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

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

2. Контрольные вопросы

1. **Для чего нужен конструктор?**
   Для совершения действий во время инициализации объекта.

2. **Сколько типов конструкторов существует в C++?**
   Три типа конструкторов.

3. **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.

4. **Для чего используется конструктор без параметров, с параметрами, копирования?**
   Для инициализации объекта «по умолчанию», объекта с требуемыми значениями для аналогичного тому, который уже существует.

5. **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.

6. **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.

7. **Перечислить свойства деструкторов.**
   Не имеет аргументов и возвращаемого значения.

   Не наследуется.

   Не может быть объявлен как const или static.

   Может быть виртуальным.

8. **К каким атрибутам имеют доступ методы класса?**
   Неограниченный доступ ко всем компонентам класса.

9. **Что представляет собой указатель this?**
   Указатель на текущий экземпляр класса.

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

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

11. **Какое значение возвращает конструктор?**
    Никакое.

12. **Какие методы создаются по умолчанию?**
    Конструктор по умолчанию, конструктор копирования, деструктор по умолчанию, оператор присваивания.

13. **Какое значение возвращает деструктор?**
    Никакое.

3. 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()
{}
```