**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: Тимофеев Алексей Владимирович

Группа: 80-207

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2020

1. Постановка задачи

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 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;

|  |  |  |  |
| --- | --- | --- | --- |
|  | 5-угольник | 6-угольник | 8-угольник |

1. Описание программы

Создаем родительский класс - Figure, далее с помощью наследования создаем создаем дочерние классы - Pentagon Octagon и Hexagon. Создаем для хранения координат точки структуру, она имеет два поля: x и y. В родительском классе храним массив этих точек в пространстве protected. Название фигуры получаем, вычисляя длину массива точек. Функция S() служит для получения площади фигуры. Функция Center() служит для получения координат центра фигуры. В конструкторе инициализируем фигуру по установленным формулам с помощью данной нам точки и длины стороны. Это сделано чтобы не было возможности некорректно задать точки. Также я перегрузил оператор << для выдачи структуры координат и для выдачи вектора структур координат. Программа проверяет входные значения, если длина стороны фигуры отрицательная, выводится соответствующее сообщение. Фигуры задаются с помощью верхней точки и длины стороны.

1. Набор тестов

Тест 1: 8 углов у фигуры

0 0 4 первые две цифры это координаты точки, последняя это длина стороны

5 углов у фигуры

1 1 3 первые две цифры это координаты точки, последняя это длина стороны

Тест 2: 5 углов у фигуры

-1 0 3 первые две цифры это координаты точки, последняя это длина стороны

6 углов у фигуры

1 2 1 первые две цифры это координаты точки, последняя это длина стороны

8 углов у фигуры

2 -3 2 первые две цифры это координаты точки, последняя это длина стороны

1. Результаты выполнения тестов

PS D:\Ycheba2kurs\OOp\OOPlab\laba3> .\a.exe

Enter:

Тест 1

1 - add shape

2 - remove element by index

3 - print a sequence of shapes

4 - display the total area of the figures

5 - end the program

Enter the command:1

Enter the number of corners in the shape:8

Enter point coordinates and side length: 0 0 4

Enter the command:3

8-angle

Coordinates: (0,0) (3.69552,-1.53073) (-3.69552,-1.53073) (5.22625,-5.22625) (-5.22625,-5.22625) (3.69552,-7.39104) (-3.69552,-7.39104) (0,-8)

Center of the figure: (0,-4)

Square: 77.2548

Enter the command:1

Enter the number of corners in the shape:5

Enter the coordinates of the top point of the point and the length of the side: 1 1 3

Enter the command:3

8-angle

Coordinates: (0,0) (3.69552,-1.53073) (-3.69552,-1.53073) (5.22625,-5.22625) (-5.22625,-5.22625) (3.69552,-7.39104) (-3.69552,-7.39104) (0,-8)

Center of the figure: (0,-4)

Square: 77.2548

5-angle

Coordinates: (1,1) (-1.21006,-2.82793) (3.21006,-2.82793) (-0.275976,-6.31396) (2.27598,-6.31396)

Center of the figure: (1,-3.45676)

Square: 61.9372

Enter the command:4

total area: 139.192

Enter the command:2

Enter index:1

Enter the command:3

5-angle

Coordinates: (1,1) (-1.21006,-2.82793) (3.21006,-2.82793) (-0.275976,-6.31396) (2.27598,-6.31396)

Center of the figure: (1,-3.45676)

Square: 61.9372

Enter the command:5

Тест 2

PS D:\Ycheba2kurs\OOp\OOPlab\laba3> .\a.exe

Enter:

1 - add shape

2 - remove element by index

3 - print a sequence of shapes

4 - display the total area of the figures

5 - end the program

Enter the command:1

Enter the number of corners in the shape:5

Enter the coordinates of the top point of the point and the length of the side: -1 0 3

Enter the command:1

Enter the number of corners in the shape:6

Enter point coordinates and side length: 1 2 1

Enter the command:3

5-angle

Coordinates: (-1,0) (-3.21006,-3.82793) (1.21006,-3.82793) (-2.27598,-7.31396) (0.275976,-7.31396)

Center of the figure: (-1,-4.45676)

Square: 61.9372

6-angle

Coordinates: (1,2) (0.133975,1.5) (1.86603,1.5) (0.133975,0.5) (1.86603,0.5) (1,0)

Center of the figure: (1,1)

Square: 2.59808

Enter the command:1

Enter the number of corners in the shape:8

Enter point coordinates and side length: 2 -3 2

Enter the command:3

5-angle

Coordinates: (-1,0) (-3.21006,-3.82793) (1.21006,-3.82793) (-2.27598,-7.31396) (0.275976,-7.31396)

Center of the figure: (-1,-4.45676)

Square: 61.9372

6-angle

Coordinates: (1,2) (0.133975,1.5) (1.86603,1.5) (0.133975,0.5) (1.86603,0.5) (1,0)

Center of the figure: (1,1)

