КПІ ім. Ігоря Сікорського

Кафедра ІПІ

ЗВІТ

про виконання комп’ютерного практикуму № 5

з кредитного модуля

«Основи програмування-2. Методології програмування»

Тема: Успадкування та поліморфізм

Варіант №3

Виконала:

студентка 1-го курсу

гр. ІП-з21 ФІОТ

Гавриленко Даяна Юріївна

Київ 2023

1. Умова завдання:

Изображение выглядит как текст, Шрифт, снимок экрана, документ

Автоматически созданное описание

1. Текст програми на мові C#:

Program.cs

namespace Lab5  
{  
 public class Program  
 {  
 public static void Main(string[] args) {  
 PointOnPlane point = new PointOnPlane(new Random().Next(-9, 9), new Random().Next(-9, 9));  
 LineService lineService = new LineService();  
   
 TLine[] lineOnPlanes = lineService.GenerateRandom2DLines(10);  
 TLine[] perpendicularToFirst = lineOnPlanes.Where(l => l.CheckPerpendicularTo(lineOnPlanes.First())).ToArray(); // знаходяться прямі, які перпендикулярні до першої прямої  
 TLine[] linesWithPoint = perpendicularToFirst.Where(l => l.CheckPointOnLine(point)).ToArray(); // знаходяться прямі, які проходять через точку  
 TLine[] randomLineInSpaces = lineService.GenerateRandom3DLines(10);  
 TLine[] lineInSpaces =  
 {  
 new LineInSpace((0, 1, 2), (7, 8, 9)),   
 new LineInSpace((3, 3, -6)),   
 new LineInSpace((4, 4, -8))  
 };  
  
 lineService.DisplayLines(lineOnPlanes, "Lines on Plane");  
 lineService.DisplayLines(perpendicularToFirst, "Lines Perpendicular to First");  
 lineService.DisplayLines(linesWithPoint, $"Lines Perpendicular to First with Point: {point}");  
 lineService.DisplayLines(randomLineInSpaces, "Random Lines in Space");  
   
 Console.WriteLine($"Random Line in Space Perpendicular to All:\n{lineService.FindPerpendicularToAll(randomLineInSpaces)}"); // шукає пряму, яка перпендикулярна до всіх прямих в масиві randomLineInSpaces  
   
 lineService.DisplayLines(lineInSpaces, "Lines in Space");  
   
 Console.WriteLine($"Line in Space Perpendicular to All:\n{lineService.FindPerpendicularToAll(lineInSpaces)}"); // шукає пряму, яка перпендикулярна до всіх прямих в масиві lineInSpaces  
 }  
 }  
}

LineService.cs

namespace Lab5;  
  
public class LineService  
{  
 public TLine? FindPerpendicularToAll(TLine[] lines) // перевіряє, чи є серед усіх прямих така пряма, яка є перпендикулярною до всіх інших прямих  
 {  
 for (int index = 0; index < lines.Length; index++)  
 {  
 TLine perpendicularLine = lines[index];  
 bool isPerpendicular =  
 lines.Where((line, j) => index != j).All(line => lines[index].CheckPerpendicularTo(line));  
 if (isPerpendicular) return perpendicularLine;  
 }  
  
 return null;  
 }  
  
 public void DisplayLines(TLine[] lines, string prefix = "") // виводить прямі на консоль  
 {  
 Console.WriteLine(prefix);  
 Array.ForEach(lines, Console.WriteLine);  
 Console.WriteLine();  
 }  
  
 public LineOnPlane GetRandomLine2D(int minRange, int maxRange) // генерує випадкові прямі на площині  
 {  
 Random random = new Random();  
 var pointA = (random.Next(minRange, maxRange), random.Next(minRange, maxRange));  
 var pointB = (random.Next(minRange, maxRange), random.Next(minRange, maxRange));  
 return new LineOnPlane(pointA, pointB);  
 }  
  
