Балтийский государственный технический университет  
«ВОЕНМЕХ»  им. Д. Ф. Устинова

Факультет “O” Естественнонаучный

Кафедра О7 «Информационные системы и программная инженерия»

**Практическая работа №1**по дисциплине «Программирование на ЯВУ»  
на тему «КЛАССЫ: ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ»

Выполнил:  
Студент Костров Г. Ю. Группа О712Б  
Преподаватель: Васюков В.М.

Санкт-Петербург  
2022 г

**Постановка задачи:**

Описать класс в соответствии с индивидуальным вариантом задания и реализовать все его методы. Каждый класс должен содержать, помимо указанных в варианте методов, конструктор с параметрами, конструктор копирования, деструктор, методы ввода с клавиатуры, установки и получения значений полей, вывода этих значений на экран. В каждом методе класса, включая конструкторы и деструктор, предусмотреть отладочную печать сообщения, содержащего имя метода. Написать программу для тестирования всех методов класса, выбор метода должен осуществляться с помощью меню.

**Вариант 12**

Класс Треугольник

Поля: длина одной из сторон и величины прилежащих к ней углов

Методы: вычисление высоты, проведенной из заданного угла,определение типа

(остроугольный, прямоугольный, тупоугольный),вычисление всех углов, вычисление длин всех сторон

Операторы: перегрузка операции ^ для обозначения операции определения подобия двух

треугольников

**Текст программы:**

**Файл triangle.h**

#ifndef TRIANGLE\_H

#define TRIANGLE\_H

#include "iostream"

using namespace std;

class triangle

{

private:

double side;

double angle\_1;

double angle\_2;

public:

string name;

triangle();///конструктор по умолчанию

triangle(string, double, double, double);///конструктор с параметром

triangle(const triangle &);///конструктор копирования

~triangle(); // деструктор

// Геттеры

double getSide();

double getAngle\_1();

double getAngle\_2();

// Сеттеры

void setSide(double);

void setAngle\_1(double);

void setAngle\_2(double);

// Методы

void set\_triangle\_properties();

void calculation\_all\_angles();

void calculating\_all\_sides();

void calculating\_height();

void type();

void find\_similarity(triangle\* &, int, int);

friend ostream& operator<< (ostream &out, const triangle &output);

friend istream& operator>> (istream &in, triangle &input);

friend string operator^ (const triangle& tr\_1, const triangle& tr\_2);//вычисление площади

};

#endif

**Файл menu.h**

#include "triangle.h"

#ifndef MENU\_H

#define MENU\_H

void print\_triangle\_menu();

void print\_main\_menu(int);

void print\_all\_triagles(triangle, int);

void init\_array(triangle\* &, const int );

void free\_array(triangle\* );

void realloc\_array(triangle\* &, int& );

void fill\_array(triangle\* &, const int );

void triangle\_menu(triangle\* &, int, int);

int get\_variant(int);

void menu();

#endif // MENU\_H

**Файл triangle.cpp**

#include "triangle.h"

#include "memory.h"

#include "menu.h"

#include "cmath"

triangle::triangle()

{

this -> name = "example" ;

this->side = 5;

this->angle\_1 = 60;

this->angle\_2 = 30;

}

///конструктор с параметром

triangle::triangle(string name, double side, double angle\_1, double angle\_2)

{

this->name = name;

this->side = side;

this->angle\_1 = angle\_1;

this->angle\_2 = angle\_2;

}

// конструктор копирования

triangle::triangle(const triangle &src)

{

this->name = src.name;

this->side = src.side;

this->angle\_1 = src.angle\_1;

this->angle\_2 = src.angle\_2;

}

// деструктор

triangle::~triangle()

{

this->name = "";

this->side = 0;

this->angle\_1 = 0;

this->angle\_2 = 0;

}

// геттеры

double triangle::getSide()

{

return this->side;

}

double triangle::getAngle\_1()

{

return this->angle\_1;

}

double triangle::getAngle\_2()

{

return this->angle\_2;

}

// сеттеры

void triangle::setSide(double side)

{

this->side = side;

}

void triangle::setAngle\_1(double angle\_1)

{

this->angle\_1 = angle\_1;

}

void triangle::setAngle\_2(double angle\_2)

{

this->angle\_2 = angle\_2;

