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 № 1**

**по учебной дисциплине «Системное программирование»**

**Тема: «**Проектирование классов**»**

Выполнил(а) студент(ка)

специальности 09.02.03

Информационные системы  
 и программирование

4 курса группы 42919/3

Волков Артемий Вячеславович

Преподаватель

Молькова Лолита Юрьевна

Санкт-Петербург,

2023

**Цели:**

1. Ознакомление с процессом классификации на примере моделирования классов предметной области.
2. Ознакомление с диаграммами классов и взаимодействия UML

**Практическое задание**

* 1. **Индивидуальное задание (50%)**

• Реализовать массив данных в соответствии с вариантом задания.

• Определить метод установки свойств (при недопустимых аргументах функции возвращать «false» и выдавать текст ошибки на экран).

• Написать демонстрационную программу, в которой показать работу с переменными.

Вариант 5.

Пользователь вводит массив Треугольников Свойства: сторона и два прилежащих к ней угла Операции:

• уменьшение/увеличение размера стороны (из свойств) в заданное кол-во раз;

• вычисление длины медианы, принадлежащие любой стороне;

• определение подобен ли треугольник данному (указанному по индексу массива);

• определение значений сторон.

Пользователь выбирает элемент массива и выполняемую операцию.

**Ход работы:**

Создание класса представляющий треугольник:

class Triangle

{

private:

double PI = 3.141592653589793;

double ab;

double bc;

double ac;

double A;

double B;

double C;

void recalculateAngles() {

// Расчет углов на основе измененных сторон

A = acos((ab \* ab + ac \* ac - bc \* bc) / (2 \* ab \* ac)) \* (180.0 / PI);

B = acos((ac \* ac + bc \* bc - ab \* ab) / (2 \* ac \* bc)) \* (180.0 / PI);

C = acos((ab \* ab + bc \* bc - ac \* ac) / (2 \* ab \* bc)) \* (180.0 / PI);

}

//подгон сторон

void recalculateSides(const string& changedSide) {

if (changedSide == "ab") {

bc = sqrt(ac \* ac + ab \* ab - 2 \* ac \* ab \* cos(C \* PI / 180));

}

else if (changedSide == "bc") {

ab = sqrt(ac \* ac + bc \* bc - 2 \* ac \* bc \* cos(B \* PI / 180));

}

else if (changedSide == "ac") {

ab = sqrt(bc \* bc + ac \* ac - 2 \* bc \* ac \* cos(A \* PI / 180));

}

}

void changeSideByAngle(const string& angleName) {

if (angleName == "A") {

//A = newAngleValue;

ab = sqrt(ac \* ac + bc \* bc - 2 \* ac \* bc \* cos(C \* PI / 180));

}

else if (angleName == "B") {

//B = newAngleValue;

bc = sqrt(ab \* ab + ac \* ac - 2 \* ab \* ac \* cos(A \* PI / 180));

}

else if (angleName == "C") {

//C = newAngleValue;

ac = sqrt(ab \* ab + bc \* bc - 2 \* ab \* bc \* cos(B \* PI / 180));

}

}

bool areAnglesValid(double newA, double newB, double newC) {

// Проверка, что сумма углов равна 180 градусов

return (newA + newB + newC) == 180;

}

bool isTriangleValid(double ab, double bc, double ac) {

// Проверка на неравенство треугольника

return (ab + bc > ac) && (ab + ac > bc) && (ac + bc > ab);

}

/// <summary>

/// Присваивает переданное значение стороне ab если это возможно

/// </summary>

/// <param name="ab">Новое значение ab</param>

/// <returns>true - если изменения возможны, false в противном случае</returns>

bool setSideAB(double ab) {

if (isTriangleValid(ab, this->bc, this->ac))

{

this->ab = ab;

recalculateSides("ab");

recalculateAngles();

}

else return false;

return true;

}

/// <summary>

/// Присваивает переданное значение стороне bc если это возможно

/// </summary>

/// <param name="bc">Новое значение bc</param>

/// <returns>true - если изменения возможны, false в противном случае</returns>

bool setSideBC(double bc) {

if (isTriangleValid(this->ab, bc, this->ac))

{

this->bc = bc;

recalculateSides("bc");

recalculateAngles();

}

else return false;

return true;

