find\_x(self):

        return round(self.\_x0 + self.\_a1 \* math.sin(self.\_t), 4)

    def get\_x(self):

        return self.\_x

    def \_find\_y(self):

        return round(self.\_y0 + self.\_a2 \* math.cos(self.\_t), 4)

    def get\_y(self):

        return self.\_y

    def get\_distance(self, ox, oy):

        return round(math.sqrt((self.\_x - ox) \*\* 2 + (self.\_y - oy) \*\* 2), 4)

**lab4.py:**

from point import pointX, pointXY

k = int(input("Input k: "))

n = int(input("Input n: "))

t = float(input("Input t: "))

coords = []

distances = []

xpo = []

xypo = []

for i in range(k):

    obj = pointX(t)

    xpo.append(obj)

    print(obj)

    for j in range(i):

        dist = obj.get\_distance(xpo[j].get\_x(), xpo[j].get\_y())

        coordinates = [obj.get\_x(), obj.get\_y(), xpo[j].get\_x(), xpo[j].get\_y()]

        coords.append(coordinates)

        distances.append(dist)

for i in range(n):

    obj = pointXY(t)

    xypo.append(obj)

    print(obj)

    for j in range(i):

        dist = obj.get\_distance(xypo[j].get\_x(), xypo[j].get\_y())

        coordinates = [obj.get\_x(), obj.get\_y(), xypo[j].get\_x(), xypo[j].get\_y()]

        coords.append(coordinates)

        distances.append(dist)

for i in range(k):

    for j in range(n):

        dist = xpo[i].get\_distance(xypo[j].get\_x(), xypo[j].get\_y())

        coordinates = [xpo[i].get\_x(), xpo[i].get\_y(), xypo[j].get\_x(), xypo[j].get\_y()]

        coords.append(coordinates)

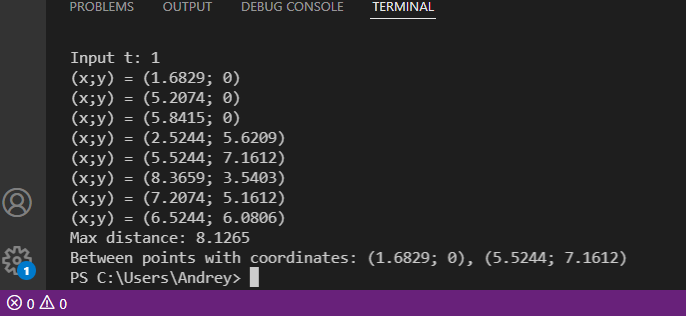
        distances.append(dist)

count = distances.index(max(distances))

print(f'Max distance: {distances[count]}')

print(f'Between points with coordinates: ({coords[count][0]}; {coords[count][1]}), ({coords[count][2]}; {coords[count][3]})')

**Скріншот результатів програми на Python :**

****

**Посилання на GitHub:**

**https://github.com/Andrey7904/KPI/tree/main/OP-2/Lab4**

**Висновок:**

Під час лабораторної роботи я вивчив механізми створення і використання класів та об’єктів. Оскільки програма правильно створює базовий клас та два класи нащадки, правильно знаходить найбільшу відстань між двома матеріальними рухомими точками, то вона працює правильно.