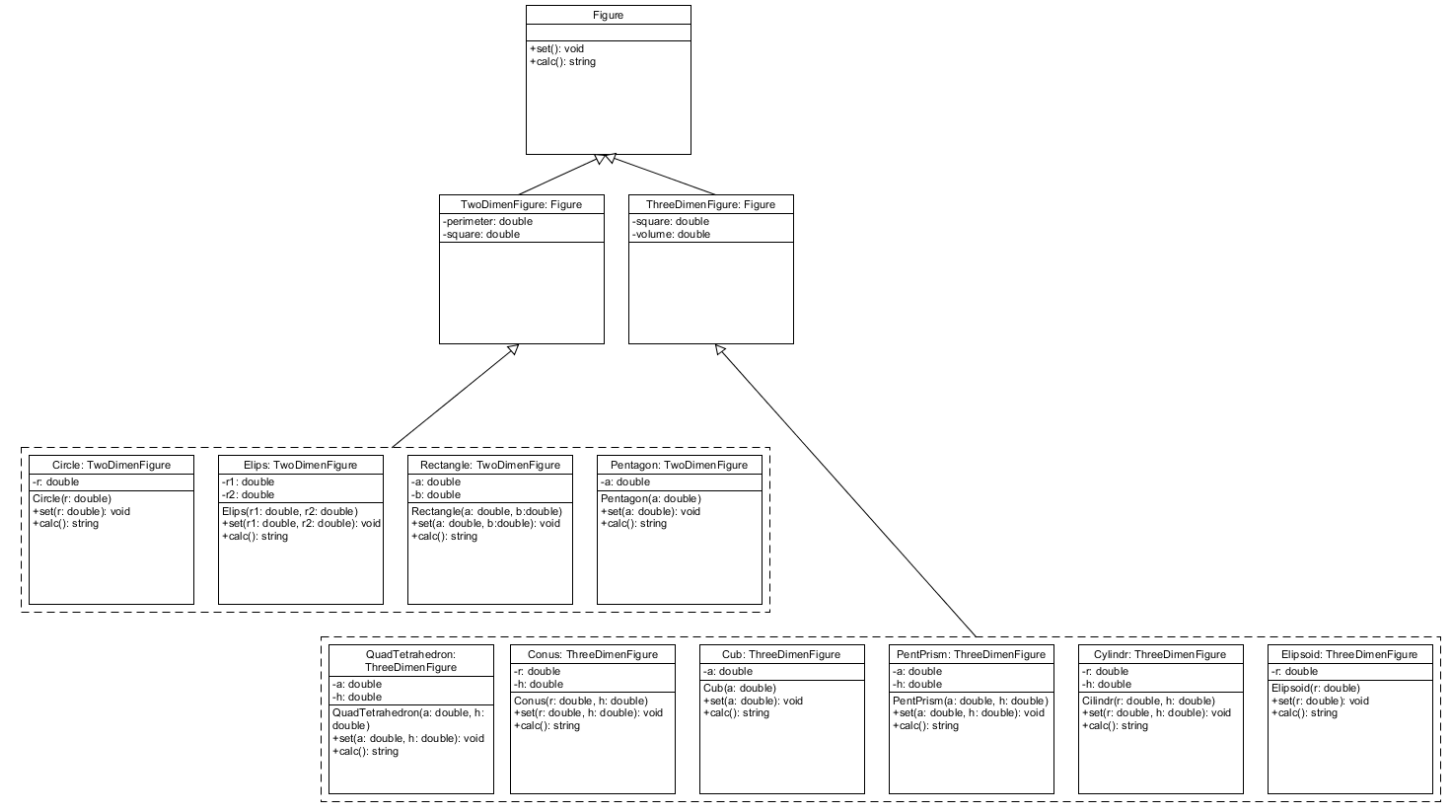
Figure
+set(): void
+calc(): string
TwoDimenFigure: Figure
-perimeter: double
-square: double
ThreeDimenFigure: Figure
-square: double
-volume: double
Circle: TwoDimenFigure
-r: double
Circle(r: double)
+set(r: double): void
+calc(): string
Elips: TwoDimenFigure
-r1: double
-r2: double
Elips(r1: double, r2: double)
+set(r1: double, r2: double): void
+calc(): string
Rectangle: TwoDimenFigure
-a: double
-b: double
Rectangle(a: double, b:double)
+set(a: double, b:double): void
+calc(): string
Pentagon: TwoDimenFigure
-a: double
Pentagon(a: double)
+set(a: double): void
+calc(): string
QuadTetrahedron: ThreeDimenFigure
-a: double
-h: double
QuadTetrahedron(a: double, h: double)
+set(a: double, h: double): void
+calc(): string
Conus: ThreeDimenFigure
-r: double
-h: double
Conus(r: double, h: double)
+set(r: double, h: double): void
+calc(): string
Cub: ThreeDimenFigure
-a: double
Cub(a: double)
+set(a: double): void
+calc(): string
PentPrism: ThreeDimenFigure
-a: double
-h: double
PentPrism(a: double, h: double)
+set(a: double, h: double): void
+calc(): string
Cylindr: ThreeDimenFigure
-r: double
-h: double
Cilindr(r: double, h: double)
+set(r: double, h: double): void
+calc(): string
Elipsoid: ThreeDimenFigure
-r: double
Elipsoid(r: double)
+set(r: double): void
+calc(): string

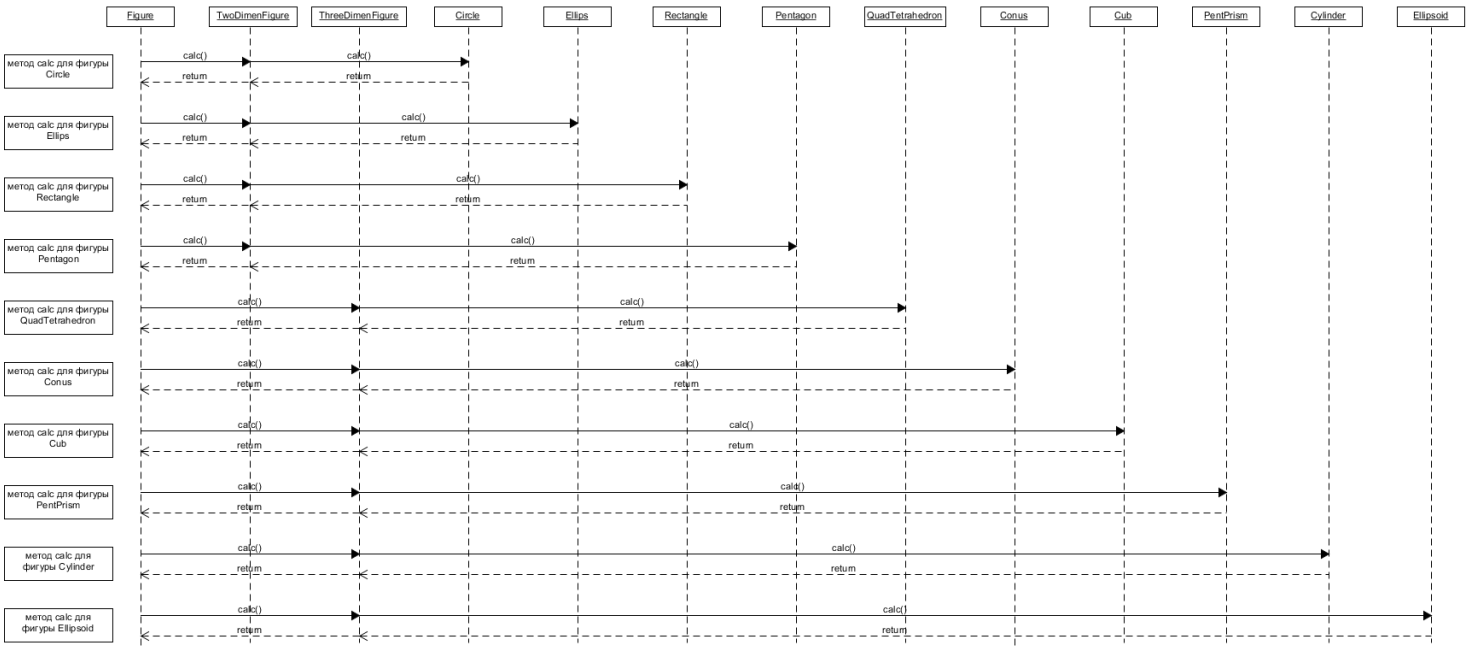
Figure
TwoDimenFigure
ThreeDimenFigure
Circle
Ellips
Rectangle
Pentagon
QuadTetrahedron
Conus
Cub
PentPrism
Cylinder
Ellipsoid
метод calc для фигуры Circle
метод calc для фигуры Ellips
метод calc для фигуры Rectangle
метод calc для фигуры Pentagon
метод calc для фигуры QuadTetrahedron
метод calc для фигуры Conus
метод calc для фигуры Cub
метод calc для фигуры PentPrism
метод calc для фигуры Cylinder
метод calc для фигуры Ellipsoid
calc()
return

```
#define _USE_MATH_DEFINES

#include <iostream>
#include <cmath>

using namespace std;

class Figure
{
  public:
    virtual void set(double);
    virtual void set(double, double);
    virtual string calc();
};

class TwoDimenFigure: public Figure
{
  protected:
    double perimeter;
    double square;
};

class ThreeDimenFigure: public Figure
{
  protected:
    double square;
    double volume;
};

class Circle: public TwoDimenFigure
{
  private:
    double r;
  public:
```

```