}

/// <summary>

/// Присваивает переданное значение стороне ac если это возможно

/// </summary>

/// <param name="ac">Новое значение ac</param>

/// <returns>true - если изменения возможны, false в противном случае</returns>

bool setSideAC(double ac) {

if (isTriangleValid(this->ab, this->bc, ac))

{

this->ac = ac;

recalculateSides("ac");

recalculateAngles();

}

else return false;

return true;

}

public:

Triangle()

{

}

Triangle(double ab, double bc, double ac)

{

if (isTriangleValid(ab, bc, ac))

{

this->ab = ab;

this->ac = ac;

this->bc = bc;

}

else throw new exception("Существования такого треугольника невозможно!");

recalculateAngles();

}

/// <summary>

/// Вычисление медианы

/// </summary>

/// <param name="sideName">Сторона к которой требуется найти значение медианы</param>

/// <returns>Значение медианы или -1 в случае передачи некорректного параметра</returns>

double calculateMedian(const string& sideName) {

if (sideName == "ab") {

return 0.5 \* sqrt(2 \* (bc \* bc + ac \* ac) - ab \* ab);

}

else if (sideName == "bc") {

return 0.5 \* sqrt(2 \* (ab \* ab + ac \* ac) - bc \* bc);

}

else if (sideName == "ac") {

return 0.5 \* sqrt(2 \* (ab \* ab + bc \* bc) - ac \* ac);

}

else {

return -1;

}

}

//Проверка на подобность двух треугольников

static bool areTrianglesSimilar(Triangle& triangle1, Triangle& triangle2) {

// Проверка соотношения длин сторон

double ratio1 = triangle1.ab / triangle2.ab;

double ratio2 = triangle1.bc / triangle2.bc;

double ratio3 = triangle1.ac / triangle2.ac;

// Проверка, что соотношение длин сторон одинаково

return (ratio1 == ratio2) && (ratio2 == ratio3);

}

/// <summary>

/// Возврат стороны AB и двух прилежащих углов

/// </summary>

/// <returns>Сторона AB, угол A, угол B</returns>

map<string, double> getABandCorners()

{

map<string, double> dictionary;

dictionary["AB"] = ab;

dictionary["A"] = A;

dictionary["B"] = B;

return dictionary;

}

/// <summary>

/// Возврат стороны AC и двух прилежащих углов

/// </summary>

/// <returns>Сторона AC, угол A, угол C</returns>

map<string, double> getACandCorners()

{

map<string, double> dictionary;

dictionary["AC"] = ac;

dictionary["A"] = A;

dictionary["C"] = C;

return dictionary;

}

/// <summary>

/// Возврат стороны BC и двух прилежащих углов

/// </summary>

/// <returns>Сторона BC, угол B, угол C</returns>

map<string, double> getBCandCorners()

{

map<string, double> dictionary;

dictionary["BC"] = bc;

dictionary["B"] = B;

dictionary["C"] = C;

return dictionary;

}

/// <summary>

/// Увеличивает или уменьшает значение поля AB в несколько раз

/// </summary>

/// <param name="pow">Во сколько раз уменьшается или увеличивается поле</param>

/// <param name="numberOperation">1 - увеличить поле, 2 - уменьшить поле</param>

/// <returns>Если геометрически возможно существование фигуры с новым значением стороны, то действие было произведено - true, в противном случае - false</returns>

bool setSideABpow(double pow, int numberOperation)

{

if (numberOperation == 1) return setSideAB(ab \* pow);

else return setSideAB(ab / pow);

}

/// <summary>

/// Увеличивает или уменьшает значение поля AС в несколько раз

/// </summary>

/// <param name="pow">Во сколько раз уменьшается или увеличивается поле</param>

/// <param name="numberOperation">1 - увеличить поле, 2 - уменьшить поле</param>

/// <returns>Если геометрически возможно существование фигуры с новым значением стороны, то действие было произведено - true, в противном случае - false</returns>

bool setSideACpow(double pow, int numberOperation)

{

if (numberOperation == 1) return setSideAC(ac \* pow);

else return setSideAC(ac / pow);

}

/// <summary>

/// Увеличивает или уменьшает значение поля BC в несколько раз

/// </summary>

/// <param name="pow">Во сколько раз уменьшается или увеличивается поле</param>

/// <param name="numberOperation">1 - увеличить поле, 2 - уменьшить поле</param>

/// <returns>Если геометрически возможно существование фигуры с новым значением стороны, то действие было произведено - true, в противном случае - false</returns>

bool setSideBCpow(double pow, int numberOperation)

