Министерство высшего образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет» (ПНИПУ)**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

ОТЧЁТ

по лабораторной работе №2

Тема: «Классы и объекты. Инкапсуляция.»

Выполнил

Студент группы РИС-22-2б

Шальнев А.В.

Проверил доц. Кафедры ИТАС

Полякова Ольга Андреевна

Пермь 2023

# Постановка задачи

1. Определить пользовательский класс.
2. Определить в классе следующие конструкторы: без параметров, с параметрами, копирования.
3. Определить в классе деструктор.
4. Определить в классе компоненты-функции для просмотра и установки полей данных (селекторы и модификаторы).
5. Написать демонстрационную программу, в которой продемонстрировать все три случая вызова конструктора-копирования, вызов конструктора с параметрами и конструктора без параметров.

ВАРИАНТ 15:

Пользовательский класс ЗАРПЛАТА

- ФИО – string

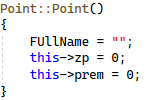
- Оклад – double

- Премия (% от оклада) – int

**Контрольные вопросы**

1. *Для чего нужен конструктор?*

Конструктор позволяет избежать ошибок, которые связаны с использованием в программе неинициализированных переменных.



здесь мы обнуляем атрибуты объекта класса Point

1. *Сколько типов конструкторов существует*

Существует три типа конструкторов. А именно, конструктор с параметрами, конструктор без параметров и конструктор копирования.

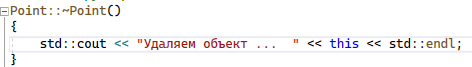






1. *Для чего нужен деструктор? В каких случаях деструктор используется явно?*

Деструктор служит для удаления ресурсов, выделенных ранее под объект, в том числе и локальные атрибуты. Деструктор исползуется явно в тех случаях, когда динамические данные, или же указатели, нужно очистить.



4-5. *Для чего конструктор без параметров? С параметрами? Копирования?*

Конструктор копирования применяется в трех случаях: для создания нового объекта, копированием существующего объекта; при передаче объекта в функцию по значению и при возврате объекта из функции.

Для создания экземпляра класса без начальных значений применяется конструктор без параметров.

Для инициализации объекта с определенными параметрами применяется конструктор с параметрами.

1. *Перечислите свойства конструкторов*
2. Конструктор не возвращает значение, даже типа void. Нельзя получить указатель на конструктор.
3. Класс может иметь несколько конструкторов с разными параметрами для разных видов инициализации (при этом используется механизм перегрузки).
4. Конструктор, вызываемый без параметров, называется конструктором по умолчанию.
5. Параметры конструктора могут иметь любой тип, кроме этого же класса. Можно задавать значения параметров по умолчанию, но их может содержать только один из конструкторов.
6. Если программист не указал ни одного конструктора, компилятор создает его автоматически. Такой конструктор вызывает конструкторы по умолчанию для полей класса. В случае, когда класс содержит константы или ссылки, при попытке создания объекта класса будет выдана ошибка, поскольку их необходимо инициализировать конкретными значениями, а конструктор по умолчанию этого делать не умеет.
7. Конструкторы не наследуются.
8. Конструкторы нельзя описывать с модификаторами const, virtual и static.
9. Конструкторы глобальных объектов вызываются до вызова функции main. Локальные объекты создаются, как только становится активной область их действия. Конструктор запускается и при создании временного объекта (на пример, при передаче объекта из функции).
10. *Свойства деструкторов*
    * 1. не имеет аргументов и возвращаемого значения;
      2. не наследуется;
      3. не может быть объявлен как const или static (далее);
      4. может быть виртуальным (далее).
11. *К каким атрибутам имеют доступ методы класса*

Методы класса имеют доступ о всем элементам класса вне зависимости от спецификатора доступа

1. *Что представляет собой запись: this*

this - Это указатель на текущий объект

1. *Как разница между методами, определенными внутри класса и вне класса*

Между методами, определенными внутри класса и вне класса нет разницы.

