Минобрнауки РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тульский государственный университет»

Институт прикладной математики и компьютерных наук
Кафедра информационной безопасности

Языки программирования

Отчет по выполнению лабораторной работы № _

_____

Вариант №__

Выполнил

_____

Проверил

_____

Тула 202_

Цель лабораторной работы:

Изучить основные понятия ООП: «объект», «класс», «инкапсуляция»; познакомиться со способами описания классов и объектов в языке C++; познакомиться с возможностью перегрузки операторов и использования конструкторов объектов класса; разработать приложения по своим вариантам заданий.

Задание на работу:

Написать тексты h-файлов и cpp-файлов для класса Treugolnik (треугольник). Описание класса:

Класс	Элементы данных	Интерфейс
Treugolnik	x1, y1, x2, y2, x3, y3	Конструкторы, функции move, square, операции =, <, > (сравнение площади), =*(изменить пропорции в некоторое число раз) , <<, >>

Ход выполнения работы

Текст программы.

Текст файла «triangle.h»:

```
#include <utility>
#include <stdio.h>
#include <iostream>
#include <cmath>
struct point {    double x, y;
std::pair<double, double> len_to(point b)    {
        std::pair<double, double> ret;
        ret.first = this->x - b.x;
        ret.second = this->y - b.y;
     return ret;    }
point multiply(point adj, double len){ point tmp = *this;
        tmp.x = len_to(adj).first * len;
        tmp.y = len_to(adj).second * len;
        *this = tmp;        return tmp;    }
point operator+(point b)    {     point tmp;
        tmp.x = this->x + b.x;        tmp.y = this->y + b.y;
        return tmp;    }
point operator=(point b)    {     point tmp;
        this->x = tmp.x = b.x;        this->y = tmp.y = b.y;
        return tmp;    }
point operator+=(point b)    {    point tmp = *this;
        tmp = tmp + b;        *this = tmp;
        return tmp;    }
point(double a = 0, double b = 0) {x = a; y = b; return *this;}
```

```
point operator*=(double mlt)    {    point tmp = *this;
        tmp.x *= mlt;    tmp.y *= mlt;
        *this = tmp;     return tmp;    }
void prt(){printf("(%f, %f)", this->x, this->y);}
friend std::istream & operator>>(std::istream &input, point &a)
{       input >> a.x >> a.y;        return input;    }
};

class triangle  {
private:
    point A;
    point B;
    point C;
    point adj;
public:
    triangle (point a, point b, point c)    {
        this->A = a;
        this->B = b;
        this->C = c;
        this->adj.x = 0;
        this->adj.y = 0;    }
    void move(int pt, point xy)    {
        switch (pt)        {
        case 1:        {   A = A + xy;     break;     }
        case 2:        {   B = B + xy;     break;     }
        case 3:        {   C = C + xy;     break;     }
        case 4:        {   adj = xy;       break;     }
        default:        {
            this->A = A + xy;
            this->B = B + xy;
            this->C = C + xy;
            break;        }
        }
    }
    long double space()    {
        long double sp, tr[4];
tr[0]=sqrt(pow(this->A.x-this->B.x,2)+pow(this->A.y-this->B.y,2));
tr[1]=sqrt(pow(this->B.x-this->C.x,2)+pow(this->B.y-this->C.y,2));
tr[2]=sqrt(pow(this->A.x-this->C.x,2)+pow(this->A.y-this->C.y,2));
tr[3]=(tr[0]+tr[1]+tr[2]);
sp = sqrt(tr[3]/2*(tr[3]/2-tr[0])*(tr[3]/2-tr[1])*(tr[3]/2-tr[2]));
            return sp;    }
```

```
    bool print_point(int point_number)    {
        switch(point_number)        {
            case 1:      A.prt();      break;
            case 2:      B.prt();      break;
            case 3:      C.prt();      break;
            default:
                printf("\nОшибка в выборе номера точки\n");
                return 1;
                break;  }
        return 0;
    }
    triangle operator=(triangle t)    {
        this->A = t.A;
        this->B = t.B;
        this->C = t.C;
        this->adj = t.adj;    return *this;}
    bool operator<(triangle t)    {
if(abs(this->space()-t.space())<0.001||t.space()-this->space()<0)
            return 0;
        return 1;    }
    bool operator>(triangle t)    {
if (abs(t.space()-this->space())<0.001||this->space()-t.space()<0)
            return 0;
        return 1;    }
    triangle operator*=(double resize_to)    {
        A.multiply(adj, resize_to);
        B.multiply(adj, resize_to);
        C.multiply(adj, resize_to);   return *this; }
friend std::istream & operator>>(std::istream & input, triangle & a)
    {   input >> a.A.x >> a.A.y >> a.B.x >> a.B.y >> a.C.x >> a.C.y;
        return input;    }
friend std::ostream & operator<<(std::ostream & output, const
triangle & a)    {
        output << "{"
        << "(" << a.A.x << ", " << a.A.y << "), "
        << "(" << a.B.x << ", " << a.B.y << "), "
        << "(" << a.C.x << ", " << a.C.y << ")"
        << "}";
        return output;    }
};
```