{

if (numberOperation == 1) return setSideBC(bc \* pow);

else return setSideBC(bc / pow);

}

double getAB()

{

return ab;

}

double getAC()

{

return ac;

}

double getBC()

{

return bc;

}

double getA()

{

return A;

}

double getB()

{

return B;

}

double getC()

{

return C;

}

};

Код главной функции main для работы с консолью:

void clearBufCin(string text)

{

cout << text + "\r\nВведите значения заново." << endl;

cin.clear();

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

}

void getCommands()

{

cout << "-------------------------------------------------------------------------------------" << endl;

cout << "1 - уменьшение/увеличение размера стороны (из свойств) в заданное кол-во раз;" << endl <<

"2 - вычисление длины медианы, принадлежащей любой стороне;" << endl

<< "3 - определение подобен ли другой треугольник данному (указанному по индексу массива);" << endl <<

"4 - определение значений сторон" << endl << "5 - выбрать треугольник;" << endl << "6 - посмотреть команды;" << endl << "7 - завершить программу." << endl;

cout << "-------------------------------------------------------------------------------------" << endl;

}

int chooseTriangle(Triangle triangles[], int size)

{

cout << "-----------------------------------------------------------" << endl;

cout << "Выберите треугольник с которым будут производиться действия" << endl;

int number;

while (true)

{

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

{

cout << "Треугольник " << i+1 << ": AB - " << triangles[i].getAB() << ", AC - " << triangles[i].getAC() << ", BC - " << triangles[i].getBC() << endl;

}

cin >> number;

if (cin.fail() || number > size || number < 1)

{

cout << "Некорректный ввод, введите значение еще раз." << endl;

cin.clear();

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

continue;

}

return number - 1;

}

}

int main()

{

setlocale(LC\_ALL, "Russian");

int index = 0;

Triangle\* triangles;

int count = 1;

while(true)

{

cout << "Укажите кол-во треугольников которое требуется ввести" << endl;

cin >> count;

if (count < 1 || cin.fail())

{

cout << "Указано некорректное значение" << endl;

cin.clear();

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

continue;

}

triangles = new Triangle[count];

break;

}

while (index < count)

{

int ab;

int ac;

int bc;

cout << "Введите значение треугольника " << index+1 << " :" << endl << endl;

while (true)

{

cout << "Введите значение стороны ab" << endl;

cin >> ab;

if (cin.fail())

{

clearBufCin("Некорректно введено значение стороны ab");

continue;

}

break;

}

while (true)

{

cout << "Введите значение стороны ac" << endl;

cin >> ac;

if (cin.fail())

{

clearBufCin("Некорректно введено значение стороны ac");

continue;

}

break;

}

while (true)

{

cout << "Введите значение стороны bc" << endl;

cin >> bc;

if (cin.fail())

{

clearBufCin("Некорректно введено значение стороны bc");

continue;

}

break;

}

if (ab + bc < ac || ab + ac < bc || ac + bc < ab)

{

cout << "Такой треугольник не может существовать!\r\nВведите значения заново" << endl;

cin.clear();

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

continue;

}

triangles[index] = Triangle(ab, bc, ac);

index += 1;

}

index = chooseTriangle(triangles, count);

getCommands();

int command;

while (true)

{

cin >> command;

if (cin.fail())

{

cout << "Команда введена некорректно, введите номер ещё раз." << endl;

cin.clear();

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

continue;

}

switch (command)

{

case 1:

cout << "--------------------" << endl;

int operation; //номер операции умножить - 1, разделить - 2

int side; //номер стороны с которой производится операция

while (true)

{

cout << "Укажите действие:" << endl << "1 - увеличить значение стороны." << endl << "2 - уменьшить значение стороны." << endl;

cin >> operation;

if (cin.fail() || operation < 1 || operation > 2)

{

cout << "Операция указана некорректно, введите номер ещё раз." << endl;

cin.clear();

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

continue;

}

break;

}

while (true)

{

cout << "Выберите сторону треугольника с которой требуется провести операцию." << endl;

cout << "Треугольник " << index+1 << endl << "1: AB - " << triangles[index].getAB() << ";" << endl << "2: AC - " << triangles[index].getAC() << ";" << endl << "3: BC - " << triangles[index].getBC() << "." << endl;

cin >> side;

if (cin.fail() || side < 1 || side > 3)

