МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное образовательное учреждения высшего образования

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт компьютерных технологий и информационной безопасности

Кафедра математического обеспечения и применения ЭВМ

**ЛАБОРАТОРНАЯ РАБОТА № 2-3**

по дисциплине

**«Объектно-ориентированное программирование»**

на тему:

**«Классы и объекты С++»**

*Вариант № 6*

Выполнил:

Студент группы

КТбо2-7

Шубенков С. С.

Проверил:

Тарасов С. А.

Оценка

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

Таганрог 2020

# **1 ФОРМУЛИРОВКА ЗАДАНИЯ**

Создать класс Triangle (треугольник), задав в нем длины двух сторон и угол между ними. Реализовать методы вычисления площади и периметра. На его основе создать классы, описывающие равносторонний, равнобедренный и прямоугольный треугольники с присущими им методами вычисления площади и периметра.

При выполнении работы необходимо: − разработать соответствующие классы, конструкторы, поля и методы; − поля класса сделать закрытыми; для чтения и изменения их значений определить открытые методы; – предусмотреть во всех вариантах консольный ввод данных для создания объектов и консольный вывод результатов. – во всех вариантах необходимо использовать хотя бы один раз блоки try catch. Это можно сделать для контроля арифметических ошибок, для проверки существования файлов и т. п. Отсутствие такового карается дополнительным уменьшением оценки. – во всех перечисленных вариантах запрещено пользоваться контейнерами STL, кроме string.

# **2 СПЕЦИФИКАЦИЯ КЛАССОВ**

class ITriangle

{

public:

virtual double Perimeter() const = 0;

virtual double Area() const = 0;

virtual double GetSide(const int num) const = 0;

virtual double GetAngle() const = 0;

virtual ~ITriangle() = default;

};

class Triangle : public ITriangle

{

public:

double Perimeter() const override;

double Area() const override;

double GetAngle() const override { return \_angle; }

double GetSide(const int num) const override;

Triangle() : \_fSide(0), \_sSide(0), \_angle(0) {}

Triangle(const double fSide, const double sSide, const double angle);

private:

double \_fSide;

double \_sSide;

double \_angle;

};

class RightTriangle : public ITriangle

{

public:

virtual double Perimeter() const override;

virtual double Area() const override

double GetAngle() const override { return 90; }

double GetSide(const int num) const override;

RightTriangle() : \_fSide(0), \_sSide(0) {}

RightTriangle(const double fSide, const double sSide);

private:

double \_fSide;

double \_sSide;

};

class EquilateralTriangle : public ITriangle

{

public:

double Perimeter() const override;

double Area() const override;

double GetAngle() const override { return 60; }

double GetSide(const int num) const override;

EquilateralTriangle() : \_side(0) {}

EquilateralTriangle(const double side);

private:

double \_side;

};

class IsoscelesTriangle : public ITriangle

{

public:

double Perimeter() const override;

double Area() const override;

double GetAngle() const override { return \_angle; }

double GetSide(const int num) const override;

IsoscelesTriangle() = default;

IsoscelesTriangle(const double Side, const double angle);

private:

double \_side;

double \_angle;

};

class Factory

{

public:

static ITriangle\* CreateTriangle(const double fSide, const double sSide, const double angle)

};

class Console

{

public:

void Run();

~Console();

private:

ITriangle\*\* \_results = nullptr;

int \_menuItemNumber = -1;

int \_numberTr = 0;

int \_numberTrNow = 0;

static const int \_numPointMenu = 8;

bool \_warMes = true;

const string \_menu[\_numPointMenu] =

{

"Create a triangle",

"Calculate the perimeter of the triangle",

"Calculate the area of the triangle",

"Compare two triangles by perimeter",

"Compare two triangles by area",

"View created triangles",

"Expand an array of triangles",

"Complete the program"

};

const string \_errInput = "Incorrect input. Try again\n\n";

void SetNumtTr();

void Input(double& v);

void CreateTriangle();

bool PrintArrayTr();

void PrintTr(const ITriangle& tr);

void CalculateValue(const int mode);

void CompareTriangle(const int mode);

};