Square: 2.59808

8-angle

Coordinates: (2,-3) (3.84776,-3.76537) (0.152241,-3.76537) (4.61313,-5.61313) (-0.613126,-5.61313) (3.84776,-6.69552) (0.152241,-6.69552) (2,-7)

Center of the figure: (2,-5)

Square: 19.3137

Enter the command:4

total area: 83.849

Enter the command:2

Enter index:3

Enter the command:4

total area: 64.5353

Enter the command:3

5-angle

Coordinates: (-1,0) (-3.21006,-3.82793) (1.21006,-3.82793) (-2.27598,-7.31396) (0.275976,-7.31396)

Center of the figure: (-1,-4.45676)

Square: 61.9372

6-angle

Coordinates: (1,2) (0.133975,1.5) (1.86603,1.5) (0.133975,0.5) (1.86603,0.5) (1,0)

Center of the figure: (1,1)

Square: 2.59808

Enter the command:5

1. Листинг программы

**figure.hpp**

#pragma once

#include <iostream>

#include <vector>

#include <string>

struct Pair {

double x;

double y;

Pair() {

x = 0;

y = 0;

}

Pair(double a, double b) {

x = a;

y = b;

}

};

class Figure {

protected:

std::vector<Pair> points;

public:

Figure();

Figure(double x1, double y1, int c);

virtual double S();

virtual Pair Center();

virtual std::vector<Pair> Coord();

int GetSize(){

return points.size();

}

};

Figure::Figure() {

}

Figure::Figure(double x1, double y1, int c) {

points.emplace\_back(Pair(x1,y1));

}

double Figure::S() {

return 0;

}

Pair Figure::Center() {

}

std::vector<Pair> Figure::Coord() {

return points;

}

**hexagon.cpp**

#pragma once

#include "figure.hpp"

#include <cmath>

class Hexagon: public Figure {

private:

int length;

public:

Hexagon() : Figure() { length = 0; }

Hexagon(double x1, double y1, int length);

double S() override;

Pair Center() override;

std::vector<Pair> Coord() override;

};

Hexagon::Hexagon(double x1, double y1, int c) {

points.push\_back(Pair(x1, y1));

points.push\_back(Pair(x1 - (double)c \* sqrt(3) / 2, y1 - (double) c / 2));

points.push\_back(Pair(x1 + (double)c \* sqrt(3) / 2, y1 - (double) c / 2));

points.push\_back(Pair(x1 - (double)c \* sqrt(3) / 2, y1 - (double) c / 2 - c));

points.push\_back(Pair(x1 + (double)c \* sqrt(3) / 2, y1 - (double) c / 2 - c));

points.push\_back(Pair(x1, y1 - 2 \* c));

length = c;

}

double Hexagon::S() {

double s = 3 \* sqrt(3) \* length \* length / 2;

return s;

}

Pair Hexagon::Center() {

Pair out(0,0);

out.x = points[0].x;

out.y = points[0].y - (double)length;

return out;

}

std::vector<Pair> Hexagon::Coord() {

return points;

}

**octagon.hpp**

#pragma once

#include "figure.hpp"

#include <cmath>

class Octagon: public Figure {

private:

int length;

public:

Octagon() : Figure() { length = 0; }

Octagon(double x1, double y1, int c);

double S() override;

Pair Center() override;

std::vector<Pair> Coord() override;

};

Octagon::Octagon(double x1, double y1, int c) {

points.push\_back(Pair(x1, y1));

points.push\_back(Pair(x1 + (double)c \* cos(M\_PI / 8), y1 - (double)c \* sin(M\_PI / 8)));

points.push\_back(Pair(x1 - (double)c \* cos(M\_PI / 8), y1 - (double)c \* sin(M\_PI / 8)));

points.push\_back(Pair(x1 + (double)c \* sqrt(1 / sqrt(2) + 1), y1 - (double)c \* sqrt(1 / sqrt(2) + 1)));

points.push\_back(Pair(x1 - (double)c \* sqrt(1 / sqrt(2) + 1), y1 - (double)c \* sqrt(1 / sqrt(2) + 1)));

points.push\_back(Pair(x1 + (double)c \* cos(M\_PI / 8), y1 - 2 \* (double)c \* cos(M\_PI / 8)));

points.push\_back(Pair(x1 - (double)c \* cos(M\_PI / 8), y1 - 2 \* (double)c \* cos(M\_PI / 8)));

points.push\_back(Pair(x1, y1 - 2 \* (double)c ));

length = c;

}

std::vector<Pair> Octagon::Coord() {

return points;

}

Pair Octagon::Center() {

Pair out(0,0);

out.x = points[0].x;

out.y = points[0].y - (double)length;

return out;

}

double Octagon::S() {

double s = 2 \* length \* length \* (1 + sqrt(2));

return s;

}

**pentagon.hpp**

#pragma once

#include "figure.hpp"

#include <cmath>

class Octagon: public Figure {

private:

int length;

public:

Octagon() : Figure() { length = 0; }

Octagon(double x1, double y1, int c);

double S() override;

Pair Center() override;

std::vector<Pair> Coord() override;