Текст файла «triangle.cpp»:

```
#include <stdio.h>
#include <iostream>
#include "triangle.h"
#include <vector>
std::vector<triangle> plane;
void menu();
void basic();
void edit();
```

```
void comparison();
void creation();
void prt();
void transform(int, triangle *);
void prt_menu(int);
void multiply(triangle *);
enum print_edit{tr_num = 1,  tr_not_exist,};
int main() {
    menu();
    return 0; }
void menu() {    int x = 1;
    while (x)    {
        printf(
"Доступные операции:\n"
"1\tБазовые операции (создание, выведение координат или площади)\n"
"2\tИзменение координат треугольников\n"
"3\tСравнение треугольников по площади\n"
"0\tВыход из программы\n");
        scanf("%d", &x);
        switch (x)        {
        case 1:    basic();       break;
        case 2:    edit();        break;
        case 3:    comparison();
        default:
            break;
        }
    }
}
void basic() {    int x = 1;
    printf(
        "Доступные операции:\n"
        "1\tСоздание нового треугольника\n"
        "2\tВыведение координат и площади n-го треугольника\n"
        "0\tВернуться в предыдущее меню\n");
    scanf("%d", &x);
    switch (x)    {
    case 1:     creation();     break;
    case 2:     prt();          break;
    default:    break;
    }
}
void edit(){
    int x = 1, trn;    triangle *j;
another_triangle:;
    prt_menu(tr_num);    scanf("%d", &trn);    getchar();
    if (trn > plane.size() || trn <= 0)    {
        prt_menu(tr_not_exist);
        goto another_triangle;
    }
```

```
    j = &plane[trn - 1];
    printf(
        "Доступные операции:\n"
        "1-3\tПеренесение i-й точки треугольника\n"
        "4\tПеренесение %d-го треугольника\n"
        "5\t\"Умножение\" %d-го треугольника на число\n"
        "0\tВернуться в предыдущее меню\n",
        trn);
    scanf("%d", &x);
    if (!x)        return;
    if (x < 5)        transform(x, j);
    else        multiply(j);    }
void comparison(){    int tr[2];
another_tr:;
    printf("Введите номера сравниваемых треугольников:\n");
    scanf("%d%d", &tr[0], &tr[1]);    tr[0]--; tr[1]--;
    if (tr[0] >= plane.size() || tr[1] >= plane.size())    {
printf("Один из введённых номеров треугольников не содержится в
памяти\n");
        goto another_tr;    }
    triangle u[2] = {plane[tr[0]], plane[tr[1]]};
    std::cout
        << "Треугольник 1 :\n"
        << u[0] << "\nПлощадь: " << u[0].space()
        << std::endl
        << "Треугольник 2 :\n"
        << u[1] << "\nПлощадь: " << u[1].space()
        << std::endl;
    if (u[0] > u[1])
        printf("Площадь треугольника 1 > площади треугольника 2\n");
    else if (u[0] < u[1])
        printf("Площадь треугольника 2 > площади треугольника 1\n");
    else
        printf("Площадь треугольника 2 = площади треугольника 1\n");}

void creation() {    triangle temp;
    printf("Введите для каждой точки треугольника координаты x и y (в
общей сложности 6):\n");
    std::cin >> temp;    plane.push_back(temp); }
void prt() {    int x;
another_trg:;
    prt_menu(tr_num);    scanf("%d", &x);    getchar();
    if (x > plane.size() || x <= 0)    {
        prt_menu(tr_not_exist);
        goto another_trg;    }
std::cout << plane[x - 1] << "\nПлощадь: " << plane[x - 1].space();
    prt_menu(0);     }
```

```
void transform(int point_num, triangle *j){    int tr_n;
    if (point_num == 4)    {    point a;
printf("Введите, на какое расстояние нужно перенести треугольник (x и
y):\n");
        std::cin >> a;        (*j).move(5, a);    }
    else    {
        point a;
        printf("Начальные координаты точки:\t");
        (*j).print_point(point_num);
        printf("\nВведите, на какое расстояние перенести точку\n");
        std::cin >> a;
        (*j).move(point_num, a);
        printf("Новые координаты точки:\t");
        (*j).print_point(point_num);
        prt_menu(0);    }  }
void multiply(triangle *a) {
    triangle n = *a;    point adj;
    double newsize;  std::cout << "Начальные координаты:\n" << n;
printf("\nВведите, от какой точки рассчитывать умножение (x и
y):\n");    std::cin >> adj;    n.move(4, adj);
    printf("Введите, на что умножить:\n");
    std::cin >> newsize;    n *= newsize;
    std::cout << "Новые координаты:\n" << n << std::endl;
    *a = n;     }
void prt_menu(int a)  {
    switch (a)    {
    case 0:      printf("\n");                                  break;
    case 1:      printf("Введите номер треугольника:\n");    break;
    case 2:
printf("Номеров треугольников с таким номером не существует в
памяти\n");       break;
    default:     printf("ошибка функции prt_menu");          break;
    }
}
```

Описание функций в файле «triangle.cpp».

Название функции	Принимаемое значение	Возвращаемое значение	За что отвечает
main		int	Вызов функции menu
menu			Вывод основного меню и вызов самих подфункций (базовые операции, изменение треугольника, сравнение по площади)
basic			Вывод меню базовых операций с треугольниками и вызов самих подфункций (создание, вывод координат и площади)
edit			Вывод меню операций с треугольником и вызов самих подфункций (перенос точек, перенос треугольника, "умножение" треугольника на число)
comparison			Сравнение двух треугольников по площади
creation			Создание треугольника
prt			Вывод координат и площади треугольника
transform	int point_num, triangle *j		Изменение координат точки или всего треугольника
multiply	triangle *a		"Умножение" треугольника на число (рассчитывается от указанной точки)
prt_menu	int a		Вспомогательная функция для вывода повторяющихся строк

Описание класса “point”.

Элемент структуры	Принимаемое значение (функции) / Тип (элементы)	Возвращаемое значение (функции)	Роль
x, y	double		Координаты точки
len_to	point b	Пара элементов типов double, double	Вычисление пары чисел – расстояния от одной точки до другой по x и по y
multiply	point adj, double len	point	"Умножение" координат точки от заданной точки на какое-либо значение
operator+	point b	point	Сложение координат точек
operator=	point b	point	Приравнивание координат точек
operator+=	point b	point	Сложение координат второй точки с первой и приравнивание координат первой точки этой "точке"
point (конструктор)	double a, double b (по умолчанию оба равны 0)	point	Конструирование элемента типа point с заданными координатами
operator*=	double mlt	point	Умножение координат точки на заданное значение
prt			Вывод координат точки
friend istream & operator>>	istream &input, point & a	istream	Перегрузка оператора >> (ввода) для cin
friend ostream & operator<<	ostream &output, point & a	ostream	Перегрузка оператора << (вывода) для cin

Описание класса “triangle”.

Элемент структуры	Принимаемое значение (функции) / Тип (элементы)	Возвращаемое значение (функции)	Роль
A, B, C, adj	point		Точки треугольника (adj используется при "умножении")
triangle (конструктор)		triangle	Конструктор треугольника, создающий треугольник со всеми нулевыми координатами
triangle (конструктор)	point a, point b, point c	triangle	Конструктор треугольника, принимающий значения-точки
move	int pt, point xy		Изменение координат точек треугольника (в том числе всего треугольника)
space		long double	Вычисление площади треугольника
print_point	int point_number	bool	Вывод координат точки
operator=	triangle t	triangle	Приравнивание координат первого треугольника второму
operator<	triangle t	bool	Сравнение площадей треугольников, если 1<2 возвращает 1
operator>	triangle t	bool	Сравнение площадей треугольников, если 1>2 возвращает 1
operator*=	double resize_to	triangle	"Умножение" координат треугольника на число
friend istream & operator>>	istream & input, triangle & a	istream	Перегрузка оператора >> (ввода) для cin
friend ostream & operator<<	ostream & output, triangle & a	ostream	Перегрузка оператора << (вывода) для cin

