Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Уфимский государственный авиационный технический университет»

Кафедра ВВТиС

Отчет по лабораторной работе №6  
по дисциплине «Программирование»

«…»

Группа ПМИ-150

Выполнил \_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_ Поташин А.В.

(дата) (подпись) (ФИО)

Проверил \_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_

(дата) (подпись) (ФИО)

Уфа 2019

# 1. Цели и задачи

Целью лабораторной работы является ознакомление с описанием и использованием структур, а также понять принципы создания классов.

Для достижения цели необходимо выполнить следующие задачи:

1. Создать класс Окружность, имеющий поля:  
 - координаты центра,  
 - радиус.  
 Определить  
 - пустой конструктор и конструктор с полным заполнением полей,  
 - деструктор (с очисткой памяти),  
 - простые методы доступа к полям (set- и get-методы),  
 - метод вывода полной информации об объекте класса,  
 - метод расчета длины окружности.  
 Класс должен быть разработан с соблюдением всех принципов объектноориентированного программирования Создать объекты данного класса в функции main() и продемонстрировать работу всех методов класса.   
2. Создать public-производный класс Круг, имеющий дополнительно поля:  
 - цвет заливки.  
 Определить  
 - пустой конструктор и конструктор с полным заполнением полей,  
 - деструктор (с очисткой памяти),  
 - простые методы доступа к полям (set- и get-методы),  
 - метод вывода полной информации об объекте класса (перегрузка метода родителя).  
 Создать объекты данного класса в функции main() и продемонстрировать работу всех методов класса (в т.ч. унаследованных).

# 2. Теоретическая часть

## Что такое класс данных?

## Что входит в состав класса?

## Чем классы отличаются от структур и объединений?

## Классы и структуры – что из них поддерживает наследование?

## Из каких двух частей состоит описание класса в C++?

## Для чего необходим и когда вызывается конструктор класса?

## Для чего необходим и когда вызывается деструктор класса?

## При помощи чего в классах обеспечивается инкапсуляция (сокрытие) внутренней структуры данных?

## Опишите назначение модификаторов видимости private, protected и public. Общие черты и различия.

## Что такое наследование и иерархия?

## Как и для чего используется служебное слово virtual?

## Что такое абстрактный класс?

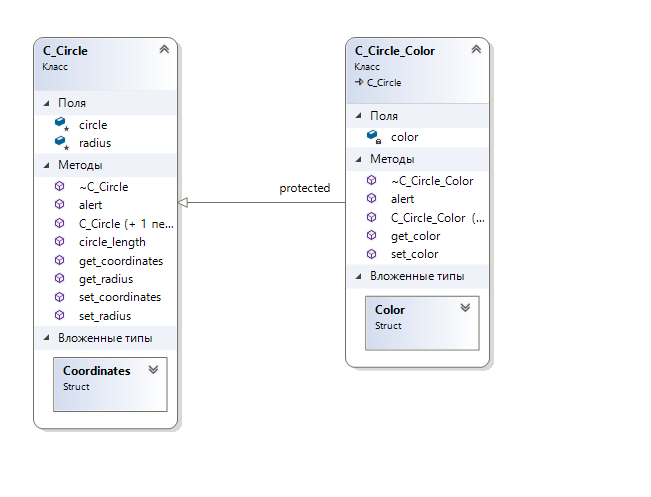
## Почему конструктор не может быть виртуальным, а деструктор почти всегда является виртуальным? 2.1. Ответы на контрольные вопросы

## Классы данных - это классы, которые содержат только поля и простейшие методы для доступа к ним.

1. Поля и методы.
2. Способом создания и модификатором доступа к своим полям.
3. И классы и структуры.
4. Объявление названия и описания содержания(поля и методы).
5. Вызывается при создании объекта класса, для “заполнения” полей начальными значениями.
6. Для отчистки памяти и уничтожения объекта.
7. При помощи разных модификаторов доступа.
8. Private – для использования внутри класса, protected – доступ для наследников, public – для общего использования.
9. Наследование это создание нового класса на основе уже существующего. Иерархия – “наследственные” связи между классами.
10. Указывает на возможность различной реализации в потомках.
11. класс, который не предполагает создания экземпляров.
12. Деструктор базового класса не может вызвать деструктор производного класса.

