**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: Ильиных Вадим Максимович

Группа: 80-201

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2020

1. **Постановка задачи**

**Вариант №23**

|  |  |  |
| --- | --- | --- |
| **Фигура №1** | **Фигура №2** | **Фигура №3** |
| 6-угольник | 8-угольник | треугольник |

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.

1. **Описание программы**

В родительском классе некоторые функции объявлены чисто виртуальными, которые переопределяются в классах-наследниках. К фигуре могут быть применены методы: вычислить квадрат стороны, вычислить центр фигуры, вычислить площадь, распечатать координаты вершин, координаты центра, площадь.

Все объекты добавляются в std::vector<Figure\*>. Далее в зависимости от действия над ними всеми выполняется необходимое действие. Также реализована возможность удаления элемента по заданному индексу.

1. **Ссылка на github**
2. **Набор тестов**

**test\_01.txt:**

1 (выбор фигуры)

3 (выбор треугольника)

0 0 1 0 0 1 (координаты треугольника)

2 (показ функций с фигурами)

2 (рассчитать центр)

2 (показ функций с фигурами)

3 (распечатать центр)

2 (показ функций с фигурами)

4 (рассчитать площадь каждой фигуры)

1 (выбор фигуры)

1 (выбор шестиугольника)

0 0 4 0 0 -4 5 9 10 -8 8 8 (координаты)

2 (показ функций с фигурами)

4 (рассчитать площадь каждой фигуры)

2 (показ функций с фигурами)

5 (рассчитать общую площадь)

3 (удалить фигуру по индексу)

1 (индекс)

2 (показ функций с фигурами)

1 (распечатать координаты фигур)

0 (выход)

**test\_02.txt:**

1 (выбор фигуры)

2 (восьмиугольник)

0 0 1 3 5 9 0 -5 8 0 14 10 9 9 -5 -5 (координаты)

3 (удаление фигуры по индексу)

4 (индекс)

1 (выбор фигуры)

3 (треугольник)

5 0 -1 0 0 6 (координаты)

2 (показ функций с фигурами)

1 (распечатать координаты)

2 (показ функций с фигурами)

2 (рассчитать центр)

2 (показ функций с фигурами)

3 (распечатать центр каждой фигуры)

2 (показ функций с фигурами)

5 (рассчитать общую площадь)

2 (показ функций с фигурами)

4 (рассчитать площадь каждой фигуры)

3 (удаление фигуры по индексу)

1 (индекс)

2 (показ функций с фигурами)

1 (распечатать координаты)

2 (показ функций с фигурами)

4 (рассчитать площадь каждой фигуры)

2 (показ функций с фигурами)

5 (рассчитать общую площадь)

0 (выход)

1. **Результаты выполнения тестов**

Все тесты успешно пройдены, программа выдает правильные ответы,

1. **Листинг программы**

#include <iostream>

#include <vector>

class Figure{

protected:

double area;

double x\_centre, y\_centre;

std::vector <double> x;

std::vector <double> y;

public:

virtual void calcCenter() = 0;

virtual double calcArea() = 0;

virtual void printCord() = 0;

bool is\_correct\_coordinates(){

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

for(int j = i+1; j < x.size(); ++j){

if(x[i] == x[j] && y[i] ==y[j]){

std::cout << "Wrong coordinates!" << std::endl;

return false;

}

}

}

return true;

}

void printCenter(){

std::cout << "(" << x\_centre << "," << y\_centre << ")"<< std::endl;

}

Figure(){}

virtual ~Figure(){}

};

class Hexagon: public Figure{

public:

Hexagon(){}

Hexagon(double x1, double y1, double x2, double y2, double x3, double y3,

double x4, double y4, double x5, double y5, double x6, double y6){

x.push\_back(x1);

x.push\_back(x2);

x.push\_back(x3);

x.push\_back(x4);

x.push\_back(x5);

x.push\_back(x6);

y.push\_back(y1);

y.push\_back(y2);

y.push\_back(y3);

y.push\_back(y4);

y.push\_back(y5);

y.push\_back(y6);

}

~Hexagon(){}

double calcArea()override{

area = 0;

for(int i = 0; i < x.size()-1; ++i){

area += x[i]\*y[i+1] - x[i+1]\*y[i];

}

area += x[x.size()-1]\*y[0] - y[x.size()-1]\*x[0];

area \*= 0.5;

if(area < 0)

area \*= -1;

return area;

}

void calcCenter()override{

x\_centre = 0;

y\_centre = 0;

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

x\_centre += x[i];

y\_centre += y[i];

}

x\_centre /= x.size();

y\_centre /= y.size();

}

void printCord(){

std::cout << "Hexagon coordinates:" << std::endl;

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

std::cout << "(" << x[i] << "," << y[i] << ")" << std::endl;

}

}

};

class Octagon: public Figure{

public:

Octagon(){}

Octagon(double x1, double y1, double x2, double y2, double x3, double y3, double x4, double y4,

double x5, double y5, double x6, double y6, double x7, double y7, double x8, double y8){

x.push\_back(x1);

x.push\_back(x2);

x.push\_back(x3);

x.push\_back(x4);

x.push\_back(x5);

x.push\_back(x6);

x.push\_back(x7);

x.push\_back(x8);

y.push\_back(y1);

y.push\_back(y2);

y.push\_back(y3);

y.push\_back(y4);

y.push\_back(y5);

y.push\_back(y6);

y.push\_back(y7);

y.push\_back(y8);

