**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: Федоров Антон Сергеевич

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 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 30:** 5-угольник, 6-угольник, 8-угольник

1. **Описание программы**

В программе реализован абстрактный класс Figure, который имеет 2 protected поля: Center (Cord) и Radius (double), конструктор Figure(Cord center, double radius), метод GetCenter() и два виртуальных метода: double Area() (найти объем), print() (печать вершин).

От Figure наследуются три класса: Pentagon, Hexagon и Octagon. Виртуальные методы реализованы для них аналогично, так как отличается только количество вершин. Печать вершин реализована в цикле, где к координатам центра поочередно прибавляются радиус-векторы. Площадь считается по треугольникам с общей вершиной в центре. Суммарная площадь фигур не равна площади объединения, поэтому для пересекающегося и непересекающегося набора фигур суммарная площадь одинакова.

Реализован также класс Cord, для удобной работы с координатами. Для данного класса перегружены операторы умножения, сложения, вычитания и оператора побитового сдвига (<<) для вывода в std::cout.

Ввод данных осуществляется в цикле while, в котором, исходя от типа выбранной операции, печатаются данные вектора std::vector<Figure\*>, добавляются новые или удаляются элементы.

Ввод фигуры осуществляется вводом центра по двум координатам и радиуса описанной окружности.

Операции:

1. -1 - удаление по индексу
2. 0 - печать данных фигур, находящихся в векторе std::vector<Figure\*>
3. 1 - добавить в конец пятиугольник
4. 2 - добавить в конец шестиугольник
5. 3 - добавить в конец восьмиугольник
6. **Набор тестов**

**Тест №1 Тестирование проверки корректности ввода.**

svfd

**Тест №2 Попытка удалить элемент по неправильному индексу**

-1

-1

**Тест №3 Печать пустого вектора**

0

**Тест №4 Тестирование работы вектора со всеми видами фигур**

1

0 0 1

2

1.5 0.5 2.25

3

0 -1.5 3

1

1 1 1

0

-1

1

-1

0

0

1. **Результаты выполнения тестов**

**Тест №1 Тестирование проверки корректности ввода.**

insert operation type:svfd

Process finished with exit code 0

**Тест №2 Попытка удалить элемент по неправильному индексу**

insert operation type:-1

insert index to delete:-1

invalid input

Process finished with exit code -1

**Тест №3 Печать пустого вектора**

insert operation type:0

summary area: 0

insert operation type:^D

Process finished with exit code 0

**Тест №4 Тестирование работы вектора со всеми видами фигур**

insert operation type:1

insert pentagon center and radius:0 0 1

insert operation type:2

insert hexagon center and radius:1.5 0.5 2.25

insert operation type:3

insert octagon center and radius:0 -1.5 3

insert operation type:1

insert pentagon center and radius:1 1 1

insert operation type:0

figure number 0

center:(1,1)

cords: (1,2), (1.95106,1.30902), (1.58779,0.190983), (0.412215,0.190983), (0.0489435,1.30902),

area: 2.37764

figure number 1

center:(1,1)

cords: (1,2), (1.95106,1.30902), (1.58779,0.190983), (0.412215,0.190983), (0.0489435,1.30902),

area: 2.37764

figure number 2

center:(1,1)

cords: (1,2), (1.95106,1.30902), (1.58779,0.190983), (0.412215,0.190983), (0.0489435,1.30902),

area: 2.37764

figure number 3

center:(1,1)

cords: (1,2), (1.95106,1.30902), (1.58779,0.190983), (0.412215,0.190983), (0.0489435,1.30902),

area: 2.37764

summary area: 9.51057

insert operation type:-1

insert index to delete:1

insert operation type:-1

insert index to delete:0

insert operation type:0

figure number 0

center:(1,1)

cords: (1,2), (1.95106,1.30902), (1.58779,0.190983), (0.412215,0.190983), (0.0489435,1.30902),

area: 2.37764

figure number 1

center:(1,1)

cords: (1,2), (1.95106,1.30902), (1.58779,0.190983), (0.412215,0.190983), (0.0489435,1.30902),

area: 2.37764

summary area: 4.75528

insert operation type:^D

Process finished with exit code 0

1. **Листинг программы**

**CMakeLists.txt**

cmake\_minimum\_required(VERSION 3.17)

project(oop\_exercise\_03)

set(CMAKE\_CXX\_STANDARD 14)

add\_executable(oop\_exercise\_03 main.cpp Figure.cpp Figure.h Cord.cpp Cord.h Pentagon.cpp Pentagon.h Hexagon.cpp Hexagon.h

Octagon.cpp Octagon.h)

**Figure.h**

//Федоров Антон Сергеевич М8О-207Б-19

//Вариант 30 5-угольник, 6-угольник, 8-угольник

#ifndef **OOP\_EXERCISE\_03\_FIGURE\_H**

#define **OOP\_EXERCISE\_03\_FIGURE\_H**

#include "Cord.h"

#include <iostream>

#include <math.h>

const double PI = 3.14159265358979323846264338327950288419716939937510;

class Figure{

protected:

Cord Center;

double Radius = 0;

public:

Figure(Cord center, double radius);

Cord GetCenter();

virtual double Area() = 0;

virtual void print() = 0;

};

#endif

**Figure.cpp**

//Федоров Антон Сергеевич М8О-207Б-19

//Вариант 30 5-угольник, 6-угольник, 8-угольник

#include "Figure.h"

#include "math.h"

Figure::Figure(Cord center, double radius){

Center = center;

Radius = radius;

}

Cord Figure::GetCenter(){

return Center;

}

**Pentagon.h**

//Федоров Антон Сергеевич М8О-207Б-19

//Вариант 30 5-угольник, 6-угольник, 8-угольник

#ifndef OOP\_EXERCISE\_03\_PENTAGON\_H

#define OOP\_EXERCISE\_03\_PENTAGON\_H

#include "Figure.h"

class Pentagon: public Figure{

public:

Pentagon(Cord center, double radius);

double Area() override;

void print() override;

};

#endif

**Pentagon.cpp**

//Федоров Антон Сергеевич М8О-207Б-19

//Вариант 30 5-угольник, 6-угольник, 8-угольник

#include "Pentagon.h"

Pentagon::Pentagon(Cord center, double radius):Figure(center,radius){};

double Pentagon::Area(){

return 5 \* 0.5 \* sin((double) 72 / 180 \* PI) \* Radius \* Radius;

}

void Pentagon::print(){

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

Cord radVector(Radius \* sin((double) (i \* 72) / 180 \* PI), Radius \* cos((double) (i \* 72) / 180 \* PI));

std::cout << Center + radVector << ", ";

}

}

**Hexagon.h**

//Федоров Антон Сергеевич М8О-207Б-19

//Вариант 30 5-угольник, 6-угольник, 8-угольник

#ifndef OOP\_EXERCISE\_03\_HEXAGON\_H

#define OOP\_EXERCISE\_03\_HEXAGON\_H

#include "Figure.h"

class Hexagon: public Figure{

public:

Hexagon(Cord center, double radius);

double Area() override;

void print() override;

};

#endif

**Hexagon.cpp**

//Федоров Антон Сергеевич М8О-207Б-19

//Вариант 30 5-угольник, 6-угольник, 8-угольник

#include "Hexagon.h"

Hexagon::Hexagon(Cord center, double radius):Figure(center,radius){};

double Hexagon::Area(){

return 6 \* 0.5 \* sin((double) 60 / 180 \* PI) \* Radius \* Radius;

}

void Hexagon::print(){

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

Cord radVector(Radius \* sin((double) (i \* 60) / 180 \* PI), Radius \* cos((double) (i \* 60) / 180 \* PI));

std::cout << Center + radVector << ", ";

}

}

**Octagon.h**

//Федоров Антон Сергеевич М8О-207Б-19

//Вариант 30 5-угольник, 6-угольник, 8-угольник

#ifndef OOP\_EXERCISE\_03\_OCTAGON\_H

#define OOP\_EXERCISE\_03\_OCTAGON\_H

#include "Figure.h"

class Octagon: public Figure{

public:

Octagon(Cord center, double radius);

double Area() override;

void print() override;

};

#endif

**Octagon.cpp**

//Федоров Антон Сергеевич М8О-207Б-19

//Вариант 30 5-угольник, 6-угольник, 8-угольник

#include "Octagon.h"

Octagon::Octagon(Cord center, double radius):Figure(center,radius){};

double Octagon::Area(){

return 8 \* 0.5 \* sin((double) 45 / 180 \* PI) \* Radius \* Radius;

}

void Octagon::print(){

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

Cord radVector(Radius \* sin((double) (i \* 45) / 180 \* PI), Radius \* cos((double) (i \* 45) / 180 \* PI));

std::cout << Center + radVector << ", ";

}

}

**main.cpp**

*//Федоров Антон Сергеевич М8О-207Б-19*

*//Вариант 30 5-угольник, 6-угольник, 8-угольник*

#include <iostream>

#include <vector>

#include "Pentagon.h"

#include "Hexagon.h"

#include "Octagon.h"

void SummaryData(std::vector<Figure\*> figuresPtr){

double summaryArea = 0;

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

std::cout << "figure number " << i << "\n";

std::cout << "center:" << figuresPtr[i]->GetCenter() << "\n";

std::cout << "cords: ";

figuresPtr[i]->print();

std::cout << "\n";

std::cout << "area: " << figuresPtr[i]->Area() << "\n";

summaryArea += figuresPtr[i]->Area();

}

std::cout << "summary area: " << summaryArea << "\n";

}

int main(){

std::vector<Figure\*> figuresPtr;

Cord curCenter;

double curRadius;

double x,y;

int type;

std::cout << "insert operation type:";

while (std::cin >> type){

if (type == 1){

std::cout << "insert pentagon center and radius:";

if (!(std::cin >> curCenter.X >> curCenter.Y >> curRadius)){

std::cout << "invalid input";

return -1;

}

Pentagon pentagon(curCenter, curRadius);

figuresPtr.push\_back(&pentagon);

}

else if (type == 2){

std::cout << "insert hexagon center and radius:";

if (!(std::cin >> curCenter.X >> curCenter.Y >> curRadius)){

std::cout << "invalid input";

return -1;

}

Hexagon hexagon(curCenter, curRadius);

figuresPtr.push\_back(&hexagon);

}

else if (type == 3){

std::cout << "insert octagon center and radius:";

if (!(std::cin >> curCenter.X >> curCenter.Y >> curRadius)){

std::cout << "invalid input";

return -1;

}

Octagon octagon(curCenter, curRadius);

figuresPtr.push\_back(&octagon);

}

else if (type == 0){

SummaryData(figuresPtr);

}

else if (type == -1){

int index;

std::cout << "insert index to delete:";

if (!(std::cin >> index)){

std::cout << "invalid input";

return -1;

}

if (index < 0 && index >= figuresPtr.size()){

std::cout << "invalid input";

return -1;

}

figuresPtr.erase(figuresPtr.begin() + index);

}

std::cout << "insert operation type:";

}

return 0;

}

**Выводы**

Я научился использовать наследование для написания схожих классов, использовать абстрактные классы и виртуальные функции.

**Список литературы**

1. RIP Tutorial C++ Наследование конструкторов

URL:[https://riptutorial.com/ru/cplusplus/example/28809/наследование-конструкторов](https://riptutorial.com/ru/cplusplus/example/28809/%D0%BD%D0%B0%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5-%D0%BA%D0%BE%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D0%BE%D1%80%D0%BE%D0%B2)

(дата обращения: 18.10.2020).

1. Урок №154. Базовое наследование

URL:<https://ravesli.com/urok-154-bazovoe-nasledovanie-v-c/>

(дата обращения: 18.10.2020).