**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 С++

Студент: Попов Илья Павлович

Группа: 80-206

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2021

# Постановка задачи

## Разработать классы согласно варианту задания, классы должны наследоваться от базового класса Figure.

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.

Программа должна иметь следующие возможности:

* 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.
* вычислять общую площадь фигур в массиве.
* удалять из массива фигуру по индексу.

**Вариант 20:**

1. Трапеция
2. Ромб
3. Пятиугольник

# Описание программы

Базовым классом программы является класс Figure, с реализованными виртуальными методами:

## *class Figure {//родительский класс фигуры с виртуальными методами*

*public:*

*virtual double Area() const = 0;//подсчет площади*

*virtual Point Geometric\_center() const = 0;//поиск геом центра*

*virtual void Coordinates() const = 0;//вывод координат точек*

*virtual ~Figure() {}*

*};*

От этого базового класса наследуются три класса фигур:

*class Trapezoid : public Figure {...}*

*class Rhomb : public Figure {...}*

*class Pentagon : public Figure {...}*

Фигуры считываются со стандартного ввода поточечно(пользователь последовательно вводит пары координат точек фигуры), далее происходит проверка введенных значений, и, если фигура считана корректно, она отправляется в вектор vector<Figure\*>. Для взаимодействия с этим вектором предусмотрены функции:

1. add - добавить фигуру
2. del - удалить фигуру
3. show - распечатывать для каждой фигуры в массиве все ее характеристики
4. sum - посчитать суммарную площадь всех фигур в массиве
5. menu - вывести меню повторно

# Набор тестов

## Тест №1:

Демонстрирует корректность работы программы для фигур у которых точки заданы целыми/дробными координатами, демонстрирует работу функций взаимодействия с этим вектором.

*Доступные функции:*

*add - добавить фигуру*

*del - удалить фигуру*

*show - распечатывать для каждой фигуры в массиве все ее характеристики*

*sum - посчитать суммарную площадь всех фигур в массиве*

*menu - вывести меню повторно*

*exit - выход*

*add*

*Выберите тип фигуры:*

*t - трапеция*

*r - ромб*

*p - пятиугольник*

*t*

*Введите в порядке последовательного обхода фигуры точки в виде x y*

*0 0*

*0 4*

*2 4*

*6 0*

*Фигура успешно добавлена*

*add*

*Выберите тип фигуры:*

*t - трапеция*

*r - ромб*

*p - пятиугольник*

*r*

*Введите в порядке последовательного обхода фигуры точки в виде x y*

*0 0*

*-1 2*

*0 4*

*1 2*

*Фигура успешно добавлена*

*add*

*Выберите тип фигуры:*

*t - трапеция*

*r - ромб*

*p - пятиугольник*

*p*

*Введите в порядке последовательного обхода фигуры точки в виде x y*

*0 0*

*-4 3*

*-2 4*

*2 4*

*4 3*

*Фигура успешно добавлена*

*show*

*Площадь фигуры с индексом 0: 16*

*Геометрический центр фигуры с индексом 0: x = 2 y = 2*

*Координаты фигуры с индексом 0: (0, 0) (0, 4) (2, 4) (6, 0)*

*--------------------------------------------------------------*

*Площадь фигуры с индексом 1: 4*

*Геометрический центр фигуры с индексом 1: x = 0 y = 2*

*Координаты фигуры с индексом 1: (0, 0) (-1, 2) (0, 4) (1, 2)*

*--------------------------------------------------------------*

*Площадь фигуры с индексом 2: 18*

*Геометрический центр фигуры с индексом 2: x = 0 y = 2.8*

*Координаты фигуры с индексом 2: (0, 0) (-4, 3) (-2, 4) (2, 4) (4, 3)*

*--------------------------------------------------------------*

*sum*

*38*

*menu*

*Доступные функции:*

*add - добавить фигуру*

*del - удалить фигуру*

*show - распечатывать для каждой фигуры в массиве все ее характеристики*

*sum - посчитать суммарную площадь всех фигур в массиве*

*menu - вывести меню повторно*

*exit - выход*

*exit*

*Завершение работы*

## Тест №2:

Демонстрирует устойчивость программы к ошибкам

*sadad*

*Некорректный ввод!*

*add*

*Выберите тип фигуры:*

*t - трапеция*

*r - ромб*

*p - пятиугольник*

*h*

*Некорректный ввод!*

*add*

*Выберите тип фигуры:*

*t - трапеция*

*r - ромб*

*p - пятиугольник*

*t*

*Введите в порядке последовательного обхода фигуры точки в виде x y*

*0.9 -gggg*

*Ошибка! Некорректный ввод.*

*add*

*Выберите тип фигуры:*

*t - трапеция*

*r - ромб*

*p - пятиугольник*

*t*

*Введите в порядке последовательного обхода фигуры точки в виде x y*

*0 0*

*1 1*

*2 2*

*3 3*

*Введенная фигура не является трапецией*

*add*

*Выберите тип фигуры:*

*t - трапеция*

*r - ромб*

*p - пятиугольник*

*r*

*Введите в порядке последовательного обхода фигуры точки в виде x y*

*0 0*

*1 1*

*2 2*

*3 3*

*Введенная фигура не является ромбом*

# Результаты выполнения тестов

Представлены выше, с целью упростить прочтение.

# Листинг программы

*/\**

*Попов Илья*

*Группа М80-206Б-20*

*Вариант 20:*

*Трапеция*

*Ромб*

*5-угольник*

*Разработать классы согласно варианту задания, классы должны наследоваться от базового класса Figure.*

*Все классы должны поддерживать набор общих методов :*

*Вычисление геометрического центра фигуры.*

*Вывод в стандартный поток вывода std::cout координат вершин фигуры;*

*Вычисление площади фигуры.*

*Программа должна иметь следующие возможности :*

*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.*

*вычислять общую площадь фигур в массиве.*

*удалять из массива фигуру по индексу.*

*\*/*

*#include <iostream>*

*#include <cmath>*

*#include <string>*

*#include <vector>*

*using namespace std;*

*bool is\_number(const string& s) {//проверка корректности вводимых координат*

*bool point = false;*

*for (int i = 0; i < s.length(); ++i) {*

*if (s[i] == '-' && i == 0) {*

*continue;*

*}*

*else if (s[i] == '.') {*

*if ((i == 0 || i == s.length() - 1) || point) {*

*return false;*

*}*

*else {*

*point = true;*

*}*

*}*

*else if (s[i] < '0' || s[i] > '9') { return false; }*

*}*

*return true;*

*}*

*class Point {//класс точки, с перегруженными операторами ввода и вывода*

*private:*

*double x;*

*double y;*

*public:*

*Point() : x(0), y(0) {}*

*Point(double \_x, double \_y) : x(\_x), y(\_y) {}*

*double getX() const { return x; }*

*double getY() const { return y; }*

*friend istream& operator>> (istream& in, Point& p);*

*};*

*istream& operator>> (istream& in, Point& p) {*

*string \_x, \_y; cin >> \_x >> \_y;*

*if (!is\_number(\_x) || !is\_number(\_y)) {*

*cout << "Ошибка! Некорректный ввод.\n";*

*}*

*p.x = stod(\_x); p.y = stod(\_y);*

*return in;*

*}*

*ostream& operator<< (ostream& out, const Point& p) {*

*cout << "(" << p.getX() << ", " << p.getY() << ") ";*

*return out;*

*}*

*double fabss(double a) {*

*if (a >= 0) { return a; }*

*else { return -a; }*

*}*

*double calc\_area(const vector<Point>& v) {//функция, считающая площадь для всех фигур*

*double cur\_res = 0;*

*double res = 0;*

*int i;*

*for (i = 0; i < v.size() - 1; i++) {*

*cur\_res += v[i].getX() \* v[i + 1].getY() - v[i].getY() \* v[i + 1].getX();*

*}*

*cur\_res += v[i].getX() \* v[0].getY() - v[i].getY() \* v[0].getX();*

*res = fabss(cur\_res) / 2;*

*return res;*

*}*

*Point calc\_geom\_center(const vector<Point>& v) {//функция, считающая геом центр для всех фигур*