    Circle(double r): r(r){}
    void set(double r){this->r = r;}
    string calc();
};

string Circle::calc()
{
  square = M_PI*r*r;
  perimeter = 2*M_PI*r;
  string answer =  "S=" + to_string(square) + ", P=" + to_string(perimeter);
  return answer;
}

class Elips: public TwoDimenFigure
{
  private:
    double r1, r2;
  public:
    Elips(double r1, double r2): r1(r1), r2(r2) {}
    void set(double r1, double r2){this->r1 = r1; this->r2 = r2;}
    string calc();
};

string Elips::calc()
{
  if(r1 == r2)
  {
    Circle t(r1);
    return t.calc();
  }
  square = M_PI*r1*r2;
  perimeter = 4*(M_PI*r1*r2+r1-r2)/(r1+r2);
  string answer =  "S=" + to_string(square) + ", P=" + to_string(perimeter);
  return answer;
```

```
}

class Rectangle: public TwoDimenFigure
{
  private:
    double a, b;
  public:
    Rectangle(double a, double b): a(a), b(b) {}
    void set(double a, double b){this->a = a; this->b = b;}
    string calc();
};

string Rectangle::calc()
{
  square = a*b;
  perimeter = 2*(a+b);
  string answer =  "S=" + to_string(square) + ", P=" + to_string(perimeter);
  return answer;
}

class Pentagon: public TwoDimenFigure
{
  private:
    double a;
  public:
    Pentagon(double a): a(a) {}
    void set(double a){this->a = a;}
    string calc();
};

string Pentagon::calc()
{
  double p = 3*a/2;
  square = sqrt(p*pow(p-a, 3));
```

```
  perimeter = a*5;
  string answer =  "S=" + to_string(square) + ", P=" + to_string(perimeter);
  return answer;
}

class QuadTetrahedron: public ThreeDimenFigure
{
  private:
    double a, h;
  public:
    QuadTetrahedron(double a, double h): a(a), h(h) {}
    void set(double a, double h){this->a = a; this->h = h;}
    string calc();
};

string QuadTetrahedron::calc()
{
  if(h == 0)
  {
    Rectangle t(h, h);
    return t.calc();
  }
  volume = a*a*h/3;
  double l = sqrt(h*h+a*a/4);
  square = a*a+l*a*2;
  string answer =  "V=" + to_string(volume) + ", S=" + to_string(square);
  return answer;
}

class Conus: public ThreeDimenFigure
{
  private:
    double r, h;
  public:
```

```
    Conus(double r, double h): r(r), h(h) {}
    void set(double r, double h){this->r = r; this->h = h;}
    string calc();
};

string Conus::calc()
{
  if(r == 0)
  {
    Rectangle t(h, h);
    return t.calc();
  }
  if(h == 0)
  {
    Circle t(r);
    return t.calc();
  }
  volume = M_PI*r*r*h/3;
