МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

**«ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**Лабораторная работа № 5:**

"Наследование. Виртуальные функции. Полиморфизм."

Выполнила: студентка гр.РИС-23-3Б

Мокроусова Ангелина Андреевна

Проверила: доцент кафедры ИТАС

Полякова Ольга Андреевна

Пермь 2024

**Постановка задачи:**

Общая часть:

* Определить абстрактный класс
* Определить иерархию классов, в основе которой будет находиться абстрактный класс.(см. лабораторную работу №4)
* Определить в классе деструктор
* Определить класс Вектор, элементами которого будут указатели на объекты иерархии классов
* Перегрузить для класса Вектор операцию вывода объектов с помощью потоков
* В основной функции продемонстрировать перегруженные операции и полиморфизм Вектора

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:**

Object: Базовый абстрактный класс, определяющий базовые методы для вывода и заполнения объекта.

Person: Класс, представляющий человека с именем и возрастом.

Employee: Класс, представляющий сотрудника с именем, возрастом, должностью и зарплатой.

Класс Vector:

* Хранит массив указателей на объекты класса Object.
* Имеет методы для добавления объектов в вектор и заполнения его данными.
* Перегружает операторы вывода и ввода для вывода и заполнения содержимого вектора.

Основная программа:

* Запрашивает у пользователя размер вектора.
* Создает вектор заданного размера.
* Заполняет вектор объектами, позволяя пользователю выбирать типы объектов (Person или Employee).
* Выводит содержимое вектора на экран.

**Код на С++ с комментариями:**

// Подключаем необходимые библиотеки

#include <iostream>

#include <vector>

#include <string>

using namespace std;

// Базовый класс объектов

class Object

{

public:

// Чисто виртуальные методы, которые должны быть переопределены в производных классах

virtual void Show(ostream& stream) = 0;

virtual void Fill(istream& stream) = 0;

// Виртуальный деструктор

virtual ~Object() {}

};

// Класс Person наследуется от Object

class Person : public Object

{

protected:

// Защищенные члены класса

string Name;

int Age;

public:

// Конструкторы

Person() {}

Person(string name, int age) { setName(name); setAge(age); }

// Виртуальный деструктор

virtual ~Person() {}

// Переопределенные методы

void Show(ostream& stream) override

{

// Выводим информацию о человеке

stream << "Имя: " << Name << ", возраст: " << Age;

}

void Fill(istream& stream) override

{

// Вводим информацию о человеке

cout << "Имя: ";

stream >> Name;

cout << "Возраст: ";

stream >> Age;

}

// Геттеры и сеттеры

void setName(string name) { Name = name; }

void setAge(int age) { Age = age; }

string getName() { return Name; }

int getAge() { return Age; }

};

// Класс Employee наследуется от Person

class Employee : public Person

{

private:

// Приватные члены класса

string Position;

double Salary;

public:

// Конструкторы

Employee() {}

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

{

setPosition(position);

setSalary(salary);

}

// Виртуальный деструктор

virtual ~Employee() {}

// Геттеры и сеттеры

void setPosition(string position) { Position = position; }

void setSalary(double salary) { Salary = salary; }

// Метод для расчета зарплаты с учетом бонуса

double calculateSalary(double bonusPercentage)

{

return Salary \* (1 + bonusPercentage / 100);

}

// Переопределенные методы

void Show(ostream& stream) override

{

// Выводим информацию о сотруднике

stream << "Имя: " << getName() << ", возраст: " << getAge() << ", должность: " << Position << ", оклад: " << Salary;

}

void Fill(istream& stream) override

{

// Вводим информацию о сотруднике

Person::Fill(stream);

stream.ignore();

cout << "Должность: ";

getline(stream, Position);

cout << "Оклад: ";

stream >> Salary;

}

};

// Класс Vector для хранения объектов

class Vector

{

public:

// Конструктор

Vector(int length)

{

elements = new Object \* [length];

Length = length;

cur\_Length = 0;

}

// Деструктор

~Vector()

{

// Удаляем все объекты из вектора

for (int i = 0; i < cur\_Length; i++)

delete elements[i];

// Удаляем массив указателей на объекты

delete[] elements;

Length = 0;

cur\_Length = 0;

}

// Метод для добавления объекта в вектор

bool TryAdd(Object\* element)

{

// Проверяем, есть ли свободное место в векторе

if (cur\_Length >= Length)

return false;

// Добавляем объект в вектор

elements[cur\_Length] = element;

cur\_Length++;

return true;

}

// Перегрузка оператора вывода для вывода содержимого вектора

friend ostream& operator<<(ostream& stream, Vector& vector);

// Перегрузка оператора ввода для заполнения вектора объектами

friend istream& operator>>(istream& stream, Vector& vector);

private:

// Массив указателей на объекты

Object\*\* elements;

// Размер вектора

int Length;

// Текущая длина вектора

int cur\_Length;

};

// Перегрузка оператора вывода для вывода содержимого вектора

ostream& operator<<(ostream& stream, Vector& vector)

{

// Выводим заголовок

stream << "Содержимое вектора:" << endl;