}

// методы

void triangle::set\_triangle\_properties() {

int variant;

do {

system("cls");

cout << "1. Change triangle name" << endl;

cout << "2. Change triangle side" << endl;

cout << "3. Change angle #1" << endl;

cout << "4. Change angle #2" << endl;

cout << string ( 25, '-' ) << endl;

cout << "5. Exit" << endl << endl;

cout << ">" ;

variant = get\_variant(5);

switch (variant) {

case 1:

cout << "Enter new name" << endl;

cin >> this->name;

break;

case 2:

double side;

cout << "Enter new triangle side" << endl;

cin >> side;

setSide(side);

break;

case 3:

double angle\_1;

cout << "Enter new triangle angle" << endl;

cin >> angle\_1;

setAngle\_1(angle\_1);

break;

case 4:

double angle\_2;

cout << "Enter new triangle angle" << endl;

cin >> angle\_2;

setAngle\_2(angle\_2);

break;

case 5:

cout << "Exit" << endl;

break;

}

} while (variant != 5);

}

void triangle::calculation\_all\_angles() {

cout << "Angle #1 = " << this-> angle\_1 << endl;

cout << "Angle #2 = " << this-> angle\_2 << endl;

double angle\_3 = (180 - this->angle\_1 - this->angle\_2);

cout << "Angle #3 = " << angle\_3 << endl;

}

void triangle::calculating\_all\_sides() {

double side\_2 = (this->side \* sin(this->angle\_1 \* M\_PI / 180));

double side\_3 = (this->side \* sin(this->angle\_2 \* M\_PI / 180));

cout << "Side #1 = " << this-> side << endl;

cout << "Side #2 = " << side\_2 << endl;

cout << "Side #3 = " << side\_3 << endl;

}

void triangle::calculating\_height() {

double height\_1 = (this->side \* sin(this->angle\_2 \* M\_PI / 180));

cout << "Height from angle\_1" << " = " << height\_1 << endl;

double height\_2 = (this->side \* sin(this->angle\_1 \* M\_PI / 180));

cout << "Height from angle\_2" << " = " << height\_2 << endl;

}

void triangle::type() {

double angle\_3 = (180 - this->angle\_1 - this->angle\_2);

if (this->angle\_1 == 90 || this->angle\_2 == 90 || angle\_3 == 90) {

cout << "Triangle is right" << endl;

}

else if (this->angle\_1 == this->angle\_2 || this->angle\_1 == angle\_3 || this->angle\_2 == angle\_3) {

cout << "Triangle is isosceles" << endl;

}

else if (this->angle\_1 > 90 || this->angle\_2 > 90 || angle\_3 > 90) {

cout << "Triangle is obtuse" << endl;

}

else {

cout << "Triangle is scalene" << endl;

}

}

void triangle::find\_similarity(triangle\* &list, int N, int current) {

cout << "Choose triangle" << endl;

cout << "> ";

system("cls"); // очищаем экран

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

cout << i + 1 << ". " << list[i].name << endl;

}

cout << string ( 25, '-' ) << endl;

cout << N + 1 << ". Exit" << endl << endl;

cout << ">";

int var = get\_variant(N + 1);

int count = N + 1;

if (var < count) {

cout << (list[current] ^ list[var-1]) << endl;

}

}

// оператор вывода

ostream& operator<< (ostream &out, const triangle &output)

{

out << "Name: " << output.name << endl;

out << "side: " << output.side << endl;

out << "angle\_1: " << output.angle\_1 << endl;

out << "angle\_2: " << output.angle\_2 << endl;

return out;

}

// оператор ввода

istream& operator>> (istream &in, triangle &input)

{

int flag = 0;

do {

cout << "side: ";

int side\_1;

in >> side\_1 ;

if (side\_1 > 0) {

input.side = side\_1;

flag++;

}

} while(flag != 1);

flag = 0;

do {

cout << "angle\_1: ";

int ang\_1;

in >> ang\_1 ;

if (ang\_1 > 1 && ang\_1 < 180) {

input.angle\_2 = ang\_1;

flag++;

}

}while(flag != 1);

flag = 0;

do {

cout << "angle\_2: ";

int ang\_2;

in >> ang\_2 ;

if (ang\_2 > 1 && ang\_2 < 180) {

input.angle\_2 = ang\_2;

flag++;

}

}while(flag != 1);

return in;