  double l = sqrt(r*r+h*h);
  square = M_PI*r*(r+l);
  string answer =  "V=" + to_string(volume) + ", S=" + to_string(square);
  return answer;
}

class Cub: public ThreeDimenFigure
{
  private:
    double a;
  public:
    Cub(double a): a(a) {}
    void set(double a){this->a = a;}
    string calc();
};
```

```
string Cub::calc()
{
  volume = a*a*a;
  square = 6*a*a;
  string answer =  "V=" + to_string(volume) + ", S=" + to_string(square);
  return answer;
}

class PentPrism: public ThreeDimenFigure
{
  private:
    double a, h;
  public:
    PentPrism(double a, double h): a(a), h(h) {}
    void set(double a, double h){this->a = a; this->h = h;}
    string calc();
};

string PentPrism::calc()
{
  if(a == 0)
  {
    Rectangle t(h, h);
    return t.calc();
  }
  if(h == 0)
  {
    Pentagon t(a);
    return t.calc();
  }
  double p = 3*a/2;
  double sba = sqrt(p*pow(p-a, 3));
  volume = sba*h;
  square = sba*2+5*h*a;
```

```
  string answer =  "V=" + to_string(volume) + ", S=" + to_string(square);
  return answer;
}

class Cylindr: public ThreeDimenFigure
{
  private:
    double r, h;
  public:
    Cylindr(double r, double h): r(r), h(h) {}
    void set(double r, double h){this->r = r; this->h = h;}
    string calc();
};

string Cylindr::calc()
{
  if(r == 0)
  {
    Rectangle t(h, h);
    return t.calc();
  }
  if(h == 0)
  {
    Circle t(r);
    return t.calc();
  }
  volume = M_PI*r*r*h;
  square = M_PI*r*r*2+2*M_PI*r*h;
  string answer =  "V=" + to_string(volume) + ", S=" + to_string(square);
  return answer;
}

class Elipsoid: public ThreeDimenFigure
{
```

```
  private:
    double r;
  public:
    Elipsoid(double r): r(r) {}
    void set(double r){this->r = r;}
    string calc();
};

string Elipsoid::calc()
{
  volume = 4*M_PI*r*r*r/3;
  square = 4*M_PI*r*r;
  string answer =  "V=" + to_string(volume) + ", S=" + to_string(square);
  return answer;
}

void add(string name, Figure *all, int num)
{
  if(name == "круг")
  {
    double r;
    cin >> r;
    all[num] = Circle(r);
  }
  if(name == "эллипс") ;
  if(name == "прямоугольник") ;
  if(name == "четырехугольный тетраэдр") ;
  if(name == "конус") ;
  if(name == "куб") ;
  if(name == "призма пятиугольная") ;
  if(name == "цилиндр") ;
  if(name == "эллипсоид") ;
}
```

```
int main()
{
   Figure *all;
   return 0;
}
```