# ЛАБОРАТОРНАЯ РАБОТА № 24

# ИСПОЛЬЗОВАНИЕ РАЗЛИЧНЫХ СВОЙСТВ НАСЛЕДОВАНИЯ ДЛЯ РАСШИРЕНИЯ И УТОЧНЕНИЯ КЛАССОВ

Цель работы:

Применить на практике знания:

* по созданию производных классов, образованных от одного базового;
* по реализации конструкторов и методов производных классов;
* по вызову методов базовых классов.

## КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Наследование является одной из главных особенностей ООП. Механизм наследования классов позволяет строить иерархии, в которых производные классы получают элементы родительских, или базовых, классов и могут дополнять их или изменять их свойства.

### Простое наследование

Простым называется наследование, при котором производный класс имеет одного родителя. Для различных методов класса существуют разные правила наследования — например, конструкторы и операция присваивания в производном классе не наследуются, а деструкторы наследуются. При описании класса в его заголовке перечисляются все классы, являющиеся для него базовыми. Возможность обращения к элементам этих классов регулируется с помощью ключей доступа

private protected public, и:

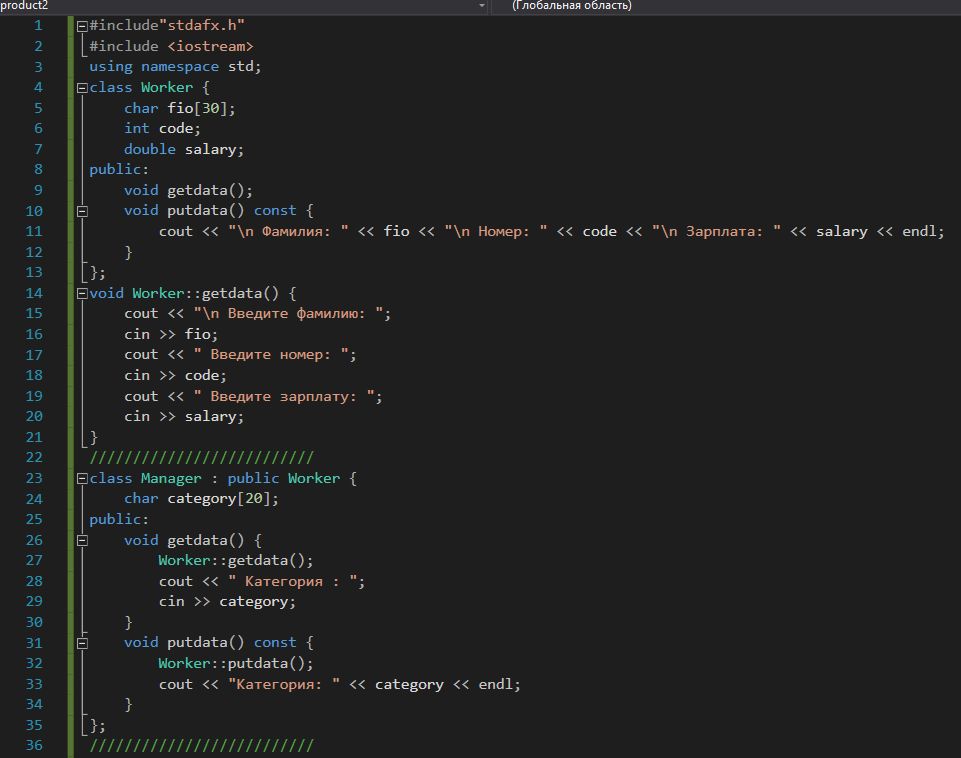
class имя: [private | protected | public] базовый\_класс {тело класса};

