Министерство образования и науки Российской федерации
ФГБОУ ВПО "Ярославский государственный
педагогический университет имени К.Д. Ушинского"

Вычислительная геометрия для студентов направления "Педагогическое образование"(профиль "Информатика и информационные технологии в образовании")

Часть 1

Учебное пособие

Ярославль
2013

УДК
ББК
Д48

Печатается по решению
редакционно-издательского совета
ЯГПУ им. К.Д. Ушинского

Рецензент: к.физ.-мат. н., доцент кафедры
теоретической информатики ЯрГУ
Ю.А. Белов

Вычислительная геометрия для студентов направления "Педагогическое образование"(профиль "Информатика и информационные технологии в образовании") Часть 1: учебное пособие/сост.: Н.И. Заводчикова, Н.П. Федотова - Ярославль: Изд-во ЯГПУ, 2013. - 71с.

ISBN

Данная работа предназначена для студентов дневного и заочного отделений педагогических вузов, специализирующихся в области информатики. Работа составлена на основе опыта преподавания данного курса на физико-математическом факультете ЯГПУ для студентов специальности "Информатика направления "Педагогическое образование профиль "Информатика и информационные технологии в образовании".

УДК
ББК
Составители:
к. пед. н., ст. преподаватель Н.И. Заводчикова;
к. ф.-м. н., ассистент Н.П. Федотова
ISBN
Ярославский государственный педагогический университет имени К.Д. Ушинского, 2013

Содержание

Введение

Вычислительная геометрия — это раздел математики (информатики), изучающий алгоритмы решения геометрических задач. Такие задачи возникают в компьютерной графике, проектировании, черчении и др. Очевидным местом возникновения таких задач является визуализация, т.е. изображение различных объектов на экране. Другие примеры менее очевидны, но не менее важны. Например, нахождение выпуклой оболочки (т.е. наименьшего выпуклого множества, содержащего данные точки) играет существенную роль в задачах поиска оптимального решения.

Исходными данными в указанных задачах могут быть множество точек, набор отрезков, многоугольник (заданный, например, списком своих вершин в порядке обхода против часовой стрелки) и т. п. Результатом может быть либо ответ на какой-то вопрос (пересекаются ли эти два отрезка?), либо какой-то геометрический объект (например, наименьший выпуклый многоугольник, содержащий данные точки).

Основная задача вычислительной геометрии состоит в разработке алгоритмов работы с геометрическими объектами, которые не используют тригонометрические функции и вещественную арифметику.

Необходимость изучения курса вычислительной геометрии будущими инженерам и программистам очевидна, однако является целесообразным включить данный раздел дискретной математики и в программу подготовки будущих учителей информатики.

Целью подобного курса является знакомство студентов с основными алгоритмами вычислительной геометрии. Основная задача курса - углубление математического образования и развитие практических навыков в области прикладной математики. Студенты должны быть готовы использовать полученные в этой области знания, как при изучении смежных дисциплин, так и в профессиональной деятельности, в частности при обучении информатике старшеклассников средней школы.

Исходя из специфики педагогического вуза, основное содержание курса в подготовке будущих учителей информатики должны занимать комбинаторные алгоритмы вычислительной геометрии. На наш взгляд, необходимо рассмотреть следующие вопросы:

- предмет вычислительной геометрии, описание основных геометрических объектов в программировании;

- взаимное расположение точек и фигур на плоскости и в пространстве;
- решение задач школьных олимпиад по программированию;
- алгоритмы построения выпуклой оболочки;
- алгоритмы триангуляции полигонов и набора точек;
- алгоритм нахождения пересечения выпуклых многоугольников;
- задачи об изотетичных прямоугольниках;
- метод "Разделяй и властвуй";
- методы вращающейся и движущейся прямой.

Первым трем пунктам и посвящена первая часть пособия. Во второй части будут рассмотрены оставшиеся вопросы.

Так же представляется необходимым рассмотреть некоторые аспекты реализации основных геометрических объектов и алгоритмов работы с ними на различных языках программирования, с использованием различных технологий программирования, поэтому в данном пособии описаны примеры на двух языках программирования Паскаль и C#.

Изучение курса вычислительной геометрии позволит будущим учителям информатики получить не только представление об энергично развивающемся разделе дискретной математики, но и необходимую подготовку для работы с одаренными школьниками, участвующими в олимпиадах по программированию.

1 Описание основных геометрических объектов на плоскости и в пространстве

1.1 Точка, вектор

Положение любой точки P в пространстве (в частности, на плоскости) может быть определено при помощи той или иной системы координат. Числа, определяющие положение точки, называются координатами этой точки.

Наиболее употребительные координатные системы - **декартовы прямоугольные**. Иногда на плоскости применяют полярные системы координат, а в пространстве - цилиндрические или сферические системы координат.

Полярными координатами точки P называются радиус-вектор ρ - расстояние от точки P до заданной точки O (полюса) и полярный угол φ - угол между прямой OP и заданной прямой, проходящей через полюс (полярной осью). Полярный угол считается положительным при отсчете от полярной оси против часовой стрелки и отрицательным при отсчете в обратную сторону.

Формулы для перехода от полярных координат к декартовым:

$$x = \rho * cos(\varphi), y = \rho * sin(\varphi) \tag{1}$$

и обратно:

$$\rho = \sqrt{x^2 + y^2}, \varphi = \text{arctg}(y/x) = \text{arcsin}(y/\rho).$$

Цилиндрические координаты точки P определяются следующим образом: ρ и φ - полярные координаты проекции точки P на основную плоскость (обычно xOy), z - аппликата - расстояние от точки P до основной плоскости.

Формулы для перехода от цилиндрических координат к декартовым:

$$x = \rho * \cos(\varphi), y = \rho * \sin(\varphi), z = z$$

и обратно:

$$\rho = \sqrt{x^2 + y^2}, \varphi = \text{arctg}(y/x) = \text{arcsin}(y/\rho).$$

Сферические координаты точки P определяются следующим образом: ρ - длина радиус-вектора, φ - долгота, θ - полярное расстояние. Если давать сферическим координатам значения в следующих пределах: $0 \leq \rho < -\infty$; $-\pi < \varphi \leq \pi$; $0 \leq \theta \leq \pi$, то получаются однозначно все точки пространства.

Формулы перехода от сферических координат к декартовым:

$$x = \rho * \sin(\varphi) * cos(\theta), y = \rho * \sin(\varphi) * \sin(\theta), z = \rho * \cos(\varphi)$$

и обратно:

$$\rho = \sqrt{(}x^2 + y^2 + z^2), \varphi = \text{arctg}(y/x), \theta = \text{arctg}(\sqrt{((}x^2 + y^2)/z)).$$

Описание точки в программе зависит от языка программирования. В языке Паскаль для описания следует использовать записи.

```
type point=record
  x:real;
```

```
  y:real;
  z:real;(при необходимости)
end;
```

При написании программы на объектно-ориентированном языке для описания геометрических объектов следует использовать классы. Класс, описывающий точку можно реализовать по-разному. Простейшим примером, аналогичным приведенному на Паскале варианту является следующий.

```
public class Point
{
  public float X;
  public float Y;
  public Point(float x, float y){
    this.X = x;
    this.Y = y;
  }
}
```

Вектор задается своими координатами.

На Паскале определим для вектора тип vector

```
type vector=record
  x:real;
  y:real;
  z:real;(при необходимости)
end;
```

На C# класс, описывающий вектор можно реализовать по-разному. Простейшим примером, аналогичным приведенному на Паскале варианту является следующий.

```
public class Vector
{
  public float X;
  public float Y;
  public Vector(float x, float y){
    this.X = x;
    this.Y = y;
  }
}
```

Напомним, что если точки P_1 и P_2 имеют координаты (x_1, y_1, z_1) и (x_2, y_2, z_2) соответсвенно, то вектор $\vec{P_1P_2}$ имеет координаты $(x_2 - x_1, y_2 - y_1, z_2 - z_1)$.

1.2 Описание геометрических объектов произвольной размерности

Одни и те же геометрические объекты средствами одного и того же языка программирования можно реализовать по-разному. Рассмотрим варианты реализации точки (ситуация с реализацией векторов аналогична). Возникает вопрос с размерностью. Как лучше поступить:

- разрабатывать классы (записи и функции) для каждой размерности отдельно - класс (запись) для точек на плоскости, класс (запись) для точек в трехмерном пространстве...

- разрабатывать класс (запись и функции) для точки в пространстве произвольной размерности?

Ответ на этот вопрос неоднозначен. Следует учесть такие моменты:

- чаще всего требуется решить конкретную задачу, для которой размерность не варьируется;

- неэффективно переписывать одни и те же методы для Точек разной размерности;

- удобство использования общего класса для любой размерности;

- потеря памяти при описании общего класса;

- объем кода; использование библиотек классов;

- имеет значение язык программирования; на одном языке лучше разрабатывать так, а на другом - иначе.

Рассмотрим некоторые моменты более подробно. Если реализовывать общий класс для точек пространства произвольной размерности, то координаты точки необходимо хранить в какой-то структуре данных.

Использование списков слишком усложнит программу. При использовании массивов возникает вопрос о потере памяти.

В языке Pascal размерность массива приходится указывать в разделе типов, память выделяется на этапе компиляции, поэтому часть памяти будет выделена, но не использована, кроме того, все равно размерность пространства будет ограничена некоторой константой. Исходя из всего этого следует вывод о том, что при написании программы на Паскале лучше использовать отдельную запись для точек каждой размерности и

разрабатывать однотипные процедуры и функции. Пример записи приведен в пункте 1.1. Данный вариант реализации не плох, поскольку абсолютное большинство прикладных задач возникает в двух- или трехмерных пространствах.

В отличие от Паскаля в языке C# описание переменных возможно в любом месте программы, размерность массива необязательно указывать при его описании, память выделяется на этапе исполнения программы, т.е. динамически, поэтому проблем с потерей памяти не возникает и есть возможность разработать общий класс для точек произвольной размерности. Во-первых, это сократит программный код. Во-вторых, можно будет использовать пространства любой размерности.Рассмотрим некоторые детали реализации подобного класса.

Класс „Точка пространства R^n“должен содержать как минимум одно поле - массив координат точки. Длину массива, т.е. размерность пространства можно и не включать в список полей класса, поскольку у массива есть свойство Длина. Поскольку постоянное обращение к этому свойству снизит эффективность программы, включим в поля класса размерность пространства. Чтобы не загромождать код, поля класса сделаем открытыми, хотя напомним, что хорошим стилем программирования является написание закрытых полей и открытых свойств, обеспечивающих доступ к полям и целостность данных.

```
// класс „Точка пространства R^n “
public class PointRN
{
  // размерность пространства
  public int n;
  // массив координат
  public float[] coords;
}
```

Для создания экземпляров класса, как известно, используется конструктор. Простейшими конструкторами являются конструктор по умолчанию и конструктор, принимающий в качестве параметров значения полей.

```
public PointRN (){}
public PointRN (int n, float[] c)
{
```

```
    this.n = n;
    this.coords = c;
}
```

Если мы ограничимся этими конструкторами, то пользователю придется самостоятельно выделять память для массива координат. Используя второй конструктор, экземпляр создается так:

```
float[] c = new float[2] { 1, 2 };
PointRN P = new PointRN(2, c);
```

При использовании первого конструктора по умолчанию экземпляр точки на плоски создается следующим образом.

```
PointRN P = new PointRN();
float[] c = new float[2] { 1, 2 };
P.coords= c;
P.n = 2;
```

Очевидно,это слишком громоздко для создания точки на плоскости. Поэтому реализуем еще несколько конструкторов для удобства пользователя. Первое, что можно сделать для пользователя, - это снять с него задачу выделения памяти. Напишем конструктор, который принимает в качестве параметра одно целое число - размерность пространства, инициализирует соответствующее поле и выделяет место для массива координат.

```
public PointRN(int n)
{
    this.n = n;
    this.coords = new float[n];
}
```

Создание экземпляра класса с использованием данного конструктора будет похоже на инициализацию переменной типа Точка в Паскале.

```
// создание точки в пространстве с координатами (1, 2, 3)
PointRN P3 = new PointRN(3);
P3.coords[0] = 1;
P3.coords[1] = 2;
P3.coords[2] = 3;
// создание точки на плоскости с координатами (1, 2)
PointRN P2 = new PointRN(2);
P2.coords[0] = 1;
P2.coords[1] = 2;
```

Чтобы пользователю было еще удобнее работать с нашим классом, напишем еще два конструктора для создания точки на плоскости и для создания точки в трехмерном пространстве. Конструктор, создающий точку на плоскости, будет принимать два вещественных параметра - координаты и точки, инициализировать поле Размерность, выделять место для массива длины 2, инициализировать массив координат точки.

```
public PointRN(float x, float y)
{
  this.n = 2;
  this.coords = new float[2];
  this.coords[0] = x;
  this.coords[1] = y;
}
```

Создание экземпляра класса с использованием этого конструктора займет всего одну строчку.

```
PointRN P2 = new PointRN(1,2);
```

Аналогичным образом создаем конструктор для точки трехмерного пространства.

```
public PointRN(float x, float y, float z)
{
  this.n = 3;
  this.coords = new float[3];
  this.coords[0] = x;
  this.coords[1] = y;
  this.coords[1] = z;
}
```

Создание экземпляра класса: PointRN P3 = new PointRN(1,2,3);

Напишем также метод класса, который создает строковое представление точки вида (1, 2, 3, 4) для дальнейшего вывода ее на консоль. Т.е. напишем явное преобразование типа Точка в тип Строка для удобства использования при печати, чтобы печать выполнялась автоматически и не было необходимости печатать все координаты по-отдельности.

```
public override string ToString()
{
  // в переменной temp накапливаем строковое представление
  // для экономии памяти используем класс StringBuilder, а не String
```

```
        StringBuilder temp = new StringBuilder();
        // заносим открывающую скобку
        temp.Append("(");
        // заносим все координаты кроме последней и запятые после них
        for (int i = 0; i < n-1; i++) temp.Append(coords[i]+ ");
        // заносим последнюю координату
        temp.Append(coords[n-1]);
        // заносим закрывающую скобку
        temp.Append(") ");
        return temp.ToString();
    }
```

Теперь для печати точки P пользователю достаточно написать Console.WriteLine(P); не задумываясь, о том, пространства какой размерности эта точка, какие у нее координаты, как к ним обратиться.

Приведем код класса полностью.

```
public class PointRN
{
    public int n;
    public float[] coords;

    public PointRN(int n, float[] c)
    {
        this.n = n;
        this.coors = c;
    }
    public PointRN(int n)
    {
        this.n = n;
        this.coords = new float[n];
    }
    public PointRN(float x, float y)
    {
        this.n = 2;
        this.coords = new float[2];
        this.coords[0] = x;
        this.coords[1] = y;
```

```
    }
    public PointRN(float x, float y, float z)
    {
        this.n = 3;
        this.coords = new float[3];
        this.coords[0] = x;
        this.coords[1] = y;
        this.coords[1] = z;
    }
    public override string ToString()
    {
        StringBuilder temp = new StringBuilder();
        temp.Append("(");
        for (int i = 0; i < n-1; i++) temp.Append(coords[i]+ ");
        temp.Append(coords[n-1]);
        temp.Append(") ");
        return temp.ToString();
    }
}
```

Приведем код программы, которая работает с созданным классом.

```
class Program
{
    static void Main(string[] args)
    {
        float[] c = new float[2]  1, 2 ;
        PointRN P = new PointRN(2, c);
        Console.WriteLine(P);

        PointRN P2 = new PointRN(4,4);
        Console.WriteLine(P2);              PointRN P3 = new PointRN(3);
        P3.coords[0] = 1;
        P3.coords[1] = 2;
        P3.coords[2] = 3;
        Console.WriteLine(P3);

        PointRN P4 = new PointRN(1,2);
```

```
            Console.WriteLine(P4);

            Console.ReadKey();
        }
    }
```

1.3 Использование модулей и библиотек классов

Если задача подразумевает работу с различными геометрическими объектами, то код часто является достаточно объемным и его необходимо структурировать. Разумеется речь идет не только о написании процедур, функций или методов. Их обычно становиться настолько много, что необходимо хранить их отдельно от разрабатываемой программы. Кроме того, необходимо реализовать возможность повторного использования кода в полной мере, т.е. и при решении других задач, при написании других программ. Для этого в языке Паскаль используются модули, а в C# - библиотеки классов. Рассмотрим, как их реализовать.

