Документация на проект „Геометрия“

Изготвил: Христо Вригазов, Компютърни науки, първи курс, фак. Номер 81133

Проектът е изграден от следните класове (структури):

1. Клас Point, представящ точка в тримерното пространство с член данни и член функции:

```
private:
    double x; // представя x координатата на точката
    double y; // представя y координатата на точката
    double z; // представя z координатата на точката
public:
    Point(); // конструктор по подразбиране – точка (0, 0, 0)
    Point(double, double, double); // конструктор с параметри
    Point(const Line&, const Line&); // конструктор като пресечна точка на две прави
    Point(const Line&, const Plane&); // конструктор като пресечна точка на права и равнина
    ~Point(); // деструктор
    Point(const Point& other); // копи конструктор
    Point& operator=(const Point& other); // оператор за присвояване
    // селектори за координатите
    const double getX() const;
    const double getY() const;
    const double getZ() const;
   // мутатори за координатите
    void setX(double);
    void setY(double);
    void setZ(double);
    Point& translateWith(const Vector&); // транслация на точка с вектор
    friend ostream& operator<< (ostream& os, const Point&); // оператор за извеждане
    bool belongsToLine(const Line&); // проверка дали точка лежи на дадена права
    bool belongsToPlane(const Plane&); // проверка дали точката лежи на равнина
    Point projectionToPlane(const Plane&); //намиране на проекцията на точка в равнина
```

2. Клас Vector, който е наследник на клас Point, тъй като всяка точка може да се разглежда като своя радиус-вектор. Член данни и член функции:

```
public:
    Vector(); // конструктор по подразбиране
    Vector(double, double, double); // конструктор по три координати
    Vector(const Point&, const Point&); // конструктор по дадени краища на вектора
    Vector(const Point&); // конструиране на вектор като радиус-вектор на дадена точка
    Vector(const Plane&); // конструиране на вектор като нормален вектор на равнина
    double dotProduct(const Vector&) const; // скаларно произведение с друг вектор
    Vector crossProduct(const Vector&) const; // векторно произведение с друг вектор
    ~Vector(); // деструктор
    Vector(const Vector& other); // копи конструктор
```

Vector& operator=(const Vector& other); // оператор за присвояване
void normalize(); // мутатор за нормиране на вектора
double length() const; // селектор за дължината на вектора
bool isCollinearWith(const Vector&) const; // проверка за колинеарност с друг вектор
friend double angleBetweenVectors(const Vector&, const Vector&); // функция, която връща ъгъла между два вектора
Vector generatePerpendicular() const; // функция, която генерира перпендикулярен вектор на дадения

3. Клас Line, представящ права в тримерното пространство чрез параметрично уравнение от вида P + Vs, където P е точка от права, V е вектор от правата, а s е реален параметър. Член данни и член функции:

private:
Point p;
Vector v;
public:
Line(); // конструктор по подразбиране правата Ox
Line(const Point&, const Vector&); // конструктор по точка и вектор
Line(const Point& , const Point&); // права през две точки
Line(const Point&, const Plane&); // перпендикулярна на равнина през дадена точка
Line(const Plane&, const Plane&); // пресечница на две равнини
~Line(); // деструктор
Line(const Line& other); // копи конструктор
Line& operator=(const Line& other); // оператор за присвояване
Мутатори:
void setPoint(Point);
void setVector(Vector);
Селектори:
const Point& getPoint() const;
const Vector& getVector() const;

friend ostream& operator<< (ostream& os, const Line& l); // извеждане на уравнението на правата
void representAsPlaneIntersection(Plane& p, Plane& r); // представяне на правата като пресечница на произвони две равнини
State stateWithPlane(const Plane&) const; // функция за намиране на взаимното положение на права и равнина
State stateWithLine(const Line&) const; // функция за намиране на взаимното положение на две прави

4. Клас Plane, представящ равнина в пространството чрез уравнение от вида Ax + By + Cz + D = 0.

