МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

ЭЛЕКТРОТЕХНИЧЕСКИЙ ФАКУЛЬТЕТ

КАФЕДРА «ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И АВТОМАТИЗИРОВАННЫХ СИСТЕМ»

**ОТЧЁТ**

**«ЛАБОРАТОРНАЯ №2:**

**КЛАССЫ — МЕТОДЫ»**

Дисциплина: «Программирование»

Выполнил:

Студент группы ИВТ-21-2б

Безух Владимир Сергеевич

Проверил:

Доцент кафедры ИТАС

Полякова Ольга Андреевна

Пермь, 2022

Содержание

[1. Постановка задачи 3](#_Toc100772848)

[2. Контрольные вопросы 4](#_Toc100772849)

[3. Исходный код 6](#_Toc100772850)

# Постановка задачи

1. Реализовать класс Employee с полями имени, зарплаты и премии.

# 

# Контрольные вопросы

1. **Для чего нужен конструктор?**

Для совершения действий во время инициализации объекта.

1. **Сколько типов конструкторов существует в C++?**

Три типа конструкторов.

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

Для совершения действий во время завершения времени жизни объекта. Деструктор описывается явно, например, в тех случаях, когда нужно вручную высвободить динамическую память.

1. **Для чего используется конструктор без параметров, с параметрами, копирования?**

Для инициализации объекта «по умолчанию», объекта с требуемыми значениями для аналогичного тому, который уже существует.

1. **В каких случаях вызывается конструктор копирования?**

В тех случаях, когда новый объект создаётся путём копирования существующего: при описании нового объекта с инициализацией другим объектом, при передаче объекта в функцию по значению, при возврате объекта из функции.

1. **Перечислить свойства конструкторов.**

Конструктор не возвращает значение, нельзя получить указатель на конструктор.

Класс может иметь несколько конструкторов с параметрами, реализованных через механизм перегрузки.

Конструктор без параметров называется конструктором по умолчанию.

Параметры конструктора не могут быть типа этого класса.

Конструкторы не наследуются.

Конструкторы нельзя описывать с модификаторами const, virtual и static.

1. **Перечислить свойства деструкторов.**

Не имеет аргументов и возвращаемого значения.

Не наследуется.

Не может быть объявлен как const или static.

Может быть виртуальным.

1. **К каким атрибутам имеют доступ методы класса?**

Неограниченный доступ ко всем компонентам класса.

1. **Что представляет собой указатель this?**

Указатель на текущий экземпляр класса.

1. **Какая разница между методами определенными внутри класса и вне класса?**

Методы, описанные вне класса, содержат в описании класса только свой прототип.

1. **Какое значение возвращает конструктор?**

Никакое.

1. **Какие методы создаются по умолчанию?**

Конструктор по умолчанию, конструктор копирования, деструктор по умолчанию, оператор присваивания.

1. **Какое значение возвращает деструктор?**

Никакое.

# Исходный код

#include <iostream>

class FullName

{

public:

friend std::istream& operator>>(std::istream& input, FullName& full\_name)

{

input >> full\_name.surname\_ >> full\_name.forename\_ >> full\_name.patronym\_;

return input;

}

friend std::ostream& operator<<(std::ostream& output, const FullName& full\_name)

{

output << full\_name.surname\_ << ' ' << full\_name.forename\_ << ' ' << full\_name.patronym\_;

return output;

}

private:

std::string surname\_;

std::string forename\_;

std::string patronym\_;

};

class Employee

{

public:

Employee() { FullName personal\_name\_; double salary\_; double bonus\_percentage\_; }

Employee(FullName personal\_name = FullName(), double salary = 0.0, double bonus\_percentage = 0.0) :

personal\_name\_(personal\_name), salary\_(salary), bonus\_percentage\_(bonus\_percentage) {}

Employee(const Employee& copy) :

personal\_name\_(copy.personal\_name\_), salary\_(copy.salary\_), bonus\_percentage\_(copy.bonus\_percentage\_) {}

~Employee() {}

FullName getName() { return personal\_name\_; }

double getSalary() { return salary\_; }

double getBonusPercentage() { return bonus\_percentage\_; }

void setName(const FullName& personal\_name) { personal\_name\_ = personal\_name; }

void setSalary(const double& salary) { salary\_ = salary; }

void setBonusPercentage(const double& bonus\_percentage) { bonus\_percentage\_ = bonus\_percentage; }

friend std::istream& operator>>(std::istream& input, Employee& employee)

{

input >> employee.personal\_name\_ >> employee.salary\_ >> employee.bonus\_percentage\_;

return input;

}

friend std::ostream& operator<<(std::ostream& output, const Employee& employee)

{

output << employee.personal\_name\_ << ' ' << employee.salary\_ << ' ' << employee.bonus\_percentage\_;

return output;

}

private:

FullName personal\_name\_;

double salary\_;

double bonus\_percentage\_;

};

int main()

{}