|  |  |
| --- | --- |
|  | **ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ** |
| 1 | Создать класс точек в пространстве, заданных координатами. Создать метод, возвращающий расстояние до другой точки в пространстве. С помощью метода определить попадает ли точка в сферу с центром в начале координат и заданным радиусом. |
| 2 | Создать класс цилиндров, заданных высотой цилиндра и радиусом основания. Создать методы, возвращающие значение объема цилиндра, площади его поверхности и сравнивающие объемы двух цилиндров. |
| 3 | Создать класс конусов, заданных высотой конуса и радиусом его основания. Создать методы, возвращающие значение объема конуса и площадь его поверхности, а также метод сравнения объемов конусов. |
| 4 | Создать класс треугольников, заданных своими сторонами. Создать методы, возвращающие периметр треугольника, его площадь и сравнивают периметры. |
| 5 | Создать класс прямоугольных треугольников, заданных своими катетами. Создать методы, возвращающие значение площади треугольника, его гипотенузы и сравнивают величины их площадей. |
| 6 | Создать класс точек в пространстве, заданных своими координатами. С помощью метода определить попадает ли точка внутрь шара с центом в начале координат и радиусом R. |
| 7 | Создать класс шаров, заданных координатами центра и радиусом. Создать методы, возвращающие значение объема шара, площадь его поверхности и сравнивают их объемы. |
| 8 | Создать класс шаров, заданных координатами центра и радиусом. С помощью метода определить попадает ли шар в сферу с центром в начале координат и заданным радиусом. |
| 9 | Создать класс прямоугольников, заданных своими сторонами. Создать методы, возвращающими его площадь, периметр и сравнивают периметры. |
| 10 | Создать класс шаров, заданных координатами центра и радиусом. Создать методы, возвращающие расстояние между центрами шаров, их объем и сравнивают объемы. |
| 12 | Создать класс шаров, заданных координатами центра и радиусом. Создать методы, возвращающие расстояние между центрами шаров, их объем и сравнивают значения объемов. |
| 13 | Создать класс прямоугольных параллелепипедов, заданных своими сторонами. Создать методы, возвращающие значение объема параллелепипеда, площади его поверхности и сравнивают их объемы. |
| 14 | Создать класс прямоугольников, заданных своими сторонами. Создать методы, возвращающими его площадь, периметр и сравнивают их площади. |
| 15 | Создать класс треугольников, заданных своими сторонами. Создать методы, возвращающие периметр треугольника, его площадь и сравнивают их площади. |
| 16 | Создать класс прямоугольников, заданных своими сторонами. Создать методы, возвращающими его площадь, периметр и сравнивают периметры. |
| 17 | Создать класс шаров, заданных координатами центра и радиусом. Создать методы, возвращающие расстояние между центрами шаров, их объем и сравнивают объемы. |

**Задача**

**Базовый класс- класс точек в пространстве.**

Поля: координаты точки в пространстве - целые числа

Методы:

1. Расстояние между двумя точками
2. Попадает ли точка внутрь сферы с центром в начале координат и заданным радиусом.
3. Попадает ли точка внутрь сферы с заданными координатами центра и радиусом.

**Производный класс-класс сфер в пространстве**

Поле: радиус сферы

1. Расстояние между центрами двух сфер
2. Пересекаются ли сферы
3. Площадь поверхности сферы
4. Объем шара, ограниченного сферой

S = 2 π r H {\displaystyle S=2\pi rH} .

**Пример. Создать класс точек в пространстве, заданных координатами. Создать метод, возвращающий расстояние до другой точки в пространстве. С помощью метода определить попадает ли точка в сферу с центром в начале координат и заданным радиусом.**

#include <math.h>

#include <iostream.h>

// Класс точки

**class Point**

{

**public:**

// Конструктор по умолчанию

**Point();**

// Конструктор

**Point(double x, double y, double z);**

// Расстояние до другой точки

// Ни поля объекта, изменены, поэтому const

**double distanceTo(const Point &p) const;**

// Определить расположена ли точка внутри сферы с центром в начале

//координат и радиусом r

// Поля объекта изменены не будут, потому const

**bool insideSphere(double r) const;**

**private:**

// Координаты точки

**double x;**

**double y;**

**double z;**

};

**Point::Point()**

{

x = y = z = 0.;

}

**Point::Point(double x, double y, double z)**

{

this->x = x;

this->y = y;

this->z = z;

}

**double Point::distanceTo(const Point &p) const**

{

double dx, dy, dz;

dx = this->x - p.x;

dy = this->y - p.y;

dz = this->z - p.z;

return sqrt(dx \* dx + dy \* dy + dz \* dz);

}

**bool Point::insideSphere(double r) const**

{

double d;

Point zero;

d = this->distanceTo(zero);

return d <= r;

}

int main()

{

double x, y, z, r;

cout << " Enter the coordinates of the sphere \n";

cin >> x >> y >> z;

cout << " Enter the radius of the sphere \n";

cin >> r;

Point p(x, y, z);

if (p.insideSphere(r))

{

cout << " Point inside the sphere \n";

}

else {

cout << " Point outside the sphere \n";

}

return 0;

}