Московский Авиационный Институт

(Национальный Исследовательский Университет)

Кафедра 806 «Вычислительная информатика и программирование»

Факультет: «Информационные технологии и прикладная математика»

Лабораторная работа

Дисциплина: «Объектно-ориентированное программирование»

III семестр

Задание 3: «Наследование, полиморфизм»

|  |  |
| --- | --- |
| Группа: | М8О-108Б-18, №6 |
| Студент: | Васильева Василиса Евгеньевна |
| Преподаватель: | Журавлёв Андрей Андреевич |
| Оценка: |  |
| Дата: | 28.10.2019 |

Москва, 2019

1. **Задание**

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;

*Вариант 6: пятиугольник, шестиугольник, восьмиугольник*

1. **Адрес репозитория на GitHub**

https://github.com/vasilisavasileva/oop\_excercise\_3

1. **Код программы на С++**

Vertex.h

#pragma once

#include<iostream>

struct Vertex {

double x, y;

};

std::istream& operator>> (std::istream& is, Vertex& l);

std::ostream& operator<< (std::ostream& os, const Vertex& l);

Vertex.cpp

#include"Vertex.h"

std::istream& operator>> (std::istream& is, Vertex& l) {

return is >> l.x >> l.y;

};

std::ostream& operator<< (std::ostream& os, const Vertex& l) {

return os << l.x << ' ' << l.y;

};

figure.h

#pragma once

#include<iostream>

#include<cmath>

#include"Vertex.h"

class Figure {

public:

virtual Vertex calculateCenter() const = 0;

virtual double calculateArea() const = 0;

virtual void printVertex(std::ostream&) const = 0;

};

std::ostream& operator<< (std::ostream& os, const Figure& f);

figure.cpp

#include"figure.h"

std::ostream& operator<< (std::ostream& os, const Figure& f) {

f.printVertex(os);

return os;

};

figures.h

#pragma once

#include<stdio.h>

#include"figure.h"

class Pentagon : public Figure {

private:

Vertex v[5];

public:

Pentagon();

Pentagon(std::istream& is);

double calculateArea() const override;

Vertex calculateCenter() const override;

void printVertex(std::ostream&) const override;

};

class Hexagon : public Figure {

private:

Vertex v[6];

public:

Hexagon();

Hexagon(std::istream& is);

double calculateArea() const override;

Vertex calculateCenter() const override;

void printVertex(std::ostream&) const override;

};

class Octagon : public Figure {

private:

Vertex v[8];

public:

Octagon();

Octagon(std::istream& is);

double calculateArea() const override;

Vertex calculateCenter() const override;

void printVertex(std::ostream&) const override;

};

figures.cpp

#include"figures.h"

#include<cmath>

Pentagon::Pentagon() {};

Pentagon::Pentagon(std::istream& is) {

Vertex l;

for (int i = 0; i < 5; i++) {

is >> l.x >> l.y;

v[i] = l;

}

};

double Pentagon::calculateArea() const{

double Area = 0;

for (int i = 0; i < 5; i++) {

Area += (v[i].x) \* (v[(i + 1)%5].y) - (v[(i + 1)%5].x)\*(v[i].y);

}

Area \*= 0.5;

return abs(Area);

};

Vertex Pentagon::calculateCenter() const {

Vertex center;

double xCenter = 0;

double yCenter = 0;

for (int i = 0; i < 5; i++) {

xCenter += v[i].x;

yCenter += v[i].y;

}

xCenter = xCenter / 5;

yCenter = yCenter / 5;

center.x = xCenter;

center.y = yCenter;

return center;

};

void Pentagon::printVertex(std::ostream& os) const {

os << "Pentagon:\n";

for (int i = 0; i < 5; i++) {

os << v[i] << std::endl;

}

os << '\b';

};

Hexagon::Hexagon() {};

Hexagon::Hexagon(std::istream& is) {

Vertex l;

for (int i = 0; i < 6; i++) {

is >> l.x >> l.y;

v[i] = l;

}

};

double Hexagon::calculateArea() const {

double Area = 0;

for (int i = 0; i < 6; i++) {

Area += (v[i].x) \* (v[(i + 1) % 6].y) - (v[(i + 1) % 6].x) \* (v[i].y);

}

Area \*= 0.5;

return abs(Area);

};

Vertex Hexagon::calculateCenter() const {

Vertex center;

double xCenter = 0;

double yCenter = 0;

for (int i = 0; i < 6; i++) {

xCenter += v[i].x;

yCenter += v[i].y;

}

xCenter = xCenter / 6;

yCenter = yCenter / 6;

center.x = xCenter;

center.y = yCenter;

return center;

};

void Hexagon::printVertex(std::ostream& os) const {

os << "Hexagon:\n";

for (int i = 0; i < 6; i++) {

os << v[i] << std::endl;

}

os << '\b';

};

Octagon::Octagon() {};

