

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Московский государственный технологический университет «СТАНКИН»

(ФГБОУ ВО «МГТУ «СТАНКИН»)

Институт цифровых интеллектуальных систем  
Кафедра электротехники, электроники и автоматики

Образовательная программа 15.03.06

«Мехатроника и робототехника»

Дисциплина «Программирование и алгоритмизация»

Отчет по лабораторной работе №1

«Структуры данных:

связный список, очередь и стек.»

Вариант №14

Выполнил:

Студент АДБ-23-11 \_\_\_\_\_\_\_\_\_\_ Анохин В.И.

(дата) (подпись)

Принял:

к.т.н., доцент \_\_\_\_\_\_\_\_\_\_ Соколов С.В.

(дата) (подпись)

Москва 2024

**Введение**

Цель работы: изучить принципы наследования и полиморфизма в языке С++ на примере создания структуры классов для хранения данных геометрических фигур

**Ход работы**

Следуя методичке, создаём класс FigureBase, от которого будем наследовать все остальные классы. В нём лежит весь функционал, не зависящий от конкретного типа фигуры: хранение имён, создание некоторых полей и т. д. Наследуем от него классы Rectangle, Triangle и Circle. От класса Rectangle наследуем класс Square. Некоторые функции в методичке отсутствуют, они дописаны мной для упрощения дальнейшей работы. Перейдём к рассмотрению заданий.

**Реализовать класс, описывающий геометрическую фигуру окружности**

#pragma once

#include "FigureBase.h"

class Circle :

public FigureBase

{

public:

Circle() { radius = 0; };

~Circle() { };

void pushRadius(int r)

{

this->radius = r;

this->computeArea();

this->computePerimeter();

}

void pushPointList(vector<Point> v)

{

this->pointList = v;

this->computeBoundary();

this->computeArea();

this->computePerimeter();

};

double GetArea()

{

return this->area;

}

protected:

int radius;

const int pointCount = 1;

bool checkForPointValidity()

{

return ((this->pointList).size() == pointCount);

}

void computeBoundary()

{

vector<Point> v = this->pointList;

if ((v.size() > 0) && checkForPointValidity())

{

auto it = v.begin();

int min\_x = (\*it).x - radius;

int min\_y = (\*it).y - radius;

int max\_x = (\*it).x + radius;

int max\_y = (\*it).y + radius;

this->Box = BoundingBox{ Point{min\_x, max\_y}, Point{max\_x, min\_y} };

};

};

void computeArea()

{

double r = this->radius;

this->area = (3.141529 \* r \* r);

};

void computePerimeter()

{

double r = this->radius;

this->perimeter = (2 \* 3.141529 \* r);

}

};

**Дополнить базовый класс и производные от него объекты методами, возвращающими их площадь и периметр.**

В базовый класс добавим поля для площади и периметра и заглушки функций для обновления их значений. Функции с названиями Get….() просто возвращают значения поля класса, подсчёты идут в другой функции.

double computeArea(){}

double computePerimeter(){}

double GetArea()

{

return this->area;

}

double GetPerimeter()

{

return this->perimeter;

}

В каждом производном классе эти функции обновляются по-своему, в зависимости от типа фигуры. К примеру, вот так они выглядят для класса Rectangle:

void computeArea()

{

int dx = this->Getdx();

int dy = this->Getdy();

this->area = dx \* dy;

};

void computePerimeter()

{

int dx = this->Getdx();

int dy = this->Getdy();

this->perimeter = 2\*(dx + dy);

};

int Getdx()

{

Point p1 = this->Box.A;

Point p2 = this->Box.B;

int dx = abs(p1.x - p2.x);

return dx;

}

int Getdy()

{

Point p1 = this->Box.A;

Point p2 = this->Box.B;

int dy = abs(p1.y - p2.y);

return dy;

}

**Реализовать класс Triangle для работы с треугольником**

Класс Triangle наследуется от FigureBase и почти ничего кроме вышеупомянутых двух функций не переопределяет. Полный код представлен ниже:

#pragma once

#include "FigureBase.h"

#include <math.h>

using namespace std;

class Triangle : public FigureBase

{

public:

Triangle() {};

~Triangle() {};

virtual void UselessFunction() //Дебаг. Если вы это читаете, я забыл её вырезать.

{

cout << "Knock-Knock-Knock";

}

int Getdx()

{

Point p1 = this->Box.A;

Point p2 = this->Box.B;

int dx = abs(p1.x - p2.x);

return dx;

}

int Getdy()

{

Point p1 = this->Box.A;

Point p2 = this->Box.B;

int dy = abs(p1.y - p2.y);

return dy;

}

double Getdz(Point a1, Point a2)

{

Point p1 = a1;

Point p2 = a2;

int dx = abs(p1.x - p2.x);

int dy = abs(p1.y - p2.y);

double dz = sqrt(dx \* dx + dy \* dy);

return dz;

}

void pushPointList(vector<Point> v)

{

this->pointList = v;

this->computeBoundary();

this->computeArea();

this->computePerimeter();

};

protected:

const int pointCount = 3;

bool checkForPointValidity()

{

return ((this->pointList).size() == pointCount);

}

void computeArea()

{

double l1 = Getdz(pointList[0], pointList[1]);

double l2 = Getdz(pointList[1], pointList[2]);

double l3 = Getdz(pointList[2], pointList[0]);

double p = (l1 + l2 + l3) / 2.0;

this->area = sqrt((p)\*(p-l1)\*(p-l2)\*(p-l3));

};

void computePerimeter()

{

double l1 = Getdz(pointList[0], pointList[1]);

double l2 = Getdz(pointList[1], pointList[2]);

double l3 = Getdz(pointList[2], pointList[0]);

this->perimeter = l1 + l2 + l3;

};

};

**Вывод**

Мы изучили принципы наследования и полиморфизма в языке С++ на примере создания структуры классов для хранения данных геометрических фигур. Полный код прилагается в архиве вместе с отчётом.