Московский авиационный институт
(Национальный исследовательский университет)

Институт: «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»
Дисциплина: «Объектно-ориентированное программирование»

Лабораторная работа № 3

Тема: Механизмы наследования в C++

Студент: Ивенкова Любовь Васильевна
Группа: 80-208
Преподаватель: Чернышов Л.Н.
Дата:
Оценка:

Москва, 2020

1. Постановка задачи

Вариант 28.

Разработать классы *Трапеция, Ромб, 5-угольник;* классы должны наследоваться от базового класса Figure. Фигуры являются фигурами вращения. Все классы должны поддерживать набор общих методов:

1. Вычисление геометрического центра фигуры;
2. Вывод в стандартный поток вывода std::cout координат вершин фигуры;
3. Вычисление площади фигуры.

Создать программу, которая позволяет:

- Вводить из стандартного ввода std::cin фигуры, согласно варианту задания.
- Сохранять созданные фигуры в динамический массив std::vector<Figure*>.
- Вызывать для всего массива общие функции (1-3 см. выше).Т.е. распечатывать для каждой фигуры в массиве геометрический центр, координаты вершин и площадь.
- Необходимо уметь вычислять общую площадь фигур в массиве.
- Удалять из массива фигуру по индексу.

2. Описание программы

1) Создаю абстрактный класс Figure с виртуальными методами:
   - ~Figure() - виртуальный деструктор;
   - void InputCoordinates() - ввод координат;
   - bool Check() - проверка верности входных данных
   - void GeomCenter() - вычисление геометрического центра фигуры;
   - double Area() - вычисление площади фигуры;
   - void Print() - печать координат фигуры;
2) Создаю для Figure три дочерних класса: Trapeze, Rhombus, Pentagon. В них переозаписываю все методы, указанные в родительском классе, а так же создаю два вектора координат x и y: CoordinatesX и CoordinatesY;

3) На вход поступают координаты фигур в виде пар “x y”, разделённых пробелами. У трапеции они должны быть введены в порядке от самой левой координаты до самой правой. У ромба - по кругу. У 5-угольника - в любом порядке. В начале вывожу меню.
4) Площадь трапеции считаю как: S=½h(a+b). Площадь ромба как: S=ah. Площадь 5-угольника как: $\frac{1}{2}\left|\sum_{i=1}^{n}\det\begin{pmatrix} x_i & y_i \\ x_{i+1} & y_{i+1} \end{pmatrix}\right|$, где $x_{n+1} = x_1$, $y_{n+1} = y_1$.
5) В основном теле программы main создаю вектор указателей на класс Figure. По мере работы программы заполняю его фигурами и изменяю его.
6) Ны выход поступают или результаты выполнения методов, или сообщение об ошибке.

3. Набор тестов

test1.t

```
1 1
0 0 1 1 2 1 3 0
2
4
5
1 3
1 0 3 0 4 1 2 3 0 1
6
7
8
9
3 0
8
0
```

test2.t

```
1 1
0 0 0 1 2 1 3 0
3
0
```

5
1 2
0 0 1 1 2 1 2 0

4. Результаты выполнения тестов

Меню:
1 - Ввести фигуру
2 - Вывести координаты фигуры
3 - Удалить фигуру по индексу
4 - Найти геометрический центр фигуры
5 - Найти площадь фигуры
6 - Найти геом. центры всех введённых фигур
7 - Найти площади всех введённых фигур
8 - Вывести координаты всех введённых фигур
9 - Найти общую площадь всех введённых фигур
10 - Меню
0 - Выход

Таблица 1: Набор тестов №1

Команды	Вывод
1 1 0 0 1 1 2 1 3 0	>>>>> Какую фигуру вы хотите ввести? 1. Трапеция 2. Ромб 3. Пятиугольник >>> Введите координаты трапеции (от самой левой до самой правой) Фигура введена
2	>>>>> (0; 0) (1; 1) (2; 1) (3; 0)
4	>>>>> (1.5; 0.5)

