МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

**«ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**Лабораторная работа № 4:**

"Простое наследование. Принцип подстановки."

Выполнила: студентка гр.РИС-23-3Б

Мокроусова Ангелина Андреевна

Проверила: доцент кафедры ИТАС

Полякова Ольга Андреевна

Пермь 2024

**Постановка задачи:**

Общая часть:

* Определить пользовательский класс
* Определить в нем все 3 типа конструкторов
* Определить в классе деструктор
* Определить в классе компоненты-функции для просмотра и установки полей данных(селекторы и модификаторы)
* Перегрузить операции присваивания, ввода вывода с помощью потоков
* Определить производный класс
* Написать программув которой продемонстрировать создание объектов и работу перегруженныхопераций
* Реализовать функции, получающие и возвращающие объект базового класса. Продемонстрировать принцип подстановки

Задание варианта:

Базовый класс – PERSON

* имя (string)
* возраст (int).

Определить методы изменения полей.

Произольный класс – EMPLOYEE

* должность (string)
* оклад (double)

Определить методы изменения полей и вычисления зарплаты сотрудника по формуле: оклад + премия (%от окладка (вводится с клавиатуры))

**Анализ задачи:**

Базовый класс PERSON:

Поля данных:

* Имя (строка)
* Возраст (целое число)

Методы:

* Конструкторы: без параметров, с параметрами, копирования
* Деструктор
* Селекторы и модификаторы для полей данных

Методы для изменения полей данных (например, setName() и setAge())

Производный класс EMPLOYEE:

Поля данных:

* Должность (строка)
* Оклад (вещественное число)

Методы:

* Конструкторы: без параметров, с параметрами, копирования
* Деструктор
* Селекторы и модификаторы для полей данных
* Методы для изменения полей данных
* Метод для вычисления зарплаты сотрудника

Внешнии функции:

* Функция createPerson будет задавать значения полей класса с консоли
* Функция printPerson будет выводить информацию о классе

Основная функция: будет задавать процент премии, создавать и выводить производный класс, подсчитывать и выводить на экран итогувую зарплату сотрудника, удалять класс после отработки.

**Код на С++ с комментариями:**

#include<iostream>

using namespace std;

// Базовый класс PERSON

class Person {

protected:

string name;

int age;

public:

// Конструктор по умолчанию

Person() {}

// Конструктор с параметрами

Person(string name, int age) : name(name), age(age) {}

// Конструктор копирования

Person(const Person& other) : name(other.name), age(other.age) {}

// Деструктор

~Person() {}

// Селектор для имени

string getName() const { return name; }

// Модификатор для имени

void setName(string name) { this->name = name; }

// Селектор для возраста

int getAge() const { return age; }

// Модификатор для возраста

void setAge(int age) { this->age = age; }

};

// Производный класс EMPLOYEE

class Employee : public Person {

private:

string position;

double salary;

public:

// Конструктор по умолчанию

Employee() {}

// Конструктор с параметрами

Employee(string name, int age, string position, double salary) : Person(name, age), position(position), salary(salary) {}

// Конструктор копирования

Employee(const Employee& other) : Person(other), position(other.position), salary(other.salary) {}

// Деструктор

~Employee() {}

// Селектор для должности

string getPosition() const { return position; }

// Модификатор для должности

void setPosition(string position) { this->position = position; }

// Селектор для оклада

double getSalary() const { return salary; }

// Модификатор для оклада

void setSalary(double salary) { this->salary = salary; }

// Метод для расчета зарплаты с учетом премии

double calculateSalary(double bonusPercentage) {

return salary + (salary \* bonusPercentage / 100);

}

};

// Функция для создания объекта производного класса и возврата его как объекта базового класса

Employee\* createPerson() {

string name;

int age;

string position;

double salary;

cout << "Введите имя: ";

cin >> name;

cout << "Введите возраст: ";

cin >> age;

cout << "Введите должность: ";

cin >> position;

cout << "Введите оклад: ";

cin >> salary;

return new Employee(name, age, position, salary);

}

// Функция для вывода информации о производном классе с использованием объекта базового класса

void printPerson(Employee\* person) {

cout << "Имя: " << person->getName() << endl;

cout << "Возраст: " << person->getAge() << endl;

if (dynamic\_cast<Employee\*>(person)) {

Employee\* employee = dynamic\_cast<Employee\*>(person);

cout << "Должность: " << employee->getPosition() << endl;

cout << "Оклад: " << employee->getSalary() << endl;