 public LineInSpace GetRandomLine3D(int minRange, int maxRange) // генерує випадкові прямі в просторі  
 {  
 Random random = new Random();  
 var pointA = (random.Next(minRange, maxRange), random.Next(minRange, maxRange), random.Next(minRange, maxRange));  
 var pointB = (random.Next(minRange, maxRange), random.Next(minRange, maxRange), random.Next(minRange, maxRange));  
 return new LineInSpace(pointA, pointB);  
 }  
  
 public LineOnPlane[] GenerateRandom2DLines(int count) // генерує масив  
 {  
 var lines = new LineOnPlane[count];  
 for (int i = 0; i < count; i++)  
 {  
 lines[i] = GetRandomLine2D(-9, 9);  
 }  
  
 return lines;  
 }  
  
 public LineInSpace[] GenerateRandom3DLines(int count) // генерує масив  
 {  
 var lines = new LineInSpace[count];  
 for (int i = 0; i < count; i++)  
 {  
 lines[i] = GetRandomLine3D(-9, 9);  
 }  
  
 return lines;  
 }  
}

Point.cs

namespace Lab5;  
  
public abstract class Point { }

PointOnPlane.cs

namespace Lab5;  
  
public class PointOnPlane : Point  
{  
 public int X { get; protected set; }  
 public int Y { get; protected set; }  
  
 public PointOnPlane(int x = 0, int y = 0)  
 {  
 X = x;  
 Y = y;  
 }  
  
 public PointOnPlane((int x, int y) coordinates)  
 {  
 X = coordinates.x;  
 Y = coordinates.y;  
 }  
   
 public override string ToString() => $"({X}, {Y})";  
   
 public static bool operator ==(PointOnPlane point1, PointOnPlane point2) // для порівняння точок на площині  
 {  
 return point1.X == point2.X && point1.Y == point2.Y;  
 }  
  
 public static bool operator !=(PointOnPlane point1, PointOnPlane point2) // для порівняння точок на площині  
 {  
 return point1.X != point2.X || point1.Y != point2.Y;  
 }  
  
}

PointInSpace.cs

namespace Lab5;  
  
public class PointInSpace : PointOnPlane  
{  
 public int Z { get; protected set; }  
  
 public PointInSpace(int x = 0, int y = 0, int z = 0) : base(x, y)  
 {  
 Z = z;  
 }  
  
 public PointInSpace((int x, int y, int z) coordinates)  
 {  
 X = coordinates.x;  
 Y = coordinates.y;  
 Z = coordinates.z;  
 }  
  
 public override string ToString() => $"({X}, {Y}, {Z})";  
  
 public static bool operator ==(PointInSpace point1, PointInSpace point2) // для порівняння точок в просторі  
 {  
 return point1.X == point2.X && point1.Y == point2.Y && point1.Z == point2.Z;  
 }  
  
 public static bool operator !=(PointInSpace point1, PointInSpace point2) // для порівняння точок в просторі  
 {  
 return point1.X != point2.X || point1.Y != point2.Y || point1.Z != point2.Z;  
 }  
}

TLine.cs

namespace Lab5;  
  
public abstract class TLine  
{  
 public abstract bool CheckParallelTo(TLine line);  
 public abstract bool CheckPerpendicularTo(TLine line);  
 public abstract bool CheckPointOnLine(Point point);  
}

