Федеральное государственное автономное образовательное учреждение высшего образования «Пермский национальный исследовательский политехнический университет», ПНИПУ

ЛАБОРАТОРНАЯ РАБОТА

КЛАССЫ И ОБЪЕКТЫ. ИСПОЛЬЗОВАНИЕ КОНСТРУКТОРОВ.

Выполнил: студент группы РИС-23-3б

Артем Владимирович Швецов

Проверила: доцент кафедры ИТАС

Ольга Андреевна Полякова

Пермь 2024

**Постановка задачи**

1. Определить пользовательский класс.
2. Определить в классе конструкторы: без параметров, с параметрами, копирования.
3. Определить в классе деструктор.
4. Определить в классе методы для просмотра и установки полей данных.

Для варианта 15:

Пользовательский класс – ЗАРПЛАТА

ФИО – string

Оклад – double

Премия – int

**Анализ задачи**

1. Создается новый файл с расширением .h, где определяется новый класс Pay.
2. Создается парный .cpp файл, в котором определяются методы нового класса.

**Код**

Pay.h

#pragma once

#include <string>

class Pay {

std::string name;

double salary;

int bonus;

public:

Pay();

Pay(std::string, double, int);

Pay(const Pay&);

~Pay();

std::string getName ();

void setName(std::string);

double getSalary();

void setSalary(double);

int getBonus();

void setBonus(int);

void Show();

};

Pay.cpp

#pragma once

#include <string>

#include <iostream>

#include "Pay.h"

using namespace std;

Pay::Pay() {

name = "empty";

salary = 0;

bonus = 0;

cout << "Constructor no\_values for object: " << this << endl;

}

Pay::Pay(string name = "empty", double salary = 0, int bonus = 0) {

this->name = name;

this->salary = salary;

this->bonus = bonus;

cout << "Constructor have\_values for object: " << this << endl;

}

Pay::Pay(const Pay& other) {

this->name = other.name;

this->salary = other.salary;

this->bonus = other.bonus;

cout << "Constructor copy\_values for object: " << this << endl;

}

Pay::~Pay() {

cout << "Destructor for object: " << this << endl;

}

std::string Pay::getName() {

return this->name;

}

void Pay::setName(std::string name) {

this->name = name;

}

double Pay::getSalary() {

return this->salary;

}

void Pay::setSalary(double salary) {

this->salary = salary;

}

int Pay::getBonus() {

return this->bonus;

}

void Pay::setBonus(int bonus) {

this->bonus = bonus;

}

void Pay::Show() {

cout << "Name: " << this->name << endl <<

"Salary: " << this->salary << endl <<

"Bonus: " << this->bonus << "%" << endl << endl;

}

Main.cpp

#include <iostream>

#include <string>

#include "Pay.h"

using namespace std;

int main() {

Pay p1;

p1.Show();

Pay p2("Abibik Jmishevich", 10000.45, 10);

p2.Show();

Pay p3;

p3.setName("Oleg");

p3.setSalary(2525.25);

p3.setBonus(400);

cout << "Name: " << p3.getName() << endl <<

"Salary: " << p3.getSalary() << endl <<

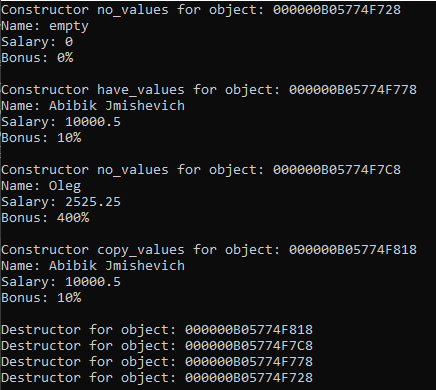
"Bonus: " << p3.getBonus() << "%" << endl << endl;

Pay p4(p2);

p4.Show();