Как известно, в языке Паскаль имеются модули, содержащие структуры данных, процедуры и функции для работы со стандартными объектами. Например, модуль CRT. Этот модуль подключается с помощью директивы Uses CRT, после этого можно использовать , например, стандартную процедуру очистки экрана clrscr. С ранних версий Паскаль позволяет аналогичным образом создавать и подключать свои модули. Модули хранятся в папке UNITS и представляют собой файлы на языке Паскаль. Модуль имеет следующую структуру:

- ключевое слово unit и название модуля;
- раздел interface, содержащий описание типов,
- объявление процедур и функций с модификаторами доступа;
- раздел implementation, содержащий реализацию этих процедур и функций.

Приведем пример простейшего модуля, содержащего описание точки на плоскости и процедуры печати координат этой точки. Создаем новый файл MyGeometry.pas.

```
unit MyGeometry;
interface
  type Point = record x:real; y:real; end;
```

```
  procedure PrintPoint(p:Point);
implementation
  procedure PrintPoint(p:Point);
  begin
    write('(',p.x,',',p.y,')');
  end;
end.
```

Следует обратить внимание на такие моменты. Модули нельзя исполнять. Лучше, чтобы название файла совпадало с тем именем, которое пишется после слова unit. Модуль нужно сохранить в папке UNITS. После этого для использования модуля в разрабатываемой программе достаточно использовать директиву USES. Приведем пример простейшей программы, использующей созданный модуль.

```
uses MyGeometry;
var p:Point;
begin
  p.x:=1; p.y:=2;
  PrintPoint(p);
end.
```

Заметим, что в данном коде нет описания типа Point, есть только описание переменной и инициализация полей.

В языке C# для реализации механизма повторного использования кода используются библиотеки классов. Создадим библиотеку классов, аналогичную только что рассмотренному модулю на Паскале. Библиотека будет содержать всего один класс для описания точек на плоскости и такой же метод печати их координат. Для этого:

- создаем новый проект,
- выбираем тип - библиотека классов,
- даем проекту имя - MyGeometry,
- добавляем класс Point, содержащий приведенный ниже код.

```
using System;
using System.Text;
namespace MyGeometry
{
  public class Point
  {
```

```
        public float X;
        public float Y;
        public Point(float x, float y)
        {
        this.X = x;
        this.Y = y;
        }
        public override string ToString()
        {
          return String.Format("("+ X +  "+ Y + ") ");
        }
      }
    }
```

Сохраняем библиотеку классов в любом месте. Для использования созданной библиотеки в разрабатываемой программе необходимо:

- создать новый проект,
- выбрать тип - консольное приложение,
- дать новому проекту имя,
- в меню Проект выбрать пункт "Добавить ссылку в открывшемся диалоговом окне на вкладке Обзор создать ссылку на созданную библиотеку классов,
- в начале программы используя директиву Using подключить MyGeometry,
- написать программный код.

Приведем пример кода, аналогичного коду на Паскале.

```
using System;
using System.Text;
using MyGeometry;
namespace ConsoleApplication1
{
  class Program
    {
      static void Main(string[] args)
      {
        Point P = new Point(1, 0);
        Console.WriteLine("Точка P: "+ P.ToString());
```

```
            Console.ReadKey();
        }
    }
}
```

1.4 Прямая на плоскости

1. **Общее уравнение прямой**

Прямая на плоскости задается уравнением вида $Ax + By + C = 0$. Таким образом, если точка (x_1, y_1) принадлежит прямой, то $Ax_1 + By_1 + C = 0$.

На Паскале определим для прямой тип line

```
type line=record
   A:real;
   B:real;
   C:real;
end;
```

На C# определим класс Line.

```
public class Line
{
   public float A;
   public float B;
   public float C;
   public Point(float a, float b, float c){
      this.A = a;
      this.B = b;
      this.C = c;
   }
}
```

Благодаря свойству прямой разделять пространство на два подпространства уравнение f(x,y)=0, позволяет с минимальными вычислительными затратами (два умножения и два сложения) определить принадлежность точки прямой, а также ориентацию двух точек относительно прямой.

Если $f(x_1, y_1) \cdot f(x_2, y_2) > 0$, то точки (x_1, y_1) и (x_2, y_2) лежат по одну сторону от прямой. Если $f(x_1, y_1) \cdot f(x_2, y_2) < 0$, то точки (x_1, y_1) и (x_2, y_2) лежат по разные стороны от прямой.

Также напомним, что (A, B) – координаты нормального вектора к прямой, $(-B, A)$ – координаты направляющего вектора к прямой. Зная эти соотношения легко построить уравнения прямой параллельной данной и перпендикулярной данной.

Замечание: при решении задач вычислительной геометрии сравнение вещественных величин необходимо производить с заданной точностью. То есть вместо проверки равенства $Ax + By + C = 0$ необходимо осуществлять проверку неравенства $abs(Ax + By + C) < eps$, где eps некоторая константа, задающая точность.

Так, например, на Паскале функция определения принадлежности точки прямой имеет вид:

```
function pl(p:point;l:line):boolean;
begin
pl:=abs(l.a*p.x+l.b*p.y+p.c)<eps;
end;
```

Рассмотрим некоторые аспекты решения задачи о принадлежности точки A прямой l на плоскости на языке C#

Проблемы с точностью вычислений решаются так же, как и в Паскале. Это особенность предметной области, а не среды программирования. В отличие от записей, потребуется разработать классы для точки и прямой на плоскости. Примеры уже приводились ранее. Кроме того, есть класс, содержащий саму программу (точку входа). Поскольку мы работаем с несколькими классами, возникает два естественных вопроса.

1. Как обеспечить взаимодействие классов?

2. В каком из трех классов программировать метод, решающий поставленную задачу?

Первый вопрос решается просто. Достаточно сделать классы открытыми, и тогда экземпляры этих классов можно будет создавать в любом месте кода.

Ответ на второй вопрос неоднозначен в общем случае. В любом из трех классов есть возможность написать метод, решающий задачу. Если задача типичная и, скорее всего, будет неоднократно использоваться при решении других задач, то лучше программировать метод, не в основной программе, а в наиболее подходящем классе, описывающем геометрический объект. Тогда, в последствие можно будет поместить этот метод вместе с классом в библиотеку классов и использовать повторно. Если выбирать один класс, то в данном случае - это класс прямой.

Тем не менее, хорошим стилем программирования принято считать избыточный код, т.е. предоставление пользователю возможности использовать различные методы различных классов для решения одной и той же задачи, поэтому вполне уместно написать метод, решающий данную задачу во всех трех классах. Приведем все три варианта метода еще по одной причине - нужно уметь реализовать любой из вариантов, поскольку задачи бывают разные. Обращаем внимание на то, что вызов методов осуществляется по-разному и что один метод является статическим, а два - динамическими.

```
// принадлежит ли точка A прямой l
using System;
using System.Text;
namespace ConsoleApplication1
{
  public class Point
  {
    public float X;
    public float Y;
    public Point(float x, float y)
    {
      this.X = x;
      this.Y = y;
    }
    public override string ToString()
    {
      return String.Format("("+ X +  "+ Y + ") ");
    }
    // реализация динамического метода в классе точки
    public bool on_line(Line l)
    {
      float eps = 0.0000001f;
      if (l.A * this.X + l.B * this.Y + l.C < eps) return true;
      else return false;
    }
  }
  public class Line
```

```
{
   public float A;
   public float B;
   public float C;
   public Line(float a, float b, float c)
   {
      this.A = a;
      this.B = b;
      this.C = c;
   }
   public override string ToString()
   {
      return String.Format( + A + "x + "+ B + "y +"+ C + - 0 ");
   }
   // реализация динамического метода в классе прямой
   public bool contains(Point p)
   {
      float eps = 0.0000001f;
      if (this.A * p.X + this.B * p.Y + this.C < eps) return true;
      else return false;
   }
}
class Program
{
   // реализация статического метода в классе программы
   public static bool point_on_line(Point p, Line l)
   {
      float eps = 0.0000001f;
      if (l.A * p.X + l.B * p.Y + l.C < eps) return true;
      else return false;
   }
   static void Main(string[] args)
   {
      Point A = new Point(0, 0);
      Console.WriteLine("Точка A: "+ A.ToString());
      Line l = new Line(1, 0, 0);
```

```
            Console.WriteLine("Прямая l:"+ l.ToString());
            // вызов метода класса программы
            if (point_on_line(A, l)) Console.WriteLine("точка A принадлежит
            прямой l");
            else Console.WriteLine("точка A не принадлежит прямой l");
            // вызов метода класса прямой
            if (l.contains(A)) Console.WriteLine("точка A принадлежит
            прямой l");
            else Console.WriteLine("точка A не принадлежит прямой l");
            // вызов метода класса точки
            if (A.on_line(l)) Console.WriteLine("точка A принадлежит
            прямой l");
            else Console.WriteLine("точка A не принадлежит прямой l");
            Console.ReadKey();
        }
    }
}
```

2. **Уравнение прямой, проходящей через две точки**

Если даны точки (x_1, y_1) и (x_2, y_2), то вычислить коэффициенты прямой можно используя следующие формулы:

$A = y_2 - y_1$;

$B = x_1 - x_2$;

$C = x_1 \cdot (y_1 - y_2) + y_1 \cdot (x_2 - x_1)$.

3. **Параметрическое уравнение прямой**

Любой вектор приложенный к точке $P_1(x_1, y_1)$ и заканчивающийся в произвольной точке $P(x, y)$, лежащий на прямой, можно получить из вектора P_1P_2 путем умножения на некоторое вещественное число t. Тогда для каждой из координат в отдельности справедливо:

$(x - x_1) = t \cdot (x_2 - x_1)$;

$(y - y_1) = t \cdot (y_2 - y_1)$.

Выразив отсюда x и y получаем систему параметрических уравнений, которой удовлетворяет каждая точка прямой:

$$\begin{cases} x = x_1 + t(x_2 - x_1) \\ y = y_1 + t(y_2 - y_1) \end{cases}$$

Наоборот, если координаты (x, y) точки удовлетворяют этим соотношениям, вектор P_1P коллинеарен P_1P_2 и, значит, точка P лежит на прямой P_1P_2. Таким образом, система уравнений, где параметр t пробегает всю действительную ось задает прямую P_1P_2. Эта же система, но с введенными ограничениями на значения t, будет задавать и отрезок P_1P_2 или луч. Для отрезка t лежит в интервале от 0 до 1, для луча в интервале от 0 до бесконечности.

1.5 Прямая в пространстве

1. **Задание прямой по двум точкам**. Если прямая проходит через точки $M_1(x_1, y_1, z_1)$ и $M_2(x_2, y_2, z_2)$, то уравнения прямой имеют следующий вид:

$$\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1} = \frac{z - z_1}{z_2 - z_1}.$$

Однако, при программировании использовать уравнения в виде отношения нельзя, так как может возникнуть деление на 0. Поэтому используют уравнения

$(x - x_1) \cdot (y_2 - y_1) = (x_2 - x_1) \cdot (y - y_1)$;

$(y - y_1) \cdot (z_2 - z_1) = (z - z_1) \cdot (y_2 - y_1)$.

2. **Параметрическое задание прямой** Аналогично случаю на плоскости получаем параметрические уравнения прямой в пространстве:

$$\begin{cases} x = x_1 + t(x_2 - x_1) \\ y = y_1 + t(y_2 - y_1) \\ z = z_1 + t(z_2 - z_1). \end{cases}$$

3. **Прямая как пересечение двух плоскостей** Две не параллелные и не совпадающие плоскости однозначно задают прямую.

1.6 Плоскость

1. **Общее уравнение плоскости** имеет вид: $Ax+By+Cz+D = 0$, причем (A, B, C) координаты нормального вектора плоскости

2.**Задание плоскости по трем точкам**

Если плоскость проходит через точки $M_1(x_1, y_1, z_1)$, $M_2(x_2, y_2, z_2)$ и $M_3(x_3, y_3, z_3)$, то уравнение плоскости можно найти из следующих

соображений. Точка $P(x, y, z)$ принадлежит плоскости, если вектора $M_1\vec{M}_2, M_2\vec{M}_3, M_3\vec{P}$ компланарны, т.е. их смешанное произведение равно 0.

$$\begin{vmatrix} x_2 - x_1 & y_2 - y_1 & z_2 - z_1 \\ x_3 - x_2 & y_3 - y_2 & z_3 - z_2 \\ x - x_3 & y - y_3 & z - z_3 \end{vmatrix} = 0$$

таким образом, можно найти коэффициенты для общего уравнения плоскости

$$A = (y_2 - y_1) \cdot (z_3 - z_2) - (y_3 - y_2) \cdot (z_2 - z_1);$$
$$B = (x_3 - x_2) \cdot (z_2 - z_1) - (x_2 - x_1) \cdot (z_3 - z_2);$$
$$C = (x_2 - x_1) \cdot (y_3 - y_2) - (x_3 - x_2) \cdot (y_2 - y_1).$$

Так же плоскость можно задать с помощью двух неколлинеарных направляющих для плоскости векторов и точки.

3.**Параметрическое задание плоскости**

Из указанных выше соображений можно найти и параметрические уравнения плоскости. Т.к.вектора $M_1\vec{M}_2, M_1\vec{M}_3, M_1\vec{P}$ компланарны,то вектор $M_1\vec{P}$ является линейной комбинацией векторов $M_1\vec{M}_2$ и $M_1\vec{M}_3$, а значит

$$\begin{cases} x - x_1 = u \cdot (x_2 - x_1) + v \cdot (x_3 - x_1) \\ y - y_1 = u \cdot (y_2 - y_1) + v \cdot (y_3 - y_1) \\ z - z_1 = u \cdot (z_2 - z_1) + v \cdot (z_3 - z_1). \end{cases}$$

На Паскале определим для плоскости тип plane:

```
type plane=record
  a:real;
  b:real;
  c:real;
  d:real;
end;
```

На C# определим для плоскости класс Plane:

```
public class Plane
{
    public float A;
    public float B;
    public float C;
    public float D;
    public Plane(float a, float b, float c, float d)
```

```
        {
            this.A = a;
            this.B = b;
            this.C = c;
            this.D = d;
        }
    }
```

1.7 Отрезок

Отрезок с концами в точках P_1 и P_2 задается координатами своих концов.

На Паскале определим для отрезка следующий тип:

```
type piece=record
  beg:point;
  en:point;
end;
```

На C# определим класс Piece

```
public class Piece
    {
        public Point Beg;
        public Point En;
        public Piece(Point b, Point e)
        {
            this.Beg = b;
            this.En = e;
        }
    /*
```

Лучше добавить этот конструктор по умолчанию, чтобы пользователю не нужно было создавать экземпляры класса Точка. Ведь некоторые пользователи могут об этом просто забыть, и тогда при обращении к полям класса возникнет ошибка.

```
    */
    public Piece()
    {
        this.Beg = new Point();
        this.En = new Point();
    }
```

}

Также выше было сказано о параметрическом способе задания отрезка.

1.8 Луч

Луч можно задавать координатами начала луча и вектором задающим направление. Однако, как правило, но луч задают началом P_1 и точкой на луче P_2. Также выше было сказано о параметрическом способе задания луча.

