**1.**

//Пищик Е.С. М8О-206Б-19 Лабораторная работа №4 Вариант 16

//8-угольник, треугольник, квадрат

#include <iostream>

#include <string>

#include <utility>

#include <cmath>

#include <algorithm>

#include <fstream>

//Константа "pi"

#define PI 3.14159265

//Шаблонные классы фигур

//Класс квадрата

template <typename T, size\_t SIZE=4>

struct Square

{

    using type = T;

    std::pair<T,T> v\_arr[SIZE];

    size\_t size = SIZE;

    std::pair<T,T>\* begin() { return &v\_arr[0]; }

    std::pair<T,T>\* end() { return &v\_arr[SIZE]; }

};

//Класс треугольника

template <typename T, size\_t SIZE=3>

struct Triangle

{

    using type = T;

    std::pair<T,T> v\_arr[SIZE];

    size\_t size = SIZE;

    std::pair<T,T>\* begin() { return &v\_arr[0]; }

    std::pair<T,T>\* end() { return &v\_arr[SIZE]; }

};

//Класс восьмиугольника

template <typename T, size\_t SIZE=8>

struct Octagon

{

    using type = T;

    std::pair<T,T> v\_arr[SIZE];

    size\_t size = SIZE;

    std::pair<T,T>\* begin() { return &v\_arr[0]; }

    std::pair<T,T>\* end() { return &v\_arr[SIZE]; }

};

// Печать любой фигуры

template <typename T>

typename std::enable\_if<(sizeof(T::v\_arr)>0),void>::type print(T& fig)

{

    for(auto v: fig)

    {

        std::cout << "(" << v.first << "," << v.second << ") ";

    }

    std::cout << std::endl;

}

// Печать tuple фигур

template <typename T,size\_t index>

typename std::enable\_if<index>=std::tuple\_size<T>::value,void>::type print\_tuple(T&)

{

    std::cout << std::endl;

}

template <typename T,size\_t index>

typename std::enable\_if<index<std::tuple\_size<T>::value,void>::type print\_tuple(T& tuple)

{

    auto v = std::get<index>(tuple);

    std::cout << "Figure " << index << ": ";

    print(v);

    print\_tuple<T,index+1>(tuple);

}

//Нахождение площади фигуры

template<typename T>

typename std::enable\_if<(sizeof(T::v\_arr)>0),double>::type square(T& fig)

{

    double side = sqrt(pow((fig.v\_arr[1].first - fig.v\_arr[0].first), 2) + pow((fig.v\_arr[1].second - fig.v\_arr[0].second), 2));

    for(size\_t i = 1; i < fig.size; ++i)

    {

        side = side < sqrt(pow((fig.v\_arr[i].first - fig.v\_arr[0].first), 2) + pow((fig.v\_arr[i].second - fig.v\_arr[0].second), 2)) ? side : sqrt(pow((fig.v\_arr[i].first - fig.v\_arr[0].first), 2) + pow((fig.v\_arr[i].second - fig.v\_arr[0].second), 2));

    }

    if(fig.size == 3) { return sqrt(3.0)/4.0\*pow(side,2); }

    else if(fig.size == 4) { return pow(side,2); }

    else if(fig.size == 8) { return 2\*pow(side,2)/tan(PI/8); }

    else { return 0.0; }

}

//Нахождение суммарной площади в tuple

template <typename T,size\_t index>

double square\_tuple(T& tuple)

{

    auto item = std::get<index>(tuple);

    double value = square(item);

    if constexpr ((index+1) < std::tuple\_size<T>::value)

    {

        return value + square\_tuple<T,index+1>(tuple);

    }

    return value;

}

int main()

{

    std::string name = "start";

    std::cout << "Enter q or Q or quit or Quit for close programm" << std::endl;

    //Бесконечный дикл до ввода ключевого слова для выхода

    while(true)

    {

        std::cout << "Enter test file name: ";

        std::cin >> name;

        if (name == "Q" || name == "q" || name == "Quit" || name == "quit") break;

        //поток ввода из файла, имя файла хранится в string переменной name

        std::ifstream finp(name.c\_str());

        std::string s11, s12, s21, s22, s31, s32, s41, s42;

        std::string t11, t12, t21, t22, t31, t32;

        std::string p11, p12, p21, p22, p31, p32, p41, p42, p51, p52;

        std::string p61, p62, p71, p72, p81, p82;

