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: |  |
| Студент: |  |
| 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 динамического массива (std::vector) фигуру по заданному пользователем индексу;

*Вариант 22: пятиугольник, шестиугольник, восьмиугольник*

1. **TextCases**

|  |
| --- |
| **Test 1.**  Commands:  0. Exit  1. Add figure  2. Function call for all elements  3. Delete figure by index  Input number of chosen function: 1  1. Input the pentagon  2. Input the hexagon  3. Input the octagon  Input the number of chosen figure: 1  Input coordinates of vertexes of pentagon:  Input the coordinate 'x': 213  Input the coordinate 'y': 213  Input the coordinate 'x': 123  Input the coordinate 'y': 123  Input the coordinate 'x': 123  Input the coordinate 'y': 123  Input the coordinate 'x': 12  Input the coordinate 'y': 123  Input the coordinate 'x': 21  Input the coordinate 'y': 123  Commands:  0. Exit  1. Add figure  2. Function call for all elements  3. Delete figure by index  Input number of chosen function: 1  1. Input the pentagon  2. Input the hexagon  3. Input the octagon  Input the number of chosen figure: 2  Input coordinates of vertexes of hexagon:  Input the coordinate 'x': 231  Input the coordinate 'y': 123  Input the coordinate 'x': 123  Input the coordinate 'y': 123  Input the coordinate 'x': 3212  Input the coordinate 'y': 123  Input the coordinate 'x': 12  Input the coordinate 'y': 213  Input the coordinate 'x': 12  Input the coordinate 'y': 123  Input the coordinate 'x': 12  Input the coordinate 'y': 123  Commands:  0. Exit  1. Add figure  2. Function call for all elements  3. Delete figure by index  Input number of chosen function: 1  1. Input the pentagon  2. Input the hexagon  3. Input the octagon  Input the number of chosen figure: 3  Input coordinates of vertexes of octagon:  Input the coordinate 'x': 21  Input the coordinate 'y': 24  Input the coordinate 'x': 12  Input the coordinate 'y': 124  Input the coordinate 'x': 214  Input the coordinate 'y': 214  Input the coordinate 'x': 21  Input the coordinate 'y': 213  Input the coordinate 'x': 124  Input the coordinate 'y': 12  Input the coordinate 'x': 241  Input the coordinate 'y': 12  Input the coordinate 'x': 241  Input the coordinate 'y': 124  Input the coordinate 'x': 12  Input the coordinate 'y': 124  Commands:  0. Exit  1. Add figure  2. Function call for all elements  3. Delete figure by index  Input number of chosen function: 2  1. Count the area of figures  2. Count the center of figures  3. Print coordinates of figures  Input the number of chosen function: 3  The coordinates of 1 figure.  Pentagon:  Coordinate 'x': 213; Coordinate 'y': 213  Coordinate 'x': 123; Coordinate 'y': 123  Coordinate 'x': 123; Coordinate 'y': 123  Coordinate 'x': 12; Coordinate 'y': 123  Coordinate 'x': 21; Coordinate 'y': 123  The coordinates of 2 figure.  Hexagon:  Coordinate 'x': 231; Coordinate 'y': 123  Coordinate 'x': 123; Coordinate 'y': 123  Coordinate 'x': 3212; Coordinate 'y': 123  Coordinate 'x': 12; Coordinate 'y': 213  Coordinate 'x': 12; Coordinate 'y': 123  Coordinate 'x': 12; Coordinate 'y': 123  The coordinates of 3 figure.  Octagon:  Coordinate 'x': 21; Coordinate 'y': 24  Coordinate 'x': 12; Coordinate 'y': 124  Coordinate 'x': 214; Coordinate 'y': 214  Coordinate 'x': 21; Coordinate 'y': 213  Coordinate 'x': 124; Coordinate 'y': 12  Coordinate 'x': 241; Coordinate 'y': 12  Coordinate 'x': 241; Coordinate 'y': 124  Coordinate 'x': 12; Coordinate 'y': 124 |