{

cout << "Номер указан некорректно, введите значение ещё раз." << endl;

cin.clear();

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

continue;

}

break;

}

while (true)

{

double value;

if (operation == 1) cout << "Укажите во сколько раз должна быть увеличена сторона." << endl;

else cout << "Укажите во сколько раз должна быть уменьшена сторона." << endl;

cin >> value;

if (cin.fail())

{

cout << "Значение указано некорректно, введите число ещё раз." << endl;

cin.clear();

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

continue;

}

bool check;

if (side == 1)

{

check = triangles[index].setSideABpow(value, operation);

}

else if (side == 2)

{

check = triangles[index].setSideACpow(value, operation);

}

else

{

check = triangles[index].setSideBCpow(value, operation);

}

if (check)

{

cout << "-----------------" << endl;

cout << "Сторона изменена!" << endl;

cout << "Треугольник " << index + 1 << endl << "1: AB - " << triangles[index].getAB() << ";" << endl << "2: AC - " << triangles[index].getAC() << ";" << endl << "3: BC - " << triangles[index].getBC() << "." << endl;

}

else

{

string sideStr;

if (side == 1) sideStr = "AB";

else if(side == 2) sideStr = "AC";

else sideStr = "BC";

if(operation == 1) cout << "Невозможно увеличить сторону " << sideStr << " на указанное значение (" << value << "). Такой треугольник не может существовать!" << endl;

else cout << "Невозможно уменьшить сторону " << sideStr << " на указанное значение (" << value << "). Такой треугольник не может существовать!" << endl;

continue;

}

getCommands();

break;

}

break;

case 2:

cout << "Треугольник " << index + 1 << endl <<

"1: AB - " << triangles[index].getAB() << "," << " Медиана AB - " << triangles[index].calculateMedian("аb") << endl <<

"2: AC - " << triangles[index].getAC() << "," << " Медиана AС - " << triangles[index].calculateMedian("ас") << endl <<

"3: BC - " << triangles[index].getBC() << "," << " Медиана BC - " << triangles[index].calculateMedian("bc") << endl;

getCommands();

break;

case 3:

cout << "Выберите треугольник для сравнения из ниже приведенных:" << endl;

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

{

if (i == index) continue;

cout << "----------------" << endl;

cout << "Треугольник " << i + 1 << endl <<

"1: AB - " << triangles[index].getAB() << ", " << "A - " << triangles[index].getA() << ";" << endl <<

"2: AC - " << triangles[index].getAC() << ", " << "B - " << triangles[index].getB() << ";" << endl <<

"3: BC - " << triangles[index].getBC() << ", " << "C - " << triangles[index].getC() << "." << endl;

cout << "----------------" << endl;

}

while (true)

{

int numberTriangle;

cin >> numberTriangle;

if (cin.fail() || numberTriangle < 1 || numberTriangle > count || numberTriangle == index + 1)

{

cout << "Значение указано некорректно, введите номер ещё раз." << endl;

cin.clear();

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

continue;

}

//numberTriangle = -1;

if (Triangle::areTrianglesSimilar(triangles[index], triangles[numberTriangle - 1]))

cout << "Треугольник " << index + 1 << " подобен треугольнику " << numberTriangle << endl;

else cout << "Треугольник " << index + 1 << " не подобен треугольнику " << numberTriangle << endl;

cout << "--------------------------------------------" << endl;

cout << "Треугольник " << index + 1 << endl <<

"1: AB - " << triangles[index].getAB() << ", " << "A - " << triangles[index].getA() << ";" << endl <<

"2: AC - " << triangles[index].getAC() << ", " << "B - " << triangles[index].getB() << ";" << endl <<

"3: BC - " << triangles[index].getBC() << ", " << "C - " << triangles[index].getC() << "." << endl;

cout << "--------------------------------------------" << endl;

cout << "Треугольник " << numberTriangle << endl <<

"1: AB - " << triangles[numberTriangle - 1].getAB() << ", " << "A - " << triangles[numberTriangle - 1].getA() << ";" << endl <<

"2: AC - " << triangles[numberTriangle - 1].getAC() << ", " << "B - " << triangles[numberTriangle - 1].getB() << ";" << endl <<

"3: BC - " << triangles[numberTriangle - 1].getBC() << ", " << "C - " << triangles[numberTriangle - 1].getC() << "." << endl;

break;