// Перебираем объекты в векторе

for (int i = 0; i < vector.cur\_Length; i++)

{

// Выводим информацию об объекте

vector.elements[i]->Show(stream);

stream << endl;

}

return stream;

}

// Перегрузка оператора ввода для заполнения вектора объектами

istream& operator>>(istream& stream, Vector& vector)

{

// Вводим количество объектов в векторе

int choice;

for (int i = 0; i < vector.Length; i++)

{

// Вводим тип объекта (Person или Employee)

stream.ignore();

cout << "Выберите тип объекта (1 - Person, 2 - Employee): ";

stream >> choice;

// Создаем объект в зависимости от выбранного типа

Object\* obj = nullptr;

if (choice == 1)

{

obj = new Person;

}

else if (choice == 2)

{

obj = new Employee;

}

// Если объект создан, заполняем его данными и добавляем в вектор

if (obj != nullptr)

{

obj->Fill(stream);

vector.TryAdd(obj);

}

// Выводим разделитель

cout << "----------------------" << endl;

}

return stream;

}

int main()

{

// Устанавливаем локаль для вывода на русском языке

setlocale(LC\_ALL, "Russian");

// Вводим размер вектора

int Length;

cout << "Введите размер вектора: ";

cin >> Length;

// Создаем вектор

Vector vector(Length);

// Заполняем вектор объектами

cin >> vector;

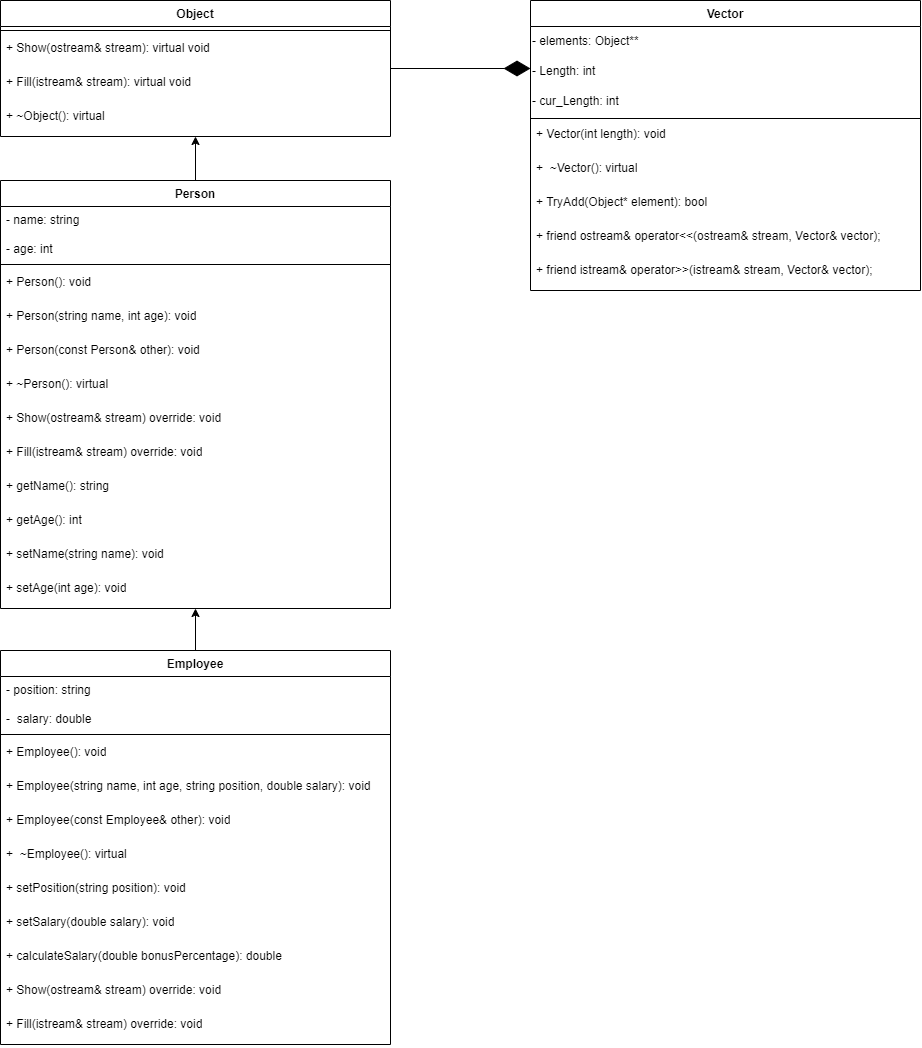
// Выводим содержимое вектора

cout << vector << endl;