Octagon::Octagon(std::istream& is) {

Vertex l;

for (int i = 0; i < 8; i++) {

is >> l.x >> l.y;

v[i] = l;

}

};

double Octagon::calculateArea() const {

double Area = 0;

for (int i = 0; i < 8; i++) {

Area += (v[i].x) \* (v[(i + 1) % 8].y) - (v[(i + 1) % 8].x) \* (v[i].y);

}

Area \*= 0.5;

return abs(Area);

};

Vertex Octagon::calculateCenter() const {

Vertex center;

double xCenter = 0;

double yCenter = 0;

for (int i = 0; i < 8; i++) {

xCenter += v[i].x;

yCenter += v[i].y;

}

xCenter = xCenter / 5;

yCenter = yCenter / 5;

center.x = xCenter;

center.y = yCenter;

return center;

};

void Octagon::printVertex(std::ostream& os) const {

os << "Octagon:\n";

for (int i = 0; i < 8; i++) {

os << v[i] << std::endl;

}

os << '\b';

};

main.cpp

#include"figure.h"

#include"figures.h"

#include<stdio.h>

#include<vector>

void printMenu() {

std::cout << "Доступные команды:" << std::endl;

std::cout << "0. Выход" << std::endl;

std::cout << "1. Добавить фигуру" << std::endl;

std::cout << "2. Вызвать функцию для всех фигур" << std::endl;

std::cout << "3. Удалить фигуру по индексу" << std::endl;

std::cout << "4. Вывести это меню" << std::endl;

}

int main() {

setlocale(LC\_ALL, "rus");

Figure\* s;

std::vector<Figure\*> v1;

double x1, x2, x3, x4, x5, x6, x7, x8, y1, y2, y3, y4, y5, y6, y7, y8;

printMenu();

while (true) {

std::cout << "Номер: ";

int k;

std::cin >> k;

std::vector<Figure\*> next;

switch (k) {

case 0:

for (size\_t i = 0; i < v1.size(); i++) {

delete v1[i];

}

return 0;

case 1:

std::cout << "1. Пятиугольник" << std::endl;

std::cout << "2. Шестиугольник" << std::endl;

std::cout << "3. Восьмиугольник" << std::endl;

std::cout << "Номер" << std::endl;

int a;

std::cin >> a;

if (a < 1 || a > 3) {

std::cout << "Неверный номер" << std::endl;

break;

}

switch (a) {

case 1:

std::cout << "Введите координаты: ";

s = new Pentagon(std::cin);

break;

case 2:

std::cout << "Введите координаты: ";

s = new Hexagon(std::cin);

break;

case 3:

std::cout << "Введите координаты: ";

s = new Octagon(std::cin);

}

v1.push\_back(s);

case 2:

std::cout << "1. Посчитать площадь" << std::endl;

std::cout << "2. Посчитать центр" << std::endl;

std::cout << "3. Распечатать координаты" << std::endl;

std::cout << "Номер: ";

int b;

std::cin >> b;

if (b < 1 || b > 5) {

std::cout << "Неверный номер" << std::endl;

break;

}

switch (b) {

case 1:

std::cout << "Areas:" << std::endl;

for (int i = 0; i < v1.size(); i++) {

std::cout << (\*v1[i]).calculateArea() << std::endl;

}

break;

case 2:

std::cout << "Centers:" << std::endl;

for (int i = 0; i < v1.size(); i++) {

std::cout << (\*v1[i]).calculateCenter() << std::endl;

}

break;

case 3:

for (int i = 0; i < v1.size(); i++) {

(\*v1[i]).printVertex(std::cout);

std::cout << std::endl;

}

break;

}

break;

case 3:

std::cout << "Индекс: \n";

size\_t id;

std::cin >> id;

if (id < 0 || id > v1.size() - 1) {

std::cout << "Индекс выходит за границы массива" << std::endl;

break;

}

delete v1[id];

v1.erase(v1.begin() + id);

break;

case 4:

printMenu();

break;

}

}

}

1. **Результаты выполнения тестов**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Фигура** | **Координаты** | **Центр** | **Площадь** |
| 1. | Пятиугольник | [0,1] [0,2] [2,2] [3,1] [2,0] | [1.4, 1.2] | 4 |
| 2. | Шестиугольник | [0,0] [1,1] [2,2] [3,3] [4,4] [4,0] | [2.33333, 1.66667] | 8 |
| 3. | Восьмиугольник | [0,0] [1,1] [2,2] [3,3] [4,4] [5,5] [6,6] [6,0] | [5.4] [4.2] | 18 |

1. **Объяснение результатов работы программы**

Программа просит на вход координаты выбранной фигуры. В зависимости от выбора пункта меню, программа высчитывает центр фигуры или площадь, а также выводит их.

1. **Вывод**

Я познакомилась с принципом наследования классов, узнала о преимуществах такого способа. Абстрактно-описанный метод в родительском классе может быть переопределен в классах-наследниках в зависимости от специфики конкретного подкласса. Это упрощает код и делает его более удобным и понятным.