

Балтийский государственный технический университет
«ВОЕНМЕХ» им. Д. Ф. Устинова
Факультет “О” Естественнoнаучный
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_triangles(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;
    } while (variant < 4);
}

```

```

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_triangles(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 << "Triangle 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

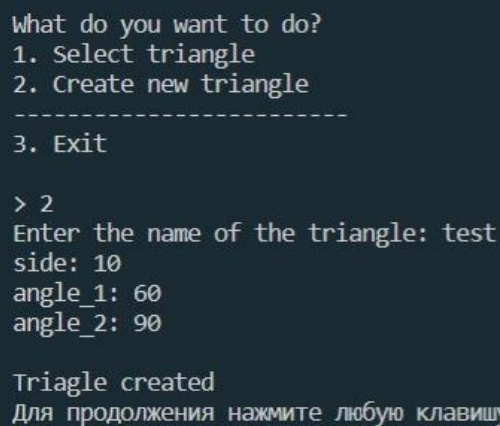
```
#include <iostream>
#include "triangle.h"
#include "menu.h"
```

```
using namespace std;
```

```
int main()
{
    menu();

    return 0;
}
```

Результат работы программы:

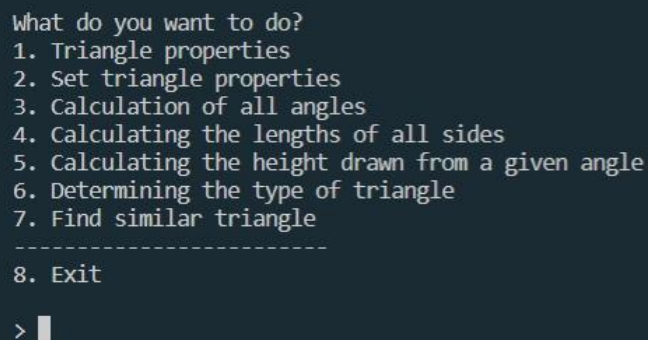


```
What do you want to do?
1. Select triangle
2. Create new triangle
-----
3. Exit

> 2
Enter the name of the triangle: test
side: 10
angle_1: 60
angle_2: 90

Triagle created
Для продолжения нажмите любую клавишу . . .
```

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



```
What do you want to do?
1. Triangle properties
2. Set triangle properties
3. Calculation of all angles
4. Calculating the lengths of all sides
5. Calculating the height drawn from a given angle
6. Determining the type of triangle
7. Find similar triangle
-----
8. Exit

> 
```

Рисунок 2 - меню

```
Name: test  
side: 10  
angle_1: 60  
angle_2: 90
```

```
Для продолжения нажмите любую клавишу . . .
```

Рисунок 3 - вывод введенных данных

```
Angle #1 = 60  
Angle #2 = 90  
Angle #3 = 30
```

```
Для продолжения нажмите любую клавишу . . .
```

Рисунок 4 – вычисление и всех углов треугольника

```
1. Change triangle name  
2. Change triangle side  
3. Change angle #1  
4. Change angle #2  
-----  
5. Exit
```

Рисунок 5 – меню для установки значений треугольника

```
Side #1 = 10  
Side #2 = 8.66025  
Side #3 = 10
```

```
Для продолжения нажмите любую клавишу . . .
```

Рисунок 6 – Вычисление длин всех сторон треугольника

```
Triangle is right
```

```
Для продолжения нажмите любую клавишу . . .
```

Рисунок 7 - тип треугольника

```
1. test
2. Main triangle
-----
3. Exit
>|
```

Рисунок 8 – выбор треугольника

```
1. test
2. Main triangle
-----
3. Exit

>2
Triangles are similar
Для продолжения нажмите любую клавишу . . . |
```

Рисунок 9 – определение подобия 2 треугольников