}

string operator^ (const triangle &tr\_1, const triangle &tr\_2)

{

string result;

double tr\_1\_angle\_3 = (180 - tr\_1.angle\_1 - tr\_1.angle\_2);

double tr\_2\_angle\_3 = (180 - tr\_2.angle\_1 - tr\_2.angle\_2);

double mas[3];

if (tr\_1.angle\_1 == tr\_2.angle\_1 ) {

if (tr\_1.angle\_2 == tr\_2.angle\_2) {

result = "Triangles are similar";

} else if (tr\_1.angle\_2 == tr\_2\_angle\_3) {

result = "Triangles are similar";

} else {

result = "Triangles are not similar";

}

} else if (tr\_1.angle\_1 == tr\_2.angle\_2) {

if (tr\_1.angle\_2 == tr\_2.angle\_1) {

result = "Triangles are similar";

} else if (tr\_1.angle\_2 == tr\_2\_angle\_3) {

result = "Triangles are similar";

} else {

result = "Triangles are not similar";

}

} else if (tr\_1.angle\_1 == tr\_2\_angle\_3) {

if (tr\_1.angle\_2 == tr\_2.angle\_1) {

result = "Triangles are similar";

} else if (tr\_1.angle\_2 == tr\_2.angle\_2) {

result = "Triangles are similar";

} else {

result = "Triangles are not similar";

}

} else {

result = "Triangles are not similar";

}

return result;

}

**Файл menu.cpp**

#include <iostream>

#include "menu.h"

#include "triangle.h"

#include "string"

using namespace std;

void print\_triangle\_menu() {

system("cls"); // очищаем экран

cout << "What do you want to do?" << endl;

cout << "1. Triangle properties" << endl;

cout << "2. Set triangle properties" << endl;

cout << "3. Calculation of all angles" << endl;

cout << "4. Calculating the lengths of all sides" << endl;

cout << "5. Calculating the height drawn from a given angle" << endl;

cout << "6. Determining the type of triangle" << endl;

cout << "7. Find similar triangle" << endl;

cout << string ( 25, '-' ) << endl;

cout << "8. Exit" << endl << endl;

cout << "> ";

}

void print\_main\_menu() {

system("cls"); // очищаем экран

cout << "What do you want to do?" << endl;

cout << "1. Select triangle" << endl;

cout << "2. Create new triangle" << endl;

cout << string ( 25, '-' ) << endl;

cout << "3. Exit" << endl << endl;

cout << "> ";

}

void print\_all\_triagles(triangle\* &list, int N) {

cout << "Choose triangle PLS!" << endl;

cout << "> ";

system("cls"); // очищаем экран

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

cout << i + 1 << ". " << list[i].name << endl;

}

cout << string ( 25, '-' ) << endl;

cout << N + 1 << ". Exit" << endl << endl;

}

void init\_array(triangle\* &P, const int N){ P = new triangle [N]; } // инициализация массива

// void free\_array(triangle\* P){

// delete []P;

// }

void realloc\_array(triangle\* &P, int& N){

triangle\* new\_ptr = new triangle [N + 1];

memmove(new\_ptr,P, sizeof(triangle) \* N); //Копирование старых данных в новое формирование

// free\_array(P);

P = new\_ptr;

}

void fill\_array(triangle\* &P, const int N) {

// P[N] = triangle();

string name;

int flag = 0;

cout << "Enter the name of the triangle: ";

getline(cin, name);

P[N].name = name;

cin >> P[N];

cout << endl << "Triagle created" << endl;

}

void menu () {

int variant\_menu;

triangle \*list = 0;

int N = 0;

init\_array(list,N);

do {

print\_main\_menu();

variant\_menu = get\_variant(3);

switch (variant\_menu) {

case 1: {

// выбор треугольника

if (N == 0) {

cout << "There are no triangles" << endl;

system("pause");

} else {

print\_all\_triagles(list, N);

cout << ">";

int count = N + 1;

int var;

var = get\_variant(count);

if (var < count) {

triangle\_menu(list, var, N);

}

}

break;

}

case 2: {

realloc\_array(list,N);

fill\_array(list,N);

N++;

system("pause");

break;

}

case 3:

break;

}

} while (variant\_menu != 3);