5	>>>>> 2
1 3 1 0 3 0 4 1 2 3 0 1	>>>>> Какую фигуру вы хотите ввести? 1. Трапеция 2. Ромб 3. Пятиугольник >>> Введите координаты фигуры Фигура введена
6	>>>>> Геом. центр 1 фигуры: (1.5; 0.5) Геом. центр 2 фигуры: (2; 1)
7	>>>>> Площадь 1 фигуры: 2 Площадь 2 фигуры: 7
8	>>>>> Координаты 1 фигуры: (0; 0) (1; 1) (2; 1) (3; 0) Координаты 2 фигуры: (1; 0) (3; 0) (4; 1) (2; 3) (0; 1)
9	>>>>> 9
3 0	>>>>> Введите индекс эл-та, который вы хотите удалить: >>> Эл-т удалён!

9	>>>>> 7
0	>>>>> Программа завершена!

Таблица 2: Набор тестов №2

Команда	Вывод
1 2 0 0 1 0 2 -1 1 -1	>>>>> Какую фигуру вы хотите ввести? 1. Трапеция 2. Ромб 3. Пятиугольник >>> Введите координаты ромба (по часовой стрелке) Это не ромб!
5	>>>>> Нет введённых фигур
3 0	>>>>> Введите индекс эл-та, который вы хотите удалить: >>> Индекс больше количества введённых фигур!
9	>>>>> Нет введённых фигур