# **4 ДИАГРАММА КЛАССОВ**

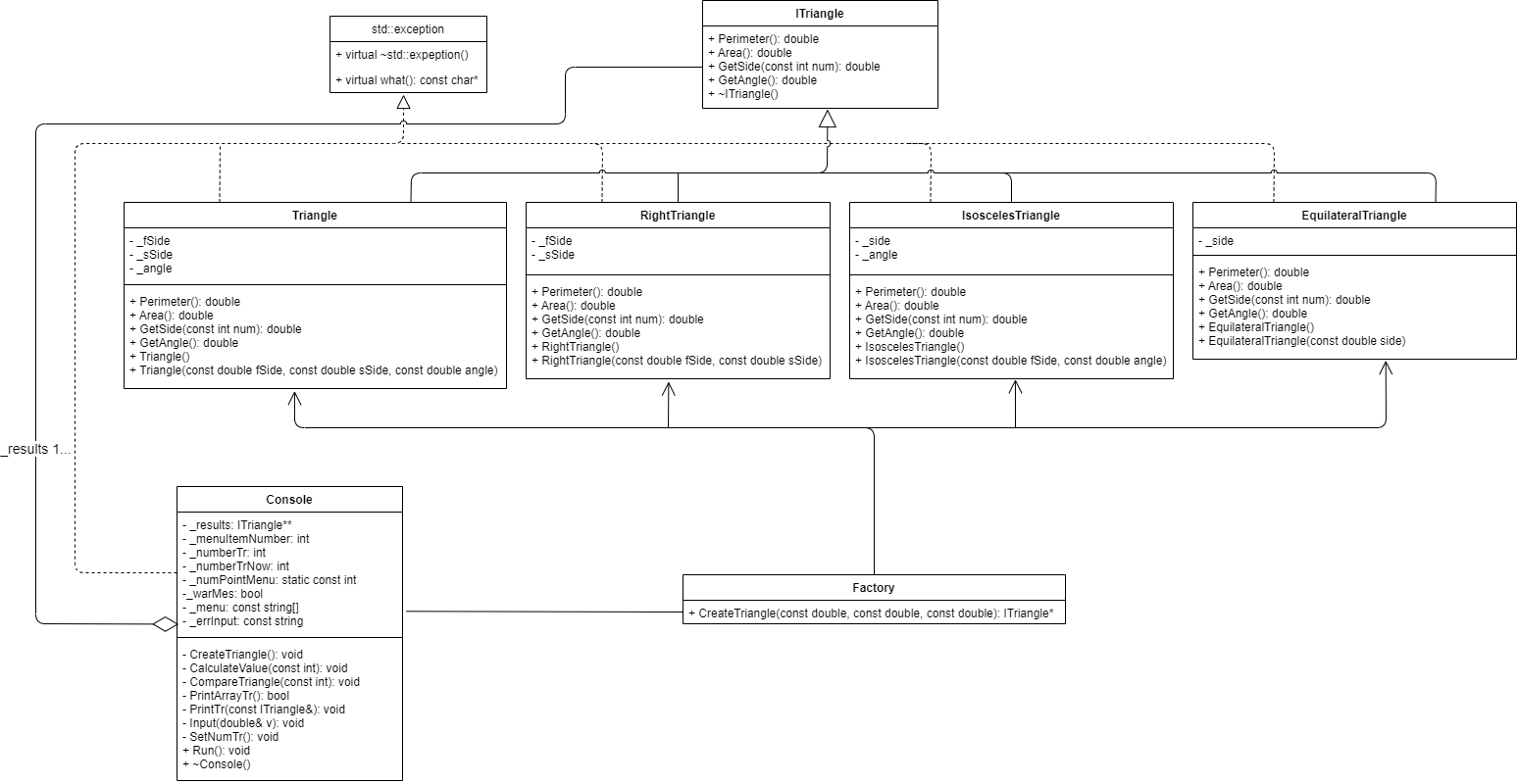
UML диаграмма классов изображена на рисунке 1.

