МИНИСТЕРСТВО НАУКИ и высшего образования

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

Федеральное государственное автономное образовательное учреждение

высшего образования

«Санкт-Петербургский политехнический университет Петра Великого»

(ФГАОУ ВО «СПбПУ»)

**Институт среднего профессионального образования**

**Отчёт по лабораторной работе № 2**

**по учебной дисциплине «Системное программирование»**

**Тема: «Реализация классов»**

Выполнил студент

специальности 09.02.07

Информационные системы и

программирование

IV курса группы 42919/4

Давлетшин Артем Рустэмович

Преподаватель

Молькова Лолита Юрьевна

Санкт-Петербург

2023

# **Цели работы:**

* Изучение структуры класса, механизм создания и использования, описание членовданных класса и методов доступа к ним.

**Общее задание:**

Задание 1:

Код заголовочного файла Worker.h:

#include "string"

#pragma once

using namespace std;

class Worker

{

public:

int age;

string name;

string getInfo();

void eat(double weight);

double getWeight();

void walk();

void work();

void dance();

private:

double weight;

int mood = 10;

};

Код файла реализации Worker.cpp:

#include "Worker.h"

#include <string>

using namespace std;

string Worker::getInfo()

{

string age = "Возраст: " + to\_string(this->age) + "\n";

string name = "Имя: " + this->name + "\n";

string mood = "Настроение: " + to\_string(this->mood) + "\n";

string weight = "Вес: " + to\_string(this->weight) + "\n";

return name + age + mood + weight;

}

void Worker::eat(double weight){

if (weight >= 10) {

this->age += 1;

this->weight += weight / 2;

}

else {

this->weight += weight;

}

walk();

walk();

dance();

dance();

dance();

}

double Worker::getWeight(){

return this->weight;

}

void Worker::dance(){

this->mood += 2;

}

void Worker::walk(){

this->mood += 1;

}

void Worker::work(){

if (this->mood - 2 < 0) this->mood = 0;

else this->mood -= 2;

}

# **Индивидуальное задание:**

Задание 2:

Пользователь вводит массив треугольников.

Свойства: две стороны и угол между ними

* Увеличение/уменьшение размера угла (из свойств) в заданно количество раз;
* Вычисление длинны биссектрисы, принадлежащей любому углу;
* Вычисление длин отрезков, на которые биссектриса делит любую сторону;
* Определение значений углов.

Пользователь выбирает элемент массива и выполняемую операцию.

Задание 2:

* Перечислить несколько однотипных объектов и конкретные значений их свойств.
* Выделить классы из группы однотипных объектов

Предметная область: Машиностроение

# **Ход работы**:

Исходя из индивидуального задания создадим класс в диаграмме классов UML. Реализация класса представлена на рисунке 1.



Рисунок 1 – Класс MyTriangle

Реализуем класс Triangle. Вынесем сигнатуры методов и поля в заголовочный файл Triangle.h. Реализуем логику в файле Triangle.cpp.

Код заголовочного файла:

class Triangle {

private:

double sideA;

double sideB;

double sideC;

void getSides(int angleNumber, double sides[3]);

public:

Triangle(double sideA, double sideB, double angle);

void increaseAngle(int angleNumber, double deltaAngle);

double getAngle(int angleNumber);

double getBisectorLength(int angleNumber);

void getLengthOfSegmentsDividedByBisectors(int angleNumber, double result[2]);

string toString();

};

Код файла реализации:

#define \_USE\_MATH\_DEFINES

#include "Triangle.h"

#include <iostream>

#include <cmath>

#include <unordered\_map>

#include <array>

Triangle::Triangle(double sideA, double sideB, double angle) {

this->sideA = sideA;

this->sideB = sideB;

double radianAngle = angle \* M\_PI / 180.0;

this->sideC = sqrt(pow(this->sideA, 2) + pow(this->sideB, 2) - 2 \* this->sideB \* this->sideA \* cos(radianAngle));

}

double Triangle::getAngle(int angleNumber) {

double p = -1;

switch (angleNumber) {

case 0:

p = (sideB \* sideB - sideC \* sideC + sideA \* sideA) / (2.0 \* sideA \* sideB);

return acos(p - static\_cast<int>(p));

case 1:

p = (sideB \* sideB + sideC \* sideC - sideA \* sideA) / (2.0 \* sideC \* sideB);

return acos(p - static\_cast<int>(p));

case 2:

p = (-sideB \* sideB + sideC \* sideC + sideA \* sideA) / (2.0 \* sideA \* sideC);

return acos(p - static\_cast<int>(p));

default:

return p;

}

}

void Triangle::increaseAngle(int angleNumber, double deltaAngle) {

double sides[3];

getSides(angleNumber, sides);

double currentAngle = getAngle(angleNumber);

double newAngle = currentAngle + deltaAngle;

int newSide = sqrt(pow(sides[1], 2) + pow(sides[2], 2) - 2 \* sides[1] \* sides[2] \* cos(newAngle));

if (newSide <= 0) {

std::cout << "Invalid angle increase. New side length is non-positive." << std::endl;

return;