5. Листинг программы

```
/*  Ивенкова Любовь Васильевна, М8О-208Б-19

    https://github.com/Li-Iven/oop_exercise_03

    Вариант 28: Трапеция, Ромб, 5-угольник

Разработать классы согласно варианту задания, классы должны наследоваться от
базового класса Figure. Фигуры являются фигурами вращения.
```

```
Все классы должны поддерживать набор общих методов:
1.  Вычисление геометрического центра фигуры;
2.  Вывод в стандартный поток вывода std::cout координат вершин фигуры;
3.  Вычисление площади фигуры.

Создать программу, которая позволяет:
•   Вводить из стандартного ввода std::cin фигуры, согласно варианту задания.
•   Сохранять созданные фигуры в динамический массив std::vector<Figure*>.
•  Вызывать для всего массива общие функции (1-3 см. выше).Т.е. распечатывать
для каждой фигуры в массиве геометрический центр,
    координаты вершин и площадь.
•   Необходимо уметь вычислять общую площадь фигур в массиве.
•   Удалять из массива фигуру по индексу.
*/

#include <iostream>
#include <algorithm>
#include <cmath>
#include <vector>

using namespace std;

class Figure {

public:
    virtual ~Figure() {};
    virtual void InputCoordinates() = 0;
    virtual void GeomCenter() = 0;
    virtual double Area() = 0;
    virtual bool Check() = 0;
    virtual void Print() = 0;
};

class Trapeze : public Figure { //ориентация точек по x
public:
    Trapeze() {
        CoordinatesX.resize(4);
        CoordinatesY.resize(4);
    }
    ~Trapeze() {};
    void InputCoordinates() {
        std::cout << "\nВведите координаты трапеции (от самой левой до самой
правой)\n";
        for (int i = 0; i < 4; i++) {
            std::cin >> CoordinatesX[i] >> CoordinatesY[i];
        }
    }
    double distance(int x1, int y1, int x2, int y2) {
        return sqrt(pow((x2 - x1), 2) + pow((y2 - y1), 2));
    }
    bool Check() {
        double  LateralSide1  =  distance(CoordinatesX[0],  CoordinatesY[0],
CoordinatesX[1], CoordinatesY[1]);
        double  LateralSide2  =  distance(CoordinatesX[2],  CoordinatesY[2],
CoordinatesX[3], CoordinatesY[3]);
        if ((CoordinatesY[3] - CoordinatesY[0]) != 0 || (CoordinatesY[2] -
CoordinatesY[1]) != 0 || (LateralSide1 != LateralSide2) || LateralSide1==0
|| LateralSide2==0)
            return false;
        else return true;
    }
```

```
    void GeomCenter() {
        double centerX = 0, centerY = 0;
        for (int i = 0; i < 4; i++) {
            centerX += CoordinatesX[i];
        }
        for (int i = 0; i < 4; i++) {
            centerY += CoordinatesY[i];
        }
        std::cout << "(" << centerX/4 << "; " << centerY/4 << ")\n";
    }
    void Print() {
        for (int i = 0; i < 4; i++) {
            std::cout << "(" << CoordinatesX[i] << "; " << CoordinatesY[i]
<< ")\n";
        }
    }
    double Area() {
        int footing1 = CoordinatesX[3] - CoordinatesX[0];
        int footing2 = CoordinatesX[2] - CoordinatesX[1];
        int high = abs(CoordinatesY[3] - CoordinatesY[1]);
        double area = 0.5 * (footing1+footing2) * high;
        return area;
    }
protected:
    std::vector<int> CoordinatesX;
    std::vector<int> CoordinatesY;
};

class Rhombus : public Figure { //ориентация точек по y
public:
    Rhombus() {
        CoordinatesX.resize(4);
        CoordinatesY.resize(4);
    }
    ~Rhombus() {};
    void InputCoordinates() {
        std::cout << "\nВведите координаты ромба (по часовой стрелке)\n";
        for (int i = 0; i < 4; i++) {
            std::cin >> CoordinatesX[i] >> CoordinatesY[i];
        }
    }
    double distance(int x1, int y1, int x2, int y2) {
        return sqrt((x2 - x1)* (x2 - x1) + (y2 - y1)* (y2 - y1));
    }
    bool Check() {
        double LateralSide0 = distance(CoordinatesX[0], CoordinatesY[0],
CoordinatesX[1], CoordinatesY[1]);
        double LateralSide1 = distance(CoordinatesX[1], CoordinatesY[1],
CoordinatesX[2], CoordinatesY[2]);
        double LateralSide2 = distance(CoordinatesX[2], CoordinatesY[2],
CoordinatesX[3], CoordinatesY[3]);
        if (LateralSide0==LateralSide1==LateralSide2)
            return true;
        else return false;
    }
    void GeomCenter() {
        double centerX = 0, centerY = 0;
        for (int i = 0; i < 4; i++) {
            centerX += CoordinatesX[i];