Рисунок 1 – Диаграмма классов

**ЛИСТИНГ ПРОГРАММЫ**

ITriangle.h

#pragma once

#define M\_PI 3.14159265358979323846 //пи

class ITriangle

{

public:

virtual double Perimeter() const = 0; //периметр

virtual double Area() const = 0; //площадь

virtual double GetSide(const int num) const = 0; //получение стороны

virtual double GetAngle() const = 0; //получение угла

virtual ~ITriangle() = default;

};

Triangle.h

#pragma once

#include <algorithm>

#include <cmath>

#include "ITriangle.h"

class Triangle : public ITriangle

{

public:

double Perimeter() const override; //периметр

double Area() const override; //площадь

double GetAngle() const override { return \_angle; } //угол

double GetSide(const int num) const override; //стороны треугольника

Triangle() : \_fSide(0), \_sSide(0), \_angle(0) {}

Triangle(const double fSide, const double sSide, const double angle);

private:

double \_fSide; //первая сторона

double \_sSide; //вторая сторона

double \_angle; //угол между сторонами

};

Triangle.cpp

#include "Triangle.h"

double Triangle::Perimeter() const

{

return \_fSide + \_sSide + GetSide(3);

}

double Triangle::Area() const

{

return \_fSide \* \_sSide \* sin(M\_PI \* \_angle / 180) / 2 ;

}

double Triangle::GetSide(const int num) const

{

switch (num)

{

case 1:

return \_fSide;

case 2:

return \_sSide;

case 3:

return sqrt(\_fSide \* \_fSide + \_sSide \* \_sSide - 2 \* \_fSide \* \_sSide \* cos(M\_PI \* \_angle / 180));

default:

throw std::exception("Incorrect side number");

}

}

Triangle::Triangle(const double fSide, const double sSide, const double angle)

{

if (fSide <= 0 || sSide <= 0)

{

throw std::exception("Impermissible side length");

}

if (angle <= 0 || angle >= 180)

{

throw std::exception("Impermissible angle");

}

\_fSide = fSide;

\_sSide = sSide;

\_angle = angle;

}

RightTriangle.h

#pragma once

#include <algorithm>

#include <cmath>

#include "ITriangle.h"

class RightTriangle : public ITriangle //прямоугольный треугольник

{

public:

virtual double Perimeter() const override;

virtual double Area() const override;

double GetAngle() const override { return 90; }

double GetSide(const int num) const override;

RightTriangle() : \_fSide(0), \_sSide(0) {}

RightTriangle(const double fSide, const double sSide);

private:

double \_fSide; //первый катет

double \_sSide; //второй катет

};

RightTriangle.cpp

#include "RightTriangle.h"

double RightTriangle::Perimeter() const

{

return \_fSide + \_sSide + GetSide(3);

}

double RightTriangle::Area() const

{

return \_fSide \* \_sSide / 2;

}

double RightTriangle::GetSide(const int num) const

{

switch (num)

{

case 1:

return \_fSide;

case 2:

return \_sSide;

case 3:

return sqrt(pow(std::max(\_fSide, \_sSide), 2) - pow(std::min(\_fSide, \_sSide), 2));

default:

throw std::exception("Incorrect side number");

}

}

RightTriangle::RightTriangle(const double fSide, const double sSide)

{

if (fSide <= 0 || sSide <= 0)

{

throw std::exception("Impermissible side length");

}

\_fSide = fSide;

\_sSide = sSide;

}

EquilateralTriangle.h

#pragma once

#include <algorithm>

#include <cmath>

#include "ITriangle.h"

class EquilateralTriangle : public ITriangle //равностороний треугольник

{

public:

double Perimeter() const override;

double Area() const override;

double GetAngle() const override { return 60; }

double GetSide(const int num) const override;

EquilateralTriangle() : \_side(0) {}

EquilateralTriangle(const double side);

private:

double \_side; //сторона

};

EquilateralTriangle.cpp

#include "EquilateralTriangle.h"

double EquilateralTriangle::Perimeter() const

{

return \_side \* 3;

}

double EquilateralTriangle::Area() const

{

return \_side \* \_side \* sqrt(3) / 4;

}

double EquilateralTriangle::GetSide(const int num) const

{

switch (num)

{

case 1:

case 2:

case 3:

return \_side;

default:

throw std::exception("Incorrect side number");

}

}

EquilateralTriangle::EquilateralTriangle(const double side)

{

if (side <= 0)

{

throw std::exception("The triangle isn't equilateral");

}

\_side = side;