1.9 Многоугольник

Широкий класс алгоритмов вычислительной геометрии составляют алгоритмы работы с многоугольниками. Прежде чем переходить к их изучению, введем некоторые полезные определения.

Многоугольник — замкнутая ориентированная ломанная без самопересечений и самокасаний, а также ограниченная ею область.

Выпуклый многоугольник — многоугольник, две любые внутренние точки которого можно соединить отрезком, целиком в нём лежащим.

Выпуклая вершина — вершина, внутренний угол при которой меньше, либо равен 180 градусов.

Внутренняя вершина — вершина, внутренний угол при которой больше 180 градусов.

Свойства выпуклых многоугольников:

- любая вершина выпуклого многоугольника выпукла

- пересечение двух выпуклых многоугольников либо пусто, либо является отрезком, либо является выпуклым многоугольником

В памяти компьютера многоугольник можно представить двумя способами.

Во-первых, с помощью данного целого числа n (количество вершин) и массива координат вершин многоугольника.

Во-вторых, в виде кольца вершин, хранящегося в кольцевом списке с двойными связями. Каждый узел соответствует вершине и связан со своими двумя соседями. Следуя связям, можно обойти границу многоугольника в любом направлении, а включая или удаляя узлы (редактируя связи соответствующим образом), можно создавать многоугольники и динамически их модифицировать.

```
type mng=^ ver;
   ver=record
      inf:point;
      left:mng;
      rigth:mng;
end;
```

Приведем процедуру считывания многоугольника (на плоскости) из файла.

```
procedure input(var n:integer;var mng1:mng);
var f:text;
   i:integer;
   begM,endM,p:mng;
begin
assign(f,'in.txt');
reset(f);
readln(f,n);
new(p);
readln(f,p^.inf.x,p^.inf.y);
p^.left:=nil;
p^.rigth:=nil;
begM:=p;
endM:=p;
for i:=1 to n-1 do
   begin
   new(p);
   readln(f,p^.inf.x,p^.inf.y);
   p^.left:=endM;
   p^.rigth:=nil;
   endM^.rigth:=p;
   endM:=p;
   end;
endM^.rigth:=begM;
begM^.left:=endM;
mng1:=begM;
close(f);
```

```
end;
```

Реализация класса Многоугольник с методами добавления вершины и печати списка вершин многоугольника будет на C# будет выглядеть следующим образом.

```
// класс Многоугольник
class Mng
{
  // поля класса
  public Mng left, right;
  public Point vershina;
  // конструктор класса
  public Mng()
  {
    left = null; right = null; vershina = null;
  }
  // метод добавления вершины
  public void AddVer(Point v)
  {
    if (left == null)
    {
    vershina = v; right = this; left = this;
    }
    else
    {
      Mng temp = new Mng();
      temp.vershina = v;
      this.left.right = temp;
      temp.left = this.left;
      this.left = temp;
      temp.right = this;
    }
  }
  // метод печати списка вершин многоугольника
  public void Print()
  {
    Mng first = this;
```

```
        Mng temp = this;
        Console.WriteLine(temp.vershina.ToString());
        temp = temp.right;
        while (first != temp)
        {
          Console.WriteLine(temp.vershina.ToString());
          temp = temp.right;
        }
      }
    }
```

В отличие от примера на Паскале, в приведенном далее коде на C# вершины многоугольника инициализируются в самой программе.

```
    class Program
    {
      static void Main(string[] args)
      {
        // создаем 4 вершины
        Point p1 = new Point(1, 1); Point p2 = new Point(2, 1);
        Point p3 = new Point(3, 1); Point p4 = new Point(4, 1);
      // создаем многоугольник из этих вершин
      // создаем пустой многоугольник
      Mng myMng = new Mng();
      // добавляем в него вершины
      myMng.AddVer(p1); myMng.AddVer(p2);
      myMng.AddVer(p3); myMng.AddVer(p4);
      // печатаем многоугольник
      myMng.Print();
      }
    }
```

В случае чтения многоугольника из файла программу придется немного изменить: сначала создаем пустой многоугольник, далее в цикле считываем координаты очередной вершины, создаем вершину и добавляем ее в многоугольник.

Замечание: в пространстве многоугольник задается аналогично, но необходимо следить, чтобы все вершины принадлежали одной плоскости

1.10 Ломаная

Как и многоугольник, ломаную в памяти компьютера можно представлять двумя способами. Во-первых, с помощью данного целого числа n (количество вершин) и массива координат вершин многоугольника. Во-вторых, в виде списка вершин, хранящегося в списке с двойными связями. Каждый узел соответствует вершине и связан со своими двумя соседями.

1.11 Окружность

Окружность задается координатами центра и радиусом. Уравнение окружности с центром в точке (x_1, y_1) и радиусом r имеет вид $(x - x_1)^2 + (y - y_1)^2 = r^2$. Данная формула также помогает определить лежит ли точка на окружности, внутри неё или вне.

Иногда бывает удобно использовать параметрическое задание окружности

$$\begin{cases} x = x_1 + r * cos(\varphi) \\ y = y_1 + r * sin(\varphi). \end{cases}$$

Сфера

Сфера задается координатами центра и радиусом. Уравнение сферы с центром в точке (x_1, y_1, z_1) и радиусом r имеет вид $(x - x_1)^2 + (y - y_1)^2 + (z - z_1) = r^2$ Данная формула также помогает определить лежит ли точка на сфере, внутри неё или вне.

Иногда бывает удобно использовать параметрическое задание сферы:

$$\begin{cases} x = x_1 + r * \sin(\varphi) * cos(\theta) \\ y = y_1 + r * \sin(\varphi) * \sin(\theta) \\ z = z_1 + r * \cos(\varphi). \end{cases}$$

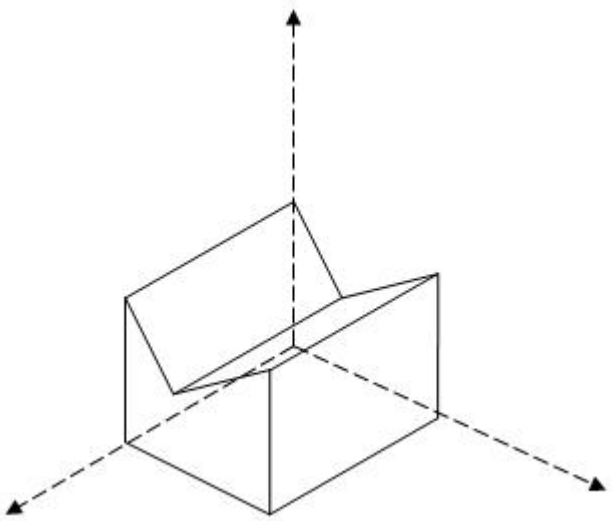
1.12 Многогранник

В общем виде многогранник задается двумя списками:

1. список вершин P, пронумерованных в произвольном порядке

2. массив списков граней G, где G_i - список номеров вершин многогранника i-той грани, перечисленных в порядке их обхода по замкнутому контуру.

Пример



Для многогранника, приведенного на рисунке, описание его вершин и граней будет следующим.

Многогранник имеет 10 вершин

$$P = \begin{bmatrix} 0 & 0 & 3 & 3 & 6 & 6 & 6 & 6 & 0 & 0 & 0 \\ 0 & 4 & 4 & 0 & 0 & 4 & 4 & 0 & 0 & 4 & 0 \\ 6 & 6 & 6 & 3 & 6 & 6 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

и 8 граней

$G = \{G_1, G_2, ...G_7\}$, где

$G_1 = \{1, 2, 3, 4, 1\}$,

$G_2 = \{3, 6, 5, 4, 3\}$,

$G_3 = \{6, 7, 8, 5, 6\}$,

$G_4 = \{1, 9, 8, 5, 4, 1\}$,

$G_5 = \{2, 10, 7, 6, 3, 2\}$,

$G_6 = \{1, 2, 10, 9, 1\}$,

$G_7 = \{9, 4, 7, 8, 9\}$

$G_8 = \{6, 7, 8, 11, 6\}$

Однако, частные виды многогранников задаются проще.

Так при описании пирамиды, как правило, в файле сначала указывают координаты вершины, противоположной основанию, затем число вершин в основании, затем координаты вершин основания. А для хранения используют один массив точек для координат вершин основания и точку - вершину, противоположную основанию.

При описании призмы достаточно указать координаты вершин одного из оснований и вектор, задающий боковую сторону.

1.13 Вопросы и задания для самопроверки

1. Перечислите способы задания точки на плоскости и в пространстве.
2. Напишите процедуру перевода координат точки из декартовой системы координат в полярную.
3. Напишите процедуру перевода координат точки из полярной системы координат в декартову.
4. Перечислите способы задания прямой на плоскости. Почему при решении задач вычислительной геометрии не используется уравнение прямой вида y=kx+b?
5. Напишите уравнение прямой, проходящей через точки:
   (a) A(1, 2) и B(3, 4);
   (b) A(2, 6) и B(3, 5);
   (c) A(7, 1) и B(4, 4);
   (d) A(-1, 4) и B(2, 8).
6. Как определить, принадлежит ли точка прямой, если прямая задана параметрически?
7. Прямая задана параметрически $\begin{cases} x = 5t + 3 \\ y = 3t + 2 \end{cases}$. Определить, принадлежит ли данной прямой точка:
   (a) M(4, 1);
   (b) M(3, 2);
   (c) M(8, 5);
   (d) M(6, 2).
8. Перечислите способы задания прямой в пространстве.
9. Принадлежит ли точка заданной плоскости?

(a) $M(5, 6, 2)$, $3x - 2y + 2z - 7 = 0$;

(b) $M(3, 6, 2)$, $2x - y + 4z - 7 = 0$;

(c) $M(4, 2, 10)$, $2x + 3y - z - 4 = 0$;

(d) $M(2, 5, 3)$, $6x - 3y + 8 = 0$.

10. Прямая задана параметрически $\begin{cases} x = -2t - \frac{8}{7} \\ y = 8t - \frac{3}{7} \\ z = 7t \end{cases}$. Определить, принадлежит ли данной прямой точка:

    (a) M(2, 7, 5);

    (b) M(-2, 0, 1);

    (c) M(1, -4, -5);

    (d) M(1, -9, $\frac{15}{2}$).

11. Сколько уравнений необходимо для того, чтобы задать прямую в пространстве? Почему для задания прямой, проходящей через две точки $M1(x1, y1, z1)$ и $M2(x2, y2, z2)$ нельзя использовать систему уравнений $\frac{x-x1}{x2-y1} = \frac{y-y1}{y2-y1} = \frac{z-z1}{z2-z1}$?

12. Перечислите способы задания плоскости.

13. Перечислите способы задания луча.

14. Перечислите способы задания многоугольника. В чем преимущество того и другого метода?

15. Перечислите способы задания окружности.

16. Перечислите способы задания сферы.

17. Напишите параметрические уравнения окружности с центром в точке O и радиусом R

    (a) $O(0, 0)$, $R = \sqrt{3}$;

    (b) $O(3, -6)$, $R = 7$.

18. Как описывается многогранник в памяти компьютера?

19. Как будет выглядеть файл с данными о кубе длина ребра которого равна 3, а три ребра лежат на координатных осях?

20. Изобразите многогранник. Множество вершин имеет вид:

$$P = \begin{bmatrix} 0 & 0 & 3 & 3 & 4 & 0 & 4 & 4 & 4 & 4 & 0 \\ 5 & 2 & 2 & 2 & 5 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 6 & 6 & 0 & 0 & 6 & 6 & 0 & 0 & 6 & 0. \end{bmatrix}$$

Множество граней имеет вид:

$G = \{G_1, G_2, ...G_7\}$, где

$G_1 = \{1, 2, 3, 5, 1\}$,

$G_2 = \{2, 3, 10, 7, 6, 2\}$,

$G_3 = \{3, 4, 5, 3\}$,

$G_4 = \{3, 4, 9, 10, 3\}$,

$G_5 = \{7, 8, 9, 10, 7\}$,

$G_6 = \{5, 4, 9, 8, 11, 1, 5\}$,

$G_7 = \{1, 2, 6, 11, 1\}$,

$G_8 = \{6, 7, 8, 11, 6\}$.

1.14 Задачи

1. Реализация основных геометрических объектов. В данном задании основная программа должна содержать только вызовы разработанных процедур, функций или методов.

   (a) Разработать структуру данных для геометрического объекта "точка плоскости". Реализовать процедуры, функции или методы для ввода и вывода точки.

   (b) Разработать структуру данных для геометрического объекта "точка трехмерного пространства". Реализовать процедуры, функции или методы для ввода и вывода точки.

   (c) (*) Разработать структуру данных для геометрического объекта "точка пространства произвольной размерности". Реализовать процедуры, функции или методы для ввода и вывода точки.

   (d) Разработать структуру данных для геометрического объекта "вектор на плоскости". Реализовать процедуры, функции или

методы для ввода, вывода вектора, вычисления координат вектора по точкам начала и конца, сложения векторов, умножения вектора на число, линейной комбинации векторов, скалярного и косого произведения векторов.

(e) Разработать структуру данных для геометрического объекта "трехмерный вектор". Реализовать процедуры, функции или методы для ввода, вывода вектора, вычисления координат вектора по точкам начала и конца, сложения векторов, умножения вектора на число, линейной комбинации векторов, скалярного, векторного и смешанного произведения векторов.

(f) (*) Разработать структуру данных для геометрического объекта "вектор пространства произвольной размерности". Реализовать процедуры, функции или методы для ввода, вывода вектора, вычисления координат вектора по точкам начала и конца, сложения векторов, умножения вектора на число, линейной комбинации векторов, скалярного, косого, векторного и смешанного произведения векторов соответствующей размерности.

(g) Разработать структуру данных для геометрического объекта "прямая на плоскости". Реализовать процедуры, функции или методы для ввода и вывода прямой, определения принадлежности точки прямой, построения прямой по двум точкам, по точке и вектору.

(h) Разработать структуру данных для геометрического объекта "прямая в трехмерном пространстве". Реализовать процедуры, функции или методы для ввода и вывода прямой, определения принадлежности точки прямой, построения прямой по двум точкам, по точке и вектору.

(i) . Ввести точки A, B, C плоскости. Построить всевозможные вектора с концами в этих точках. Вычислить значения выражений $2 \cdot AB - CA$; $AB \cdot BA$; $AC \times CA$(косое произведение); определить коллинеарны ли вектора AB и AC.

(j) . Ввести точки D, E, F, G пространства. Построить всевозможные вектора с концами в этих точках. Вычислить $2 \cdot EF + 4\ cdotFG$; $ED \times FE$(векторное произведение); $DE \cdot ED$; (DE, FE, GE); определить компланарны ли вектора DE, FE, GE.

Далее решение каждой задачи необходимо оформить в виде процедуры или функции на Паскале или метода на C#.

2. Уравнение прямой на плоскости

   (a) Найти коэффициенты уравнения прямой $Ax + By + C = 0$, проходящей через две точки $(x, y), (x_1, y_1)$.

   (b) Даны точки A, B, C своими координатами, найти уравнение прямой проходящей через точку C и параллельную прямой AB.

   (c) Даны точки A, B, C своими координатами, найти уравнение прямой проходящей через точку C и перпендикулярную прямой AB.

   (d) Определить принадлежит ли точка прямой. Прямая задана своими коэффициентами.

   (e) Даны три точки, определить лежат ли они на одной прямой.

   (f) Определить положение двух точек относительно прямой (по одну сторону от прямой, по разные, на прямой). Прямая задана своими коэффициентами.

   (g) Прямые заданы своими коэффициентами. Определить их взаимное расположение (пересекаются, не пересекаются, совпадают) .

   (h) Даны точки A, B, C, D. Определить взаимное расположение прямых AB и CD (пересекаются, не пересекаются, совпадают).

3. Параметрическое задание прямой на плоскости. Отрезок. Луч.