| **Test 2.**  Commands:  0. Exit  1. Add figure  2. Function call for all elements  3. Delete figure by index  Input number of chosen function: 1  1. Input the pentagon  2. Input the hexagon  3. Input the octagon  Input the number of chosen figure: 1  Input coordinates of vertexes of pentagon:  Input the coordinate 'x': 123  Input the coordinate 'y': 123  Input the coordinate 'x': 12312  Input the coordinate 'y': 312  Input the coordinate 'x': 4  Input the coordinate 'y': 12  Input the coordinate 'x': 12  Input the coordinate 'y': 124  Input the coordinate 'x': 12  Input the coordinate 'y': 214  Commands:  0. Exit  1. Add figure  2. Function call for all elements  3. Delete figure by index  Input number of chosen function: 1  1. Input the pentagon  2. Input the hexagon  3. Input the octagon  Input the number of chosen figure: 124  No figure with such number  Commands:  0. Exit  1. Add figure  2. Function call for all elements  3. Delete figure by index  Input number of chosen function: 12  No function with such number.  Commands:  0. Exit  1. Add figure  2. Function call for all elements  3. Delete figure by index  Input number of chosen function: 1  1. Input the pentagon  2. Input the hexagon  3. Input the octagon  Input the number of chosen figure: 2  Input coordinates of vertexes of hexagon:  Input the coordinate 'x': 214  Input the coordinate 'y': 12  Input the coordinate 'x': 12  Input the coordinate 'y': 124  Input the coordinate 'x': 124  Input the coordinate 'y': 12124  Input the coordinate 'x': 214  Input the coordinate 'y': 12  Input the coordinate 'x': 241  Input the coordinate 'y': 214  Input the coordinate 'x': 241  Input the coordinate 'y': 12  Commands:  0. Exit  1. Add figure  2. Function call for all elements  3. Delete figure by index  Input number of chosen function: 3  Input the index of figure for deleting:  2  The index is out of bounds  Commands:  0. Exit  1. Add figure  2. Function call for all elements  3. Delete figure by index  Input number of chosen function: 3  Input the index of figure for deleting:  1  Commands:  0. Exit  1. Add figure  2. Function call for all elements  3. Delete figure by index  Input number of chosen function: 3  Input the index of figure for deleting:  100  The index is out of bounds  Commands:  0. Exit  1. Add figure  2. Function call for all elements  3. Delete figure by index  Input number of chosen function: 2  1. Count the area of figures  2. Count the center of figures  3. Print coordinates of figures  Input the number of chosen function: 2  The center of figure:  The center of 1 figure.  Coordinate 'x': 2492; Coordinate 'y': 157 |