1. *Какое значение возвращает конструктор?*

Конструктор не возвращает значение.

1. *Какие методы создаются по умолчанию?*

По умолчанию создаются конструктор без параметров, копирования, деструктор.

1. *Какое значение возвращает деструктор?*

Деструктор не возвращает значение.

*Дано описание класса class Student*

*{*

*string name; int group; public:*

*student(string, int); student(const student&)*

*~student();*

*};*

*Какой метод отсутствует в описании класса?*

В описании класса отсутствует конструктор без параметров

*Какой метод будет вызван при выполнении следующих операторов: student\*s;*

*s=new student;*

При выполнении следующих операторов будет вызван конструктор без параметров

*Какой метод будет вызван при выполнении следующих операторов: student s(“Ivanov”,20);*

При выполнении следующих операторов будет вызван конструктор с параметрами

*Какие методы будут вызваны при выполнении следующих операторов: student s1(“Ivanov”,20);*

*student s2=s1;*

При выполнении следующих операторов будет вызван конструктор с параметрами, копирования

*Какие методы будут вызваны при выполнении следующих операторов: student s1(“Ivanov”,20);*

*student s2; s2=s1;*

При выполнении следующих операторов будут вызваны конструктор с параметрами и без, копирования

*Какой конструктор будет использоваться при передаче параметра в функцию print():*

*void print(student a)*

*{*

*a.show();*

*}*

При передаче параметра в функцию print() будет использоваться конструктор копирования

*Класс описан следующим образом:*

*class Student*

*{*

*string name; int age;*

*public:*

*void set\_name(string);*

*void set\_age(int );*

*…..*

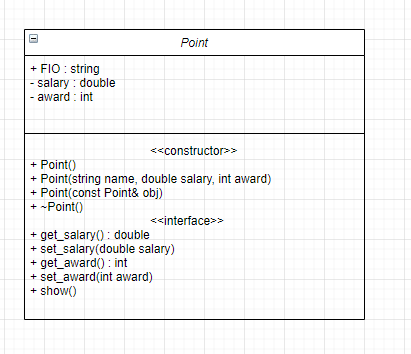
*};*

*Student p;*

*Каким образом можно присвоить новое значение атрибуту name объекта р?*

Присвоить значение атрибуту name объекта p можно с помощью сеттера p.set\_name(string YOUR TEXT);

UML таблица



Код программы

Point.h:

#pragma once

#include <string>;

class Point

{

public:

std::string FIO;

Point();

Point(std::string name, double salary, int award);

Point(const Point& obj);

~Point();

double get\_salary();

void set\_salary(double salary);

int get\_award();

void set\_award(int award);

void show();

private:

double salary;

int award;

};

Point.cpp:

#pragma once

#include "Point.h"

#include <iostream>

// Конструктор без параметров

Point::Point()

{

FUllName = "";

this->zp = 0;

this->prem = 0;

}

// Конструктор с параметрами

Point::Point(std::string name, double zp, int prem)

{

FullName = name;

this->zp = zp;

this->prem = prem;

}

// Конструктор копирования

Point::Point(const Point& obj)

{

this->FullName = obj.FullName;

this->zp = obj.zp;

this->prem = obj.prem;

}

// Деконструктор

Point::~Point()

{

std::cout << "Удаляем объект ... " << this << std::endl;

}

// Геттер зарплаты

double Point::get\_zp()

{

return zp;

}

// Сеттер зарплаты

void Point::set\_zp(double zp)

{

this->zp = zp;

}

// Геттер премии

int Point::get\_prem()

{

return prem;

}

// Сеттер премии

void Point::set\_prem(int prem)

{

this->prem = prem;

}

// Вывод локальных атрибутов объекта класса Point;

void Point::show()

{

std::cout << "FullName: " << FullName << std::endl << "zp: " << zp << std::endl << "prem: " << prem << std::endl;

}