   (a) Даны три точки A, B, C. Определить принадлежит ли точка C отрезку AB.

   (b) Даны три точки A, B, C, лежащие на одной прямой. Определить расположение точки C относительно отрезка AB (между точками A и B, вне отрезка за точкой A, вне отрезка за точкой B).

   (c) Даны три точки A, B, C, лежащие на одной прямой. Определить какая из них лежит между двумя другими.

   (d) Даны три точки A, B, C. Определить принадлежит ли точка C лучу AB.

   (e) Даны четыре точки A, B, C и D, лежащие на одной прямой. Определить пересекаются ли отрезки AB и CD.

(f) Даны четыре точки A, B, C и D, лежащие на одной прямой. Определить пересекаются ли лучи $[AB)$ и $[CD)$.

(g) Даны четыре точки A, B, C и D, лежащие на одной прямой. Определить пересекаются ли луч $[AB)$ и отрезок $[CD)$.

(h) Даны три точки A, B, C лежащие на одной прямой. Определить расположение точки C относительно луча AB.

4. Скалярное и косое произведение векторов на плоскости

(a) Угол задан тремя точками A, B, C (точка B - вершина угла). Определить вид угла (острый, тупой или прямой)

(b) Треугольник задан координатами своих вершин. Определить вид треугольника (остроугольный, тупоугольный или прямоугольный)

(c) Даны 4 точки A, B, C и D. Перпендикулярны ли прямые AB и CD?

(d) Даны три точки A, B, C, определить является ли обход $A - B - C$ обходом по часовой стрелке или против.

(e) Треугольник задан координатами вершин. Найти его площадь.

(f) Даны 4 точки A, B, C и D. Выясните, лежит ли точка D внутри угла ABC? Имеется в виду тот из углов ABC, который заметается лучом при повороте от BA к BC против часовой стрелки.

в пространстве

(g) Угол задан тремя точками A, B, C (точка B - вершина угла). Определить вид угла (острый, тупой или прямой)

(h) Даны три точки A, B, C. Определить является ли обход $A - B - C$ обходом по часовой стрелке или против.

5. Уравнение плоскости, прямая в пространстве

(a) Найти коэффициенты уравнения $A + B + Cz + D = 0$ плоскости, проходящей через три точки (x, y, z),$(x1, y1, z1)$, $(x2, y2, z2)$.

(b) Даны точки A, B, C, D своими координатами, найти уравнение плоскости проходящей через точку D и параллельную плоскости ABC.

(c) Даны точки A, B, C, D, E своими координатами, найти уравнение плоскости проходящей через точки D и и перпендикулярную плоскости ABC.

(d) Определить принадлежит ли точка плоскости.

(e) Даны четыре точки, определить лежат ли они в одной плоскости.

(f) Определить положение двух точек относительно плоскости (по одну сторону от плоскости, по разные, на плоскости). Плоскость задана своими коэффициентами.

(g) Даны три точки в пространстве, своими координатами, определить лежат ли они на одной прямой.

(h) Определить принадлежит ли данная точка прямой, заданной двумя плоскостями. Плоскости заданы своими коэффициентами.

2 Некоторые формулы и соотношения

2.1 Скалярное произведение векторов

Скалярное произведение $\vec{a} \cdot \vec{b}$ для векторов $\vec{a} = (x_a, y_a, z_a)$ и $\vec{b} = (x_b, y_b, z_b)$ определяется как $\vec{a} \cdot \vec{b} = x_a \cdot x_b + y_a \cdot y_b + z_a \cdot z_b$.

Напомним, что два вектора $\vec{a}$ и $\vec{b}$ перпендикулярны, тогда, и только тогда, когда их скалярное произведение равно нулю. Если $\vec{a} \cdot \vec{b} > 0$, то угол между векторами меньше $90°$. Если $\vec{a} \cdot \vec{b} < 0$, то угол между векторами больше $90°$.

2.2 Векторное произведение векторов

Векторным произведением неколлинеарных векторов $\vec{a}$ и $\vec{b}$ взятых в данном порядке называется вектор $\vec{p}$, длина которого численно равна площади параллелограмма, построенного на этих векторах; этот вектор перпендикулярен векторам $\vec{a}$ и $\vec{b}$ и направлен так, что базис a, b, p имеет правую ориентацию. Векторное произведение коллинеарных векторов считается равным нуль-вектору.

Если векторы $\vec{a}$ и $\vec{b}$ имеют координаты $(x_1, y_1, z_1), (x_2, y_2, z_2)$, то их векторное произведение имеет координаты

$$\left(\begin{vmatrix} y_1 & z_1 \\ y_2 & z_2 \end{vmatrix}, -\begin{vmatrix} x_1 & z_1 \\ x_2 & z_2 \end{vmatrix}, \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix}\right),$$

то есть $(y_1 \cdot z_2 - y_2 \cdot z_1, z_1 \cdot x_2 - z_2 \cdot x_1, x_1 \cdot y_2 - x_2 \cdot y_1)$.

2.3 Косое произведение векторов

Важную роль во многих геометрических алгоритмах играет определение направления поворота одного вектора к другому. Для решения этой задачи введем понятие косого произведения.

Косое произведение для двух векторов $\vec{a} = (x_a, y_a)$ и $\vec{b} = (x_b, y_b)$ определяется как $\vec{a} \times \vec{b} = \begin{vmatrix} x_a & y_a \\ x_b & y_b \end{vmatrix} = x_a \cdot y_b - y_a \cdot x_b$.

```
function kosoep(v1,v2:vektor):real;
begin
vect:=v1.x*v2.y-v2.x*v1.y;
end;
```

Из представленных определений видим, что косое произведение является z-координатой векторного, поэтому в литературе часто косое произведение векторов называют векторным (на плоскости).

Косое произведение векторов позволяет определить направление поворота от одного вектора к другому:

- если $\vec{a} \times \vec{b} > 0$, то $\vec{a}$ надо поворачивать против часовой стрелки,
- если $\vec{a} \times \vec{b} < 0$, то $\vec{a}$ надо поворачивать по часовой стрелке,
- если $\vec{a} \times \vec{b} = 0$, то векторы либо сонаправлены, либо направлены в противоположные стороны.

2.4 Смешанное произведение векторов

Определим смешанное произведение векторов $\vec{a}, \vec{b}, \vec{c}$ как $(\vec{a}, \vec{b}, \vec{c}) = (\vec{a} \times \vec{b}) \cdot \vec{c}$.

Напомним, что смешанное произведение равно объему параллелепипеда построенного на векторах $\vec{a}, \vec{b}, \vec{c}$ со знаком „плюс“, если тройка $\vec{a}, \vec{b}, \vec{c}$ - правая и со знаком „минус “, если данная тройка - левая.

Значение смешанного произведения для векторов $\vec{a} = (x_a, y_a, z_a)$, $\vec{b} = (x_b, y_b, z_b)$, $\vec{c} = (x_c, y_c, z_c)$ можно вычислить следующим образом

$$(\vec{a}, \vec{b}, \vec{c}) = \begin{vmatrix} x_a & y_a & z_a \\ x_b & y_b & z_b \\ x_c & y_c & z_c \end{vmatrix}.$$

Если вектора $\vec{a}, \vec{b}, \vec{c}$ компланарны, то их смешанное произведение равно 0.

2.5 Угол между векторами

В вычислительной геометрии как правило пользуются понятием ориентированного угла, то есть угла учитывающего взаимное расположение векторов. Ориентированный угол между векторами $\vec{a}$ и $\vec{b}$ положительный, если поворот от вектора $\vec{a}$ к вектору $\vec{b}$ совершается против часовой стрелки, и отрицательный, если наоборот. Таким образом величина ориентированного угла может меняться в интервале $(-\pi; \pi]$.

Для любых векторов $\vec{OA}, \vec{OB}, \vec{OC}$, легко вычислите величину угла AOB, зная величины углов AOC и COB: она равна их сумме с учетом знака. Может случиться, что при суммировании двух положительных (отрицательных) углов результат превзойдет π по модулю. Тогда, чтобы получить правильное значение угла, нужно отнять (добавить) 2π. Замечательно, что при этом не придется рассматривать различные случаи взаимного расположения векторов. В этом и состоит преимущество использования ориентированных углов. Как найти значение ориентированного угла? В этом случае применяется косое произведение векторов.

Угол α между прямыми, с напрвляющими векторами $\vec{a}$ и $\vec{b}$ определяетсбя следующим образом $\alpha = \arctan \frac{\vec{a} \times \vec{b}}{\vec{a} \cdot \vec{b}}$.

Величина ориентированного угла между векторами может меняться в интервале $(-\pi; \pi]$.

Пусть φ угол между векторами $\vec{a}$ и $\vec{b}$, а α угол между прямыми заданными этими векторами, тогда

$\varphi = \pi/2$, если $\vec{a} \cdot \vec{b} = 0$ и $\vec{a} \times \vec{b} > 0$;
$\varphi = -\pi/2$, если $\vec{a} \cdot \vec{b} = 0$ и $\vec{a} \times \vec{b} < 0$;
$\varphi = \alpha$, если $\vec{a} \cdot \vec{b} > 0$;
$\varphi = \pi + \alpha$, если $\vec{a} \cdot \vec{b} < 0$ и $\vec{a} \times \vec{b} \geq 0$;
$\varphi = \alpha - \pi$, если $\vec{a} \cdot \vec{b} < 0$ и $\vec{a} \times \vec{b} < 0$.

2.6 Площадь многоугольника

Ориентированная площадь треугольника ABC – это его площадь, снабженная таким же знаком, как у ориентированного угла между векторами $\vec{AB}$ и $\vec{AC}$, то есть знак зависит от порядка перечисления вершин.

Из определения косого произведения следует, что площадь треугольника с координатами вершин $(x_1, y_1); (x_2, y_2,); (x_3, y_3,)$ равна

$$1/2((x_2 - x_1) \cdot (y_3 - y_1) - (x_3 - x_1) \cdot (y_2 - y_1)).$$

Эту же величину можно вычислить другим способом. Пусть точка O произвольная точка плоскости. Площадь треугольника ABC равна площади треугольника OBC минус площади треугольников OAC и AOB, то есть если сложить ориентированные площади треугольников OAB, OBC, OCA. Это правило работает при любом выборе точки O.

Точно так же для вычисления площади любого многоугольника $A_1, A_2, ...A_n$. Нужно сложить ориентированные площади треугольников $OA_1A_2, OA_2A_3, \ldots, OA_nA_1$. В сумме получится площадь многоугольника, взятая со знаком плюс, если при обходе ломаной $A_1A_2 \ldots A_n$ внутренность многоугольника находится слева, и отрицательной, если она находится справа. Она и называется ориентированной площадью многоугольника.

2.7 Объем многогранника

Объем параллелепипеда построенного на векторах $\vec{a}, \vec{b}, \vec{c}$ равен смешанному произведению этих векторов.

Объем треугольной призмы, построенной на этих же векторах равен половине смешанного произведения, а объем треугольной пирамиды 1/6 смешанного прозведения этих векторов.

Объем произвольной призмы (пирамиды) вычисляют как сумму объемов треугольных призм (пирамид), полученных из данной триангуляцией основания.

2.8 Вопросы и задания для самопроверки

В заданиях 1-5 выберите один правильный ответ из четырех предложенных.

1. Точки P_1, P_2, P_3 лежат на одной прямой, если:
   (a) $\overrightarrow{P_1P_2} \cdot \overrightarrow{P_1P_3} > 0$;
   (b) $\overrightarrow{P_1P_2} \cdot \overrightarrow{P_1P_3} = 0$;
   (c) $\overrightarrow{P_1P_2} \times \overrightarrow{P_1P_3} > 0$;
   (d) $\overrightarrow{P_1P_2} \times \overrightarrow{P_1P_3} = 0$.

2. Угол между векторами P_1P_2 и P_1P_3 прямой, если:
   (a) $\overrightarrow{P_1P_2} \cdot \overrightarrow{P_1P_3} > 0$;

(b) $\overrightarrow{P_1P_2} \cdot \overrightarrow{P_1P_3} = 0$;

(c) $\overrightarrow{P_1P_2} \times \overrightarrow{P_1P_3} > 0$;

(d) $\overrightarrow{P_1P_2} \times \overrightarrow{P_1P_3} = 0$.

3. Угол между векторами P_1P_2 и P_1P_3 тупой, если:

(a) $\overrightarrow{P_1P_2} \cdot \overrightarrow{P_1P_3} > 0$;

(b) $\overrightarrow{P_1P_2} \cdot \overrightarrow{P_1P_3} = 0$;

(c) $\overrightarrow{P_1P_2} \times \overrightarrow{P_1P_3} > 0$;

(d) $\overrightarrow{P_1P_2} \times \overrightarrow{P_1P_3} = 0$.

4. Для того, чтобы векторы $\vec{a}$ и $\vec{b}$ были сонаправлены, необходимо, чтобы:

(a) $\vec{a} \times \vec{b} = 0$ и $\vec{a} \cdot \vec{b} > 0$;

(b) $\vec{a} \times \vec{b} > 0$ и $\vec{a} \cdot \vec{b} = 0$;

(c) $\vec{a} \times \vec{b} = 0$ и $\vec{a} \cdot \vec{b} < 0$;

(d) $\vec{a} \times \vec{b} < 0$ и $\vec{a} \cdot \vec{b} > 0$.

5. Поворот от вектора $\vec{a}$ к вектору $\vec{b}$ осуществляется по часовой стрелке, если:

(a) $\vec{a} \times \vec{b} = 0$;

(b) $\vec{a} \times \vec{b} > 0$;

(c) $\vec{a} \cdot \vec{b} < 0$;

(d) $\vec{a} \times \vec{b} < 0$.

6. Дана функция, вычисляющая ориентированную площадь треугольника с вершинами в точках $p1, p2, p3$. Найти ошибки:

```
Function SquareTrian (p1,p2:point):real;
Begin
SquareTrian:=((p2.x-p1.x)*(p3.x-p1.x)+( p2.y-p1.y)*(p3.y-p1.y))/2;
End;
```

7. Как за время $n^2 log(n)$ определить лежат ли какие-то три из данных n точек на одной прямой?

8. Опишите алгоритм сортировки, распаолагающий точки в порядке обхода по часовой стрелке, если смотреть из заданной точки P_0. Алгоритм должен использовать косое произведение и выполняться за время $nlog(n)$

9. Чтобы узнать являются ли вершины $P_1, P_2, ..., P_{n-1}$ вершинами выпуклого многоугольника, перечисленными в порядке обхода многоугольника профессор предлагает такой метод: проверить, что множество P_i, P_{i+1}, P_{i+2}, где $i+1$ и $i+2$ взяты по модулю n, не содержит одновременно левых и правых поворотов. Покажите, что этот способ не всегда дает правильный ответ, хотя и выполняется за линейное время. Как изменить его, чтобы получить правильный ответ?

10. Как за время $O(1)$ определить пересекается ли горизонтальный луч, выходящий из точки P_0 вправо с отрезком P_1P_2

11. Напишите алгоритм вычисления ориентированной площади многоугольника из N точек.

12. Опишите алгоритм вычисления объема:

    a) параллелепипеда;

    b) треугольной призмы.

2.9 Задачи

Написать программы для решения следующих задач, придумать полную систему тестов (тесты оформить в отдельных файлах, и едином документе с ответами и рисунками), а также описать формат входных и выходных данных.

1. Найти величину ориентированного угла между векторами AB и CD.

2. Составить уравнение прямой, равноудаленной от двух заданных точек (x,y),(x1,y1)

3. Угол задан вершиной и двумя точками M2(x2,y2),M1(x1,y1), которые лежат на сторонах этого угла. Найти уравнение биссектрисы этого угла.

4. Треугольник задан координатами вершин. Определить координаты и радиус вписанной окружности.

5. Треугольник задан координатами вершин. Определить координаты и радиус описанной окружности.

6. Дана окружность (координаты центра и радиус) и точка. Найти уравнение касательных к данной окружности проходящих через данную точку.

7. Дано множество точек (точки заданы своими координатами). Найти треугольник с наименьшей и наибольшей площадью, вершинами которого будут исходные точки.

8. Два круга заданы координатами центра и радиусом и дана произвольная точка. Написать программу, которая определяет, попадает ли данная точка в область пересечения данных кругов.

3 Взаимное расположение основных геометрических объектов на плоскости

3.1 Расположение точки относительно прямой или отрезка

3.1.1 Точка и прямая

Пусть прямая l задана уравнением $Ax + By + C = 0$, тогда для определения принадлежности точки $P(x_0, y_0)$ данной прямой достаточно подставить координаты точки в уравнение прямой. Равенство нулю значения полученного выражения означает, что точка принадлежит данной прямой.

Замечание: для вещественных координат или коэффициентов уравнения проверку на равенство нулю необходимо осуществлять с учетом погрешности.

Если значение выражения меньше нуля, то точка лежит в одной полуплоскости от прямой, если больше нуля - в другой.

Если прямая изначально задана двумя своими точками $P_1(x_1, y_1)$ и $P_2(x_2, y_2)$, то для решения данной задачи уравнение прямой выписывать не нужно, достаточно вычислить значение косого произведения векторов $\vec{P_1P_2} \times \vec{P_1P} = (x_0 - x_1)(y_2 - y_1) - (y_0 - y_1)(x_2 - x_1)$ и сравнить полученное выражение с нулем.

Расстояние от точки P до прямой l вычисляется по формуле

$$\rho(P, l) = \frac{|Ax_0 + By_0 + C|}{\sqrt{A^2 + B^2}}$$

Две точки P_3 и P_4 расположены по разные стороны от прямой l заданной уравнением $Ax + By + C = 0$, если при подстановке в уравнение прямой они дают разный знак, то есть

$(Ax_3 + By_3 + 0)(Ax_4 + By_4 + C) < 0.$

Если прямая задана парой точек P_1иP_2, то точки P_3 и P_4 лежат по разные стороны от неё, если

$(\vec{P_1P_2} \times \vec{P_1P_3}) \cdot (\vec{P_1P_2} \times \vec{P_1P_4}) < 0.$

3.1.2 Точка и отрезок

Для определения принадлежности точки $P(x_0, y_0)$ отрезку $[P_1P_2]$, необходимо проверить три условия:

1. Точка P принадлежит прямой P_1P_2, то есть

$$\vec{P_1P_2} \times \vec{P_1P} = (x_0 - x_1)(y_2 - y_1) - (y_0 - y_1)(x_2 - x_1) = 0.$$

2. Угол PP_1P_2 острый, то есть

$$\vec{P_1P_2} \cdot \vec{P_1P} = (x_2 - x_1)(x_0 - x_1) + (y_2 - y_1)(y_0 - y_1) > 0.$$

3. Угол PP_2P_1 острый, то есть

$$\vec{P_2P_1} \cdot \vec{P_2P} = (x_1 - x_2)(x_0 - x_2) + (y_1 - y_2)(y_0 - y_2) > 0.$$

Расстояние от точки P до отрезка $[P_1P_2]$ вычисляется по формуле

$$\rho(P, [P_1P_2]) = \frac{|Ax_0 + By_0 + C|}{\sqrt{A^2 + B^2}}$$

только в случае, если точка расположена над отрезком, то есть углы PP_1P_2 и PP_2P_1 острые, а значит

$\vec{P_1P_2} \cdot \vec{P_1P} = (x_2 - x_1)(x_0 - x_1) + (y_2 - y_1)(y_0 - y_1) \geq 0$

и

$\vec{P_2P_1} \cdot \vec{P_2P} = (x_1 - x_2)(x_0 - x_2) + (y_2 - y_1)(y_0 - y_2) \geq 0.$

При невыполнении одного из условий расстояние определяется как расстояние от точки до соотвествующего конца отрезка.

3.1.3 Точка и луч

Для определения принадлежности точки $P(x_0, y_0)$ лучу $[P_1P_2)$, достаточно проверить только первые два условия из случая взаимного расположения точки и отрезка. Для вычисления расстояния от точки до луча проверка второго условия также не требуется.

3.2 Взаимное расположение прямых, отрезков, лучей

3.2.1 Взаимное расположение двух прямых

Пусть прямые l_1 и l_2 заданы уравнениями $A_1x + B_1y + C_1 = 0$ и $A_2x + B_2y + C_2 = 0$. Коллинеарность нормальных векторов этих прямых говорит о том, что эти прямые либо параллельны, либо совпадают. Таким образом, если $\frac{A_1}{A_2} \neq \frac{B_1}{B_2}$, то прямые пересекаются. Если $\frac{A_1}{A_2} = \frac{B_1}{B_2} \neq \frac{C_1}{C_2}$, то прямые параллельны. Если $\frac{A_1}{A_2} = \frac{B_1}{B_2} = \frac{C_1}{C_2}$, то прямые совпадают.

Если прямые заданы парами точек, то находить уравнения прямых нет необходимости, достаточно проверить коллинеарность направляющих векторов.

Если прямые пересекаются, то найти точку пересечения этих прямых можно решив систему

$$\begin{cases} A_1x + B_1y + C_1 = 0 \\ A_2x + B_2y + C_2 = 0. \end{cases}$$

Таким образом точка пересечения будет иметь координаты $\left(\dfrac{B_1C_2 - B_2C_1}{A_1B_2 - A_2B_1}, \dfrac{A_2C_1 - A_1C_2}{A_1B_2 - A_2B_1}\right)$.

Точку пересечения двух прямых можно найти и другим способом (который подходит и для прямых в пространстве) Пусть прямые L_1 $(M_1(x_1, y_1, z_1)\ \overrightarrow{\alpha})$ и L_2 $(M_2(x_2, y_2, z_2)\ \overrightarrow{\beta})$ заданы параметрически

$$\begin{cases} x = x_1 + t\alpha_x \\ y = y_1 + t\alpha_y; \end{cases}$$

$$\begin{cases} x = x_2 + u\beta_x \\ y = y_2 + u\beta_y. \end{cases}$$

Точка пересечения $P(x, y)$ прямых должна удовлетворять обоим системам, то есть

$$\begin{cases} x_1 + t\alpha_x = x_2 + u\beta_x \\ y_1 + t\alpha_y = y_2 + u\beta_y. \end{cases}$$

Решая систему, находим значения параметров t и u.

$$\begin{cases} t\alpha_x - u\beta_x = x_2 - x_1 \\ t\alpha_y - u\beta_y = y_2 - y_1 \end{cases}$$

$$\begin{cases} t = \dfrac{\beta_x(y_2 - y_1) - \beta_y(x_2 - x_1)}{\alpha_y\beta_x - \alpha_x\beta_y} \\ u = \dfrac{\alpha_x(y_2 - y_1) - \alpha_y(x_2 - x_1)}{\alpha_y\beta_x - \alpha_x\beta_y} \end{cases}$$

Подставив t и u в любую из систем, задающих исходные прямые получим координаты точки пересечения этих прямых.

Описанный способ удобно использовать для определения принадлежности точки пересечения прямых отрезку или лучу, заданному теми же точками, что и прямые, указанные выше.

3.2.2 Взаимное расположение прямой и отрезка

Отрезок пересекает прямую, если его концы расположены по разные стороны от данной прямой.

3.2.3 Взаимное расположение прямой и луча

Луч $[P_1P_2)$ пересекает прямую l, если точки P_1 и P_2 расположены по разные стороны от данной прямой или они расположены по одну сторону, но точка P_2 находится по отношению к этой прямой ближе, чем точка P_1.

Также для проверки пересечения луча и прямой можно найти точку пересечения данной прямой и прямой содержащей луч, а затем посмотреть принадлежит ли эта точка лучу.

Если искать точку пересечения прямых используя их параметрическое задания, то принадлежность точки лучу не нужно проверять отдельно, достаточно посмотреть полученное значение параметра прямой, содержащей луч. Если оно находится в диапазоне от 0 до ∞, то точка пересечения принадлежит лучу.

3.2.4 Взаимное расположение двух отрезков

Отрезки $[P_1P_2]$ и $[P_3P_4]$ пересекаются тогда, когда, во-первых, пересекаются ограничивающие их прямоугольники и, во-вторых, концы каждого отрезка лежат по разные стороны от прямой, которой принадлежит другой отрезок. Первое из этих условий позволяет не рассматривать отдельно тот случай, когда отрезки лежат на одной прямой.

Обозначим x_{max1} и x_{min1} - максимальную и минимальную из x- координат первого отрезка, x_{max2} и x_{min2} - второго отрезка. Для y- координат имеем аналогично $y_{max1}, y_{min1}, y_{max2}, y_{min2}$. Тогда условия пересечения отрезков формально можно записать так:

1. $x_{max1} \geq x_{min2}$ и $x_{max2} \geq x_{min1}$ и $y_{max1} \geq y_{min2}$ и $y_{max2} \geq y_{min1}$
2. $(P_1\vec{P}_2 \times P_1\vec{P}_3) \cdot (P_1\vec{P}_2 \times P_1\vec{P}_4) < 0$ и $(P_3\vec{P}_4 \times P_3\vec{P}_1) \cdot (P_3\vec{P}_4 \times P_3\vec{P}_2) < 0$

Введем понятие расстояния между двумя непересекающимися отрезками как минимальное среди расстояний между всеми парами точек двух отрезков. Несложно понять, что оно равно расстоянию от конца одного из отрезков до другого отрезка. Поэтому для нахождения расстояния между отрезками достаточно подсчитать соответствующее расстояние для каждой из четырех концевых точек и выбрать из них минимальное.

3.2.5 Взаимное расположение двух лучей

Лучи $[P_1P_2)$ и $[P_3P_4)$ пересекаются тогда, когда, пересекаются прямые их содержащие и точка пересечения принадлежит каждому из лучей. (Проверить принадлежность точки пересечения каждому из лучей удобно, используя метод нахождения точки пересечения прямых занных параметрически)

Когда лучи лежат на одной прямой, с помощью знака скалярного произведения $P_1\vec{P}_2 \cdot P_3\vec{P}_4$ можно понять, в одну или в разные стороны они направлены. В первом случае скалярное произведение будет положительным, а во втором - отрицательным.

Если лучи сонаправлены, то определить, какой из лучей является их пересечением, можно, подсчитав значение скалярного произведения $P_1\vec{P}_2 \cdot P_1\vec{P}_3$. Когда оно больше нуля, пересечением является луч P_3P_4, в противном случае - луч P_1P_2.

В случае же противоположной направленности лучей их пересечение - либо отрезок P_1P_3, и тогда начало любого из двух лучей лежит внутри

другого луча: $\vec{P_1P_2} \cdot \vec{P_1P_3} > 0$, одна точка $P_1 = P_3$ ($\vec{P_1P_2} \cdot \vec{P_1P_3} = 0$), либо оно пусто $\vec{P_1P_2} \cdot \vec{P_1P_3} < 0$.

3.3 Окружность и прямая, луч, отрезок

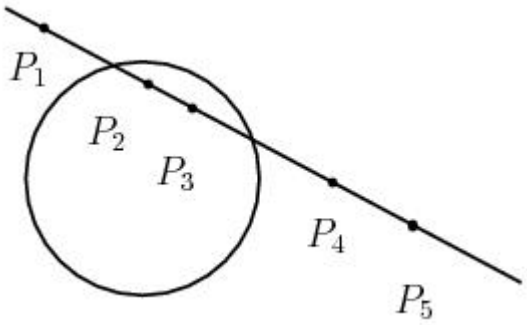
Уравнение окружности имеет вид $(x - x_0)^2 + (y - y_0)^2 = R^2$, где (x_0, y_0) - координаты центра окружности, R - радиус. Точка $P(x_1, y_1)$ лежит внутри окружности, если $(x_1 - x_0)^2 + (y_1 - y_0)^2 < R^2$ и снаружи, если $(x_1 - x_0)^2 + (y_1 - y_0)^2 > R^2$.

3.3.1 Окружность и прямая

Прямая может пересекать окружность в двух точках, касаться ее или не иметь с окружностью общих точек. Для определения конкретного случая достаточно найти расстояние от центра окружности до данной прямой. Если это расстояние меньше радиуса окружности, то прямая пересекает окружность в двух точках, если равно ему, то прямая касается окружности, а если оно больше радиуса, то общих точек нет.

3.3.2 Окружность и отрезок

Рассмотрим задачу взаимного расположения окружности и отрезка. Обозначим координаты данных концов отрезка через $(x_1, y_1), (x_2, y_2)$, координаты центра окружности через (x_0, y_0), а её радиус через R. Ход решения данной задачи может быть математическим или вычислительным. Математически мы можем задать прямую, содержащую отрезок, параметрически (используем параметр t). Подставить эти выражения в уравнение окружности и решить полученное квадратное уравнение относительно переменной t. Если корней нет, то прямая (а отрезок тем более) не пересекается с окружностью. Если корни есть, то проверяем значение корней. Пересечение есть в том и только том случае, когда хотя бы один из корней лежит в диапазоне от 0 до 1.



Другой вариант математического решения - найти расстояния от точек до центра окружности и сравнить их с радиусом окружности. Если оба расстояния меньше R, то отрезок целиком лежит внутри окружности, и они не пересекаются (например, отрезок $[P_2P_3]$ на рисунке). Если одно меньше, а другое больше R, то отрезок и окружность пересекаются (например, отрезок $[P_1P_2]$ на рисунке). В случае, когда оба расстояния больше радиуса окружности ответ может быть как положительным (отрезок $[P_1P_4]$), так и отрицательным (отрезок $[P_4P_5]$),поэтому необходимо найти расстояние от центра окружности до прямой, содержащей отрезок, сравнить его с радиусом окружности и, в случае, когда радиус больше, то есть прямая пересекает окружность, определить, лежат ли точки пересечения внутри нашего отрезка

При вычислительном подходе мы может перебирать с некоторым шагом точки отрезка, вычислять расстояния от них до центра окружности и сравнивать с радиусом. Простые случаи исследуются легко, а в сложном случае - когда отрезок пересекает окружность в двух точках, мы будем применять какой-либо из стандартных методов поиска экстремума (в нашем случае минимума) функции одной переменной (в нашем случае - расстояния от точек отрезка до центра окружности) с нужной точностью. После нахождения экстремума сделаем вывод и сформулируем ответ.

3.4 Точка и треугольник

Пусть дана точка P и треугольник ABC, необходимо определить принадлежит ли точка внутренней области треугольника. Возможны следующие подходы к решению этой задачи.

Во-первых, рассмотрим три прямые, содержащие стороны треугольника. Если для каждой из них данная точка находится по ту же сторону, что и третья вершина, то точка P расположена внутри треугольника.

Во-вторых, если сумма площадей трех треугольников, образованных

точкой P и каждой парой вершин треугольника, равна площади исходного треугольника, то заданная точка находится внутри треугольника.

Третий подход основан на том, что сумма углов при точке P, образованных лучами, идущими в вершины треугольника, равна 360^0 , только, если точка находится внутри треугольника. Во всех случаях вычисления надо проводить в пределах некоторой заданной точности.

3.5 Точка и многоугольник

Пусть $M(x, y)$ - некоторая точка плоскости. Требуется определить ее местонахождение относительно замкнутой ломаной, являющейся границей многоугольника.