1. **Адрес репозитория на GitHub**

<https://github.com/---/oop_excercise_3>

1. **Код программы на С++**

Vertex.h

#pragma once

#include<iostream>

//Класс "Вершина"

struct **Vertex**

{

using m\_vertex = std::pair<int,int>;

m\_vertex coordinates;

};

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

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

Vertex.cpp

#include"Vertex.h"

std::istream& operator>> (std::istream& is, Vertex& vertex) //ввод координат из потока

{

std::cout << "Input the coordinate 'x': ";

is >> vertex.coordinates.first;

std::cout << "Input the coordinate 'y': ";

is >> vertex.coordinates.second;

std::cout << "\n";

return is;

};

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

{

return os << "Coordinate 'x': " <<vertex.coordinates.first << "; Coordinate 'y': " << vertex.coordinates.second;

};

figure.h

#pragma once

#include<iostream>

#include<cmath>

#include"Vertex.h"

//Родительский класс "Фигура", от которого наследуются другие классы

class **Figure** {

public:

virtual Vertex ***calculateCenter***() const = 0;

virtual ~***Figure***(){};

virtual double ***calculateArea***() const = 0;

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

};

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

figures.h

#pragma once

#include<stdio.h>

#include"figure.h"

//Класс пятиугольника

class **Pentagon** : public Figure {

private:

Vertex v[5];

public:

**Pentagon**();

virtual ~***Pentagon***() override {}

**Pentagon**(std::istream& is);

double ***calculateArea***() const override;

Vertex ***calculateCenter***() const override;

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

};

//Класс шестиугольника

class **Hexagon** : public Figure {

private:

Vertex v[6];

public:

**Hexagon**();

virtual ~***Hexagon***() override {}

**Hexagon**(std::istream& is);

double ***calculateArea***() const override;

Vertex ***calculateCenter***() const override;

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

};

//Класс восьмиугольника

class **Octagon** : public Figure {

private:

Vertex v[8];

public:

**Octagon**();

virtual ~***Octagon***() override {}

**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>

//Методы классов

Pentagon::**Pentagon**() {};//Конструктор класса

Pentagon::**Pentagon**(std::istream& is) //Ввод вершин из потока

{

Vertex l;

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

{

is >> l;

v[i] = l;

}

};

double Pentagon::***calculateArea***() const//Метод вычисления площади фигуры

{

double Area = 0;

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

Area += (v[i].coordinates.first) \* (v[(i + 1)%5].coordinates.second)

- (v[(i + 1)%5].coordinates.first)\*(v[i].coordinates.second);

Area \*= 0.5;

return abs(Area);

};

Vertex Pentagon::***calculateCenter***() const//Метод вычисления центра фигуры

{

Vertex center;

double xCenter = 0;

double yCenter = 0;

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

{

xCenter += v[i].coordinates.first;

yCenter += v[i].coordinates.second;

}

xCenter = xCenter / 5;

yCenter = yCenter / 5;

center.coordinates.first = xCenter;

center.coordinates.second = yCenter;

return center;

};

void Pentagon::***printVertex***(std::ostream& os) const//Вывод вершин пятиугольника

{

os << "Pentagon:\n";

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

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

os << '\b';

};

Hexagon::**Hexagon**() {};//Конструктор класса

Hexagon::**Hexagon**(std::istream& is) //Конструктор класса

{

Vertex l;

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

is >> l;

v[i] = l;

}

};

double Hexagon::***calculateArea***() const//Метод вычисления площади шестиугольника

{

double Area = 0;

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

Area += (v[i].coordinates.first) \* (v[(i + 1)%6].coordinates.second)

- (v[(i + 1)%6].coordinates.first)\*(v[i].coordinates.second);

}

Area \*= 0.5;

return abs(Area);

};

Vertex Hexagon::***calculateCenter***() const//Метод вычисления центра фигуры шестиугольника

{

Vertex center;

double xCenter = 0;

double yCenter = 0;

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

{

xCenter += v[i].coordinates.first;

yCenter += v[i].coordinates.second;

}

xCenter = xCenter / 6;

yCenter = yCenter / 6;

center.coordinates.first = xCenter;

center.coordinates.second = yCenter;

return center;

};

void Hexagon::***printVertex***(std::ostream& os) const//Вывод вершин шестиугольника

{

os << "Hexagon:\n";

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

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

os << '\b';

};

Octagon::**Octagon**() {};

Octagon::**Octagon**(std::istream& is)//Ввод вершин восьмиугольника

{

Vertex l;

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

is >> l;

v[i] = l;

}

};

double Octagon::***calculateArea***() const//Вычисление площади восьмиугольника

{

double Area = 0;

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

Area += (v[i].coordinates.first) \* (v[(i + 1)%8].coordinates.second)

- (v[(i + 1)%8].coordinates.first)\*(v[i].coordinates.second);

Area \*= 0.5;

return abs(Area);

};

Vertex Octagon::***calculateCenter***() const//Вычисления центра восьмиугольника

{

Vertex center;

double xCenter = 0;

double yCenter = 0;

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

{

xCenter += v[i].coordinates.first;

yCenter += v[i].coordinates.second;