}

switch (angleNumber) {

case 0:

this->sideC = newSide;

break;

case 1:

this->sideA = newSide;

break;

case 2:

this->sideB = newSide;

break;

}

}

double Triangle::getBisectorLength(int angleNumber) {

double sides[3];

double angle = getAngle(angleNumber);

getSides(angleNumber, sides);

return 2 \* sides[1] \* sides[2] \* cos(angle / 2.0) / (sides[1] + sides[2]);

}

void Triangle::getSides(int angleNumber, double sides[3]) {

switch (angleNumber) {

case 0:

sides[0] = this->sideC;

sides[1] = this->sideA;

sides[2] = this->sideB;

break;

case 1:

sides[0] = this->sideA;

sides[1] = this->sideB;

sides[2] = this->sideC;

break;

case 2:

sides[0] = this->sideB;

sides[1] = this->sideC;

sides[2] = this->sideA;

break;

}

}

void Triangle::getLengthOfSegmentsDividedByBisectors(int angleNumber, double result[2]) {

double sides[3];

getSides(angleNumber, sides);

result[0] = (sides[2] \* sides[0]) / (sides[1] + sides[2]);

result[1] = (sides[1] \* sides[0]) / (sides[1] + sides[2]);

}

std::string Triangle::toString() {

return "Стороны треугольника: " + std::to\_string(sideA) + " см, " + std::to\_string(sideB) + " см, " + std::to\_string(sideC) + " см" +

"\nУглы треугольника: " +

"alpha=" + std::to\_string(getAngle(0) \* 180 / M\_PI) + "°, " +

"beta=" + std::to\_string(getAngle(1) \* 180 / M\_PI) + "°, " +

"gamma=" + std::to\_string(getAngle(2) \* 180 / M\_PI) + "°";

}

Создадим консольное приложение для работы с классом. Код консольного приложения:

int main()

{

setlocale(LC\_ALL, "Ru");

int choice;

double deltaAngle;

double bisectorLength;

double result[2];

int selectedTriangleIndex = 0;

vector<Triangle> triangles;

int numTriangles;

std::cout << "Введите количество треугольников: ";

std::cin >> numTriangles;

for (int i = 0; i < numTriangles; i++) {

double side1, side2, angle;

cout << "Введите сторону 1 для треугольника " << (i + 1) << ": ";

cin >> side1;

cout << "Введите сторону 2 для треугольника " << (i + 1) << ": ";

cin >> side2;

cout << "Введите угол между сторонами для треугольника " << (i + 1) << ": ";

cin >> angle;

triangles.push\_back(Triangle(side1, side2, angle));

}

//triangles.push\_back(Triangle(6, 7, 20));

//triangles.push\_back(Triangle(8, 9, 30));

//triangles.push\_back(Triangle(10, 11, 45));

while (true) {

cout << "Выберите треугольник (0 - " << (triangles.size() - 1) << "): ";

cin >> selectedTriangleIndex;

if (selectedTriangleIndex < 0 || selectedTriangleIndex >= triangles.size()) {

std::cout << "Неверный выбор треугольника. Попробуйте еще раз." << std::endl;

continue;

}

cout << "Выберите действие:" << endl;

cout << "1. Вывести информацию о треугольнике" << endl;

cout << "2. Увеличить один из углов" << endl;

cout << "3. Получить длину биссектрисы" << endl;

cout << "4. Получить длины отрезков, разделенных биссектрисой" << endl;

cout << "5. Выход" << endl;

cin >> choice;

switch (choice) {

case 1:

cout << triangles[selectedTriangleIndex].toString() << endl;

break;

case 2:

int angleNumber;

cout << "Выберите номер угла (0, 1 или 2): ";

cin >> angleNumber;

cout << "Введите на сколько градусов увеличить угол: ";

cin >> deltaAngle;

triangles[selectedTriangleIndex].increaseAngle(angleNumber, deltaAngle);

break;

case 3:

cout << "Введите номер угла (0, 1 или 2): ";

cin >> angleNumber;

bisectorLength = triangles[selectedTriangleIndex].getBisectorLength(angleNumber);

cout << "Длина биссектрисы: " << bisectorLength << " см" << endl;

break;

case 4:

cout << "Введите номер угла (0, 1 или 2): ";

cin >> angleNumber;

triangles[selectedTriangleIndex].getLengthOfSegmentsDividedByBisectors(angleNumber, result);

cout << "Длины отрезков, разделенных биссектрисой: " << result[0] << " см и " << result[1] << " см" << endl;

break;

case 5:

break;

default:

cout << "Неверный выбор. Попробуйте еще раз." << endl;

}

if (choice == 5)

break;

}

return 0;

}

Протестируем работу приложения.