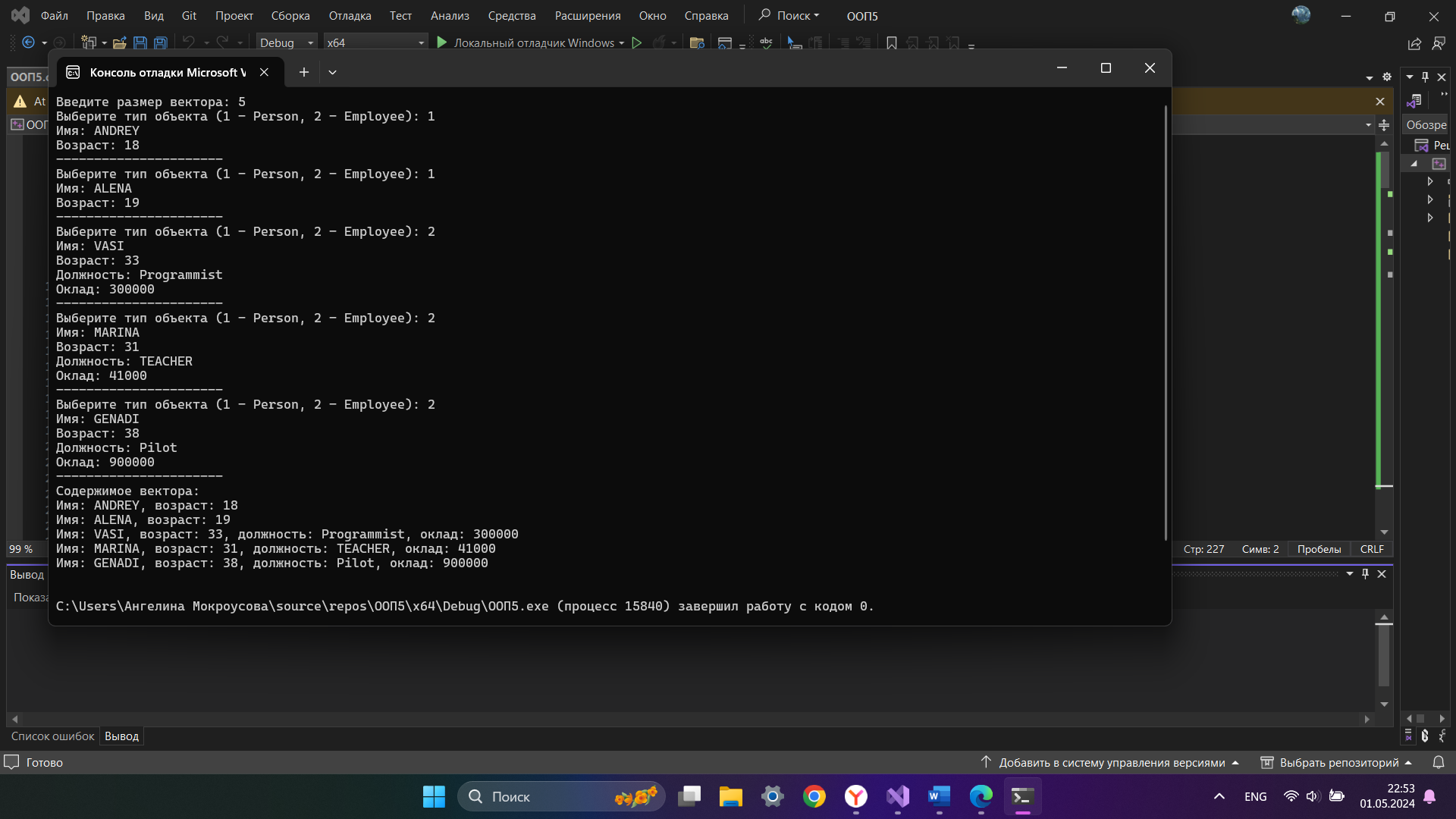
return 0;

}

**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. Какой метод называется чисто виртуальным? Чем он отличается от виртуального метода?

Чисто виртуальным методом называют метод, который вместо тела имеет нуль. В отличии от виртуального метода он не может быть вызван в явном виде без переопределения в произвольном классе (попытка приведет к ошибке). Иными словами, если виртуальный метод может быть переопределенн, то чисто виртуальный метод должен быть переопределен. В противном случае произвольный класс также будет абстрактным;

1. Какой класс называется абстрактным?

Класс является абстрактным, если в нем есть хотя бы одна чисто виртуальная функция;

1. Для чего предназначены абстрактные классы?

Абстрактные классы предназначены для определения общих понятий, которые будут конкретезированы в его потомках;

1. Что такое полиморфные функции?

Полиморфные функции – функции, которые могут работать со всеми объектами, чьи классы определенны в одной иерархии;

1. Чем полиморфизм отличается от принципа подстановки?

Принцип подстановки – принцип, согласно которому на место родителя может быть подставлен потомок. Полиморфизм же – это механизм, который позволяет программе в ходе работы самой определить, какую реализацию функции стоит использовать в данный момент;

1. Привести примеры иерархий с использованием абстрактных классов.

Абстрактный класс – триугольник с чисто виртуальным методом нахождения площади. Потомки: равнобедренный треугольник, прямоугольный треугольник, равносторонний и т.д

1. Привести примеры полиморфных функций.

В каждом переопределен метод нахождения площади в зависимости от правил геометрии (из прошлого примера)

1. В каких случаях используется механизм позднего связывания? Механизм позднего связывания используется при работа с виртуальными функциями, если вызов идет через указатель на базовый абстрактный класс.

**Скриншот с GitHab:**