# 3. ПРАКТИЧЕСКАЯ ЧАСТЬ

Диаграмма:



**Main.cpp**

#include <iostream>

#include <math.h>

#include "C\_Circle.h"

#include "C\_Circle\_Color.h"

using namespace std;

int main() {

setlocale(LC\_ALL, "Russian");

C\_Circle \*fObj = new C\_Circle();

C\_Circle\* sObj = new C\_Circle(3, 10, 5);

C\_Circle\_Color\* ffObj = new C\_Circle\_Color();

C\_Circle\_Color\* ssObj = new C\_Circle\_Color(1, 1, 1, 22, 33, 44);

fObj->alert();

sObj->alert();

ffObj->alert();

ssObj->alert();

return 0;

}

**C\_Circle.h**

#pragma once

class C\_Circle

{

public:

struct Coordinates {

float x\_coord;

float y\_coord;

};

void set\_coordinates(float, float);

void set\_radius(float);

float get\_radius();

Coordinates get\_coordinates();

double circle\_length();

void alert();

C\_Circle();

C\_Circle(float, float, float);

~C\_Circle();

protected:

float radius;

Coordinates circle;

};

**C\_Circle.cpp**

#define \_USE\_MATH\_DEFINES

#include "C\_Circle.h"

#include <iostream>

#include <math.h>

using namespace std;

void C\_Circle::set\_coordinates(float x, float y) {

this->circle.x\_coord = x;

this->circle.y\_coord = y;

};

void C\_Circle::set\_radius(float r) {

this->radius = r;

};

float C\_Circle::get\_radius() {

return this->radius;

};

C\_Circle::Coordinates C\_Circle::get\_coordinates() {

return this->circle;

};

double C\_Circle::circle\_length() {

return M\_PI \* this->radius \* 2;

};

void C\_Circle::alert() {

cout << "Радиус: " << this->get\_radius() << endl;

cout << "Х и Y координаты: " << this->get\_coordinates().x\_coord << ", " << this->get\_coordinates().y\_coord << endl;

cout << "Длинна окружности: " << this->circle\_length() << endl;

}

C\_Circle::C\_Circle()

{

set\_coordinates(4, 4);

set\_radius(2);

}

C\_Circle::C\_Circle(float x, float y, float r)

{

set\_coordinates(x, y);

set\_radius(r);

}

C\_Circle::~C\_Circle()

{

}

**C\_Circle\_Color.h**

#pragma once

#include "C\_Circle.h"

class C\_Circle\_Color :

protected C\_Circle

{

public:

struct Color

{

int r;

int g;

int b;

};

void set\_color(int, int, int);

Color get\_color();

void alert();

C\_Circle\_Color();

C\_Circle\_Color(int, int, int, int, int, int);

~C\_Circle\_Color();

private:

Color color;

};

**C\_Circle\_Color.cpp**

#include "C\_Circle\_Color.h"

#include "C\_Circle.h"

#include <iostream>

#include <math.h>

using namespace std;

void C\_Circle\_Color::set\_color(int r, int g, int b) {

this->color.r = r;

this->color.g = g;

this->color.b = b;

};

C\_Circle\_Color::Color C\_Circle\_Color::get\_color() {

return this->color;

};

void C\_Circle\_Color::alert() {

cout << "Радиус: " << this->get\_radius() << endl;

cout << "Х и Y координаты: " << this->get\_coordinates().x\_coord << ", " << this->get\_coordinates().y\_coord << endl;

cout << "Цвет(RGB): " << this->color.r << this->color.g << this->color.b << endl;

};

C\_Circle\_Color::C\_Circle\_Color()

{

set\_color(0, 0, 0);

}

C\_Circle\_Color::C\_Circle\_Color(int x, int y, int rad, int r, int g, int b) : C\_Circle(x, y ,rad)

{

set\_color(r, g, b);

}

C\_Circle\_Color::~C\_Circle\_Color()

{

}

Пример выполнения программы:

