**МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ**

**федеральное государственное автономное образовательное**

**учреждение высшего образования**

**«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»**

**(РУТ(МИИТ)**

Кафедра «Вычислительные системы, сети и информационная безопасность»

**Отчет По Дисциплине**

**«Алгоритмизация и программирование»**

*Направление:* 10.03.01*Информационная безопасность*

*Профиль:**Безопасность компьютерных систем*

Выполнила:  
студент группы УИБ-112

Орлов Андрей Витальевич

Проверил:

Старший преподаватель Цыганова Наталия Алексеевна

(должность, ФИО)

Старший преподаватель Никольская Мария Николаевна

(должность, ФИО)

Москва 2021 г.

**Задания №1–6**

Задание: выполнить все задания варианта 21.

1. Таблица имён:

|  |  |  |
| --- | --- | --- |
| Таблица имён для конструктора triangle по умолчанию | | |
| Исходные переменные | | |
| side | вещественный | массив сторон треугольника |
| xy | вещественный | массив координат треугольника |
| kind\_of\_triangle | строковый | переменная, указывающая вид треуг. |
| id | целочисленный | индекс треугольника |
| coun | целочисленный | количество треугольников в программе |
| Рабочие переменные | | |
| i | целочисленный | Переменная-счётчик в циклах |
| side[i] | вещественный | Сторона треугольника i |

|  |
| --- |
| Таблица имён для конструктора triangle с параметрами |
| Исходные переменные |
| Таблица имён совпадает с указанной выше. |

|  |
| --- |
| Таблица имён для функции перегрузки оператора присваивания |
| Исходные переменные |
| Таблица имён совпадает с указанной выше. |

|  |  |  |
| --- | --- | --- |
| Таблица имён для функции перегрузки оператора сложения | | |
| Исходные переменные | | |
| temp | triangle | Переменная для временного пользования |

|  |
| --- |
| Таблица имён для метода определения прямоугольного треугольника |
| Таблица имён совпадает с таблицей имён конструктора класса. |

|  |
| --- |
| Таблица имён для метода определения равнобедренного треугольника |
| Таблица имён совпадает с таблицей имён конструктора класса. |

|  |
| --- |
| Таблица имён для определения правильного треугольника |
| Таблица имён совпадает с таблицей имён конструктора класса |

|  |
| --- |
| Таблица имён для определения несуществующего треугольника |
| Таблица имён совпадает с таблицей имён конструктора класса |

|  |
| --- |
| Таблица имён для определения вида треугольника |
| Переменных нет. |

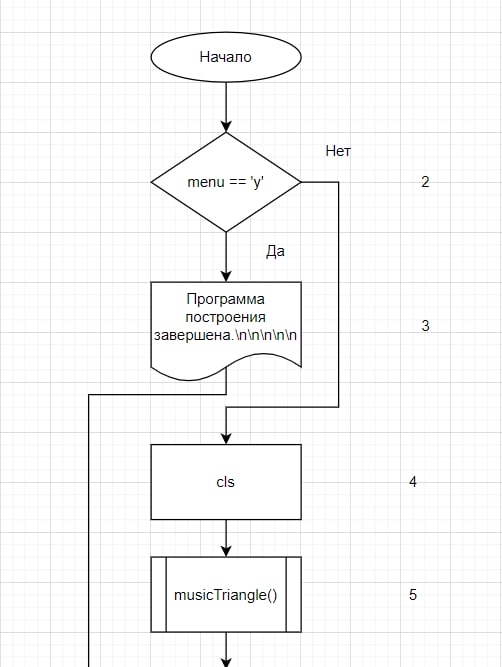
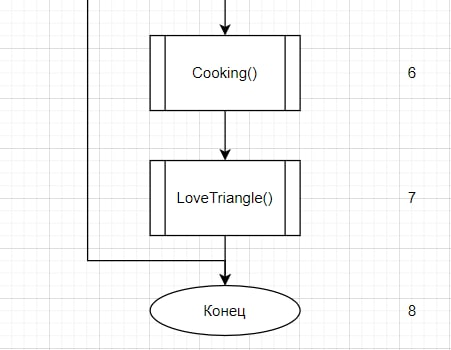
|  |
| --- |
| Таблица имён для основной программы |
| Переменных нет. |

|  |  |  |
| --- | --- | --- |
| Таблица имён для функции menu | | |
| Исходные переменные | | |
| counter | целочисленный | Количество треугольников в программе |
| my\_triangle | массив triangle | Массив треугольников в программе |
| fate | целочисленный | Переменная решения тестировщика |
| amount | целочисленный | Переменная решения тестировщика |
| interNumber | целочисленный | Переменная решения тестировщика |
| Рабочие переменные | | |
| i | целочисленный | Переменная-счётчик в циклах |
| a | целочисленный дин. массив | Массив с индексами треугольников, которые пользователь хочет сложить |

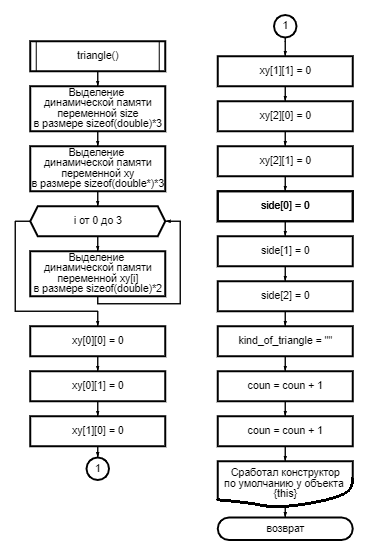
|  |  |  |
| --- | --- | --- |
| Таблица имён для функции перегрузки оператора ввода | | |
| Исходные переменные | | |
| x1, x2, x3, y1, y2, y3 | целочисленные | Координаты точек, на которых строится треугольник |
| interDecision | целочисленный | Переменная решения тестировщика |
| count | статический  целочисленный | Показатель решения тестировщика по поводу задания статической вершины |
| decidement | статический булевый | Показатель решения тестировщика по поводу задания статической вершины |
| x\_1, y\_1 | вещественный | Статические вершины треугольника |
| Рабочие переменные | | |
| os | потоковая адресация записи | Адрес потока, который будет использован для взаимодействия с консолью |
| object | triangle | Временная переменная для ввода координат треугольников |

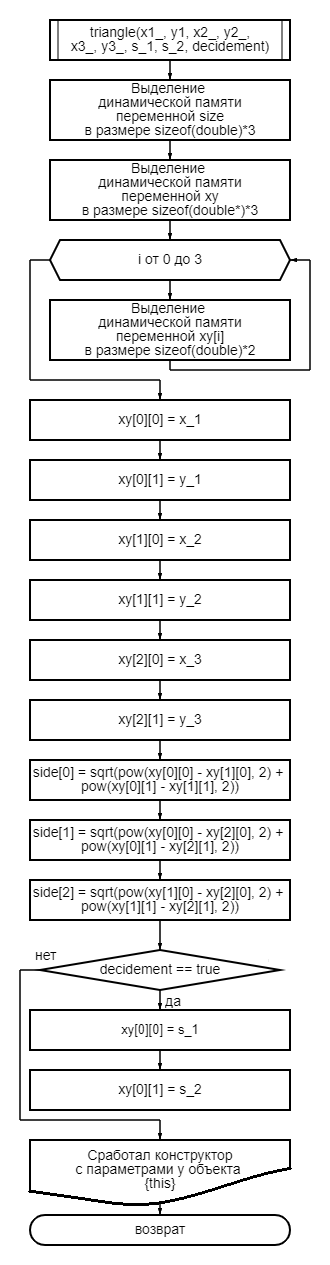
|  |  |  |
| --- | --- | --- |
| Таблица имён для функции перегрузки оператора вывода | | |
| Исходные переменные | | |
| thing | triangle | Переменная – копия выводящегося на экран пользователю треугольника |

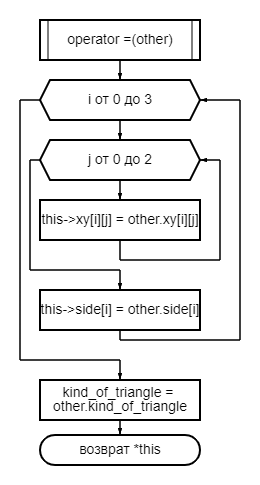
1. Блок-схемы:

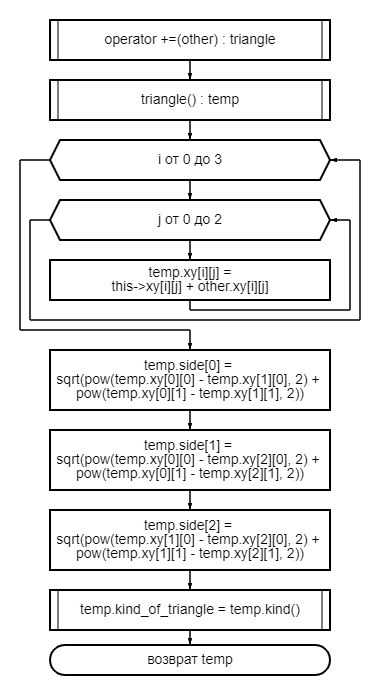
  


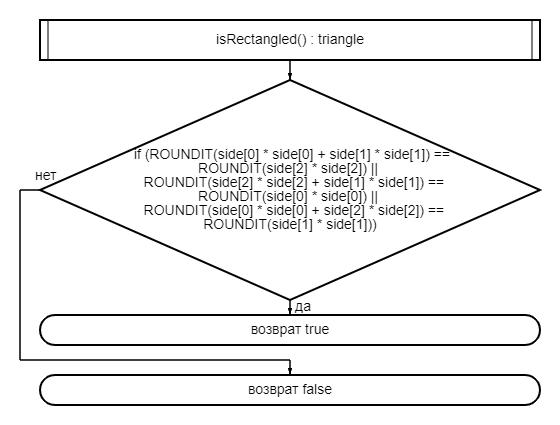
Блок-схема №1: Основная программа

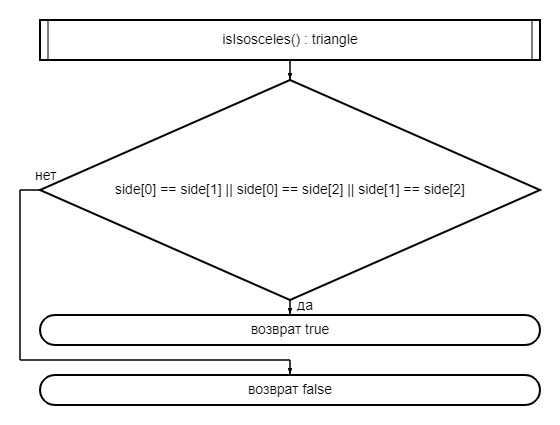
  
Блок-схема №2 – Конструктор по умолчанию

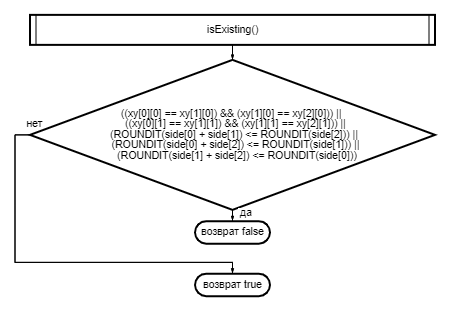
  
Блок-схема №3 – Конструктор с параметрами

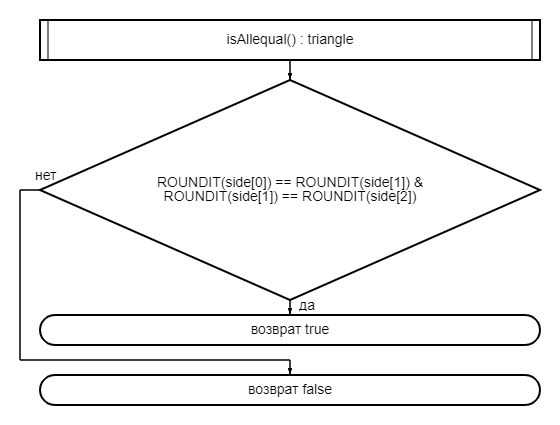
  
Блок-схема №4 – Перегрузка оператора присваивания

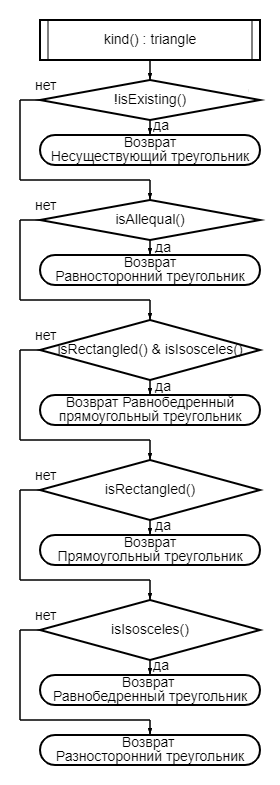
  
Блок-схема №5 – Перегрузка оператора присваивания-сложения

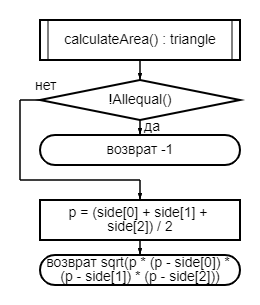
  
Блок-схема №6 – Проверка на прямоугольный треугольник

  
Блок-схема №7 – Проверка на равнобедренный треугольник

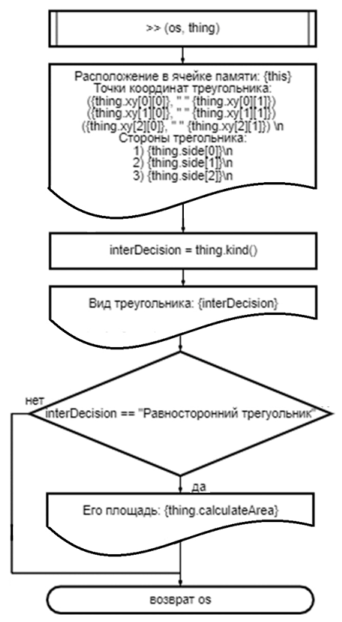
  
Блок-схема №8 – Проверка на существование треугольника

  
Блок-схема №9 – Проверка на правильный треугольник

  
Блок-схема №10 – Определение вида треугольника

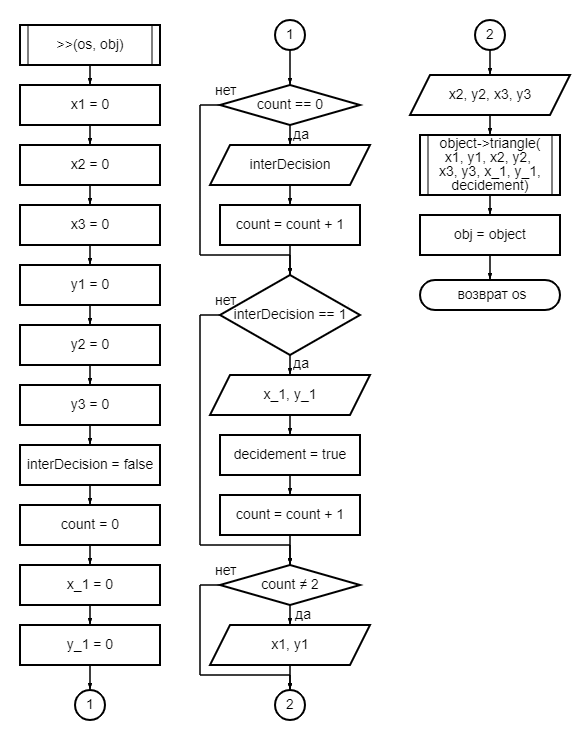


Блок-схема №11 – Подсчёт площади правильного треугольника



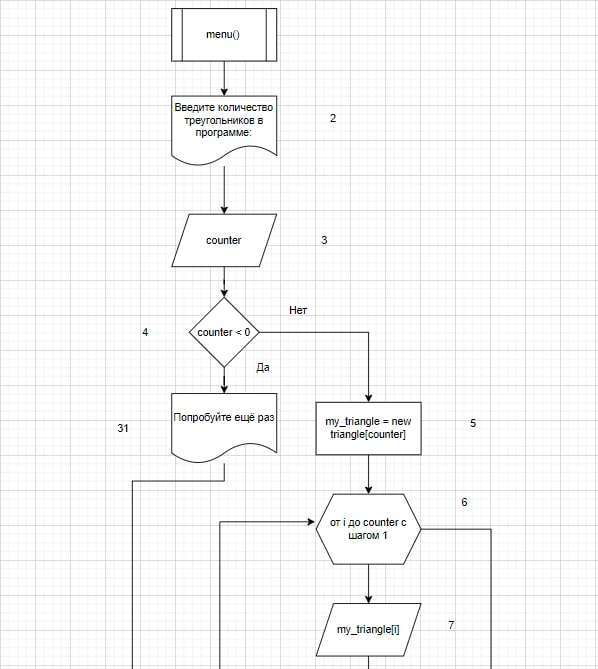
operator << (os, thing)

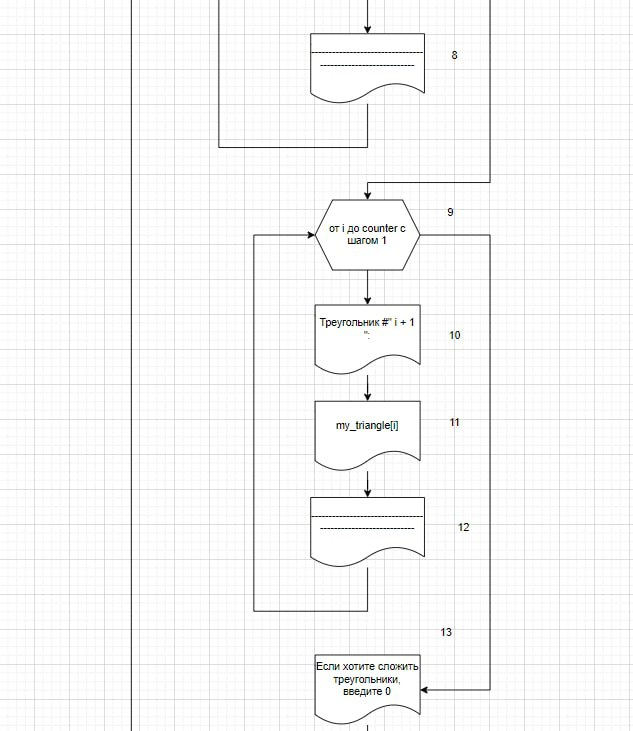
Блок-схема №12: Функция перегрузки оператора вывода



operator >>  
(os, obj)

Блок-схема №13: Функция перегрузки оператора ввода





2x2 = 02

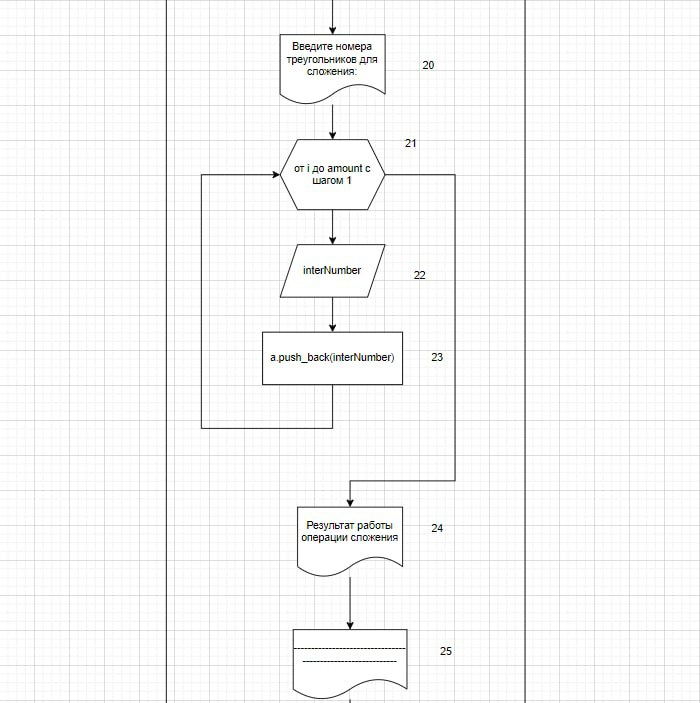
1x2 = 0

Изображение выглядит как текст, доска

Автоматически созданное описание

2x2 = 02

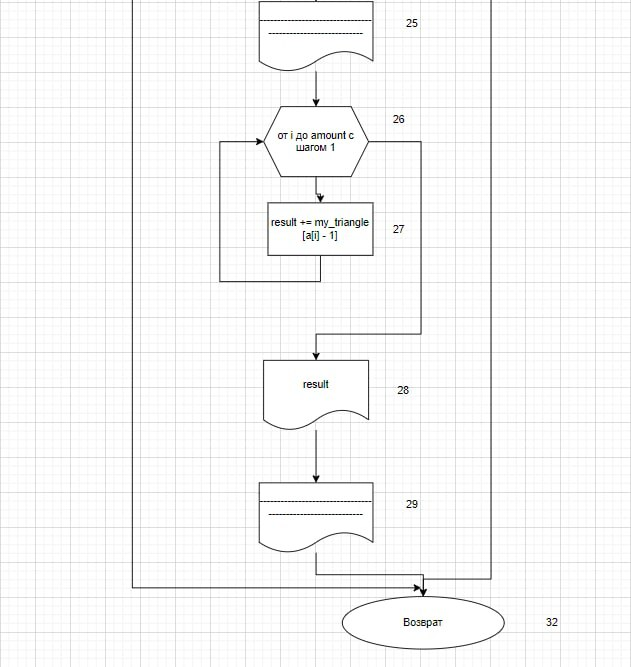
1x2 = 0



5x2 = 02

4x2 = 02

3x2 = 02

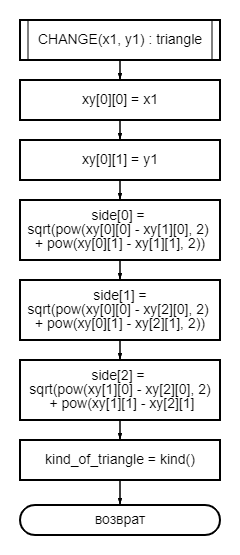


5x2 = 02

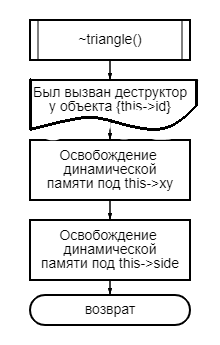
4x2 = 02

3x2 = 02

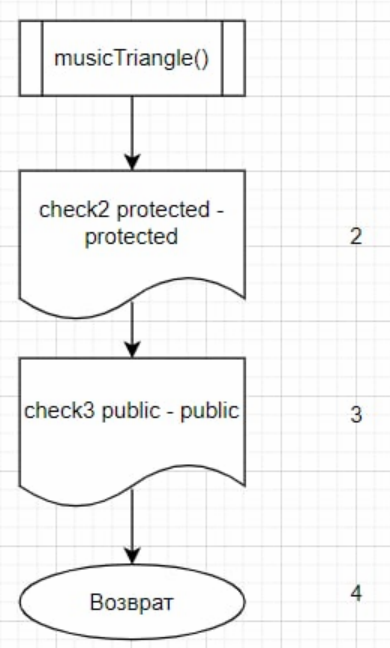
Блок-схема №14: Функция меню файла



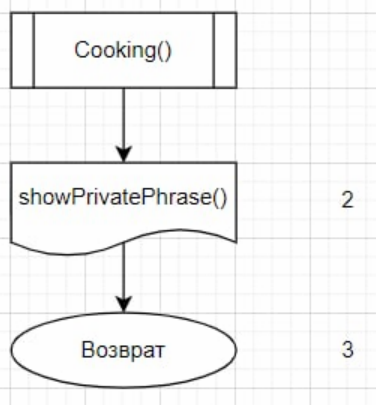
Блок-схема №15: Функция смены общей точки



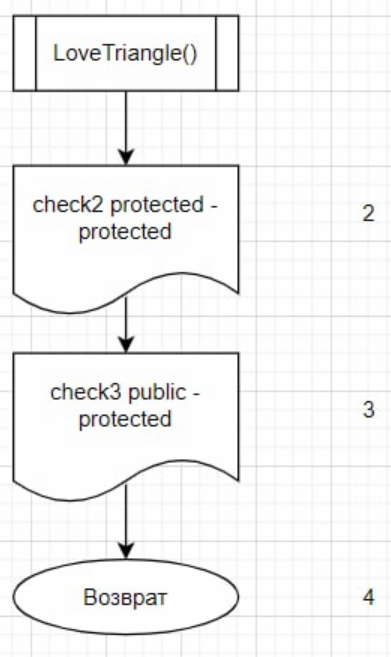
Блок-схема №16 – Деструктор класса



Блок-схема №17 – Конструктор класса musicTriangle



Блок-схема №18 – Конструктор класса Cooking



Блок-схема №19 – Конструктор класса LoveTriangle

1. Отладочный пример:

1 вариант.  
Покажем работу программы при корректном вводе координат.

Изображение выглядит как текст

Автоматически созданное описание

Покажем работу программы при выводе информации о треугольниках.

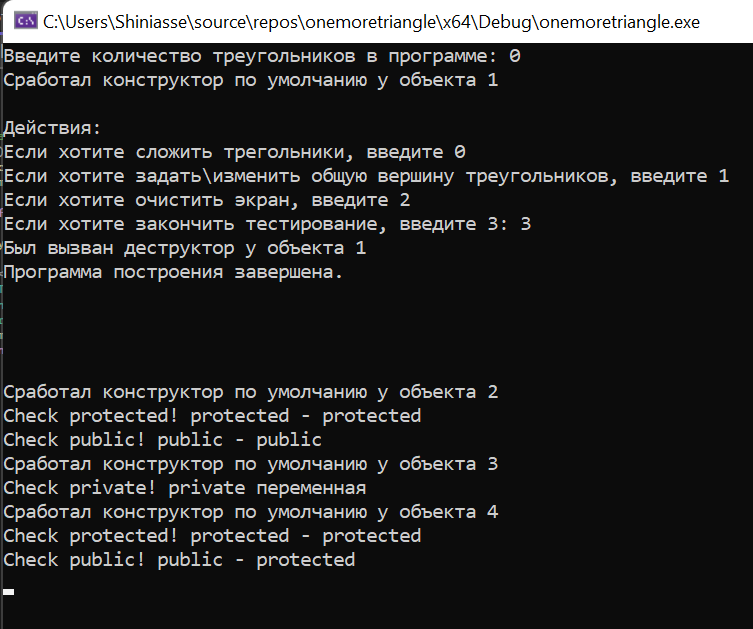
Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание  
Покажем работу программы при реализации сложения.



Покажем работу при завершении работы программы.  
  
  
  
  
Покажем реализацию наследования в программе

1. Код программы:

#include <iostream>

#include <string>

#include <cmath>

#include <vector>

#include <iomanip>

using namespace std;

// Обозначаем макрос для нахождения правильных треугольников

#define ROUNDIT(x) int((x) \* 1000 + 0.5)

static int coun = 0;

// Класс треугольников

class triangle {

private:

string check1 = "Check private!";

protected:

string check2 = "Check protected!";

public:

string check3= "Check public!";

string id = "функции";

double\*\* xy;

double\* side;

string kind\_of\_triangle;

// Конструктор по умолчанию

triangle(double x1 = 0, double y1 = 0,

double x2 = 0, double y2 = 0,

double x3 = 0, double y3 = 0) {

side = new double[3];

xy = new double\* [3];

for (int i = 0; i < 3; i++)

xy[i] = new double[2];

this->xy[0][0] = x1, this->xy[0][1] = y1,

this->xy[1][0] = x2, this->xy[1][1] = y2,

this->xy[2][0] = x3, this->xy[2][1] = y3,

this->side[0] = 0, this->side[1] = 0,

this->side[2] = 0, this->kind\_of\_triangle = "";

coun++;

id = to\_string(coun);

cout << "Сработал конструктор по умолчанию у объекта " << id << endl;

}

// Конструктор с параметрами, задаваемыми пользователем

triangle(

double x1\_, double y1\_,

double x2\_, double y2\_,

double x3\_, double y3\_,

static int s\_1,

static int s\_2,

static bool decidement) {

side = new double[3];

xy = new double\* [3];

for (int i = 0; i < 3; i++)

xy[i] = new double[2];

xy[0][0] = x1\_, xy[1][0] = x2\_, xy[2][0] = x3\_;

xy[0][1] = y1\_, xy[1][1] = y2\_, xy[2][1] = y3\_;

side[0] = sqrt(pow(xy[0][0] - xy[1][0], 2) + pow(xy[0][1] - xy[1][1], 2));

side[1] = sqrt(pow(xy[0][0] - xy[2][0], 2) + pow(xy[0][1] - xy[2][1], 2));

side[2] = sqrt(pow(xy[1][0] - xy[2][0], 2) + pow(xy[1][1] - xy[2][1], 2));

if (decidement) {

xy[0][0] = s\_1;

xy[0][1] = s\_2;

}

cout << "Сработал конструктор с параметрами у объекта " << id << endl;

}

triangle(const triangle& obj) {

side = new double[3];

xy = new double\* [3];

for (int i = 0; i < 3; i++)

xy[i] = new double[2];

this->xy[0][0] = obj.xy[0][0], this->xy[1][0] = obj.xy[1][0], this->xy[2][0] = obj.xy[2][0];

this->xy[0][1] = obj.xy[0][1], this->xy[1][1] = obj.xy[1][1], this->xy[2][1] = obj.xy[2][1];

this->side[0] = obj.side[0];

this->side[1] = obj.side[1];

this->side[2] = obj.side[2];

cout << "Заработал конструктор копирования у объекта " << id << endl;

}

// Функция перегрузки оператора присваивания

triangle& operator =(const triangle& other) {

for (int i = 0; i < 3; i++) {

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

this->xy[i][j] = other.xy[i][j];

this->side[i] = other.side[i];

}

kind\_of\_triangle = other.kind\_of\_triangle;

return \*this;

}

// Функция перегрузки оператора сложения

triangle operator +=(const triangle& other) {

triangle temp;

for (int i = 0; i < 3; i++)

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

temp.xy[i][j] = this->xy[i][j] + other.xy[i][j];

temp.side[0] = sqrt(pow(temp.xy[0][0] - temp.xy[1][0], 2) + pow(temp.xy[0][1] - temp.xy[1][1], 2));

temp.side[1] = sqrt(pow(temp.xy[0][0] - temp.xy[2][0], 2) + pow(temp.xy[0][1] - temp.xy[2][1], 2));

temp.side[2] = sqrt(pow(temp.xy[1][0] - temp.xy[2][0], 2) + pow(temp.xy[1][1] - temp.xy[2][1], 2));

temp.kind\_of\_triangle = temp.kind();

return temp;

}

// Функция для определения прямоугольного треугольника

bool isRectangled() {

if (ROUNDIT(side[0] \* side[0] + side[1] \* side[1]) == ROUNDIT(side[2] \* side[2]) ||

ROUNDIT(side[2] \* side[2] + side[1] \* side[1]) == ROUNDIT(side[0] \* side[0]) ||

ROUNDIT(side[0] \* side[0] + side[2] \* side[2]) == ROUNDIT(side[1] \* side[1]))

return true;

return false;

}

// Функция для определения равнобедренного треугольника

bool isIsosceles() {

if (side[0] == side[1] || side[0] == side[2] || side[1] == side[2])

return true;

return false;

}

// Функция для определения правильного треугольника

bool isAllequal(double eps = 0.1) {

if (ROUNDIT(side[0]) == ROUNDIT(side[1]) && ROUNDIT(side[1]) == ROUNDIT(side[2]))

return true;

return false;

}

// Функция для определения несуществующего треугольника

bool isExisting() {

if (((xy[0][0] == xy[1][0]) && (xy[1][0] == xy[2][0])) ||

((xy[0][1] == xy[1][1]) && (xy[1][1] == xy[2][1])) ||

(ROUNDIT(side[0] + side[1]) <= ROUNDIT(side[2])) ||

(ROUNDIT(side[0] + side[2]) <= ROUNDIT(side[1])) ||

(ROUNDIT(side[1] + side[2]) <= ROUNDIT(side[0])))

return false;

return true;

};

// Функция для определения вида треугольника

string kind() {

if (!isExisting())

return "Несуществующий треугольник";

if (isAllequal())

return "Равносторонний треугольник";

if (isRectangled() && isIsosceles())

return "Равнобедренный прямоугольный треугольник";

if (isRectangled())

return "Прямоугольный треугольник";

if (isIsosceles())

return "Равнобедренный треугольник";

return "Разносторонний треугольник";

}

// Функция для подсчёта площади треугольника

double calculateArea() {

if (!isAllequal())

return -1;

double p = (side[0] + side[1] + side[2]) / 2;

return sqrt(p \* (p - side[0]) \* (p - side[1]) \* (p - side[2]));

}

// Для изменения статической координаты

void CHANGE(int x1, int y1) {

xy[0][0] = x1;

xy[0][1] = y1;

side[0] = sqrt(pow(xy[0][0] - xy[1][0], 2) + pow(xy[0][1] - xy[1][1], 2));

side[1] = sqrt(pow(xy[0][0] - xy[2][0], 2) + pow(xy[0][1] - xy[2][1], 2));

side[2] = sqrt(pow(xy[1][0] - xy[2][0], 2) + pow(xy[1][1] - xy[2][1], 2));

kind\_of\_triangle = kind();

}

// Деструктор класса

~triangle() {

cout << "Был вызван деструктор у объекта " << id << endl;

free(this->xy);

free(this->side);

}

void showPrivatePhrase() {

cout << check1 << " private переменная" << endl;

}

};

// Перегрузка оператора вывода

ostream& operator <<(ostream& os, triangle& thing) {

cout << "-----------------------------------------------------------\n" << endl;

cout << "Треугольник # " << thing.id << ":\n";

cout << "\nРасположение в ячейке памяти: " << &thing << '\n';

cout << "Точки координат треугольника: \n";

cout << setprecision(5) <<

"(" << thing.xy[0][0] << ", " << thing.xy[0][1] << "), " <<

"(" << thing.xy[1][0] << ", " << thing.xy[1][1] << "), " <<

"(" << thing.xy[2][0] << ", " << thing.xy[2][1] << ")\n";

cout << "Стороны треугольника: \n";

cout << setprecision(5) <<

"1) " << thing.side[0] << '\n' <<

"2) " << thing.side[1] << '\n' <<

"3) " << thing.side[2] << '\n';

string interDecision = thing.kind();

cout << "Вид треугольника: " << interDecision << endl;

if (interDecision == "Равносторонний треугольник")

cout << "Его площадь: " << thing.calculateArea();

cout << "\n-----------------------------------------------------------" << endl;

return os;

}

// Перегрузка оператора ввода

istream& operator >>(istream& os, triangle& obj) {

double x1 = 0, x2 = 0, x3 = 0, y1 = 0, y2 = 0, y3 = 0;

int interDecision = 0;

static bool decidement = false;

static int count = 0;

static double x\_1 = 0, y\_1 = 0;

if (count == 0) {

cout << "Введите 1, чтобы задать общую вершину у треугольников: ";

cin >> interDecision;

count++;

}

if (interDecision == 1) {

cout << "Введите координату x1: "; cin >> x\_1;

cout << "Введите координату y1: "; cin >> y\_1;