*double sumX = 0;*

*double sumY = 0;*

*for (int i = 0; i < v.size(); i++) {*

*sumX += v[i].getX();*

*sumY += v[i].getY();*

*}*

*Point a(sumX / v.size(), sumY / v.size());*

*return a;*

*}*

*double dlina(const Point& a, const Point& b) {*

*return sqrt(pow((b.getX() - a.getX()), 2) + pow((b.getY() - a.getY()), 2));*

*}*

*class Figure {//родительский класс фигуры с виртуальными методами*

*public:*

*virtual double Area() const = 0;//подсчет площади*

*virtual Point Geometric\_center() const = 0;//поиск геом центра*

*virtual void Coordinates() const = 0;//вывод координат точек*

*virtual ~Figure() {}*

*};*

*class Trapezoid : public Figure {//наследуемый класс трапеция*

*Point a, b, c, d;*

*vector<Point> v;*

*public:*

*Trapezoid(Point \_a, Point \_b, Point \_c, Point \_d) : a(\_a), b(\_b), c(\_c), d(\_d) {*

*v.push\_back(\_a);*

*v.push\_back(\_b);*

*v.push\_back(\_c);*

*v.push\_back(\_d);*

*}*

*~Trapezoid() {}*

*double Area() const {*

*return calc\_area(v);*

*}*

*Point Geometric\_center() const {*

*return calc\_geom\_center(v);*

*}*

*void Coordinates() const {*

*cout << a << b << c << d;*

*}*

*};*

*class Rhomb : public Figure {//наследуемый класс ромб*

*Point a, b, c, d;*

*vector<Point> v;*

*public:*

*Rhomb(Point \_a, Point \_b, Point \_c, Point \_d) : a(\_a), b(\_b), c(\_c), d(\_d) {*

*v.push\_back(\_a);*

*v.push\_back(\_b);*

*v.push\_back(\_c);*

*v.push\_back(\_d);*

*}*

*~Rhomb() {}*

*double Area() const {*

*return calc\_area(v);*

*}*

*Point Geometric\_center() const {*

*return calc\_geom\_center(v);*

*}*

*void Coordinates() const {*

*cout << a << b << c << d;*

*}*

*};*

*class Pentagon : public Figure {//наследуемый класс пятиугольник*

*Point a, b, c, d, e;*

*vector<Point> v;*

*public:*

*Pentagon(Point \_a, Point \_b, Point \_c, Point \_d, Point \_e) : a(\_a), b(\_b), c(\_c), d(\_d), e(\_e) {*

*v.push\_back(\_a);*

*v.push\_back(\_b);*

*v.push\_back(\_c);*

*v.push\_back(\_d);*

*v.push\_back(\_e);*

*}*

*~Pentagon() {}*

*double Area() const {*

*return calc\_area(v);*

*}*

*Point Geometric\_center() const {*

*return calc\_geom\_center(v);*

*}*

*void Coordinates() const {*

*cout << a << b << c << d << e;*

*}*

*};*

*void menu() {*

*cout << "\nДоступные функции:\nadd - добавить фигуру\ndel - удалить фигуру\nshow - распечатывать для каждой фигуры в массиве все ее характеристики\nsum - посчитать суммарную площадь всех фигур в массиве\nmenu - вывести меню повторно\nexit - выход\n\n\n";*

*}*

*double koef\_nakl(const Point& a, const Point& b) {//побочная функция, проверяющая параллельность прямых*

*double k = (a.getY() - b.getY()) / (a.getX() - b.getX());*

*//cout << k << endl;*

*return k;*

*}*

*bool trapezoid\_check(const Point& a, const Point& b, const Point& c, const Point& d) {//проверка введенной фигура на то, является ли она трапецией*

*Trapezoid t(a, b, c, d);*

*if (t.Area() == 0) {*

*return false;*

*}*

*//case 1: ab || cd*

*if (koef\_nakl(a, b) == koef\_nakl(c, d)) { return true; }*

*//case 2: bc || ad*

*else if (koef\_nakl(b, c) == koef\_nakl(a, d)) { return true; }*

*return false;*

*}*

*bool rhomb\_check(const Point& a, const Point& b, const Point& c, const Point& d) {//проверка введенной фигура на то, является ли она ромбом*

*if (dlina(a, b) == dlina(b, c) && dlina(c, d) == dlina(a, d) && dlina(b, c) == dlina(c, d)) { return true; }*