}

**Решение**



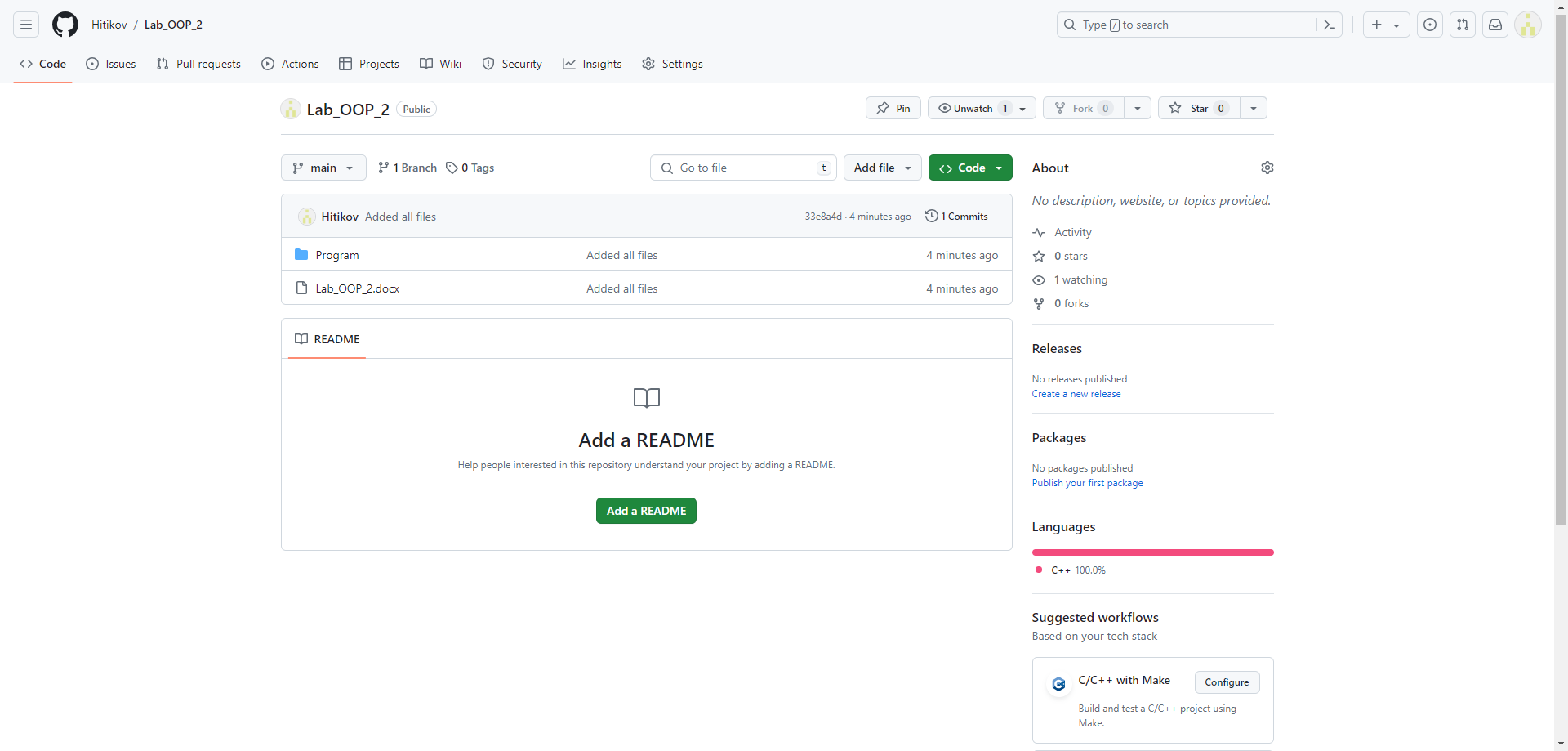
**Выводы**

В ходе выполнения работы были изучены механизмы создания экземпляров класса, а также конструкторы и деструкторы и методы работы с ними.

**Ответы на контрольные вопросы:**

1. Для выполнения заданных действий при создании экземпляра класса.
2. 3: без параметров, с параметрами, копирования
3. Для выполнения заданных действий при уничтожении экземпляра класса. Описывается явно, если была использована динамическая память, либо нужно выполнить действие не освобождающее память.
4. Для создания экземпляра с базовыми значениями атрибутов. Для создания экземпляра с значениями атрибутов соответствующими параметрам. Для создания экземпляра с значениями атрибутов идентичными значениям атрибутов уже существующего экземпляра.
5. Когда нужно сохранить копию значений атрибутов.
6. Не возвращает значения, класс может иметь несколько конструкторов, не наследуется, нельзя описывать модификаторами, без параметров – по умолчанию.
7. Имя начинается с тильды, не возвращает значения, не наследуется, нельзя описывать модификаторами
8. Ко всем
9. Указатель на объект.
10. Разный уровень доступа к атрибутам, описанные вне класса могут не иметь доступа к защищенным атрибутам.
11. Никакое.
12. Конструктор без параметров, деструктор.
13. Никакое.
14. Конструктор без параметров.
15. Конструктор без параметров.
16. Конструктор с параметрами.
17. Конструктор с параметрами, копирования.
18. Конструктор без параметров, с параметрами, копирования.
19. Никакой, при передачи объекта как параметра предполагается, что он уже создан.
20. p.set\_name(/новое значение/)

**Github**



<https://github.com/Hitikov/Lab_OOP_2>