Main.cpp

#include <iostream>

#include "Tools.h"

//#include "IShape.h"

//#include "TwoDShape.h"

//#include "ThreeDShape.h"

//#include "Circle.h"

//#include "Rectangle.h"

//#include "Sphere.h"

//#include "Cube.h"

//#include <conio.h>

int main()

{

mainMenu();

}

Tools.h

#pragma once

#include <iostream>

#include "IShape.h"

#include "TwoDShape.h"

#include "ThreeDShape.h"

#include "Circle.h"

#include "Rectangle.h"

#include "Sphere.h"

#include "Cube.h"

#include <conio.h>

const int UNDEFINED\_VALUE = 0;

const double PI = 3.14;

double inputNumber();

void mainMenu();

void showMenu();

void createCircle();

void createSphere();

void createRectangle();

void createCube();

Tools.cpp

#include "Tools.h"

double inputNumber()

{

double number;

while (1)

{

std::cin >> number;

if (std::cin.get() == '\n')

{

return number;

}

else

{

std::cout << "Incorrect input!" << std::endl;

std::cin.clear();

std::cin.ignore(256, '\n');

std::cout << "Number: ";

}

}

}

void mainMenu()

{

showMenu();

while (1)

{

switch (\_getch())

{

case '1': createCircle(); break;

case '2': createRectangle(); break;

case '3': createSphere(); break;

case '4': createCube(); break;

default:continue;

}

break;

}

}

void showMenu()

{

std::cout << "~~~~~~~~~~~~~~~~" << std::endl;

std::cout << "Choose a shape!" << std::endl;

std::cout << "1. Circle" << std::endl;

std::cout << "2. Rectangle" << std::endl;

std::cout << "3. Sphere" << std::endl;

std::cout << "4. Cube" << std::endl;

std::cout << "~~~~~~~~~~~~~~~~" << std::endl;

}

void createCircle()

{

Circle circle;

circle.printInfo();

}

void createRectangle()

{

Rectangle rectangle;

rectangle.printInfo();

}

void createSphere()

{

Sphere sphere;

sphere.printInfo();

}

void createCube()

{

Cube cube;

cube.printInfo();

}

IShape.h

#pragma once

class IShape

{

protected:

virtual void printInfo() = 0; //Вывод данных о фигуре

//Установка значений координат центра фигуры

virtual void setXCenter(double) = 0;

virtual void setYCenter(double) = 0;

virtual void setZCenter(double) = 0;

//Установка значений координат произвольной точки

virtual void setXPoint(double) = 0;

virtual void setYPoint(double) = 0;

virtual void setZPoint(double) = 0;

};

TwoDShape.h

#pragma once

#include "IShape.h"

#include "Tools.h"

class TwoDShape : public IShape

{

protected:

//Координаты центра

double m\_x\_center;

double m\_y\_center;

double m\_z\_center;

//Координаты произвольной точки

double m\_x\_point;

double m\_y\_point;

double m\_z\_point;

//Площадь

double m\_area;

public:

TwoDShape(); //Конструктор

void createCenter(); //Определение координат центра

void createCoordinates(); //Определение координат произвольной точки

//Установка значений координат центра

void setXCenter(double);

void setYCenter(double);

void setZCenter(double);

//Установка значений координат произвольной точки

void setXPoint(double);

void setYPoint(double);

void setZPoint(double);

virtual double calculateArea()=0; //Подсчет площади

};

TwoDShape.cpp

#include "TwoDShape.h"

TwoDShape::TwoDShape()

{

createCenter();

createCoordinates();

}

void TwoDShape::createCenter()

{

//Координата х центра

std::cout << "X (center) = ";

double x\_center = inputNumber();

setXCenter(x\_center);

//Координата у центра

std::cout << "Y (center) = ";

double y\_center = inputNumber();

setYCenter(y\_center);

//Координата z центра - на 2D плоскости равна 0

setZCenter(UNDEFINED\_VALUE);

}

void TwoDShape::createCoordinates()

{

//Координата х произвольной точки

std::cout << "X (point) = ";

double x\_coordinate = inputNumber();

setXPoint(x\_coordinate);

//Координата у произвольной точки

std::cout << "Y (point) = ";

double y\_coordinate = inputNumber();

setYPoint(y\_coordinate);

//Координата z произвольной точки - на 2D плоскости равна 0

setZPoint(UNDEFINED\_VALUE);

}

void TwoDShape::setXCenter(double x\_center = UNDEFINED\_VALUE)

{

m\_x\_center = x\_center;

}

void TwoDShape::setYCenter(double y\_center = UNDEFINED\_VALUE)

{

m\_y\_center = y\_center;

}

void TwoDShape::setZCenter(double z\_center = UNDEFINED\_VALUE)

{

m\_z\_center = z\_center;

}

void TwoDShape::setXPoint(double x\_coordinate = UNDEFINED\_VALUE)

{

m\_x\_point = x\_coordinate;

}

void TwoDShape::setYPoint(double y\_coordinate = UNDEFINED\_VALUE)

{

m\_y\_point = y\_coordinate;

}

void TwoDShape::setZPoint(double z\_coordinate = UNDEFINED\_VALUE)

{

m\_z\_point = z\_coordinate;

}

Circle.h

#pragma once

#include "TwoDShape.h"

class Circle : public TwoDShape

{

private:

double m\_radius; //Радиус

public:

Circle(); //Конструктор

void printInfo(); //Вывод данных о фигуре

double calculateRadius(); //Подсчет радиуса

double calculateArea(); //Подсчет площади

};

Circle.cpp

#include "Circle.h"

Circle::Circle()

{

m\_radius = calculateRadius();

m\_area = calculateArea();

}

double Circle::calculateRadius()

{

//Радиус вычисляется по теореме Пифагора, где катеты - разность координат х и разность координат у

//Нахождение квадрата разности координат х

double x\_number = m\_x\_center - m\_x\_point;

x\_number \*= x\_number;

//Нахождение квадрата разности координат у

double y\_number = m\_y\_center - m\_y\_point;

y\_number \*= y\_number;

//Радиус

double radius = sqrt(x\_number + y\_number);

return radius;

}

double Circle::calculateArea()

{

return PI \* m\_radius \* m\_radius;

}

void Circle::printInfo()

{

std::cout << "Figure: circle" << std::endl;

std::cout << "Center: (" << m\_x\_center << ";" << m\_y\_center << ")" << std::endl;

std::cout << "Point: (" << m\_x\_point << ";" << m\_y\_point << ")" << std::endl;

std::cout << "Radius: " << m\_radius << std::endl;

std::cout << "Area: " << m\_area << std::endl;

std::cout << "Volume: 0" << std::endl;

}

Rectangle.h

#pragma once

#include "TwoDShape.h"

class Rectangle:public TwoDShape

{

private:

//Стороны

double m\_length; //Длина

double m\_width; //Ширина

public:

Rectangle(); //Конструктор

void printInfo(); //Вывод данных о фигуре

double calculateLength(); //Подсчет длины

double calculateWidth(); //Подсчет ширины

double calculateArea(); //Подсчет площади

};

Rectangle.cpp

#include "Rectangle.h"

Rectangle::Rectangle()

{

m\_length = calculateLength();

m\_width = calculateWidth();

m\_area = calculateArea();

}

double Rectangle::calculateLength()

{

return 2\*fabs(m\_x\_center - m\_x\_point);

}

double Rectangle::calculateWidth()

{

return 2 \* fabs(m\_y\_center - m\_y\_point);

}

double Rectangle::calculateArea()

{

return m\_length \* m\_width;

}

void Rectangle::printInfo()

{

std::cout << "Figure: rectangle" << std::endl;

std::cout << "Center: (" << m\_x\_center << ";" << m\_y\_center << ")" << std::endl;

std::cout << "Point: (" << m\_x\_point << ";" << m\_y\_point << ")" << std::endl;

std::cout << "Length: " << m\_length << std::endl;

std::cout << "Width: " << m\_width << std::endl;

std::cout << "Area: " << m\_area << std::endl;

std::cout << "Volume: 0" << std::endl;

}

ThreeDShape.h

#pragma once

#include "IShape.h"

#include "Tools.h"

class ThreeDShape : public IShape

{

protected:

//Координаты центра

double m\_x\_center;

double m\_y\_center;

double m\_z\_center;

//Координаты произвольной точки

double m\_x\_point;

double m\_y\_point;

double m\_z\_point;

//Объем

double m\_volume;

public:

ThreeDShape(); //Конструктор

void createCenter(); //Определение координат центра

void createCoordinates(); //Определение координат произвольной точки

//Установка значений координат центра

void setXCenter(double);

void setYCenter(double);

void setZCenter(double);

//Установка значений координат произвольной точки

void setXPoint(double);

void setYPoint(double);

void setZPoint(double);

virtual double calculateVolume() = 0; //Подсчет площади

};

ThreeDShape.cpp

#include "ThreeDShape.h"

ThreeDShape::ThreeDShape()

{

createCenter();

createCoordinates();

}

void ThreeDShape::createCenter()

{

//Координата х центра

std::cout << "X (center) = ";

double x\_center = inputNumber();

setXCenter(x\_center);

//Координата у центра

std::cout << "Y (center) = ";

double y\_center = inputNumber();

setYCenter(y\_center);

//Координата z центра

std::cout << "Z (center) = ";

double z\_center = inputNumber();

setZCenter(z\_center);

}

void ThreeDShape::createCoordinates()

{

//Координата х произвольной точки

std::cout << "X (point) = ";

double x\_coordinate = inputNumber();

setXPoint(x\_coordinate);

//Координата у произвольной точки

std::cout << "Y (point) = ";

double y\_coordinate = inputNumber();

setYPoint(y\_coordinate);

//Координата z произвольной точки

std::cout << "Z (point) = ";

double z\_coordinate = inputNumber();

setZPoint(z\_coordinate);

}

void ThreeDShape::setXCenter(double x\_center = UNDEFINED\_VALUE)

{

m\_x\_center = x\_center;

}

void ThreeDShape::setYCenter(double y\_center = UNDEFINED\_VALUE)

{

m\_y\_center = y\_center;

}

void ThreeDShape::setZCenter(double z\_center = UNDEFINED\_VALUE)

{

m\_z\_center = z\_center;

}

void ThreeDShape::setXPoint(double x\_coordinate = UNDEFINED\_VALUE)

{

m\_x\_point = x\_coordinate;

}

void ThreeDShape::setYPoint(double y\_coordinate = UNDEFINED\_VALUE)

{

m\_y\_point = y\_coordinate;

}

void ThreeDShape::setZPoint(double z\_coordinate = UNDEFINED\_VALUE)

{

m\_z\_point = z\_coordinate;

}

Sphere.h

#pragma once

#include "ThreeDShape.h"

class Sphere :public ThreeDShape

{

private:

double m\_radius;

public:

Sphere(); //Конструктор

void printInfo(); //Вывод данных о фигуре

double calculateRadius(); //Подсчет радиуса

double calculateVolume(); //Подсчет площади

};

Sphere.cpp

#include "Sphere.h"

Sphere::Sphere()

{

m\_radius = calculateRadius();

m\_volume = calculateVolume();

}

double Sphere::calculateRadius()

{

//Сумма квадратов трех координат равна квадрату радиуса

//Нахождение квадрата разности координат х

double x\_number = m\_x\_center - m\_x\_point;

x\_number \*= x\_number;

//Нахождение квадрата разности координат у

double y\_number = m\_y\_center - m\_y\_point;

y\_number \*= y\_number;

//Нахождение квадрата разности координат z

double z\_number = m\_z\_center - m\_z\_point;

z\_number \*= z\_number;

//Радиус

double radius = sqrt(x\_number + y\_number + z\_number);

return radius;

}

double Sphere::calculateVolume()

{

return 4 / 3 \* PI \* m\_radius \* m\_radius \* m\_radius;

}

void Sphere::printInfo()

{

std::cout << "Figure: Sphere" << std::endl;

std::cout << "Center: (" << m\_x\_center << ";" << m\_y\_center << ";" << m\_z\_center << ")" << std::endl;

std::cout << "Point: (" << m\_x\_point << ";" << m\_y\_point << ";" << m\_z\_point << ")" << std::endl;

std::cout << "Radius: " << m\_radius << std::endl;

std::cout << "Volume: " << m\_volume << std::endl;

}

Cube.h

#pragma once

#include "ThreeDShape.h"

class Cube :public ThreeDShape

{

private:

double m\_edge; //Ребро

public:

Cube(); //Конструктор

void printInfo(); //Вывод данных о фигуре

double calculateEdge(); //Подсчет длины

double calculateVolume(); //Подсчет объема

};

Cube.cpp

#include "Cube.h"

Cube::Cube()

{

m\_edge = calculateEdge();

m\_volume = calculateVolume();

}

double Cube::calculateEdge()

{

//Сумма квадратов трех координат равна квадрату диагонали

//Нахождение квадрата разности координат х

double x\_number = m\_x\_center - m\_x\_point;

x\_number \*= x\_number;

//Нахождение квадрата разности координат у

double y\_number = m\_y\_center - m\_y\_point;

y\_number \*= y\_number;

//Нахождение квадрата разности координат z

double z\_number = m\_z\_center - m\_z\_point;

z\_number \*= z\_number;

//Диагональ

double half\_diagonal = sqrt(x\_number + y\_number + z\_number);

double edge = half\_diagonal \* sqrt(2);

return edge;

}

double Cube::calculateVolume()

{

return m\_edge \* m\_edge \* m\_edge;

}

void Cube::printInfo()

{

std::cout << "Figure: Cube" << std::endl;

std::cout << "Center: (" << m\_x\_center << ";" << m\_y\_center << ";" << m\_z\_center << ")" << std::endl;

std::cout << "Point: (" << m\_x\_point << ";" << m\_y\_point << ";" << m\_z\_point << ")" << std::endl;

std::cout << "Edge: " << m\_edge << std::endl;

std::cout << "Volume: " << m\_volume << std::endl;

}