}

getCommands();

break;

case 4:

cout << "Значения сторон и углов треугольника " << index + 1 << endl <<

"1: AB - " << triangles[index].getAB() << ", " << "A - " << triangles[index].getA() << ";" << endl <<

"2: AC - " << triangles[index].getAC() << ", " << "B - " << triangles[index].getB() << ";" << endl <<

"3: BC - " << triangles[index].getBC() << ", " << "C - " << triangles[index].getC() << "." << endl;

getCommands();

break;

case 5:

index = chooseTriangle(triangles, count);

getCommands();

break;

case 6:

getCommands();

break;

case 7:

return 0;

break;

default:

cout << "Такой команды не существует!" << endl;

break;

}

cin.clear();

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

}

return 0;

}

Протокол испытаний:

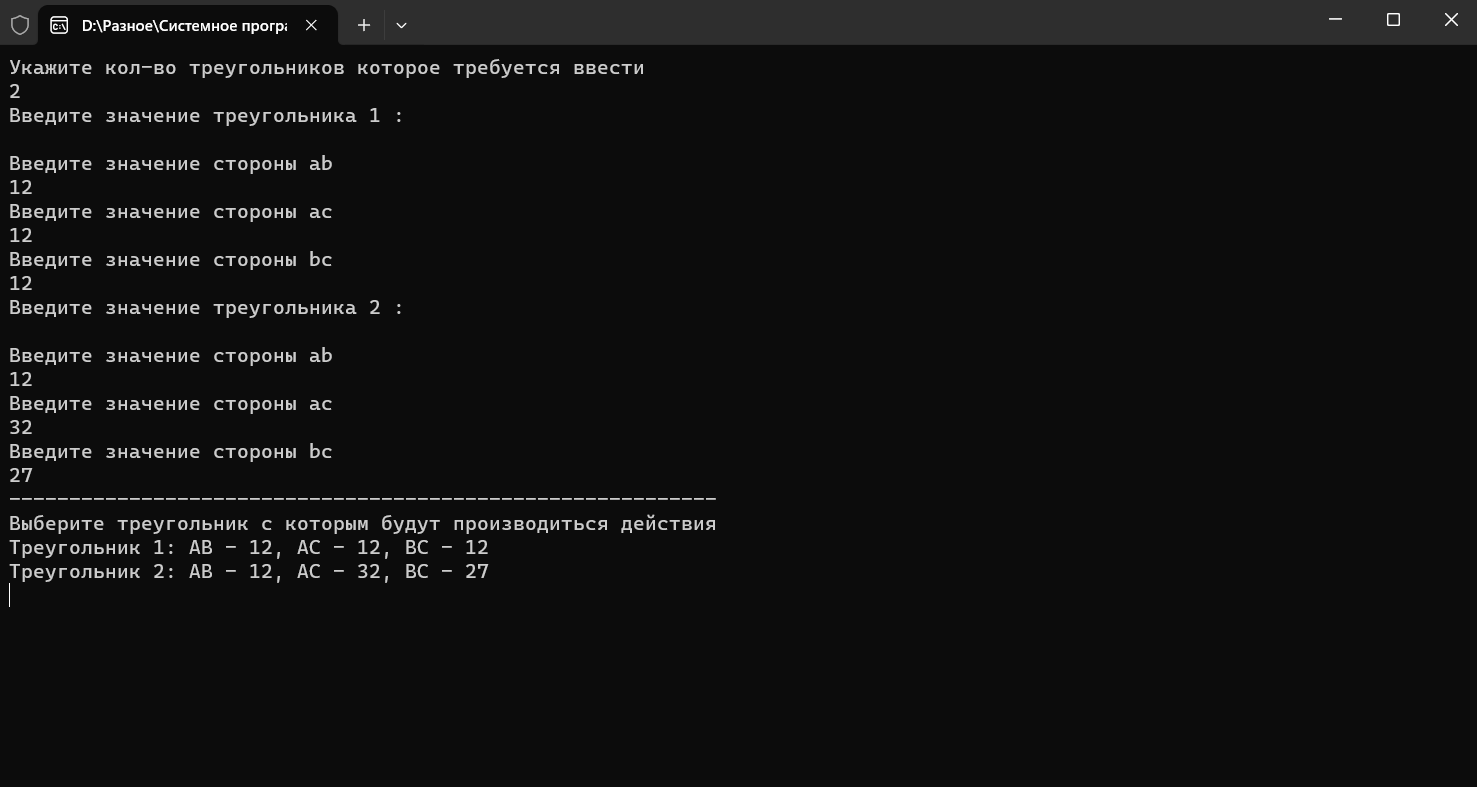


Рисунок 1 –ввод значений сторон треугольников.