private: // коефициентите в уравнението на правата
double A;
double B;
double C;
double D;

public:

Plane(); // конструктор по подразбиране – равнината Oyz

Plane(double, double, double, double); // конструктор с параметри

Plane(const Plane&, const Point&); // конструктор като успоредна на друга равнина и минаваща през точка

Plane(const Line&, const Point&); // конструтор като перпендикулярна на дадена права през дадена точка

Plane(const Vector&, const Vector&, const Point&); // успоредна на двойка неколинеарни вектора и минаваща през дадена точка

Plane(const Point&, const Point&, const Point&); // конструктор на равнина, минаваща през три точки

~Plane(); // деструктор

Plane(const Plane& other); // копи конструктор

Plane& operator=(const Plane& other); // оператор за присвояване

Селектори за коефициентите в уравнението на равнината:

const double getA() const;

const double getB() const;

const double getC() const;

const double getD() const;

Мутатори за коефициентите в уравнението на равнината:

void setA(double);

void setB(double);

void setC(double);

void setD(double);

Функции за извеждане на уравненията на правата в нормална и параметрична форма:

void showEquation() const;

void showParametricalEquation() const;

Селектор за намиране на нормалния вектор (A, B, C) на дадена права

Vector normal() const;

Функция за намиране на взаимното положение с друга равнина:

State stateWithPlane(const Plane&);

5. Клас Triangle, представящ триъгълник в тримерното пространство:

private:

Point p[3]; // представен като масив от 3 точки

public:

Triangle(); // конструктор по подразбиране

Triangle(const Point&, const Point&, const Point&); // конструктор по три точки

~Triangle(); // деструктор

Triangle(const Triangle& other); // копи конструктор

Triangle& operator=(const Triangle& other); // оператор за присвояване

Plane getPlane() const; // селектор за равнината на триъгълника

Line getLine(short) const; // селектор за някоя права, определена от две точки от триъгълника

Line getHeight(short) const; // селектор за височината, определена от някой връх

Line getMedian(short) const; // селектор за медиана
Line getSymmetral(short) const; // селектор за симетрала
Line getBisector(short) const; // селектор за ъглополовяща
Point getOrthocenter(short) const; // селектор за ортоцентър
Point getMedicenter(short) const; // селектор за медицентър
Point getDescribedCircleCenter(short) const; // селектор за центъра на описана окръжност
Point getInscribedCircleCenter(short) const; // селектор за центъра на вписана окръжност

6. Допълнителни функции и типове, нужни за осигуряването на описаната функционалност:
   6.1. Структура Solution, чрез която представяме решенията на система от 2 уравнения с решения a и b или решение на линейно уравние с решение a = b
   bool exists; // индикатор дали има решение
   bool hasExactlyOneSolution; // дали има точно 1 решение
   bool hasManySolutions; // индикатор дали има много решения
   double a;
   double b;
   6.2. Допълнителни функции в “Point.h”:
   Solution solveEquations (const Point&, const Point&); // решава система от 2 уравнения с 2 неизвестни, като коефициентите на уравненията са подадени като точки
   Solution solveLinearEquation(const double A, const double B); // решава линейно уравнение от вида Ax = B
   double det(double, double, double, double); // изчислява детерминанта 2x2
   6.3. Изброен тип State:
   Intersection = 0,
   Parallel = 1,
   Skew = 2
   6.4. Други допълнителни функции в “Angle.h” и “Distances.h”:
   double angleBetweenLinePlane(const Line&, const Plane&); // намира ъгъла между права и равнина
   double angleBetweenVectors(const Vector&, const Vector&); // намира ъгъл между два вектора
   double distancePointLine(const Plane&, const Line&); // разстояние между точка и права
   double distancePointPlane(const Plane&, const Point&); // разстояние между точка и равнина
   double distanceSkewLines(const Line&, const Line&); // разстояние между кръстосани прави