}

}

// Программа, демонстрирующая работу перегруженных операций

int main() {

setlocale(LC\_ALL, "ru");

// Ввод процента премии

cout << "Введите процент премии: ";

double bonusPercentage;

cin >> bonusPercentage;

// Создание объекта производного класса

Employee\* employee = createPerson();

cout << endl << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

// Вывод объекта производного класса

printPerson(employee);

cout << endl << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

// Подсчет итоговой зарплаты

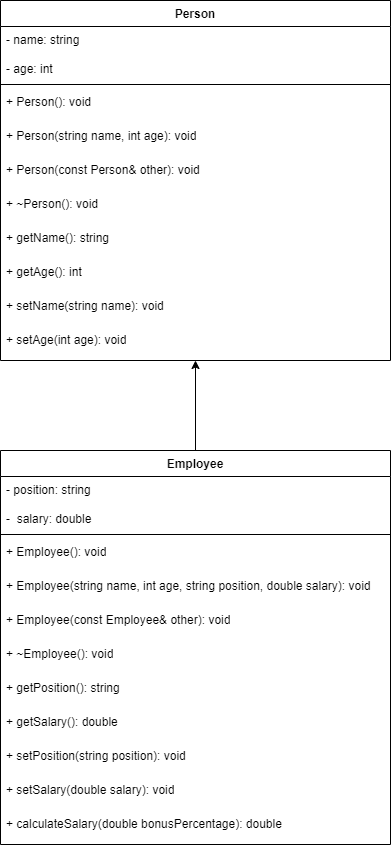
cout << "Зарплата сотрудника с учетом премии: " << employee->calculateSalary(bonusPercentage) << endl;

delete employee;

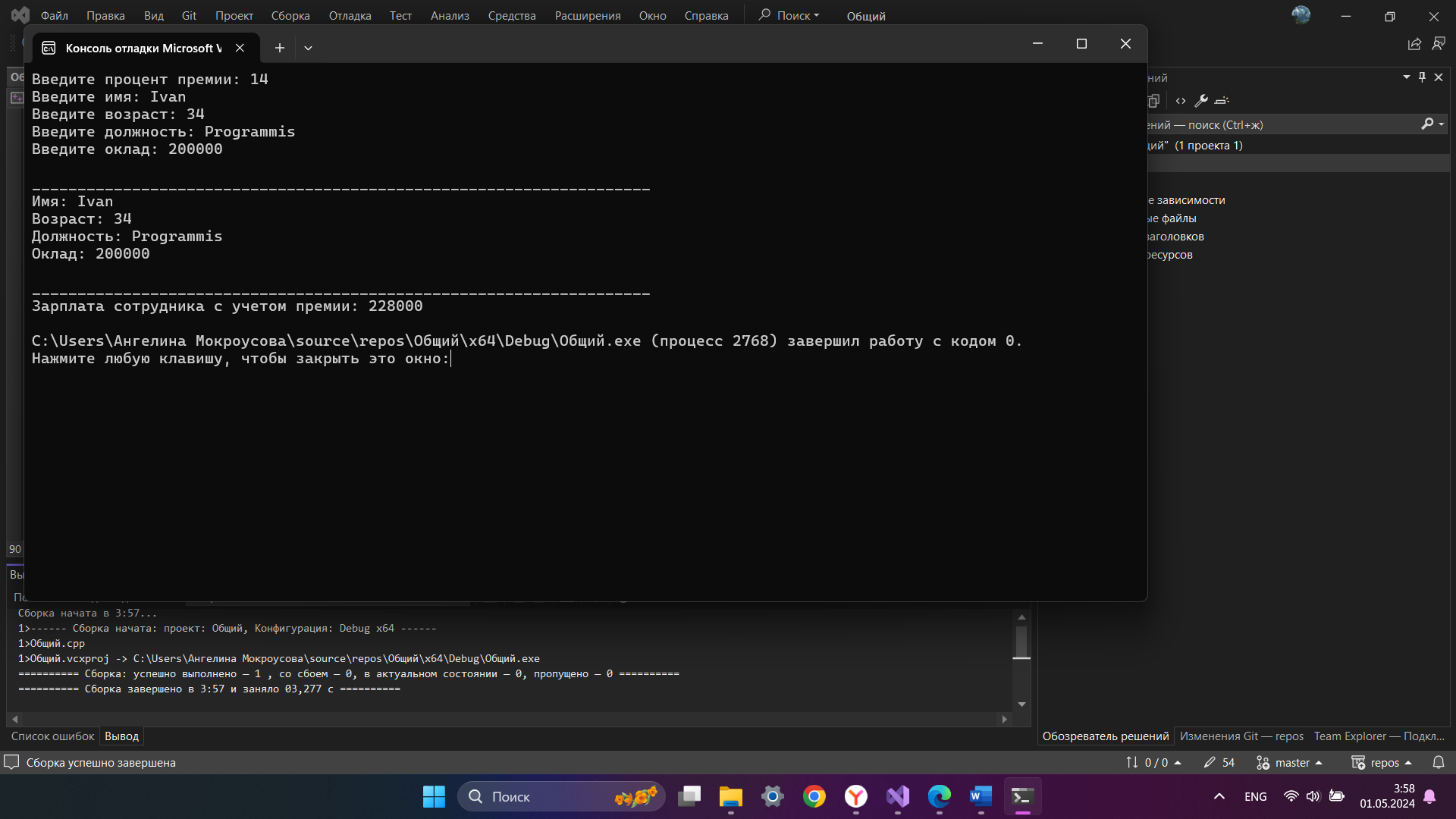
return 0;

}

**UML диаграмма:**

****

**Скриншот работы программы:**



**Вывод:**

Мною была аписана программа, которая успешно выполняет поставленную перед ней задачу. В процессе написания был повторен и закреплен на практике пройденный ранее материал.

**Ответы на вопросы:**

1. Для чего используется механизм наследования?

Механизм наследования используется в ситуациях. А) когда несколько классов имеют одинаковые поля. Таким образом можно избежать дублирования одних и тех же полей. Б) когда необходимо определить иерархию классов.

1. Каким образом наследуются компоненты класса, описанные со спецификатором public?

Если наследование публичное, то как публичные, если наследование protected, то как protected, если наследование private, то как private;

1. Каким образом наследуются компоненты класса, описанные со спецификатором private?

Private компоненты не наследуются;

1. Каким образом наследуются компоненты класса, описанные со спецификатором protected?

Если наследование публичное или protected, то остаются protected в произвольном классе. Если наследование private, то становятся private в произвольном классе;

1. Каким образом описывается производный класс?

Class <название производного класса> : <модификатор доступа> <имя класса - родителя> {<тело класса>};

1. Наследуются ли конструкторы? Нет;
2. Наследуются ли диструкторы? Нет;
3. В каком порядке конструируются объекты производных классов? Снача происходит конструирование всех базовых классов, потом главного (сначала у здания закладывается фундамент, а потом строится само здание);
4. В каком порядке уничтожаются объекты производных классов? Сначала уничтожается производный класс, потом его базовые классы (в большинстве своё фундамент здания сносят в последнюю очередь);
5. Что представляют собой виртуальные функции и механизм позднего связывания?

Виртуальные функции – функции, которые могут быть переопределены в производном классе. Механизм позднего связывания означает, что функция, которая будет вызвана, определяется на этапе самого выполнения, а не раньше;

1. Могут ли быть виртуальными конструкторы? Деструкторы? Конструкторы – нет, деструкторы – да;
2. Наследуется ли спецификатор virtual?

Да, если виртуальная функция не была переопределенная;

1. Какое отношение устанавливает между классами открытое наследование?

Произвольный класс имеет доступ ко всем полям и методам родителя, кроме приватных;

1. Какое отношение устанавливает между классами закрытое наследование?

Произвольный класс не имеет доступа к метода и полям родителя;

1. В чем заключается принцип подстановки?

Если наследование между наследниом и ребенком может быть публичным, то везде, где применяется класс – родитель, можно использовать класс – наследник. Обратное не верно;

1. Какие компонентные данные будет иметь объект х?

name, post и stage (причем частью интерфейса является только name);

**Скриншот с GitHab:**