**Билет 21 по лабораторной работе по теме 2**

Задание 1 (Python)

import math  
  
def vector\_length(vector):  
 return math.sqrt(vector[0]\*\*2 + vector[1]\*\*2)  
  
def angle\_between\_vectors(vector1, vector2):  
 dot\_product = vector1[0]\*vector2[0] + vector1[1]\*vector2[1]  
 magnitude1 = vector\_length(vector1)  
 magnitude2 = vector\_length(vector2)  
 return math.degrees(math.acos(dot\_product / (magnitude1 \* magnitude2)))  
  
def vector\_sum(vector1, vector2):  
 return [vector1[0] + vector2[0], vector1[1] + vector2[1]]  
  
def vector\_difference(vector1, vector2):  
 return [vector1[0] - vector2[0], vector1[1] - vector2[1]]  
  
vector1 = [3, 4]  
vector2 = [5, 2]  
vector3 = [1, 6]  
  
lengths = [vector\_length(vector1), vector\_length(vector2), vector\_length(vector3)]  
print("Длины векторов:", lengths)  
  
angles = [angle\_between\_vectors(vector1, vector2), angle\_between\_vectors(vector2, vector3), angle\_between\_vectors(vector3, vector1)]  
print("Углы между векторами (в градусах):", angles)  
  
sum\_12 = vector\_sum(vector1, vector2)  
sum\_23 = vector\_sum(vector2, vector3)  
sum\_31 = vector\_sum(vector3, vector1)  
diff\_12 = vector\_difference(vector1, vector2)  
diff\_23 = vector\_difference(vector2, vector3)  
diff\_31 = vector\_difference(vector3, vector1)  
print("Суммы векторов:", sum\_12, sum\_23, sum\_31)  
print("Разности векторов:", diff\_12, diff\_23, diff\_31)  
  
def are\_parallel(v1, v2):  
 return v1[0]\*v2[1] == v1[1]\*v2[0]  
  
parallel\_12 = are\_parallel(vector1, vector2)  
parallel\_23 = are\_parallel(vector2, vector3)  
parallel\_31 = are\_parallel(vector3, vector1)  
print("Параллельны ли вектора 1 и 2?", parallel\_12)  
print("Параллельны ли вектора 2 и 3?", parallel\_23)  
print("Параллельны ли вектора 3 и 1?", parallel\_31)

Задание 2 (Python)

import numpy as np  
  
A = np.random.randint(-100, 100)  
B = np.random.randint(-100, 100)  
C = np.random.randint(-100, 100)  
D = np.random.randint(-100, 100)  
  
A\_plane = np.random.randint(-100, 100)  
B\_plane = np.random.randint(-100, 100)  
C\_plane = np.random.randint(-100, 100)  
D\_plane = np.random.randint(-100, 100)  
  
angle = np.arccos(np.abs(A \* A\_plane + B \* B\_plane + C \* C\_plane) / (  
 np.sqrt(A \*\* 2 + B \*\* 2 + C \*\* 2) \* np.sqrt(A\_plane \*\* 2 + B\_plane \*\* 2 + C\_plane \*\* 2)))  
  
if np.abs(A \* A\_plane + B \* B\_plane + C \* C\_plane) == 0:  
 print("Прямая параллельна плоскости")  
  
 distance = np.abs(D\_plane) / np.sqrt(A \*\* 2 + B \*\* 2 + C \*\* 2)  
 print("Расстояние между прямой и плоскостью:", distance)  
else:  
 print("Прямая не параллельна плоскости")  
  
 x = (B\_plane \* D - B \* D\_plane) / (A \* B\_plane - A\_plane \* B)  
 y = (D\_plane \* A - D \* A\_plane) / (A \* B\_plane - A\_plane \* B)  
 z = 0  
 print("Координаты точки пересечения прямой и плоскости: ({}, {}, {})".format(x, y, z))  
  
print("Угол между прямой и плоскостью:", np.degrees(angle))

Задание 3 (C++)

#include <iostream>

#include <cmath>

#include <random>

struct Point {

double x;

double y;

};

struct Ellipse {

double a;

double b;

Point center;

double e;

Point focus1;

Point focus2;

};

double distance(Point p1, Point p2) {

return std::sqrt(std::pow(p2.x - p1.x, 2) + std::pow(p2.y - p1.y, 2));

}

bool isPointInsideEllipse(Point point, Ellipse ellipse) {

return (std::pow(point.x - ellipse.center.x, 2) / std::pow(ellipse.a, 2) +

std::pow(point.y - ellipse.center.y, 2) / std::pow(ellipse.b, 2)) <= 1;

}

int main() {

std::random\_device rd;

std::mt19937 gen(rd());

std::uniform\_real\_distribution<double> dis(-10.0, 10.0);

Ellipse ellipse;

ellipse.a = std::abs(dis(gen));

ellipse.b = std::abs(dis(gen));

ellipse.center.x = dis(gen);

ellipse.center.y = dis(gen);

Point testPoint;

testPoint.x = dis(gen);

testPoint.y = dis(gen);

if (isPointInsideEllipse(testPoint, ellipse)) {

std::cout << "Точка (" << testPoint.x << ", " << testPoint.y << ") принадлежит эллипсу." << std::endl;

}

else {

std::cout << "Точка (" << testPoint.x << ", " << testPoint.y << ") не принадлежит эллипсу." << std::endl;