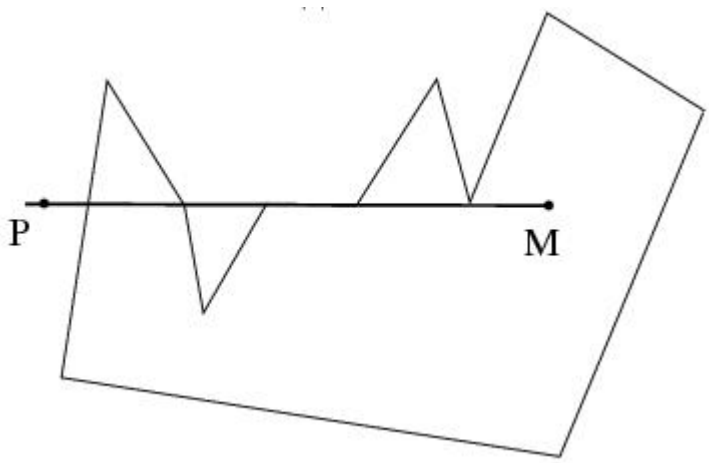
Рассмотрим сначала случай выпуклого многоугольника.

Пусть заданые своими координатами вершины многоугольника $P_0, P_1, ..., P_{n-1}$ перечислены в порядке его обхода против часовой стрелки. При таком обходе многоугольник лежит слева от границы. И, значит, если точка M лежит внутри многоугольника, то ориентированный угол между векторами $P_i\vec{M}$ и $P_i\vec{P}_{i+1}$ отрицателен. Поэтому достаточно подсчитать величину косых произведений $P_i\vec{M} \times P_i\vec{P}_{i+1}$, где $;i = 0, 1, .., n-1$ Если все полученные при этом значения отрицательны, то точка M внутренняя. Если же одно из них равно нулю, а все остальные отрицательны, то M принадлежит границе многоугольника (убедитесь, что просто равенства нулю одного из значений не достаточно). В противном же случае точка M лежит вне нашего многоугольника.

Рассмотрим теперь произвольный многоугольник.

Проведем горизонтальный луч из точки M, например, влево. Так как многоугольник ограничен, то всегда легко указать на этом луче точку $P(x, y)$, заведомо ему не принадлежащую. Далее подсчитаем количество пересечений отрезка $[PM]$ с границей многоугольника. Если оно равно нулю или четно, то точка M лежит вне многоугольника, в противном случае - внутри него. Количество пересечений отрезка $[PM]$ с границей мы подсчитаем, рассмотрев по очереди пересечение отрезка $[PM]$ с каждым из звеньев ломаной. При этом возможны следующие особые случаи.

1. Одно из звеньев ломаной целиком содержится внутри отрезка $[PM]$.
2. Звено ломаной касается отрезка $[PM]$.
3. M лежит на одном из звеньев ломаной.



В последнем случае M принадлежит границе многоугольника, и в подсчете общего числа пересечений необходимости нет.

В первом случае пересечение будем игнорировать.

Во втором - дополнительно проверим, "нижним"или "верхним"концом звено ломаной касается горизонтального отрезка $[PM]$. Если точкой касания является "нижний"конец звена, то пересечение игнорируется, а если "верхний то засчитываете". С учетом этого соглашения касание отрезка $[PM]$ границы многоугольника в одних точках игнорируется, а в других точках считается дважды. Это не изменит четности числа пересечений, а только она важна при поиске ответа на вопрос данной задачи. Если же отрезок действительно пересекает ломаную в ее вершине, то, по нашему соглашению, число пересечений как раз увеличится на единицу (пересечение с верхним ребром засчитано не будет, а с нижним - будет).

3.6 Вопросы и задания для самопроверки

1. Определите, принадлежит ли точка $M(x, y)$ отрезку AB, если:
   (a) $\overrightarrow{MA} \times \overrightarrow{MB} = 0$ и $\overrightarrow{MA} \cdot \overrightarrow{MB} < 0$;
   (b) $\overrightarrow{MA} \times \overrightarrow{MB} = 0$ и $\overrightarrow{AM} \cdot \overrightarrow{AB} > 0$ и $\overrightarrow{BM} \cdot \overrightarrow{BA} > 0$;
   (c) $\overrightarrow{MA} \times \overrightarrow{MB} \neq 0$ и $\overrightarrow{MA} \cdot \overrightarrow{MB} < 0$;
   (d) $\overrightarrow{MA} \times \overrightarrow{MB} = 0$ и $\overrightarrow{MA} \cdot \overrightarrow{MB} > 0$;
   (e) $\overrightarrow{MA} \times \overrightarrow{MB} = 0$ и $\overrightarrow{AM} \cdot \overrightarrow{AB} < 0$ и $\overrightarrow{BM} \cdot \overrightarrow{BA} > 0$.

2. Определите, принадлежит ли точка $M(x, y)$ лучу AB, если:
   (a) $\overrightarrow{MA} \times \overrightarrow{MB} = 0$ и $\overrightarrow{MA} \cdot \overrightarrow{MB} < 0$;

(b) $\overrightarrow{MA} \times \overrightarrow{MB} = 0$ и $\overrightarrow{AM} \cdot \overrightarrow{AB} > 0$ и $\overrightarrow{BM} \cdot \overrightarrow{BA} < 0$;

(c) $\overrightarrow{MA} \times \overrightarrow{MB} \neq 0$ и $\overrightarrow{MA} \cdot \overrightarrow{MB} < 0$;

(d) $\overrightarrow{MA} \times \overrightarrow{MB} = 0$ и $\overrightarrow{MA} \cdot \overrightarrow{MB} > 0$;

(e) $\overrightarrow{MA} \times \overrightarrow{MB} = 0$ и $\overrightarrow{AM} \cdot \overrightarrow{AB} < 0$ и $\overrightarrow{BM} \cdot \overrightarrow{BA} > 0$.

3. По какой формуле будет находиться расстояние d от точки $M(x, y)$ до отрезка AB, если

(a) $\overrightarrow{AM} \cdot \overrightarrow{AB} < 0$ и $\overrightarrow{BM} \cdot \overrightarrow{BA} > 0$;

(b) $\overrightarrow{AM} \cdot \overrightarrow{AB} > 0$ и $\overrightarrow{BM} \cdot \overrightarrow{BA} < 0$;

(c) $\overrightarrow{AM} \cdot \overrightarrow{AB} = 0$ и $\overrightarrow{BM} \cdot \overrightarrow{BA} > 0$;

(d) $\overrightarrow{AM} \cdot \overrightarrow{AB} > 0$ и $\overrightarrow{BM} \cdot \overrightarrow{BA} > 0$?

4. Определите взаимное расположение двух прямых, заданных своими уравнениями:

(a) $3x + 5y + 6 = 0$ и $3x + 5y + 10 = 0$;

(b) $3x + 5y + 6 = 0$ и $13x + 15y + 16 = 0$;

(c) $6x + 10y + 12 = 0$ и $3x + 5y + 6 = 0$;

(d) $3x + 5y + 6 = 0$ и $13x + 5y + 10 = 0$.

5. Найдите точку пересечения прямых

(a) $3x + 5y + 6 = 0$ и $x + 2y + 2 = 0$;

(b) $-3x + 5y + 6 = 0$ и $-x + 2y + 16 = 0$;

(c) $6x + 10y + 12 = 0$ и $3x + 5y + 6 = 0$;

(d) $3x + 5y + 6 = 0$ и $13x + 5y + 10 = 0$.

6. Определите взаимное расположение прямой $L : Ax + By + C = 0$ и отрезка P_1P_2, где $P_1(x_1, y_1), P_2(x_2, y_2)$, если:

(a) $(Ax_1 + By_1 + C) \cdot (Ax_2 + By_2 + C) = 0$;

(b) $(Ax_1 + By_1 + C) \cdot (Ax_2 + By_2 + C) > 0$;

(c) $(Ax_1 + By_1 + C) \cdot (Ax_2 + By_2 + C) < 0$.

7. Определите взаимное расположение прямой $L : Ax + By + C = 0$ и луча P_1P_2, где $P_1(x_1, y_1), P_2(x_2, y_2)$, если:

   (a) $(Ax_1 + By_1 + C)(Ax_2 + By_2 + C) < 0$;

   (b) $(Ax_1 + By_1 + C)(Ax_2 + By_2 + C) > 0$ и $\rho(P_1, L) > \rho(P_2, L)$;

   (c) $(Ax_1 + By_1 + C)(Ax_2 + By_2 + C) > 0$ и $\rho(P_1, L) < \rho(P_2, L)$;

   (d) $(Ax_1 + By_1 + C)(Ax_2 + By_2 + C) < 0$ и $\rho(P_1, L) > \rho(P_2, L)$.

8. Определите взаимное расположение двух лучей A_1B_1 и A_2B_2, где $A_1(x_1, y_1), B_1(x_2, y_2), A_2(x_3, y_3), B_2(x_4, y_4)$ если:

   (a) $\vec{A_1B_1} \times \vec{(A_2B_2)} \neq 0$;

   (b) $\vec{A_1B_1} \times \vec{(A_2B_2)} = 0$ и $\vec{A_1B_1} \cdot \vec{(A_2B_2)} > 0$ и $\vec{A_1B_1} \times \vec{(A_1A_2)} > 0$;

   (c) $\vec{A_1B_1} \times \vec{(A_2B_2)} = 0$ и $\vec{A_1B_1} \cdot \vec{(A_2B_2)} > 0$ и $\vec{A_1B_1} \times \vec{(A_1A_2)} < 0$;

   (d) $\vec{A_1B_1} \times \vec{(A_2B_2)} = 0$ и $\vec{A_1B_1} \cdot \vec{(A_2B_2)} < 0$ и $\vec{A_1B_1} \times \vec{(A_1A_2)} > 0$.

9. Определите взаимное расположение двух окружностей, с центрами в точках $O_1(x_1, y_1)$ и $O_2(x_2, y_2)$ и радиусами R_1 и R_2, при чем $R_1 > R2$

   (a) $(x_1 - x_2)^2 + (y_1 - y_2)^2 > (R_1 + R_2)^2$;

   (b) $(x_1-x_2)^2+(y_1-y_2)^2 < (R_1+R_2)^2$ и $(x_1-x_2)^2+(y_1-y_2)^2 < (R_1-R_2)^2$;

   (c) $(x_1-x_2)^2+(y_1-y_2)^2 < (R_1+R_2)^2$ и $(x_1-x_2)^2+(y_1-y_2)^2 > (R_1-R_2)^2$;

   (d) $(x_1-x_2)^2+(y_1-y_2)^2 < (R_1+R_2)^2$ и $(x_1-x_2)^2+(y_1-y_2)^2 = (R_1-R_2)^2$.

10. Что делает следующая процедура?

    a)

    ```
    Procedure PTL (v, w: point; var L:line);
    Begin
    L.A:=v.y - w.y;
    L.B:=w.x - v.x;
    L.C:=-(v.x*L.A + v.y*L.B);
    End;
    ```

    b)

    ```
    Procedure LTP (L1, L2:line; var P:point);
    ```

Var st:real;

Begin

st:=L1.A*L2.B - L2.A*L1.B;

P.X:=(L1.C*L2.B - L2.C*L1.B)/st;

P.Y:=(L1.A*L2.C - L2.A*L1.C)/st;

End;

11. Что делает следующая функция?

a)

Function CL (L1, L2:line): Boolean;

Var st:real;

Begin

st:=L1.A*L2.B - L2.A*L1.B;

CL:=abs(st)<eps;

End;

b)

Function Di(A, B:point):real;

Begin

Di:=sqrt (sqr(A.x - B.x) + sqr(A.y - B.y));

End;

c)

function rpl (L1,L2:line): boolean;

begin

if (abs(L1.a*L2.b-L2.a*L1.b)< eps) and (abs(L1.b*L2.c-L1.c*L2.b)< eps) then rpl:=true else rpl:=false;

end;

d) function RTL (L:line, p:point):real;

begin

rtl:= (abs(L.A*p.x+L.B*p.y+L.C))/(sqrt(L.A*L.A+L.B*L.B));

end;

12. Даны точки A, B, C, D. Выберите все необходимые условия для того, чтобы четырехугольник $ABCD$ являлся

    - параллелограммом

    - ромбом

    - прямоугольником

    - квадратом

    (a) $\overrightarrow{AB} = \overrightarrow{CD}$;
    (b) $\overrightarrow{AB} \cdot \overrightarrow{CD} = 0$;
    (c) $\overrightarrow{AB} \times \overrightarrow{CD} = 0$;
    (d) $\overrightarrow{AC} \cdot \overrightarrow{BD} = 0$;
    (e) $\overrightarrow{AB} \cdot \overrightarrow{CD} = 0$;
    (f) $\overrightarrow{AB} \cdot \overrightarrow{AD} = 0$.

3.7 Задачи

В задачах, где описание формата ввода-вывода отсутсвует необходимо написать процедуру или функцию на Паскале или метод на C#.

1. Определить взаимное расположение на плоскости (принадлежит, не принадлежит, если не принадлежит, то найти расстояние от точки до объекта)

    (a) точки и луча.

    Формат входных данных:

    в первой строке два числа - координаты точки;

    во второй и третьей строках по два числа - координаты начала луча и точки на луче.

    Формат выходных данных:

    одно число с точностью до 5 знаков после запятой (если принадлежит - ответ 0).

    (b) точки и отрезка.

    Формат входных данных:

    в первой строке два числа - координаты точки;

во второй и третьей строках по два числа - координаты концов отрезка.

Формат выходных данных:

одно число с точностью до 5 знаков после запятой (если принадлежит - ответ 0).

2. Определить взаимное расположение на плоскости (пересекаются, не пересекаются, если не пересекаются, то найти расстояние между геометрическими объектами):

   (a) Двух отрезков

   Формат входных данных:

   в строках по два числа - координаты концов отрезка.

   Формат выходных данных:

   в первой строке ответ в формате true/false;

   во второй строке одно число с точностью до 5 знаков после запятой (0 если отрезки пересекаются).

   (b) Луча и прямой

   Формат входных данных:

   в первых двух строках по два числа - координаты начала луча и точки на луче;

   в третьей строке - коэффициенты уравнения прямой.

   Формат выходных данных:

   впервой строке ответ в формате true/false;

   во второй строке одно число с точностью до 5 знаков после запятой (0 если луч принадлежит прямой).

   (c) Двух лучей

   Формат выходных данных:

   в первой и второй строках по два числа - в первой и третьей координаты начала луча и во второй и четвертой координаты точки на луче.

   Формат выходных данных:

   впервой строке ответ в формате true/false;

   во второй строке одно число с точностью до 5 знаков после запятой (0 если отрезки пересекаются).

(d) Отрезка и прямой

Формат входных данных:

в первых двух строках по два числа - координаты концов отрезка;

в третьей строке - коэффициенты уравнения прямой.

Формат выходных данных:

в первой строке ответ в формате true/false;

во второй строке одно число с точностью до 5 знаков после запятой (0 если отрезок принадлежит прямой).

(e) Окружности и прямой

Формат входных данных:

в первой строке три числа - координаты центра и радиус окружности;

в третьей строке - коэффициенты уравнения прямой.

Формат выходных данных:

впервой строке ответ в формате true/false;

во второй строке одно число с точностью до 5 знаков после запятой (0 если отрезок принадлежит прямой).

(f) Луча и отрезка

Формат входных данных:

В первой и второй строке по два числа - координаты концов отрезка

В третьей и четвертой строках по два числа - координаты начала луча и точки на луче

Формат выходных данных:

Впервой строке ответ в формате true/false

Во второй строке одно число с точностью до 5 знаков после запятой (0 если отрезки пересекаются).

3. Многоугольник на плоскости задан координатами вершин. Найти его площадь.

Формат входных данных:

в первой строке целое число N- количество вершин многоугольника;

в следующих N строках по два числа - координаты вершин многоугольника в порядке обхода против часовой стрелки.

Формат выходных данных:

одно число с точностью до 5 знаков после запятой.

4. Определить взаимное расположение:

   (a) Точки и треугольника

   Формат входных данных:

   в первых трех строках по два числа - координаты вершин треугольника;

   в последней строке два числа - координаты заданной точки.