};

Octagon::Octagon(double x1, double y1, int c) {

points.push\_back(Pair(x1, y1));

points.push\_back(Pair(x1 + (double)c \* cos(M\_PI / 8), y1 - (double)c \* sin(M\_PI / 8)));

points.push\_back(Pair(x1 - (double)c \* cos(M\_PI / 8), y1 - (double)c \* sin(M\_PI / 8)));

points.push\_back(Pair(x1 + (double)c \* sqrt(1 / sqrt(2) + 1), y1 - (double)c \* sqrt(1 / sqrt(2) + 1)));

points.push\_back(Pair(x1 - (double)c \* sqrt(1 / sqrt(2) + 1), y1 - (double)c \* sqrt(1 / sqrt(2) + 1)));

points.push\_back(Pair(x1 + (double)c \* cos(M\_PI / 8), y1 - 2 \* (double)c \* cos(M\_PI / 8)));

points.push\_back(Pair(x1 - (double)c \* cos(M\_PI / 8), y1 - 2 \* (double)c \* cos(M\_PI / 8)));

points.push\_back(Pair(x1, y1 - 2 \* (double)c ));

length = c;

}

std::vector<Pair> Octagon::Coord() {

return points;

}

Pair Octagon::Center() {

Pair out(0,0);

out.x = points[0].x;

out.y = points[0].y - (double)length;

return out;

}

double Octagon::S() {

double s = 2 \* length \* length \* (1 + sqrt(2));

return s;

}

**main.cpp**

#include <iostream> // Тимофеев Алексей М8О-207Б-19

#include "figure.hpp"

#include "pentagon.hpp"

#include "hexagon.hpp"

#include "octagon.hpp"

std::ostream& operator<<(std::ostream &os, Pair p) {

os << '(' << p.x << ',' << p.y << ')';

return os;

}

std::ostream& operator<<(std::ostream &os, std::vector<Pair> p) {

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

os << " " << p[i];

}

return os;

}

int main() {

std::string name;

double x1, y1;

int length, m;

std::vector<Figure \*> figure;

std::cout << "Enter:\n 1 - add shape\n 2 - remove element by index\n 3 - print a sequence of shapes\n 4 - display the total area of the figures\n 5 - end the program\n";

std::cout << "Enter the command:";

while (std::cin >> m && m < 5) {

switch (m) {

case 1: {

std::cout << "Enter the number of corners in the shape:";

std::cin >> m;

switch (m) {

case 5: {

std::cout << "Enter the coordinates of the top point of the point and the length of the side: ";

std::cin >> x1 >> y1 >> length ;

if (length <= 0 ) {

std::cout << "The entered values are not correct" << std::endl;

break;

}

Figure \* pentagon = new Pentagon(x1, y1, length);

figure.push\_back(pentagon);

break;

}

case 6: {

std::cout << "Enter point coordinates and side length: ";

std::cin >> x1 >> y1 >> length;

if (length <= 0 ) {

std::cout << "The entered values are not correct" << std::endl;

break;

}

Figure \* hexagon = new Hexagon(x1, y1, length);

figure.push\_back(hexagon);

break;

}

case 8: {

std::cout << "Enter point coordinates and side length: ";

std::cin >> x1 >> y1 >> length;

if (length <= 0 ) {

std::cout << "The entered values are not correct" << std::endl;

break;

}

Figure \* octagon = new Octagon(x1, y1, length);

figure.push\_back(octagon);

break;

}

default: {

std::cout << "Incorrect number of corners.\n Shapes available: pentagon, hexagon, octagon.\n Try again. \n";

break;

}

}

break;

}

case 2: {

int ind;

std::cout << "Enter index:";

std::cin >> ind;

if (ind <= 0 || ind >= figure.size() + 1) {

std::cout << "The entered values are not correct" << std::endl;

break;

}

ind--;

figure.erase(figure.begin() + ind);

break;

}

case 3: {

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

std::cout << std::endl;

std::cout << figure[i]->GetSize() << "-angle" << std::endl;

std::cout << "Coordinates:" << figure[i]->Coord() << std::endl;

std::cout << "Center of the figure: " << figure[i]->Center() << std::endl;

std::cout << "Square: " << figure[i]->S() << std::endl;

}

break;

}

case 4: {

double s = 0;

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

s += figure[i]->S();

}

std::cout << std::endl << "total area: " << s << std::endl;

break;

}

default:

break;

}

std::cout << "\n Enter the command:"<< std::endl;

}

}

1. **Выводы**

Я научился создавать родительский класс и дочерни классы, перегружать операторы вывода, и использовать наследование и виртуальные функции, так же познакомился с библиотекой #include <vector>.

1. Списоклитературы

ООП вкартинках/ Хабр —Habr

URL: https://habr.com/ru/post/463125/ (дата обращения: 29.10.2020)

Классы в C++ — урок 10/ code-live.ru

URL: https://code-live.ru/post/cpp-classes/#\_2 (дата обращения: 29.10.2020)