        }
        for (int i = 0; i < 4; i++) {
```

```
            centerY += CoordinatesY[i];
        }
        std::cout << "(" << centerX/4 << "; " << centerY/4 << ")\n";
    }
    void Print() {
        for (int i = 0; i < 4; i++) {
            std::cout << "(" << CoordinatesX[i] << "; " << CoordinatesY[i]
<< ")\n";
        }
    }
    double Area() {
        double  diagonal1  =  distance(CoordinatesX[1],  CoordinatesY[1],
CoordinatesX[3], CoordinatesY[3]);
        double  diagonal2  =  distance(CoordinatesX[0],  CoordinatesY[0],
CoordinatesX[2], CoordinatesY[2]);
        double area = 0.5 * diagonal1 * diagonal2;
        return area;
    }
protected:
    std::vector<int> CoordinatesX;
    std::vector<int> CoordinatesY;
};

class Pentagon : public Figure {
public:
    Pentagon() {
        CoordinatesX.resize(5);
        CoordinatesY.resize(5);
    }
    ~Pentagon() {};

    void InputCoordinates() {
        std::cout << "\nВведите координаты фигуры\n";
        for (int i = 0; i < 5; i++) {
            std::cin >> CoordinatesX[i] >> CoordinatesY[i];
        }
    }
    void GeomCenter() {
        double centerX = 0, centerY = 0;
        for (int i = 0; i < 5; i++) {
            centerX += CoordinatesX[i];
        }
        for (int i = 0; i < 5; i++) {
            centerY += CoordinatesY[i];
        }
        std::cout << "(" << centerX/5 << "; " << centerY/5 << ")\n";
    }
    bool Check() {
        double   side1   =   distance(CoordinatesX[0],   CoordinatesY[0],
CoordinatesX[1], CoordinatesY[1]);
        double   side2   =   distance(CoordinatesX[1],   CoordinatesY[1],
CoordinatesX[2], CoordinatesY[2]);
        double   side3   =   distance(CoordinatesX[2],   CoordinatesY[2],
CoordinatesX[3], CoordinatesY[3]);
        double   side4   =   distance(CoordinatesX[3],   CoordinatesY[3],
CoordinatesX[4], CoordinatesY[4]);
        double   side5   =   distance(CoordinatesX[4],   CoordinatesY[4],
CoordinatesX[0], CoordinatesY[0]);

        if (side1 != 0 && side2 != 0 && side3 != 0 && side4 != 0 && side5 !=
0) return true;
```

```
        else return false;
    }
    void Print() {
        for (int i = 0; i < 5; i++) {
            std::cout << "(" << CoordinatesX[i] << "; " << CoordinatesY[i]
<< ")\n";
        }
    }
    double distance(int x1, int y1, int x2, int y2) {
        return sqrt(pow((x2 - x1), 2) + pow((y2 - y1), 2));
    }
    double Area() {
        double    d1    =    distance(CoordinatesX[0],    CoordinatesY[0],
CoordinatesX[2], CoordinatesY[2]);
        double    d2    =    distance(CoordinatesX[2],    CoordinatesY[2],
CoordinatesX[4], CoordinatesY[4]);
        double   side1   =   distance(CoordinatesX[0],   CoordinatesY[0],
CoordinatesX[1], CoordinatesY[1]);
        double   side2   =   distance(CoordinatesX[1],   CoordinatesY[1],
CoordinatesX[2], CoordinatesY[2]);
        double   side3   =   distance(CoordinatesX[2],   CoordinatesY[2],
CoordinatesX[3], CoordinatesY[3]);
        double   side4   =   distance(CoordinatesX[3],   CoordinatesY[3],
CoordinatesX[4], CoordinatesY[4]);
        double   side5   =   distance(CoordinatesX[4],   CoordinatesY[4],
CoordinatesX[0], CoordinatesY[0]);

        double p1 = 0.5 * (d1 + side1 + side2);
        double p2 = 0.5 * (d2 + side3 + side4);
        double p3 = 0.5 * (d1 + d2 + side5);
        double area1 = sqrt(p1*(p1-d1)*(p1-side1)*(p1-side2));
        double area2 = sqrt(p2*(p2-d2)*(p2-side3)*(p2-side4));
        double area3 = sqrt(p3*(p3-d1)*(p3-d2)*(p3-side5));

        return area1 + area2 + area3;
    }
protected:
    std::vector<int> CoordinatesX;
    std::vector<int> CoordinatesY;
};

void menu() {
    cout << "Меню:" << endl;
    cout << "1 - Ввести фигуру" << endl;
    cout << "2 - Вывести координаты фигуры" << endl;
    cout << "3 - Удалить фигуру по индексу" << endl;
    cout << "4 - Найти геометрический центр фигуры" << endl;
    cout << "5 - Найти площадь фигуры" << endl;
    cout << "6 - Найти геом. центры всех введённых фигур" << endl;
    cout << "7 - Найти площади всех введённых фигур" << endl;
    cout << "8 - Вывести координаты всех введённых фигур" << endl;
    cout << "9 - Найти общую площадь всех введённых фигур" << endl;
    cout << "10 - Меню" << endl;