LineOnPlane.cs

namespace Lab5;  
  
public class LineOnPlane : TLine  
{  
 public PointOnPlane A { get; private set; }  
 public PointOnPlane B { get; private set; }  
 public PointOnPlane VectorCoord => new PointOnPlane(B.X - A.X, B.Y - A.Y);  
  
 public LineOnPlane()  
 {  
 A = new PointOnPlane();  
 B = new PointOnPlane(2);  
 }  
  
 public LineOnPlane(PointOnPlane b)  
 {  
 A = new PointOnPlane();  
 B = b;  
 }  
  
 public LineOnPlane(PointOnPlane a, PointOnPlane b) : this(b)  
 {  
 if (a != b) A = a;  
 }  
  
 public LineOnPlane((int x, int y) b)  
 {  
 A = new PointOnPlane();  
 B = new PointOnPlane(b);  
 }  
  
 public LineOnPlane((int x, int y) a, (int x, int y) b) : this(b)  
 {  
 if(a != b) A = new PointOnPlane(a);  
 }  
  
 public override bool CheckParallelTo(TLine line)  
 {  
 return line is LineOnPlane planeLine && planeLine.VectorCoord.X != 0 && planeLine.VectorCoord.Y != 0 &&  
 (decimal) VectorCoord.X / planeLine.VectorCoord.X == (decimal) VectorCoord.Y / planeLine.VectorCoord.Y;  
 }  
  
 public override bool CheckPerpendicularTo(TLine line)  
 {  
 return line is LineOnPlane planeLine &&  
 VectorCoord.X \* planeLine.VectorCoord.X + VectorCoord.Y \* planeLine.VectorCoord.Y == 0;  
 }  
  
 public override bool CheckPointOnLine(Point point)  
 {  
 if (point is not PointOnPlane point2D) return false;  
 LineOnPlane lineWithPoint = new LineOnPlane(A, point2D);  
 return CheckParallelTo(lineWithPoint);  
 }  
  
 public override string ToString() => $"A {A}\tB {B}";  
}

LineInSpace.cs

namespace Lab5;  
  
public class LineInSpace : TLine  
{  
 public PointInSpace A { get; private set; }  
 public PointInSpace B { get; private set; }  
 public PointInSpace VectorCoord => new PointInSpace(B.X - A.X, B.Y - A.Y, B.Z - A.Z);  
  
 public LineInSpace()  
 {  
 A = new PointInSpace();  
 B = new PointInSpace(2);  
 }  
  
 public LineInSpace(PointInSpace b)  
 {  
 A = new PointInSpace();  
 B = b;  
 }  
  
 public LineInSpace(PointInSpace a, PointInSpace b) : this(b)  
 {  
 if (a != b) A = a;  
 }  
  
 public LineInSpace((int x, int y, int z) b)  
 {  
 A = new PointInSpace();  
 B = new PointInSpace(b);  
 }  
  
 public LineInSpace((int x, int y, int z) a, (int x, int y, int z) b) : this(b)  
 {  
 if (a != b)   
 A = new PointInSpace(a);  
 }  
  
 public override bool CheckParallelTo(TLine line)  
 {  
 return line is LineInSpace line3D &&  
 line3D.VectorCoord.X != 0 && line3D.VectorCoord.Y != 0 && line3D.VectorCoord.Z != 0 &&  
 (decimal) VectorCoord.X / line3D.VectorCoord.X == (decimal) VectorCoord.Y / line3D.VectorCoord.Y &&  
 (decimal) VectorCoord.X / line3D.VectorCoord.X == (decimal) VectorCoord.Z / line3D.VectorCoord.Z;  
 }  
  
 public override bool CheckPerpendicularTo(TLine line)  
 {  
 return line is LineInSpace line3D &&  
 VectorCoord.X \* line3D.VectorCoord.X + VectorCoord.Y \* line3D.VectorCoord.Y +  
 VectorCoord.Z \* line3D.VectorCoord.Z == 0;  
 }  
  
 public override bool CheckPointOnLine(Point point)  
 {  
 if (point is not PointInSpace point3D) return false;  
 LineInSpace lineWithPoint = new LineInSpace(A, point3D);  
 return CheckParallelTo(lineWithPoint);  
 }  
  
 public override string ToString() => $"A {A}\tB {B}";  
}

1. Відеокопія результатів роботи програми:

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, черно-белый

Автоматически созданное описание