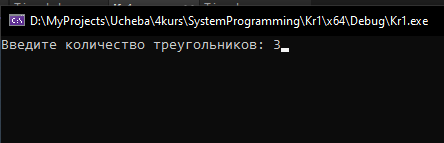


Рисунок 2 – Ввод числа треугольников

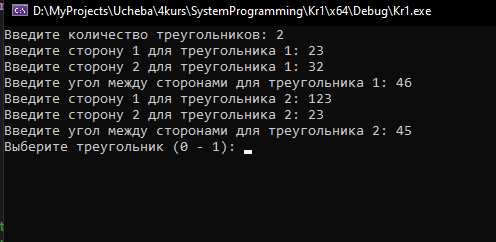


Рисунок 3 – Введенные треугольники и интерфейс выбора

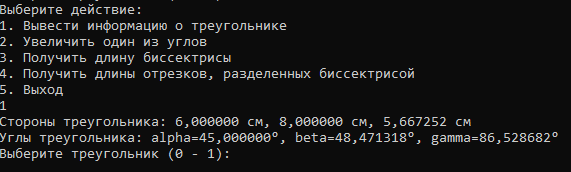


Рисунок 5 – Меню и вывод информации о треугольнике

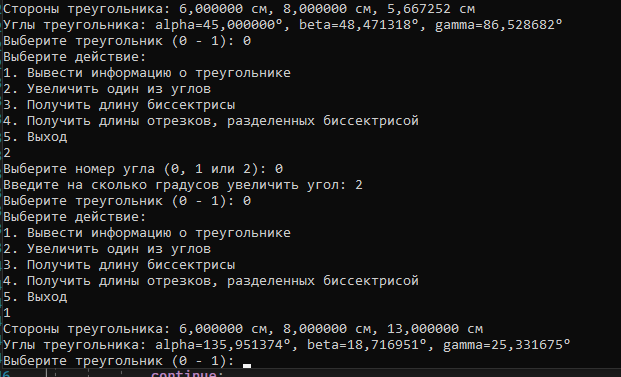


Рисунок 6 – Увеличение угла треугольника

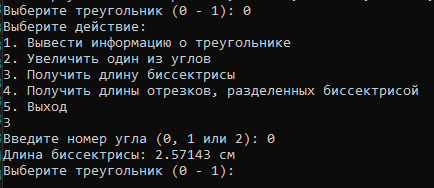


Рисунок 7 – Получение биссектрисы треугольника

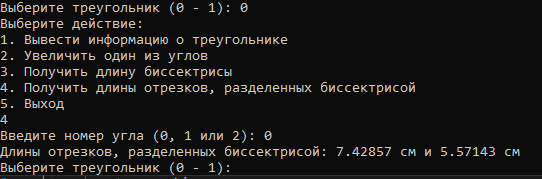


Рисунок 8 – Получение длин отрезков, разделенных биссектрисой

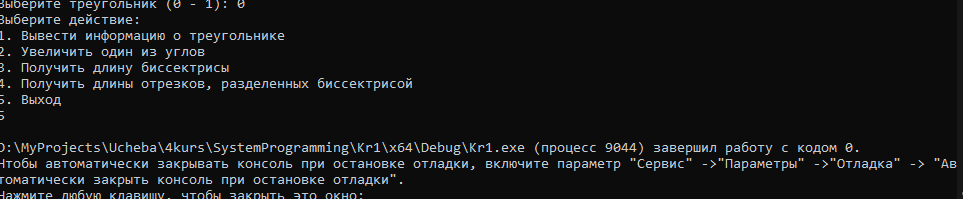


Рисунок 9 – Выход из программы

**Контрольные вопросы:**

**Вопрос 1**

В определении класса программы члены класса с ключевым словом private доступны методам этого класса.

**Вопрос 2**

Класс - шаблон или определение, описывающее структуру и поведение объектов, которые будут созданы на его основе. Класс определяет атрибуты (поля) и методы (функции), которые объекты этого класса могут содержать и выполнять.

**Вопрос 3**

class StudentGroup {

private:

int number;

public:

void add();

};

**Вопрос 4**

То, что поля класса должны быть закрытыми истинно.

**Вопрос5**

При работе с объектами применяется стрелочка при вызове методов и обращении к полям через указатель.

**Вопрос 6**

Методу класса всегда доступны данные класса, членом которого он является.

**Вопрос 7:**

Единственное формальное различие между структурой и классом состоит в том что в первом случае члены по умолчанию имеют модификатор доступа public, а во втором - private.

**Вопрос 8:**

Каждый объект хранит свою собственную копию данных, но все объекты одного класса будут использовать одинаковый набор методов класса.

**Вопрос 9:**

Необходимо переопределять операции new и delete для оптимизации и упрощения работы с памятью.

**Вопрос 10:**

Указатель This означает обращение к объекту – владельцу метода.