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

decidement = true;

count++;

}

else if (count != 2) {

cout << "Введите координату x1: "; cin >> x1;

cout << "Введите координату y1: "; cin >> y1;

}

cout << "Введите координату x2: "; cin >> x2;

cout << "Введите координату y2: "; cin >> y2;

cout << "Введите координату x3: "; cin >> x3;

cout << "Введите координату y3: "; cin >> y3;

triangle object(x1, y1, x2, y2, x3, y3, x\_1, y\_1, decidement);

obj = object;

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

return os;

}

// Меню работы программы

char menu() {

int counter;

cout << "Введите количество треугольников в программе: "; cin >> counter;

if (counter < 0) {

cout << "Попробуйте ещё раз.\n";

return '-';

}

triangle\* my\_triangle = new triangle[counter];

// Ввод треугольников

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

cin >> my\_triangle[i];

// Вывод треугольников

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

cout << my\_triangle[i];

int fate, x1, x2, amount, interNumber;

double wow, wow1;

vector<int> a; triangle result;

while (true) {

cout << "\nДействия: \n";

cout << "Если хотите сложить трегольники, введите 0\n";

cout << "Если хотите задать\\изменить общую вершину треугольников, введите 1\n";

cout << "Если хотите очистить экран, введите 2\n";

cout << "Если хотите закончить тестирование, введите 3: ";

cin >> fate;

switch (fate) {

case 0:

cout << "Введите количество складываемых треугольников: ";

while (true) {

cin >> amount;

if (amount > counter || amount < 0) {

"Неправильный ввод. Попробуйте ещё раз!";

continue;

}

break;

}

cout << "Введите номера треугольников для сложения (с единицы, через пробел): ";

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

cin >> interNumber;

a.push\_back(interNumber);

}

cout << "Результат работы операции сложения\n";

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

result = (result += my\_triangle[a[i] - 1]);

cout << result;

system("pause >> null");

break;

case 1:

cout << "Введите координату х1: "; cin >> x1;

cout << "Введите координату y1: "; cin >> x2;

cout << "Новые треугольники: \n";

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

my\_triangle[i].CHANGE(x1, x2);

cout << my\_triangle[i];

}

break;

case 2:

return 4;

case 3:

delete[] my\_triangle;

return 'y';

}

}

}

class musicTriangle : public triangle {

string melody;

public:

musicTriangle() {

cout << check2 << " protected - protected" << endl; // При наследовании protected - public мы получаем тип доступа

// protected - наследуемые поля класса доступны лишь внутри класса-потомка,

// и к ним доступ не может быть получен из других функций. Тестовое значение

// check1 же имеет только тип private, и потому недоступно даже в

// производном классе.

cout << check3 << " public - public" << endl; // Здесь мы можем видеть, что унавследованное поле класса типа public

// доступно из любой части программы, остальные две проверочные переменные

// check1/check2 имели типы protected и private, так что недоступны со

// стороны функции - точки входа в программу.

}

};

class Cooking : private triangle { // При наследовании private все наследуемые элементы становятся

string dish; // типа private - если мы попытаемся сделать производный класс на этой

// основе, то все поля дочернего класса будут нам недоступны. Также,

// все поля private дочернего класса в производном недоступны, таким образом

// мы не можем получить доступ отсюда к тестовой переменной check1.

public:

Cooking() {

showPrivatePhrase(); // Зато мы можем получить к ней доступ вот таким образом, через функцию геттера.

}

};

class LoveTriangle : protected triangle {

string guy = "Вячеслав", guy1 = "Ольга", girl = "Святослав";

public:

LoveTriangle() {

cout << check2 << " protected - protected" << endl; // При наследовании protected - protected мы получаем тип доступа

// protected - наследуемые поля класса доступны лишь внутри класса-потомка,

// и к ним доступ не может быть получен из других функций. Тестовое значение

// check1 же имеет только тип private, и потому недоступно даже в

// производном классе.

cout << check3 << " public - protected" << endl; // При наследовании public - protected мы получаем тип доступа

// protected - наследуемые поля класса доступны лишь внутри класса-потомка,

// и к ним доступ не может быть получен из других функций.

}

};

// Точка входа в программу

int main() {

setlocale(LC\_ALL, "rus");

system("chcp 1251 >> null");

while (true) {

if (menu() == 'y')

break;

system("cls");

}

cout << "Программа построения завершена.\n\n\n\n\n";

musicTriangle test1; // тут идёт наследование public-public, private-public, protected-public

Cooking test2; // тут идёт наследование public - private, private - private, rotected - private

LoveTriangle test3; // тут идёт наследование public - protected, private - protected, protected - protected

system("pause >> null"); // Все тонкости наследования объясняю устно.

return 0;

}

1. Результат выполнения работы программы:

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание  
Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

1. Вывод:

В ходе выполнения работы мы ознакомились со способом реализации объектов и методов над ними, научились перегружать стандартные операторы потокового ввода-вывода, а также стандартный оператор сложения, познакомились с конструктором копирования. Также научились декомпозировать код, изначально написанный в main на различные функции.

На контрольных примерах мы убедились, что код работает корректно.  
Был оформлен комплект документации на программный код.