   Формат выходных данных:

   ответ в формате true/false.

   (b) Точки и выпуклого многоугольника

   Формат входных данных:

   в первой строке целое число N - количество вершин многоугольника;

   в следующих N строках по два числа - координаты вершин многоугольника в порядке обхода против часовой стрелки;

   в последней строке два числа - координаты заданной точки.

   Формат выходных данных:

   ответ в формате true/false.

   (c) Точки и прямоугольника

   Формат входных данных:

   в первых четырех строках по два числа - координаты вершин прямоугольника;

   в последней строке два числа - координаты заданной точки.

   Формат выходных данных:

   ответ в формате true/false.

   (d) Определить, является ли многоугольник выпуклым

   Формат входных данных:

   в первой стоке число N количество вершин многоугольника;

   в следующих N строках по два числа - координаты вершин многоугольника в порядке обхода против часовой стрелки.

   Формат выходных данных:

ответ в формате true/false.

многоугольника в порядке обхода против часовой стрелки;

4. Взаимное расположение двух прямых

   (a) Даны точки M_1, M_2, M_3. Определить взаимное расположение точки M_1 и прямой (M_3, M_4).

   (b) Даны точки M_1, M_2, M_3, M_4. Определить взаимное расположение прямых (M_1M_2) и (M_3M_4).

5. Дано множество точек. Определить диаметр множества. (Диаметром множества точек называется наибольшее расстояние между двумя точками этого множества).

   Формат входных данных:

   в первой строке одно число - количество точек;

   далее в строках по два числа - координаты точек.

   Формат выходных данных:

   одно число с точностью до 5 знаков после запятой (если принадлежит - ответ 0).

6. Дано множество точек на плоскости. Найти среди них две ближайшие (с наименьшим расстоянием).

   Формат входных данных:

   в первой строке одно число - количество точек;

   далее в строках по два числа - координаты точек.

   Формат выходных данных:

   одно число с точностью до 5 знаков после запятой (если принадлежит - ответ 0).

7. На плоскости задано множество точек A и точка d вне его. Подсчитать количество (неупорядоченных) различных троек точек a, b, c из A таких, что четырехугольник $abcd$ является параллелограммом.

   Формат входных данных:

   в первой строке два числа - координаты точки d;

во второй строке одно целое число - количество точек множества A;

далее в строках по два числа - координаты точек.

Формат выходных данных:

одно целое число.

8. На плоскости задано множество точек A и точка d вне его. Подсчитать количество (неупорядоченных) различных троек точек a, b, c из A таких, что четырехугольник $abcd$ является ромбом.

   Формат входных данных:

   в первой строке два числа - координаты точки d;

   во второй строке одно целое число - количество точек множества A;

   далее в строках по два числа - координаты точек.

   Формат выходных данных:

   одно целое число.

9. На плоскости задано множество точек A и точка d вне его. Подсчитать количество (неупорядоченных) различных троек точек a, b, c из A таких, что четырехугольник $abcd$ прямоугольником.

   Формат входных данных:

   в первой строке два числа - координаты точки d;

   во второй строке одно целое число - количество точек множества A.

   Далее в строках по два числа - координаты точек.

   Формат выходных данных:

   одно целое число.

10. На плоскости задано множество точек A и точка d вне его. Подсчитать количество (неупорядоченных) различных троек точек a, b, c из A таких, что четырехугольник $abcd$ квадратом.

    Формат входных данных:

    в первой строке два числа - координаты точки d;

    во второй строке одно целое число - количество точек множества A;

    далее в строках по два числа - координаты точек.

Формат выходных данных:

одно целое число.

11. Клетки в круге

На клетчатой бумаге нарисовали окружность целого радиуса R с центром на пересечении линий. Найдите количество клеток, целиком лежащих внутри этой окружности.

Формат входных данных:

одно число - радиус R.

Формат выходных данных:

одно целое число.

12. Торт

На квадратном торте стоит N свечей. Можно ли одним прямолинейным разрезом разделить его на две равные по площади части, одна из которых не содержала бы ни одной свечи? Свечи считать точками, каждая из которых задана парой целочисленных координат. Начало координат находится в центре торта, оси координат параллельны краям торта; разрез не может проходить через свечу.

Формат входных данных:

в первой строке одно целое число N- количество свечей;

далее в строках по два числа - координаты свечей.

Формат выходных данных:

ответ в формате true/false.

13. Треугольник и точки

Даны координаты вершин треугольника на плоскости. Опишите алгоритм, определяющий количество целочисленных точек, попавших внутрь треугольника

Формат входных данных:

в трех строках по два числа - коодинаты вершин треугольника.

Формат выходных данных:

одно целое число.

14. "Центр масс"

    Губернатор одной из областей заключил с фирмой "НИет"контракт на подключение всех городов области к компьютерной сети. Сеть создается следующим образом: в области устанавливается станция спутниковой связи, и затем от каждого города прокладывается кабель до станции. Технология, используемая компанией, во избежание накопления ошибок требует при увеличении расстояния увеличения толщины кабеля. Стоимость кабеля, соединяющего город со станцией, при используемой компанией технологии будет равна kl^2, где k - некий коэффициент, а l - расстояние от города до станции. Требуется определить положение станции, при котором затраты компании на установку сети минимальны.

    Формат входных данных:

    в первой строке одно число - количество городов;

    далее в строках по два числа - координаты точек.

    Формат выходных данных:

    в строке два числа - координаты станции.

15. Задано множество прямых на плоскости коэффициентами своих уравнений. Подсчитать количество точек пересечения этих прямых.

    Формат входных данных:

    в первой строке целое число - количество прямых;

    в следующих строках по три числа - коэффициенты прямых.

    Формат выходных данных:

    одно целое число.

16. Дано множество отрезков. Проверить, есть ли среди них хотя бы два пересекающихся.

    Формат входных данных:

    в первой строке одно число - количество отрезков;

    далее в строках по четыре числа - координаты концов отрезка (абсцисса и ордината первого конца, абсцисса и ордината второго конца).

    Формат выходных данных:

    ответ в формате true/false.

17. Определить взаимное расположение точки и произвольного многоугольника на плоскости.

    Формат входных данных:

    в первой строке целое число N- количество вершин многоугольника;

    в следующих N строках по два числа - координаты вершин многоугольника в порядке обхода против часовой стрелки;

    в последней строке два числа - координаты заданной точки.

    Формат выходных данных:

    ответ в формате true/false.

18. Множество из $2k$ точек на плоскости задано координатами точек. Построить K непересекающихся отрезков с вершинами в данных точках.

    Формат входных данных:

    первое число - количество точек (обязательно четное);

    далее в строках по два числа - координаты точек.

    Формат выходных данных:

    в строках по 4 числа - координаты концов отрезка (абсцисса и ордината первого конца, абсцисса и ордината второго конца).

19. Биллиард

    Дано биллиардное поле размером $a \times b$ с единственной лузой, расположенной в левом нижнем углу, совпадающим с началом координат. (x, y) - координаты биллиардного шара на поле. Опишите алгоритм нахождения направлений удара по шару, обеспечивающих попадание в лузу при одном отражении от борта

    Формат входных данных:

    в одной строке два числа - координаты биллиардного шара.

    Формат выходных данных:

    в одной строке два числа - координаты точки на границе биллиардного поля.

20. Вершина прямоугольника

Даны координатs трёх вершин прямоугольника. Найдите координаты четвертой его вершины.

Формат входных данных:

в трех строках по два числа - координаты вершин прямоугольника.

Формат выходных данных:

в одной строке два числа - координаты четвертой вершины.

21. Даны три круга с вершинами в точках $A(x1, y1), B(x2, y2), C(x3, y3)$ радиусов $R1, R2, R3$ соответственно. Опишите алгоритм, определяющий, накрывают ли данные круги треугольник ABC.

    Формат входных данных:

    в трех строках по два числа - координаты точек A, B, C.

    Формат выходных данных:

    ответ в формате true/false.

22. На плоскости задано множество точек . Построить круг наименьшего радиуса, содержащий все точки данного множества.

    Формат входных данных:

    первое число - количество точек;

    далее в строках по два числа - координаты точек.

    Формат выходных данных:

    три числа - координаты центра и радиус окружности.

23. Дано множество точек, которое задает выпуклый многоугольник. Определить вписанную окружность максимального радиуса.

    Формат входных данных:

    первое число - количество вершин многоугольника;

    далее в строках по два числа - координаты вершин в порядке обхода против часовой стрелки.

    Формат выходных данных:

    три числа - координаты центра и радиус окружности.

24. Дано множество точек. Найти минимальный угол такой, чтобы 1 точка была вершиной, стороны проходили через две другие точки, а остальные лежали внутри угла.

    Формат входных данных:

    первое число - количество точек;

    далее в строках по два числа - координаты точек.

    Формат выходных данных:

    в трех строках по три числа - координаты вершины угла и координаты точек на лучах угла.

25. Построить замкнутую несамопересекающуюся ломаную с вершинами в заданных точках.

    Формат входных данных:

    первое число - количество точек;

    далее в строках по два числа - координаты точек.

    Формат выходных данных:

    в строках по два числа - координаты вершин ломаной в порядке обхода.

26. Ломаная (может быть и незамкнутая) задана координатами своих вершин в порядке их обхода. Определить, на сколько частей она разбивает плоскость.

    Формат входных данных:

    первое число - количество вершин ломаной;

    далее в строках по два числа - координаты вершин в порядке обхода.

    Формат выходных данных:

    одно целое число.

27. Наибольшая "пустая"окружность

    В черте города, граница которого известна, а дома заданы своими координатами, требуется выбрать место для строительства химического завода так, чтобы расстояние от него до ближайшего дома было максимальным.

Формат входных данных (ввод из файла):

в первой строке одно число (N) - количество вершин многоугольника, определяющего черту города;

в следующих N строках по два числа координаты вершин многоугольника;

в $N + 2$ строке одно число (M) - количество домов;

в следующих M строках по два числа координаты домов.

Формат выходных данных:

координаты расположения химического завода.

Вывод можно осуществлять на экран.

28. Кратчайшая сеть дорог

Заданы N населенных пунктов (N точек на плоскости). Необходимо проложить между ними дороги, чтобы по этим дорогам возможно было проехать из любого пункта в любой другой, а суммарная длина дорог была минимальной. Решить задачу для $N = 3$, и для $N = 4$, причем точки находятся в вершинах прямоугольника.

Формат входных данных (ввод из файла):

в первой строке одно число (N) - количество населенных пунктов;

в следующих N строках по два числа координаты вершин многоугольника.

Формат выходных данных:

координаты точек Штейнера.

29. Разделение пограничных городов

На границе двух государств, только что обретших свою независимость друг от друга, расположены два города, представляющих собой на карте два выпуклых непересекающихся многоугольника. По соглашению между президентами государств эти города отходят к разным государствам, и между ними должна быть проведена прямолинейная граница. Напишите программу, которая рассчитывает, как следует провести искомую границу.

Формат входных данных:

в первой строке находится число N - количество вершин многоугольника, соответствующего первому городу;

в последующих N строках находятся пары чисел $x_i y_i$, разделенных пробелами, - координаты вершин первого многоугольника, перечисленные вдоль обхода контура;

в следующей строке: K - количество вершин многоугольника, соответствующего второму городу;

в последующих K строках находятся пары чисел $x_i y_i$, разделенных пробелами, - координаты вершин второго многоугольника, перечисленные вдоль обхода контура.

Формат выходных данных:

в первой строке должна быть выведена пара чисел, разделенных пробелом, - координаты произвольной точки, лежащей на прямой - искомой границы между городами;

во второй строке: пара чисел, разделенных пробелом, - координаты вектора, направленного вдоль искомой границы.

4 Взаимное расположение основных геометрических объектов в пространстве

4.1 Расположение точки и других геометрических объектов

Точка $M(x, y, z)$ принадлежит прямой M_1M_2, если вектора $\overrightarrow{MM_1}$ и $\overrightarrow{MM_2}$ коллинеарны, то есть их векторное произведение равно 0.

Задача о взаимном рассположении точки и отрезка, точки и луча в пространстве решается аналогично плоскому случаю.

Точка $M(x_0, y_0, z_0)$ принадлежит плоскости ω: $Ax+By+Cz+D = 0$, если $Ax_0 + By_0 + Cz_0 + D = 0$.

Точка M принадлежит плоскости $M_1M_2M_3$, если вектора $\overrightarrow{MM_1}$, $\overrightarrow{MM_2}$ и $\overrightarrow{MM_3}$ колмпланарны, то есть их смешанное произведение равно 0.

Задача о взаимном расположении точки и треугольника (многоугольника) сводится к плоскому случаю следющим образом:

- выбирается координатная плоскость не перпендикулярная плоскости треугольника (многоугольника), пусть это будет плоскость XOY;

- точка и треугольник (многоугольник) проецируются на эту плоскость, то есть координата z всех точек обнуляется;

- решается задача о взаимном расположении точки и треугольника (многоугольника) на плоскости.

4.2 Расположение прямых в пространтсве

Пусть прямые L_1 и L_2 заданы точкой и направляющим вектором

$L_1 : M_1(x_1, y_1, z_1), \overrightarrow{\alpha}(\alpha_x, \alpha_y, \alpha_z)$;

$L_2 : M_2(x_2, y_2, z_2), \overrightarrow{\beta}(\beta_x, \beta_y, \beta_z)$.

Если $(\overrightarrow{\alpha}, \overrightarrow{\beta}, \overrightarrow{M_1M_2}) \neq 0$, то прямые L_1 и L_2 скрещиваются, то есть не лежат в одной плоскости.

Если $(\overrightarrow{\alpha}, \overrightarrow{\beta}, \overrightarrow{M_1M_2}) = 0$ и $\overrightarrow{\alpha} \times \overrightarrow{\beta} \neq 0$, то прямые L_1 и L_2 пересекаются.

Если $(\overrightarrow{\alpha}, \overrightarrow{\beta}, \overrightarrow{M_1M_2}) = 0$ и $\overrightarrow{\alpha} \times \overrightarrow{\beta} = 0$ и $\overrightarrow{\alpha} \times \overrightarrow{M_1M_2} = 0$, то прямые L_1 и L_2 совпадают.

Если $(\overrightarrow{\alpha}, \overrightarrow{\beta}, \overrightarrow{M_1M_2}) = 0$ и $\overrightarrow{\alpha} \times \overrightarrow{\beta} = 0$ и $\overrightarrow{\alpha} \times \overrightarrow{M_1M_2} \neq 0$, то прямые L_1 и L_2 параллельны.

Точка пересечения двух прямых.

Рассмотрим параметрические уравнения этих прямых

$$\begin{cases} x = x_1 + t\alpha_x \\ y = y_1 + t\alpha_y \\ z = z_1 + t\alpha_z; \end{cases}$$

$$\begin{cases} x = x_2 + u\beta_x \\ y = y_2 + u\beta_y \\ z = z_2 + u\beta_z. \end{cases}$$

Точка пересечения $P(x, y, z)$ прямых должна удовлетворять обеим системам, то есть

$$\begin{cases} x_1 + t\alpha_x = x_2 + u\beta_x \\ y_1 + t\alpha_y = y_2 + u\beta_y \\ z_1 + t\alpha_z = z_2 + u\beta_z. \end{cases}$$

Заметим, что достаточно рассмотреть решение системы из двух уравнений. Так, если $\alpha_y \cdot \beta_x - \alpha_x \cdot \beta_y \neq 0$,то можно найти решение системы из двух первых уравнений.

Подставляя найденые значения u и t в любую из систем задающих прямые, можно получить координаты точки пересечения прямых.

4.3 Расположение прямой и плоскости

Пусть прямая L задана точкой $M(x_0, y_0, z_0)$ и направляющим вектором $\overrightarrow{\alpha}(\alpha_x, \alpha_y, \alpha_z)$, плоскость φ имеет уравнение $Ax + By + Cz + D = 0$.