Результаты работы программы.

```
Доступные операции:
1       Базовые операции (создание, выведение координат или площади)
2       Изменение координат треугольников
3       Сравнение треугольников по площади
0       Выход из программы
1
Доступные операции:
1       Создание нового треугольника
2       Выведение координат и площади n-го треугольника
0       Вернуться в предыдущее меню
1
Введите для каждой точки треугольника координаты x и y (в общей
сложности 6):
0 0 1 0 1 1
Доступные операции:
1       Базовые операции (создание, выведение координат или площади)
2       Изменение координат треугольников
3       Сравнение треугольников по площади
0       Выход из программы
1
Доступные операции:
1       Создание нового треугольника
2       Выведение координат и площади n-го треугольника
0       Вернуться в предыдущее меню
2
Введите номер треугольника:
1
{(0, 0), (1, 0), (1, 1)}
Площадь: 0.5
Доступные операции:
1       Базовые операции (создание, выведение координат или площади)
2       Изменение координат треугольников
3       Сравнение треугольников по площади
0       Выход из программы
1
Доступные операции:
1       Создание нового треугольника
2       Выведение координат и площади n-го треугольника
0       Вернуться в предыдущее меню
1
Введите для каждой точки треугольника координаты x и y (в общей
сложности 6):
5 8 3 6 9 0
Доступные операции:
1       Базовые операции (создание, выведение координат или площади)
2       Изменение координат треугольников
3       Сравнение треугольников по площади
0       Выход из программы
1
```

```
Доступные операции:
1       Создание нового треугольника
2       Выведение координат и площади n-го треугольника
0       Вернуться в предыдущее меню
2
Введите номер треугольника:
2
{(5, 8), (3, 6), (9, 0)}
Площадь: 12
Доступные операции:
1       Базовые операции (создание, выведение координат или площади)
2       Изменение координат треугольников
3       Сравнение треугольников по площади
0       Выход из программы
2
Введите номер треугольника:
1
Начальные координаты треугольника:
{(0, 0), (1, 0), (1, 1)}
Доступные операции:
1-3     Перенесение i-й точки треугольника
4       Перенесение 1-го треугольника
5       "Умножение" 0-го треугольника на число
0       Вернуться в предыдущее меню
2
Начальные координаты точки:     (1.000000, 0.000000)
Введите, на какое расстояние перенести точку
9 -5
Новые координаты точки: (10.000000, -5.000000)
Новые координаты треугольника:
{(0, 0), (10, -5), (1, 1)}
Доступные операции:
1       Базовые операции (создание, выведение координат или площади)
2       Изменение координат треугольников
3       Сравнение треугольников по площади
0       Выход из программы
1
Доступные операции:
1       Создание нового треугольника
2       Выведение координат и площади n-го треугольника
0       Вернуться в предыдущее меню
2
Введите номер треугольника:
1
{(0, 0), (10, -5), (1, 1)}
Площадь: 7.5
Доступные операции:
1       Базовые операции (создание, выведение координат или площади)
2       Изменение координат треугольников
3       Сравнение треугольников по площади
0       Выход из программы
```

```
3
Введите номера сравниваемых треугольников:
1 2
Треугольник 1 :
{(0, 0), (10, -5), (1, 1)}
Площадь: 7.5
Треугольник 2 :
{(5, 8), (3, 6), (9, 0)}
Площадь: 12
Площадь треугольника 2 > площади треугольника 1
Доступные операции:
1       Базовые операции (создание, выведение координат или площади)
2       Изменение координат треугольников
3       Сравнение треугольников по площади
0       Выход из программы
2
Введите номер треугольника:
1

Начальные координаты треугольника:
{(0, 0), (10, -5), (1, 1)}
Доступные операции:
1-3     Перенесение i-й точки треугольника
4       Перенесение 1-го треугольника
5       "Умножение" 0-го треугольника на число
0       Вернуться в предыдущее меню
4
Введите, на какое расстояние нужно перенести треугольник (x и y):
5 5
Новые координаты треугольника:
{(5, 5), (15, 0), (6, 6)}
Доступные операции:
1       Базовые операции (создание, выведение координат или площади)
2       Изменение координат треугольников
3       Сравнение треугольников по площади
0       Выход из программы
2
Введите номер треугольника:
1

Начальные координаты треугольника:
{(5, 5), (15, 0), (6, 6)}
Доступные операции:
1-3     Перенесение i-й точки треугольника
4       Перенесение 1-го треугольника
5       "Умножение" 0-го треугольника на число
0       Вернуться в предыдущее меню
5
Начальные координаты:
{(5, 5), (15, 0), (6, 6)}
```

```
Введите, от какой точки рассчитывать умножение (x и y):
15 15
Введите, на что умножить:
4
Новые координаты:
{(-40, -40), (0, -60), (-36, -36)}
Новые координаты треугольника:
{(-40, -40), (0, -60), (-36, -36)}
Доступные операции:
1       Базовые операции (создание, выведение координат или площади)
2       Изменение координат треугольников
3       Сравнение треугольников по площади
0       Выход из программы
3
Введите номера сравниваемых треугольников:
2 1
Треугольник 1 :
{(5, 8), (3, 6), (9, 0)}
Площадь: 12
Треугольник 2 :
{(-40, -40), (0, -60), (-36, -36)}
Площадь: 120
Площадь треугольника 2 > площади треугольника 1
Доступные операции:
1       Базовые операции (создание, выведение координат или площади)
2       Изменение координат треугольников
3       Сравнение треугольников по площади
0       Выход из программы
0
```