        //Взятие 16 чисел для восьмиугольника из файла

        finp >> p11 >> p12 >> p21 >> p22 >> p31 >> p32 >> p41 >> p42 >> p51 >> p52 >> p61 >> p62 >> p71 >> p72 >> p81 >> p82;

        //Взятие 8 чисел для квадрата из файла

        finp >> s11 >> s12 >> s21 >> s22 >> s31 >> s32 >> s41 >> s42;

        //Взятие 6 чисел для треугольника из файла

        finp >> t11 >> t12 >> t21 >> t22 >> t31 >> t32;

        try

        {

            //Создаем три экземпляра шаблонных классов

            Square<int> sq;

            Triangle<double> tr;

            Octagon<double> ogon;

            //Инициализация

            sq.v\_arr[0].first = std::stoi(s11);

            sq.v\_arr[0].second = std::stoi(s12);

            sq.v\_arr[1].first = std::stoi(s21);

            sq.v\_arr[1].second = std::stoi(s22);

            sq.v\_arr[2].first = std::stoi(s31);

            sq.v\_arr[2].second = std::stoi(s32);

            sq.v\_arr[3].first = std::stoi(s41);

            sq.v\_arr[3].second = std::stoi(s42);

            tr.v\_arr[0].first = std::stod(t11);

            tr.v\_arr[0].second = std::stod(t12);

            tr.v\_arr[1].first = std::stod(t21);

            tr.v\_arr[1].second = std::stod(t22);

            tr.v\_arr[2].first = std::stod(t31);

            tr.v\_arr[2].second = std::stod(t32);

            ogon.v\_arr[0].first = std::stod(p11);

            ogon.v\_arr[0].second = std::stod(p12);

            ogon.v\_arr[1].first = std::stod(p21);

            ogon.v\_arr[1].second = std::stod(p22);

            ogon.v\_arr[2].first = std::stod(p31);

            ogon.v\_arr[2].second = std::stod(p32);

            ogon.v\_arr[3].first = std::stod(p41);

            ogon.v\_arr[3].second = std::stod(p42);

            ogon.v\_arr[4].first = std::stod(p51);

            ogon.v\_arr[4].second = std::stod(p52);

            ogon.v\_arr[5].first = std::stod(p61);

            ogon.v\_arr[5].second = std::stod(p62);

            ogon.v\_arr[6].first = std::stod(p71);

            ogon.v\_arr[6].second = std::stod(p72);

            ogon.v\_arr[7].first = std::stod(p81);

            ogon.v\_arr[7].second = std::stod(p82);

            //Создаем tuple из фигур

            auto tuple = std::make\_tuple(tr, sq, ogon);

            //Печать tuple фигур

            print\_tuple<decltype(tuple),0>(tuple);

            //Печать суммарной площади фигур в tuple

            std::cout << "Square: " << square\_tuple<decltype(tuple),0>(tuple) << std::endl;

        }

        catch(std::exception& ex)

        {

            std::cerr << "Incorrect data" << std::endl;

            std::cerr << ex.what() << std::endl;

        }

        finp.close();

    }

    return 0;

}

**2.** <https://github.com/Pe4enIks/OOP_Pishchik_206>

**3.**

Сначала инициализируем квадрат, потом треугольник, потом восьмиугольник

sq – объект класса Square, tr – объект класса Triangle, ogon – объект класса Octagon.

**Тест 1:**

1.41421 3.41421 2.41421 1.0 -1.0 1.0 0.0 3.41421 0.0 0.0 1.41421 0.0 -1.0 2.41421 2.41421 2.41421

0 0 1 1 0 1 1 0

1.0 0.0 0.0 0.0 0.5 0.866025

**4.**

**Результат теста:**

Figure 0: (1,0) (0,0) (0.5,0.866025)

Figure 1: (0,0) (1,1) (0,1) (1,0)

Figure 2: (1.41421,3.41421) (2.41421,1) (-1,1) (0,3.41421) (0,0) (1.41421,0)

(-1,2.41421) (2.41421,2.41421)

Square: 11.0899

**5.** В программе создается 3 шаблонных класса Square, Triangle, Octagon, также создается метафункция print для печати объекта каждого класса, внутри каждого класса написан constexpr метод вычисляющий площадь равностороннего объекта, создана метафункция print\_tuple, принимающая tuple фигур и печатающая каждую фигуру, написана метафункция square\_tuple, возвращающая сумму площадей фигур из tuple.