}

double c = std::sqrt(std::pow(ellipse.a, 2) - std::pow(ellipse.b, 2));

ellipse.focus1.x = ellipse.center.x + c;

ellipse.focus1.y = ellipse.center.y;

ellipse.focus2.x = ellipse.center.x - c;

ellipse.focus2.y = ellipse.center.y;

std::cout << "Координаты фокусов: (" << ellipse.focus1.x << ", " << ellipse.focus1.y << ") и (" << ellipse.focus2.x << ", " << ellipse.focus2.y << ")" << std::endl;

ellipse.e = c / ellipse.a;

std::cout << "Эксцентриситет: " << ellipse.e << std::endl;

double distanceToDirectrix = ellipse.a / ellipse.e;

std::cout << "Расстояние от директрис до центра: " << distanceToDirectrix << std::endl;

return 0;

}

Задание 4 (C++)

#include <iostream>

#include <cmath>

#include <random>

using namespace std;

struct Point {

double x;

double y;

double z;

};

struct Plane {

double a;

double b;

double c;

double d;

};

struct Line {

Point p;

Point v;

};

struct ParabolicCylinder {

double a;

double b;

double c;

double d;

};

double generateRandomNumber(double min, double max) {

random\_device rd;

mt19937 generator(rd());

uniform\_real\_distribution<double> distribution(min, max);

return distribution(generator);

}

bool intersectsPlaneCylinder(const Plane& plane, const ParabolicCylinder& cylinder) {

if (abs(plane.c) < 1e-6) {

return false;

}

double x = (plane.d - plane.b \* cylinder.b - plane.c \* cylinder.c) / plane.a;

double y = (cylinder.d - cylinder.a \* x \* x - cylinder.c \* plane.c) / cylinder.b;

return abs(plane.a \* x + plane.b \* y + plane.c \* plane.c - plane.d) < 1e-6;

}

bool intersectsLineCylinder(const Line& line, const ParabolicCylinder& cylinder) {

// Параметрическое уравнение прямой:

// x = line.p.x + t \* line.v.x

// y = line.p.y + t \* line.v.y

// z = line.p.z + t \* line.v.z

// Подставляем уравнение прямой в уравнение параболического цилиндра:

// a \* (line.p.x + t \* line.v.x)^2 + b \* (line.p.y + t \* line.v.y) + c \* (line.p.z + t \* line.v.z) = d

// Решаем квадратное уравнение относительно t

double a = cylinder.a \* line.v.x \* line.v.x;

double b = 2 \* cylinder.a \* line.v.x \* (line.p.x - line.v.x) + cylinder.b \* line.v.y;

double c = cylinder.a \* (line.p.x - line.v.x) \* (line.p.x - line.v.x) + cylinder.b \* line.p.y + cylinder.c \* line.p.z - cylinder.d;

double discriminant = b \* b - 4 \* a \* c;

if (discriminant < 0) {

return false;

}

if (discriminant == 0) {

return true;

}

return true;

}

bool pointBelongsToCylinder(const Point& point, const ParabolicCylinder& cylinder) {

return abs(cylinder.a \* point.x \* point.x + cylinder.b \* point.y + cylinder.c \* point.z - cylinder.d) < 1e-6;

}

int main() {

ParabolicCylinder cylinder;

cylinder.a = generateRandomNumber(-10, 10);

cylinder.b = generateRandomNumber(-10, 10);

cylinder.c = generateRandomNumber(-10, 10);

cylinder.d = generateRandomNumber(-10, 10);

Plane plane;

plane.a = generateRandomNumber(-10, 10);

plane.b = generateRandomNumber(-10, 10);

plane.c = generateRandomNumber(-10, 10);

plane.d = generateRandomNumber(-10, 10);

Line line;

line.p.x = generateRandomNumber(-10, 10);

line.p.y = generateRandomNumber(-10, 10);

line.p.z = generateRandomNumber(-10, 10);

line.v.x = generateRandomNumber(-10, 10);

line.v.y = generateRandomNumber(-10, 10);

line.v.z = generateRandomNumber(-10, 10);

Point point;

point.x = generateRandomNumber(-10, 10);

point.y = generateRandomNumber(-10, 10);

point.z = generateRandomNumber(-10, 10);

cout << "Уравнение параболического цилиндра: " << cylinder.a << "x^2 + " << cylinder.b << "y + " << cylinder.c << "z = " << cylinder.d << endl;

cout << "Уравнение плоскости: " << plane.a << "x + " << plane.b << "y + " << plane.c << "z = " << plane.d << endl;

cout << "Уравнение прямой: (x, y, z) = (" << line.p.x << ", " << line.p.y << ", " << line.p.z << ") + t \* (" << line.v.x << ", " << line.v.y << ", " << line.v.z << ")" << endl;

cout << "Точка: (" << point.x << ", " << point.y << ", " << point.z << ")" << endl;

cout << "Пересекает ли плоскость параболический цилиндр: " << (intersectsPlaneCylinder(plane, cylinder) ? "Да" : "Нет") << endl;

cout << "Пересекает ли прямая параболический цилиндр: " << (intersectsLineCylinder(line, cylinder) ? "Да" : "Нет") << endl;

cout << "Принадлежит ли точка параболическому цилиндру: " << (pointBelongsToCylinder(point, cylinder) ? "Да" : "Нет") << endl;

return 0;

}