}

xCenter = xCenter / 8;

yCenter = yCenter / 8;

center.coordinates.first = xCenter;

center.coordinates.second = 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>

//Функция вывода главного меню

void **printMenu**()

{

std::cout << "\nCommands:" << std::endl;

std::cout << "0. Exit" << std::endl;

std::cout << "1. Add figure" << std::endl;

std::cout << "2. Function call for all elements" << std::endl;

std::cout << "3. Delete figure by index" << std::endl;

}

int **main**()

{

Figure\* s;

std::vector<Figure\*> v1;

while (true)

{

printMenu();

std::cout << "Input number of chosen function: ";

int k;

std::cin >> k;

std::vector<Figure\*> next;//создание динамического массива фигур с помощью std::vector

switch (k)

{

case 0:

//Удаление всех фигур из массива

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

delete v1[i];

return 0;

case 1:

std::cout << "1. Input the pentagon" << std::endl;

std::cout << "2. Input the hexagon" << std::endl;

std::cout << "3. Input the octagon" << std::endl;

std::cout << "Input the number of chosen figure: ";

int a;

bool flag;

flag = true;

std::cin >> a;

switch (a)

{

case 1:

std::cout << "Input coordinates of vertexes of pentagon: \n";

s = new Pentagon(std*::cin*);

break;

case 2:

std::cout << "Input coordinates of vertexes of hexagon: \n";

s = new Hexagon(std*::cin*);

break;

case 3:

std::cout << "Input coordinates of vertexes of octagon: \n";

s = new Octagon(std*::cin*);

break;

default:

std::cout << "No figure with such number" << std::endl;

flag = false;

break;

}

if (flag)

v1.push\_back(s);//добавление элемента в массив

break;

case 2:

if (v1.size() == 0)

{

std::cout << "There are no elements in array\n";

break;

}

std::cout << "1. Count the area of figures" << std::endl;

std::cout << "2. Count the center of figures" << std::endl;

std::cout << "3. Print coordinates of figures" << std::endl;

std::cout << "Input the number of chosen function: ";

int b;

std::cin >> b;

switch (b) {

case 1:

//Вывод площади всех фигур

std::cout << "Area of figure:" << std::endl;

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

{

std::cout << "The area of " << i+1 << " figure.\n";

std::cout << v1[i]->*calculateArea*() << std::endl;

}

break;

case 2:

//Вывод координат центров всех фигур

std::cout << "The center of figure:" << std::endl;

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

{

std::cout << "The center of " << i+1 << " figure.\n";

std::cout << v1[i]->*calculateCenter*() << std::endl;

}

break;

case 3:

//Вывод координат x,y всех фигур

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

{

std::cout << "The coordinates of " << i+1 << " figure.\n";

v1[i]->*printVertex*(std*::cout*);

std::cout << std::endl;

}

break;

default:

std::cout << "No function with such number" << std::endl;

break;

}

break;

case 3:

std::cout << "Input the index of figure for deleting: \n";

size\_t id;

std::cin >> id;

if (id < 0 || id >= v1.size())

{

std::cout << "The index is out of bounds" << std::endl;

break;

}

else

{

delete v1[id];

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

break;

}

default:

std::cout << "No function with such number.\n";

break;

}

}

}

1. **Объяснение результатов работы программы**

Программа начинается с главного меню, в котором пользователь может выбрать добавить элемент в вектор (Add element), удалить элемент из вектора по индексу (Delete element by index), а также вывести параметры фигур, содержащиеся в динамическом массиве с помощью “Function call for all elements”. При добавлении элемента программа запросит координаты вершин для выбранной фигуры, при удалении – индекс удаляемого элемента в динамическом массиве. “Function call for all elements” выводит координаты центра фигуры, её площадь, а также координаты всех вершин.

1. **Вывод**

В данной работе было осуществлено наследование классов. Виртуальные методы в родительском классе могут быть переопределены в классах наследниках, если это требуется.