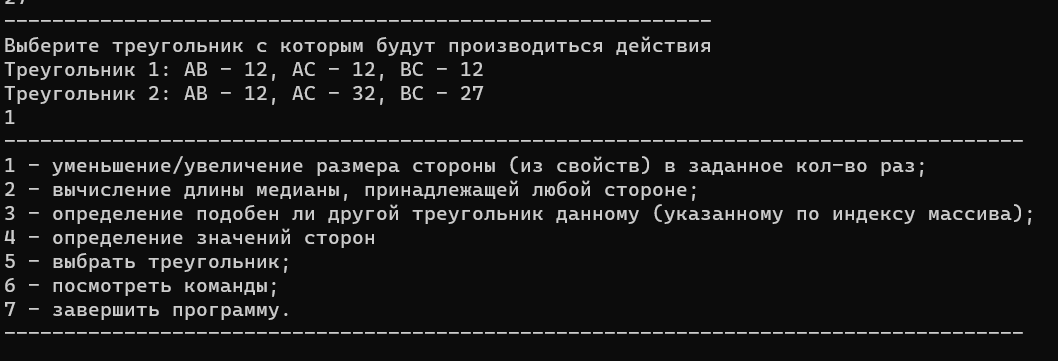


Рисунок 2 –выбор треугольник для операций.

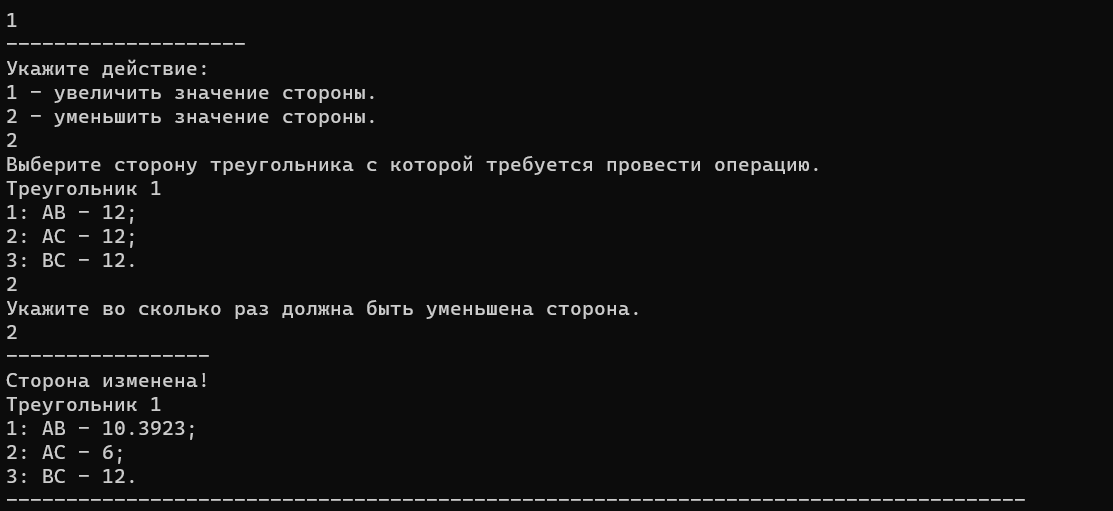


Рисунок 3 – уменьшение значения стороны треугольника в два раза.



Рисунок 4 – вычисление медианы для треугольника.

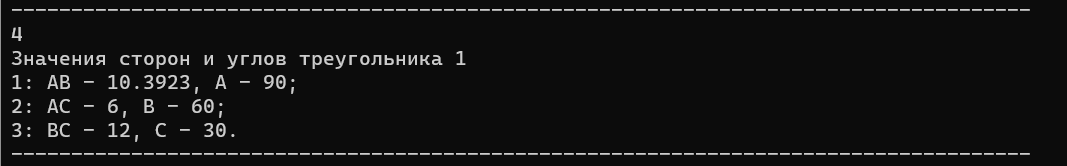


Рисунок 5 –вывод значений сторон и углов треугольника.