*return false;*

*}*

*int main() {*

*setlocale(LC\_ALL, "rus");*

*vector<Figure\*> vf;*

*string str, fig;*

*bool exit = false;*

*menu();*

*while (!exit) {*

*cin >> str;*

*if (str == "add") {*

*cout << "\nВыберите тип фигуры:\nt - трапеция\nr - ромб\np - пятиугольник\n";*

*cin >> fig;*

*if (fig == "t") {*

*Point a1, b1, c1, d1;*

*cout << "Введите в порядке последовательного обхода фигуры точки в виде x y\n";*

*cin >> a1 >> b1 >> c1 >> d1;*

*if (trapezoid\_check(a1, b1, c1, d1)) {*

*vf.push\_back(new Trapezoid(a1, b1, c1, d1));*

*cout << "Фигура успешно добавлена\n";*

*}*

*else {*

*cout << "Введенная фигура не является трапецией\n";*

*}*

*}*

*else if (fig == "r") {*

*Point a1, b1, c1, d1;*

*cout << "Введите в порядке последовательного обхода фигуры точки в виде x y\n";*

*cin >> a1 >> b1 >> c1 >> d1;*

*if (rhomb\_check(a1, b1, c1, d1)) {*

*vf.push\_back(new Rhomb(a1, b1, c1, d1));*

*cout << "Фигура успешно добавлена\n";*

*}*

*else {*

*cout << "Введенная фигура не является ромбом\n";*

*}*

*}*

*else if (fig == "p") {*

*Point a1, b1, c1, d1, e1;*

*cout << "Введите в порядке последовательного обхода фигуры точки в виде x y\n";*

*cin >> a1 >> b1 >> c1 >> d1 >> e1;*

*vf.push\_back(new Pentagon(a1, b1, c1, d1, e1));*

*cout << "Фигура успешно добавлена\n";*

*}*

*else {*

*cout << "Некорректный ввод!\n\n\n";*

*}*

*}*

*else if (str == "show") {*

*for (int i = 0; i < vf.size(); i++) {*

*cout << "Площадь фигуры с индексом " << i << ": " << vf[i]->Area() << endl;*

*Point cur = vf[i]->Geometric\_center();*

*cout << "Геометрический центр фигуры с индексом " << i << ": " << "x = " << cur.getX() << " y = " << cur.getY() << endl;*

*cout << "Координаты фигуры с индексом " << i << ": ";*

*vf[i]->Coordinates();*

*cout << "\n--------------------------------------------------------------\n";*

*}*

*}*

*else if (str == "sum") {*

*double sum = 0;*

*for (int i = 0; i < vf.size(); i++) {*

*sum += vf[i]->Area();*

*}*

*cout << sum << endl;*

*}*

*else if (str == "del") {*

*int n, i;*

*cout << "Введите индекс удаляемого элемента\n";*

*cin >> n;*

*if (vf.size() > n){*

*delete vf[n];*

*vf.erase(vf.begin() + n);*

*}*

*else {*

*cout << "Данного элемента в массиве нет!\n";*

*}*

*}*

*else if (str == "menu") {*

*menu();*

*}*

*else if (str == "exit") {*

*cout << "Завершение работы\n";*

*exit = true;*

*}*

*else {*

*cout << "Некорректный ввод!\n\n\n";*

*}*

*}*

*}*

*/\**

*\* Тестовые значения*

*///////////////////////////////////*

*add t*

*0 0*

*0 4*

*2 4*

*6 0*

*add r*

*0 0*

*-1 2*

*0 4*

*1 2*

*add p*

*0 0*

*-4 3*

*-2 4*

*2 4*

*4 3*

*//////////////////////////////////////*

*add t*

*-1 0*

*-4 5*

*8 5*

*2.7 0*

*add r*

*0 1*

*1.1 0*

*0 -1.1*

*-1 0*

*add p*

*1 1*

*7.5 1*

*8 -1*

*2 -3.5*

*-1 -1*

*\*/*

# ВЫВОДЫ

В ходе этой лабораторной работы я познакомился с таким понятием, наследование классов, виртуальные методы и многое другое. Наследование - очень удобная вещь при работе с большим количеством объектов, с ее помощью классы, имеющие общие методы можно представить потомками какого-то абстрактного класса, что существенно упрощает восприятие кода и упрощает его написание.

# ЛИТЕРАТУРА

1. Справочник по языку С++ [Электронный ресурс]. URL: https://ravesli.com (дата обращения:17.10.2021).