}

IsoscelesTriangle.h

#pragma once

#include <algorithm>

#include <cmath>

#include "ITriangle.h"

class IsoscelesTriangle : public ITriangle //равнобедренный треугольник

{

public:

double Perimeter() const override;

double Area() const override;

double GetAngle() const override { return \_angle; }

double GetSide(const int num) const override;

IsoscelesTriangle() = default;

IsoscelesTriangle(const double Side, const double angle);

private:

double \_side;

double \_angle;

};

IsoscelesTriangle.cpp

#include "IsoscelesTriangle.h"

double IsoscelesTriangle::Perimeter() const

{

return \_side \* 2 + GetSide(3);

}

double IsoscelesTriangle::Area() const

{

return \_side \* \_side \* sin(M\_PI \* \_angle / 180) / 2 ;

}

double IsoscelesTriangle::GetSide(const int num) const

{

switch (num)

{

case 1:

case 2:

return \_side;

case 3:

return \_side / (2 \* cos(M\_PI \* \_angle / 180));

default:

throw std::exception("Incorrect side number");

}

}

IsoscelesTriangle::IsoscelesTriangle(const double side, const double angle)

{

if (side <= 0)

{

throw std::exception("Impermissible side length");

}

if (angle <= 0 || angle >= 180)

{

throw std::exception("Impermissible angle");

}

\_side = side;

\_angle = angle;

}

Factory.h

#pragma once

#include "Triangle.h"

#include "EquilateralTriangle.h"

#include "RightTriangle.h"

#include "IsoscelesTriangle.h"

class Factory

{

public:

static ITriangle\* CreateTriangle(const double fSide, const double sSide, const double angle)

{

if (angle == 90)

{

return new RightTriangle(fSide, sSide);

}

if (fSide == sSide)

{

if (angle == 60)

{

return new EquilateralTriangle(fSide);

}

return new IsoscelesTriangle(fSide, angle);

}

return new Triangle(fSide, sSide, angle);

}

};

Console.h

#pragma once

#include <iostream>

#include <string>

#include <algorithm>

#include "Factory.h"

using namespace std;

class Console

{

public:

void Run(); //запуск консоли

~Console()

{

for (int i = 0; i < \_numberTrNow; i++)

{

delete \_results[i];

}

delete[] \_results;

}

private:

ITriangle\*\* \_results = nullptr; //массив треугольников

int \_menuItemNumber = -1; //номер пункта меню

int \_numberTr = 0; //размер массива

int \_numberTrNow = 0; //количество запис. в мас. треугольников

static const int \_numPointMenu = 8; //количество пунктов меню

bool \_warMes = true;

const string \_menu[\_numPointMenu] = // основное меню

{

"Create a triangle",

"Calculate the perimeter of the triangle",

"Calculate the area of the triangle",

"Compare two triangles by perimeter",

"Compare two triangles by area",

"View created triangles",

"Expand an array of triangles",

"Complete the program"

};

const string \_errInput = "Incorrect input. Try again\n\n";

void SetNumtTr(); //установка размера массива

void Input(double& v); //ввод

void Input(int& v);

void CreateTriangle(); //создание треугольника

bool PrintArrayTr(); //вывод массива

void PrintTr(const ITriangle& tr); //вывод треугольника

void CalculateValue(const int mode); //вычисление периметра и площади

//0 - периметр

//1 - площадь

void CompareTriangle(const int mode); //сравнение треугольников по пер. и пло.

};

Console.cpp

#include "Console.h"

void Console::Run()

{

SetNumtTr();

while (\_menuItemNumber != \_numPointMenu)

{

for (int i = 0; i < \_numPointMenu; i++)

{

cout << i + 1 << ". " << \_menu[i] << endl;

}

Input(\_menuItemNumber);

system("cls");

switch (\_menuItemNumber)

{

case 1:

CreateTriangle();

break;

case 2:

CalculateValue(0);

break;

case 3:

CalculateValue(1);

break;

case 4:

CompareTriangle(0);

break;

case 5:

CompareTriangle(1);

break;

case 6:

PrintArrayTr();

break;

case 7:

SetNumtTr();

break;

case 8:

break;

default:

cout << \_errInput;

}

}

}

void Console::SetNumtTr()

{

for (int i = 0; i < \_numberTrNow; i++)

{

delete \_results[i];