}

~Octagon(){}

double calcArea()override{

area = 0;

for(int i = 0; i < x.size()-1; ++i){

area += x[i]\*y[i+1] - x[i+1]\*y[i];

}

area += x[x.size()-1]\*y[0] - y[x.size()-1]\*x[0];

area \*= 0.5;

if(area < 0)

area \*= -1;

return area;

}

void calcCenter()override{

x\_centre = 0;

y\_centre = 0;

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

x\_centre += x[i];

y\_centre += y[i];

}

x\_centre /= x.size();

y\_centre /= y.size();

}

void printCord(){

std::cout << "Octagon coordinates:" << std::endl;

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

std::cout << "(" << x[i] << "," << y[i] << ")" << std::endl;

}

}

};

class Triangle: public Figure{

public:

Triangle(){}

Triangle(double x1, double y1, double x2, double y2, double x3, double y3){

x.push\_back(x1);

x.push\_back(x2);

x.push\_back(x3);

y.push\_back(y1);

y.push\_back(y2);

y.push\_back(y3);

}

~Triangle(){}

double calcArea()override{

area = 0;

for(int i = 0; i < x.size()-1; ++i){

area += x[i]\*y[i+1] - x[i+1]\*y[i];

}

area += x[x.size()-1]\*y[0] - y[x.size()-1]\*x[0];

area \*= 0.5;

if(area < 0)

area \*= -1;

return area;

}

void calcCenter()override{

x\_centre = 0;

y\_centre = 0;

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

x\_centre += x[i];

y\_centre += y[i];

}

x\_centre /= x.size();

y\_centre /= y.size();

}

void printCord(){

std::cout << "Triangle coordinates:" << std::endl;

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

std::cout << "(" << x[i] << "," << y[i] << ")" << std::endl;

}

}

};

void printMenu(){

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

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

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

std::cout << "2. Call the function for all figures" << std::endl;

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

std::cout << "4. Print Menu" << std::endl;

}

int main(){

int action, choice;

std::vector<Figure\*> v;

printMenu();

while(1){

std::cout << "Choose number of menu: ";

std::cin >> action;

switch(action){

case 0:

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

delete v[i];

}

return 0;

case 1:

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

std::cout << "\n1. Hexagon" << std::endl;

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

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

std::cout << "Choose number:" << std::endl;

std::cin >> choice;

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

std::cout << "Wrong number!" << std::endl;

break;

}

Figure \*figure;

switch(choice){

case 1:

std::cout << "Write coordinates:" << std::endl;

std::cin >> x1 >> y1 >> x2 >> y2 >> x3 >> y3 >> x4 >> y4 >> x5 >> y5 >> x6 >> y6;

figure = new Hexagon(x1, y1, x2, y2, x3, y3, x4, y4, x5, y5, x6, y6);

break;

case 2:

std::cout << "Write coordinates:" << std::endl;

std::cin >> x1 >> y1 >> x2 >> y2 >> x3 >> y3 >> x4 >> y4 >> x5 >> y5 >> x6 >> y6 >> x7 >> y7 >> x8 >> y8;

figure = new Octagon(x1, y1, x2, y2, x3, y3, x4, y4, x5, y5, x6, y6, x7, y7, x8, y8);

break;

case 3:

std::cout << "Write coordinates:" << std::endl;

std::cin >> x1 >> y1 >> x2 >> y2 >> x3 >> y3;

figure = new Triangle(x1, y1, x2, y2, x3, y3);

break;

}

if(!figure->is\_correct\_coordinates()){

delete figure;

}

else{

v.push\_back(figure);

std::cout << "Complete!" << std::endl;

}

break;

case 2:

std::cout << "\n1. Print coordinates" << std::endl;

std::cout << "2. Calculate centre" << std::endl;

std::cout << "3. Print centre" << std::endl;

std::cout << "4. Calculate area" << std::endl;

std::cout << "5. Calculate total area" << std::endl;

std::cout << "Choose number:" << std::endl;

std::cin >> choice;

if(choice < 1 || choice > 6){

std::cout << "Wrong number!" << std::endl;

break;

}

switch(choice){

case 1:

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

v[i]->printCord();

}

break;

case 2:

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

v[i]->calcCenter();

}

break;

case 3:

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

v[i]->printCenter();

}

break;

case 4:

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

std::cout << "Area is " << v[i]->calcArea() << std::endl;

}

break;

case 5:{

double res = 0;

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

res += v[i]->calcArea();

}

std::cout << "Total area is " << res << std::endl;

break;

}

}

break;

case 3:{

int idx;

std::cout << "Write index: ";

std::cin >> idx;

if(idx < 0 || idx > v.size()-1){

std::cout << "Wrong index!" << std::endl;

break;

}

v.erase(v.begin()+idx);

std::vector<Figure\*>(v).swap(v);

std::cout << "Complete!" << std::endl;

break;

}

case 4:

printMenu();

break;

default:

std::cout << "Wrong number! Try again" << std::endl;

}

}

v.clear();

return 0;

}

1. **Вывод**

Реализовал программу с абстрактным классом-родителем и классами-потомками для создания шестиугольника, восьмиугольника и треугольника. Благодаря динамическому полиморфизму на этапе работы программы определяется, какая функция должна быть вызвана для каждого объекта, на который указывает указатель типа класса-родителя.

**Список литературы**

1. Справочник по языку С++ [Электронный ресурс]. URL: [http://www.cplusplus.com/reference](http://www.cplusplus.com/reference/deque/) (дата обращения: 18.10.2020).