Прямая пересекает плоскость, если $A\alpha_x + B\alpha_y + C\alpha_z \neq 0$.

Прямая параллельна плоскости, если $A\alpha_x + B\alpha_y + C\alpha_z = 0$ и $Ax_0 + By_0 + Cz_0 + D \neq 0$.

Прямая лежит в плоскости, если $A\alpha_x + B\alpha_y + C\alpha_z = 0$ и $Ax_0 + By_0 + Cz_0 + D = 0$.

Чтобы найти точку пересечения прямой и плоскости можно использовать параметрические уравнения прямой

$$\begin{cases} x = x_0 + t\alpha_x \\ y = y_0 + t\alpha_y \\ z = z_0 + t\alpha_z. \end{cases}$$

Точка пересечения $P(x, y, z)$ должна удовлетворять данной системе и уравнению плоскости, то есть

$A(x_0 + t\alpha_x) + B(y_0 + t\alpha_y) + C(z_0 + t\alpha_z) + D = 0$

В результате решения данного уравнения находится значение параметра t. Подстановка найденного значения t в параметрические уравнения прямой дает координаты точки пересечения.

4.4 Расположение двух плоскостей

Пусть плоскость ω_1 задана уравнением $A_1x + B_1y + C_1z + D_1 = 0$, а плоскость ω_2 уравнением $A_2x + B_2y + C_2z + D_2 = 0$.

Плоскости ω_1 и ω_2 пересекаются, когда $A_1 * B_2 - A_2 * B_1 \neq 0$ или $B_1 * C_2 - B_2 * C_1 \neq 0$;

Плоскости ω_1 и ω_2 параллельны, когда $A_1 * B_2 - A_2 * B_1 = 0$ и $B_1 * C_2 - B_2 * C_1 = 0$ и $C_1 * D_2 - C_2 * D_1 \neq 0$;

Плоскости ω_1 и ω_2 совпадают, когда их коэффициенты пропорциональны, то есть $A_1 * B_2 - A_2 * B_1 = 0$ и $B_1 * C_2 - B_2 * C_1 = 0$ и $C_1 * D_2 - C_2 * D_1 = 0$;

4.5 Расположение трех плоскостей

Возможны восемь следующих случаев взаимного расположения трех плоскостей

1) плоскости имеют единственную общую точку;
2) плоскости попарно пересекаются, но не имеют общей точки;
3) плоскости попарно различны, но проходят через одну прямую;
4) две из плоскостей параллельны, а третья их пересекает;
5) три плоскости попарно параллельны;
6) две плоскости совпадают, а третья их пересекает;
7) две плоскости совпадают, а третья им параллельна;
8) все три плоскости совладают.

Для определения каждого конкретного случая необходимо проанализировать взаимное расположение каждой пары плоскостей.

4.6 Расположение прямой и треугольника, многоугольника

Для определения взаимного расположения прямой и треугольника сначала необходимо проанализировать взаимное расположение пямой и плоскости треугольника . Если прямая параллельна плоскости треугольника, то прямая и треугольник не пересекаются. Если прямая и плоскость пересекаются, то необходимо найти точку пересечения плоскости треугольника с прямой и далее проверить лежит ли точка пересечения внутри треугольника, задача о взаимном расположении треугольника и точки была рассмотрена выше. Если прямая лежит в плоскости теугольника , то задачу можно свести к плоскому случаю так же, как это было описанов пункте 4.1.

Задачи о взаимном расположении прямой и многоугольника, отрезка и треугольника, отрезка и многоугольника решаются аналогично.

4.7 Вопросы и задания для самопроверки

1. Определите взаимное расположение плоскостей α и β, уравнения которых имеют вид:

   (a) α: $5x + 3y + z + 6 = 0$
   β: $5x + 3y + z + 16 = 0$;

   (b) α: $5x + 3y + z + 6 = 0$
   β : $15x + 13y + 10z + 16 = 0$;

(c) α: $5x + 3y + z + 6 = 0$
β: $10x + 6y + 2z + 12 = 0$;

(d) α :$5x + 3y + z + 6 = 0$
β :$-3x - 5y - z + 6 = 0$.

2. Определите взаимное расположение точек A и B относительно плоскости α , уравнения которой имеет вид $5x + 3y + z + 6 = 0$, если точки A и B имеют следующие координаты:

(a) $A(1, 2, 3)$, $B(-1, -2, -3)$;

(b) $A(1, 2, 3)$, $B(-1, -2, 3)$;

(c) $A(1, 2, 3)$, $B(-1, -2, -14)$.

3. Определите взаимное расположение прямых M_1M_2 и M_3M_4, если

(a) $M_1(1, 1, 1)$, $M_2(2, 2, 2)$, $M_3(3, 4, 5)$, $M_2(4, 5, 6)$;

(b) $M_1(1, 1, 1)$, $M_2(2, 2, 2)$, $M_3(0, 0, 0)$, $M_2(4, 6, 8)$;

(c) $M_1(1, 1, 1)$, $M_2(2, 2, 2)$, $M_3(3, 4, 5)$, $M_2(4, 6, 8)$;

(d) $M_1(1, 1, 1)$, $M_2(2, 2, 2)$, $M_3(5, 5, 5)$, $M_2(10, 10, 10)$.

4. Определите взаимное расположение прямых M_1M_2 и M_3M_4

(a) $(\overrightarrow{M_1M_2}, \overrightarrow{M_3M_4}, \overrightarrow{M_1M_2}) \neq 0$;

(b) $(\overrightarrow{M_1M_2}, \overrightarrow{M_3M_4}, \overrightarrow{M_1M_2}) = 0$ и $\overrightarrow{M_1M_2} \times \overrightarrow{M_3M_4} = 0$ и $\overrightarrow{M_1M_2} \times \overrightarrow{M_1M_2} = 0$;

(c) $(\overrightarrow{M_1M_2}, \overrightarrow{M_3M_4}, \overrightarrow{M_1M_2}) = 0$ и $\overrightarrow{M_1M_2} \times \overrightarrow{M_3M_4} = 0$ и $\overrightarrow{M_1M_2} \times \overrightarrow{M_1M_2} \neq 0$;

(d) $(\overrightarrow{M_1M_2}, \overrightarrow{M_3M_4}, \overrightarrow{M_1M_2}) = 0$ и $\overrightarrow{M_1M_2} \times \overrightarrow{M_3M_4} \neq 0$.

5. Найдите точку пересечения прямых M_1M_2 и M_3M_4, если $M_1(1, 1, 1)$, $M_2(2, 2, 2)$, $M_3(0, 0, 0)$, $M_2(4, 6, 8)$.

6. Определите взаимное расположение прямой M_1M_2 и плоскости
$\alpha : 5x + y + 2z + 1 = 0$, если

(a) $M_1(0, -1, 0)$, $M_2(1, 0, -2)$;

(b) $M_1(0, -1, 0)$, $M_2(2, 5, 7)$;

(c) $M_1(1, 2, 3)$, $M_2(2, 5, 7)$;

(d) $M_1(1,2,3)$, $M_2(2,1,1)$.

7. Определите взаимное расположение прямой M_1M_2 и плоскости $\alpha : Ax + By + Cz + D = 0$, если

   (a) $A(x_2 - x_1) + B(y_2 - y_1) + C(z_2 - z_1) = 0$ и $Ax_1 + By_1 + Cz_1 + D \neq 0$;
   (b) $A(x_2 - x_1) + B(y_2 - y_1) + C(z_2 - z_1) = 0$ и $Ax_1 + By_1 + Cz_1 + D = 0$;
   (c) $A(x_2 - x_1) + B(y_2 - y_1) + C(z_2 - z_1) \neq 0$.

8. Найдите точку пересечения прямой M_1M_2, где $M_1(0,-1,0)$, $M_2(2,5,7)$ и плоскости $5x + y + 2z + 1 = 0$.

9. Определите взаимное расположение трех плоскостей

   (a) $x + 2y - 3z + 4 = 0, -x - 2y + 3z - 4 = 0, 2x + 4y - 6z + 8 = 0$;
   (b) $x + 2y - 3z + 4 = 0, -x - 2y + 3z - 4 = 0, 2x - 4y - 6z - 8 = 0$;
   (c) $x + y - 5 = 0, y - 5 = 0, x = 0$;
   (d) $x + 2y - 3z + 4 = 0, x + 3y - 5z6 = 0, 2x + 4y - 6z + 8 = 0$.

10. Определите взаимное расположение прямой L_1L_2 и треугольника ABC, если

    (a) $A(0,-1,0), B(1,0,-2), C(0,-5,2), L_1(1,2,3), L_2(2,1,1)$;
    (b) $A(0,-1,0), B(1,0,-2), C(0,-5,2), L_1(0,1,0), L_2(1,1,0)$;
    (c) $A(0,-1,0), B(1,0,-2), C(0,-5,2), L_1(0,1,0), L_2(1,-1,2.5)$.

11. Опишите алгоритм определения взаимного расположения прямой и многогранника.

12. Опишите алгоритм определения взаимного расположения плоскости и многогранника.

13. Что делает следующая функция?

    ```
    Function CP (P1, P2:plane): Boolean;
    Var st:real; st1:real;
    Begin
    st:=P1.A*P2.B - P2.A*P1.B;
    ```

st1:=P1.B*P2.C - P2.D*L1.C;

CP:=(abs(st)>eps)or (abs(st1)>eps);

End;

14. Что делает следующая функция?

    function rpl (P:point;Pl:plane): boolean;

    begin

    rp1:= (abs(Pl.a*P.x+Pl.b*P.y+Pl.c*P.z+Pl.d)< eps)

    end;

4.8 Задачи

В задачах, где описание формата ввода-вывода отсутсвует необходимо написать процедуру или функцию на Паскале или метод на C#.

1. Вычисление объемов тел

   (a) Найти объем пирамиды.

   Формат входных данных:

   первой строке три числа - координаты вершины пирамиды, противоположной основанию;

   во второй строке 1 число - количество вершин в основании;

   в следующих строках по три числа - координаты вершин основания.

   (b) Найти объем призмы

   Формат входных данных:

   в первой строке три числа - координаты вектора, задающего боковую сторону призмы;

   во второй строке 1 число - количество вершин в основании;

   в следующих строках по три числа - координаты вершин основания.

2. Определить взаимное расположение:

   (a) Точки и тетраэдра

   Формат входных данных:

в первых четырех строках по три числа - координаты вершин тетреаэдра;

в последней строке три числа - координаты заданной точки.

Формат выходных данных:

ответ в формате true/false.

(b) Точки и куба

Формат входных данных:

в восьми строках по три числа - координаты вершин прямоугольника;

в последней строке три числа - координаты заданной точки.

Формат выходных данных:

ответ в формате true/false.

(c) Точки и прямоугольного параллелепипеда

Формат входных данных:

в восьми строках по три числа - координаты вершин прямоугольника;

в последней строке три числа - координаты заданной точки.

Формат выходных данных:

ответ в формате true/false.

3. Взаимное расположение двух прямых, прямой и плоскости.

(a) Даны точки M_1, M_2, M_3, M_4, M_5. Определить взаимное расположение прямой (M_1, M_2) и плоскости $(M_3M_4M_5)$.

(b) Даны точки M_1, M_2, M_3, M_4, M_5. Найти точку пересечения прямой (M_1, M_2) и плоскости $(M_3M_4M_5)$. Известно, что прямая и плоскость пересекаются.

(c) Даны точки M_1, M_2, M_3, M_4, M_5. Определить взаимное расположение отрезка (M_1, M_2) и плоскости $(M_3M_4M_5)$.

(d) Даны точки M_1, M_2, M_3, M_4, M_5. Определить взаимное расположение луча (M_1, M_2) и плоскости $(M_3M_4M_5)$.

4. Пересечение прямой и тетраэдра, заданного координатами своих вершин.

Формат входных данных:

в первых двух строках по три числа - координаты точек прямой;

в следующих четырех строках три числа - координаты вершин тетраэдра.

Формат выходных данных:

ответ в формате true/false.

5. Определить пересекаются ли прямая и прямоугольный параллелепипед.

   Формат входных данных:

   в первых двух строках по три числа - координаты точек прямой;

   в следующих четырех строках три числа - координаты вершин основания параллелепипеда;

   в последней строке три числа - координаты вектора, задающего ребро параллелепипеда.

   Формат выходных данных:

   ответ в формате true/false.

6. Определить пересекаются ли сфера и плоскость.

   Формат входных данных:

   в первых двух строках четыре числа - координаты центра сферы и радиус;

   в следующей строке четыре числа - коэффициенты плоскости.

   Формат выходных данных:

   ответ в формате true/false.

7. Даны три плоскости своими коэффициентами, определить их взаимное расположение.

8. Определить пересекаются ли прямая и треугольник в пространстве.

   Формат входных данных:

   в первых тех строках три числа - координаты вершин треугольника;

   в следующих двух строках три числа - координаты точек прямой.

   Формат выходных данных:

   ответ в формате true/false.

9. Определить пересекаются ли отрезок и треугольник в пространстве.

   Формат входных данных:

   в первых тех строках три числа - координаты вершин треугольника;

   в следующих двух строках три числа - координаты концов отрезка.

   Формат выходных данных:

   ответ в формате true/false.

10. Пересечение плоскости и многогранника.

    Формат входных данных:

    В первой строке файла содержится количество вершин многогранника N.

    В следующих N строках перечисляются координаты вершин многогранника через пробел.

    В $N + 2$ строке указано количество граней .

    В следующих строках указаны сначало число вершин в грани, затем списки вершин принадлежащих каждой грани.

Список литературы

[1] Андреева Е.В. Геометрические задачи на олимпиадах по информатике. Информатика №14/2002

[2] Андреева Е.В., Егоров Ю.Е. Вычислительная геометрия на плоскости. Информатика №39-42/2002

[3] Волченков С.Г. Ярославские олимпиады по информатике. Сборник задач с решениями/С.Г. Волченков, П.А. Корнилов, Ю.А. Белов, и др. - М.БИНОМ. Лаборатория знаний, 2010. - 405 с.

[4] Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы. Построение и анализ. М.:МЦНМО, 2000

[5] Ласло М. Вычислительная геометрия и компьютерная графика на C++: Пер. с англ. - М.: БИНОМ, 1997. - 304 с.: ил.

[6] Никулин Е.А. компьютерная геометрия и алгоритмы машинной графики. - СПб.: БХВ-Петербург ,2003. - 560с.

[7] Ноден П., Китте К. Алгебраическая алгоритмика (с упражнениями и решениями)

[8] Окулов С.М. 100 задач по информатике. Киров: Изд-во ВГПУ, 2000

[9] Препарата Ф., Шеймос М. Вычислительная геометрия. Введение. М.:Мир, 1989- 478 с.

[10] Фокс А., Пратт М. Вычислительная геометрия. Применение в проектировании и на производстве. М.:Мир, - 304 с.

[11] Шикин Е.В., Боресков А.В., Зайцев А.А. Начала компьютерной графики. М.: Диалог-МИФИ, 1993.

Учебное издание

Заводчикова Надежда Ивановна,
Федотова Наталия Петровна

Вычислительная геометрия для студентов направления "Педагогическое образование"(профиль "Информатика и информационные технологии в образовании")

Часть 1

Учебное пособие

Редактор О. П. Новикова

Компьютерная верстка - Н.И. Заводчикова

Подписано в печать 24.11.2013. Формат 60?92/16 Объем 8 п.л.; 4,3 уч.-изд. л. Тираж 50 экз. Заказ № 342.

Издательство Ярославского государственного педагогического университета им. К. Д. Ушинского 150000, Ярославль, Республиканская ул., 108
Типография ЯГПУ 150000, Ярославль, Которосльная наб., 44 Тел.: (4852) 32-98-69, 72-64-05