}

delete[] \_results;

\_numberTr = 0;

cout << "Enter the array's size to save triangles\n";

while (\_numberTr == 0)

{

Input(\_numberTr);

if (\_numberTr != 0)

{

try

{

\_results = new ITriangle\*[\_numberTr];

/\*for (int i = 0; i < \_numberTr; i++)

{

\_results[i] = nullptr;

}\*/

system("cls");

cout << "An array of size " << \_numberTr << " is created\n\n";

\_numberTrNow = 0;

}

catch (bad\_array\_new\_length)

{

if (\_numberTr < 0)

{

cout << "Error. Enter a number greater than 0\n";

}

else

{

cout << "Error. Array size is too big. Please enter a smaller size\n";

}

\_numberTr = 0;

}

}

else

{

cout << "Enter a size greater than 0\n";

}

}

}

void Console::Input(double& v)

{

cout << "> ";

cin >> v;

if (cin.fail())

{

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

}

}

void Console::Input(int& v)

{

cout << "> ";

cin >> v;

if (cin.fail())

{

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

}

}

void Console::CreateTriangle()

{

double fir;

double sec;

double angle;

cout << "Enter the first side of the triangle\n";

Input(fir);

cout << "Enter the second side of the triangle\n";

Input(sec);

cout << "Enter the angle of the triangle\n";

Input(angle);

system("cls");

if (\_numberTrNow == \_numberTr)

{

cout << "The array is full. A new array will be created\n";

SetNumtTr();

}

try

{

\_results[\_numberTrNow] = Factory::CreateTriangle(fir, sec, angle);

\_numberTrNow++;

cout << "The triangle was created\n\n";

}

catch (exception er)

{

cout << "Error. " << er.what() << "\n\n";

}

}

bool Console::PrintArrayTr()

{

if (\_numberTrNow == 0)

{

cout << "The array is empty\n\n";

return false;

}

cout.width(10);

cout << left << "Number";

cout.width(15);

cout << left << "First Side";

cout.width(15);

cout << left << "Second Side";

cout.width(6);

cout << left << "Angle\n";

for (int i = 0; i < \_numberTrNow; i++)

{

cout.width(10);

string n = to\_string(i + 1) + ')';

cout << left << n;

PrintTr(\*\_results[i]);

cout << '\n';

}

cout << '\n';

return true;

}

void Console::PrintTr(const ITriangle& tr)

{

cout.width(15);

cout << left << tr.GetSide(1);

cout.width(15);

cout << left << tr.GetSide(2);

cout.width(6);

cout << left << tr.GetAngle();

}

void Console::CalculateValue(const int mode)

{

if (!PrintArrayTr()) return;

cout << "Enter the triangle number\n";

int num = 0;

while (num <= 0 || num > \_numberTrNow)

{

Input(num);

if (num <= 0 || num > \_numberTrNow)

cout << \_errInput;

else

{

system("cls");

switch (mode)

{

case 0:

cout << "Perimetr = ";

cout << \_results[num - 1]->Perimeter() << "\n\n";

break;

case 1:

cout << "Area = ";

cout << \_results[num - 1]->Area() << "\n\n";

break;

default:

cout << "Error. This mode doesn't exist\n\n";

}

}

}

}

void Console::CompareTriangle(const int mode)

{

if (!PrintArrayTr()) return;

int num = 0;

double value[2] = { 0, 0 };

for (int i = 0; i < 2; i++)

{

num = 0;

cout << "Enter the number of the " << i+1 << " triangle\n";

while (num <= 0 || num > \_numberTrNow)

{

Input(num);

if (num <= 0 || num > \_numberTrNow)

cout << \_errInput;

else

{

switch (mode)

{

case 0:

value[i] = \_results[num-1]->Perimeter();

break;

case 1:

value[i] = \_results[num-1]->Area();

break;

default:

cout << "Error. This mode doesn't exist\n\n";

}

}

}

}

system("cls");

cout << value[0] << ' ';

if (value[0] == value[1]) cout << '=';

else

if (value[0] > value[1]) cout << '>';

else cout << '<';

cout << ' ' << value[1] << "\n\n";

}

main.cpp

#include <vld.h>

#include "Console.h"

int main()

{

Console con;

con.Run();

return 0;

}