}

void triangle\_menu(triangle\* &P, int var, int N) {

int variant;

var--;

do {

print\_triangle\_menu();

variant = get\_variant(8);

system("cls");

switch (variant) {

case 1:

cout << P[var] << endl;

break;

case 2:

P[var].set\_triangle\_properties();

break;

case 3:

P[var].calculation\_all\_angles();

break;

case 4:

P[var].calculating\_all\_sides();

break;

case 5:

P[var].calculating\_height();

break;

case 6:

P[var].type();

break;

case 7:

P[var].find\_similarity(P,N,var);

break;

case 8:

cout << "Exit" << endl;

break;

}

variant != 8 ? system("pause") : 0;

} while (variant != 8);

}

int get\_variant(int count) {

int var;

cin.clear();

string s; // строка для считывания введённых данных

getline(cin, s); // считываем строку

// пока ввод некорректен, сообщаем об этом и просим повторить его

while (sscanf(s.c\_str(), "%d", &var) != 1 || var < 1 || var > count) {

if (s.size() != 0) {

cout << "Incorrect input. Try again: "; // выводим сообщение об ошибке

getline(cin, s); // считываем строку повторно

} else {

var = get\_variant(count);

if (var >= 1 || var <= count) {

break;

}

}

}

return var;

}

**Файл main.cpp**

#include <iostream>

#include "triangle.h"

#include "menu.h"

using namespace std;

int main()

{

menu();

return 0;

}

**Результат работы программы:**

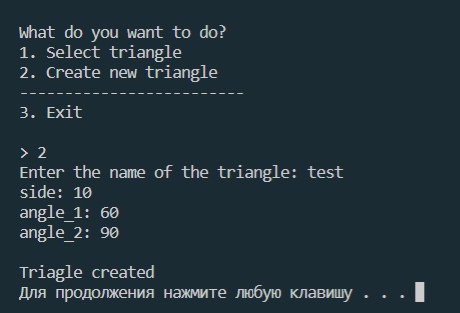


Рисунок 1 - вводим углы и сторону между ними(создание треугольника)

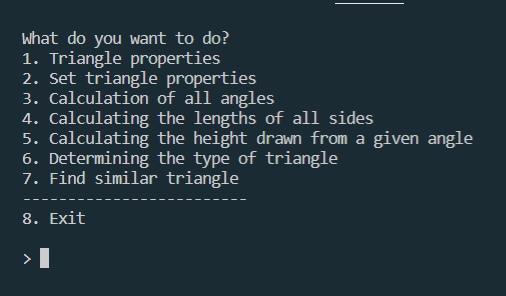


Рисунок 2 - меню

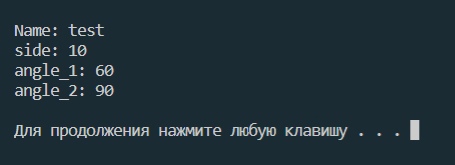


Рисунок 3 - вывод введенных данных

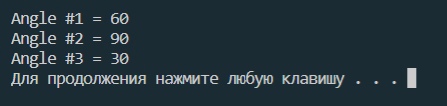


Рисунок 4 – вычисление и всех углов треугольника

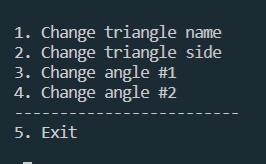


Рисунок 5 – меню для установки значений треугольника

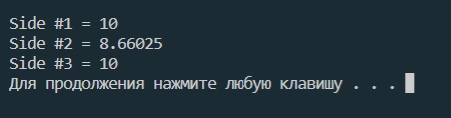


Рисунок 6 – Вычисление длин всех сторон треугольника

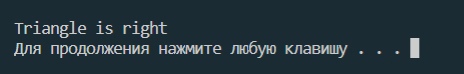


Рисунок 7 - тип треугольника

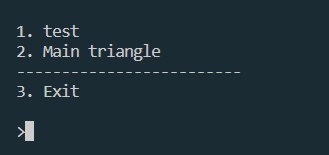


Рисунок 8 – выбор треугольника

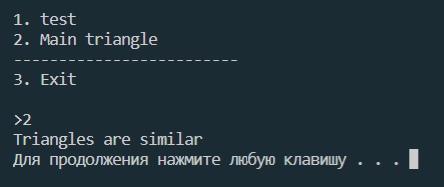


Рисунок 9 – определение подобия 2 треугольников