    cout << "0 - Выход" << endl;
}


int main()
{
    setlocale(LC_ALL, "RUS");
```

```
    vector<Figure*> figures;
    Figure* figure;

    double sum;

    int x;
    menu();
    cout << ">>>>> ";
    while (cin >> x) {
        switch (x) {
        case 0:
            return 0;
            break;
        case 1:
            cout << "\nКакую фигуру вы хотите ввести?\n 1. Трапеция\n 2.
Ромб\n 3. 5-угольник\n\n>>> ";
            int type;
            cin >> type;
            switch (type) {
            case 1:
                figure = new Trapeze();
                figure->InputCoordinates();
                if (figure->Check()) {
                    figures.push_back(figure);
                    cout << "\nФигура введена\n";
                }
                else cout << "Это не равносторонняя трапеция!\n";
                break;
            case 2:
                figure = new Rhombus();
                figure->InputCoordinates();
                if (figure->Check()) {
                    figures.push_back(figure);
                    cout << "\nФигура введена\n";
                }
                else cout << "Это не ромб!\n";
                break;
            case 3:
                figure = new Pentagon();
                figure->InputCoordinates();
                if (figure->Check()) {
                    figures.push_back(figure);
                    cout << "\nФигура введена\n";
                }
                else cout << "Это не 5-угольник!\n";
                break;
            default:
                cout << "Всего три вида фигур!\n";
                break;
            }
            break;
        case 2:
            if (figures.size() == 0) cout << "Нет введённых фигур\n";
            else figures[figures.size() - 1]->Print();
            break;
        case 3:
            size_t index;
            cout   <<   "\nВведите   индекс   эл-та,   который   вы   хотите
удалить:\n\n>>> ";
            cin >> index;
            if (index < figures.size()) {
```

```
            figures.erase(figures.begin() + index);
            cout << "\nЭл-т удалён!\n";
        }
        else cout << "Индекс больше количества введённых фигур!\n";
        break;
    case 4:
        if (figures.size() == 0) cout << "Нет введённых фигур\n";
        else figures[figures.size() - 1]->GeomCenter();
        break;
    case 5:
        if (figures.size() == 0) cout << "Нет введённых фигур\n";
        else cout << figures[figures.size() - 1]->Area();
        break;
    case 6:
        if (figures.size() == 0) cout << "Нет введённых фигур\n";
        else {
            for (size_t i = 0; i < figures.size(); i++) {
                cout << "\nГеом. центр " << i + 1 << " фигуры:\n";
                figures[i]->GeomCenter();
                cout << "\n";
            }
        }
        break;
    case 7:
        if (figures.size() == 0) cout << "Нет введённых фигур\n";
        else {
            for (size_t i = 0; i < figures.size(); i++) {
                cout << "\nПлощадь " << i + 1 << " фигуры:\n";
                cout << figures[i]->Area() << "\n";
            }
        }
        break;
    case 8:
        if (figures.size() == 0) cout << "Нет введённых фигур\n";
        else {
            for (size_t i = 0; i < figures.size(); i++) {
                cout << "\nКоординаты " << i + 1 << " фигуры:\n";
                figures[i]->Print();
                cout << "\n";
            }
        }
        break;
    case 9:
        sum = 0;
        for (auto ptr : figures) {
            sum += ptr->Area();
        }
        cout << sum;
        break;
    case 10:
        menu();
        break;
    default:
        cout << "Неизвестная команда!\n";
        menu();
        break;
    }
    std::cout << " " << std::endl;
    std::cout << ">>>>> ";
}
cout << "Программа завершена!" << std::endl;
```

```
    return 0;
}
```

6. **Выводы**

   В этой лабораторной работе я познакомилась с наследованием в C++и научилась его применять, научилась работать с абстрактными классами и виртуальными методами классов.

7. **Список литературы**

Формула площади Гаусса – URL : https://ru.wikipedia.org/wiki/Формула_площади_Гаусса

Барицентр – URL: https://ru.wikipedia.org/wiki/Барицентр