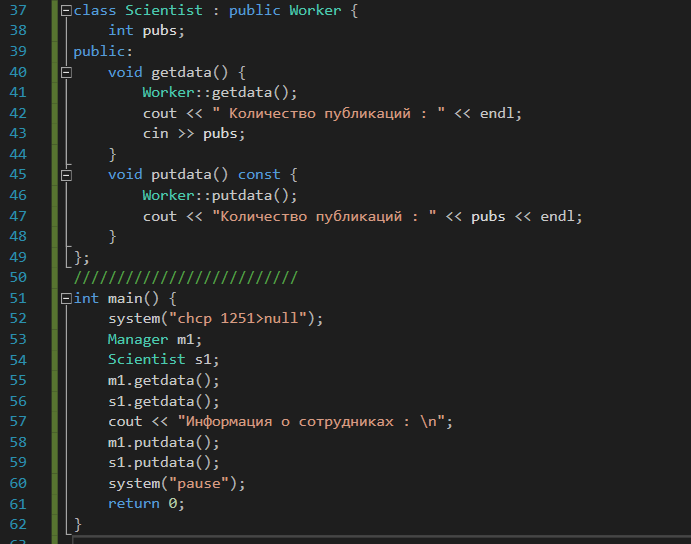
По умолчанию для классов используется ключ доступа private, а для структур - public.

## ПРИМЕРЫ

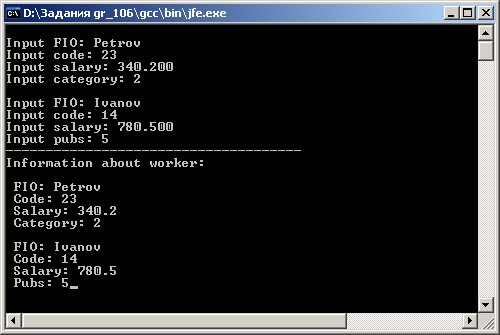
Пример. Простое наследование

Написать программу, в которой представлен класс Worker, являющийся базовым для двух классов: Manager и Scientist.





Результат работы программы представлен на рисунке 1:



*Рисунок 1*

## ЗАДАНИЯ

Для выполнения задания внесите изменения в предыдущую Лабораторную работу №22 и 23. Варианты заданий остаются прежними.

### Вариант 1

Задание: Простое наследование.

Разработайте класс Bilet– производный от класса TRAIN. Элементы класса:

* количество вагонов;
* стоимость билета.

Методы:

* конструктор без параметров (по умолчанию);
* конструктор с параметрами;
* конструктор копирования;
* деструктор;
* метод, осуществляющий ввод значений полей класса с клавиатуры;
* метод, осуществляющий вывод значений полей класса на экран.

В определение конструкторов и деструктора поместите идентифицирующую их работу информацию.

Например:

cout << “Вызван конструктор без параметров”;

### Вариант 2

Задание: Простое наследование.

Разработайте класс NOTE1 – производный от класса NOTE. Элементы класса:

* электронная почта.

Методы:

* конструктор без параметров (по умолчанию);
* конструктор с параметрами;
* конструктор копирования;
* деструктор;
* метод, осуществляющий ввод значений полей класса с клавиатуры;
* метод, осуществляющий вывод значений полей класса на экран.

В определение конструкторов и деструктора поместите идентифицирующую их работу информацию. Например:

cout << “Вызван конструктор без параметров”;

### Вариант 3

Задание: Простое наследование.

Разработайте класс REIS – производный от класса AEROFLOT. Элементы класса:

* расстояние до пункта назначения;
* стоимость билета.

Методы:

* конструктор без параметров (по умолчанию);
* конструктор с параметрами;
* конструктор копирования;
* деструктор;
* метод, осуществляющий ввод значений полей класса с клавиатуры;
* метод, осуществляющий вывод значений полей класса на экран.

В определение конструкторов и деструктора поместите идентифицирующую их работу информацию.

Например:

cout << “Вызван конструктор без параметров”;

### Вариант 4

Задание: Простое наследование.

Разработайте класс WORKER1 – производный от класса WORKER. Элемент класса:

* нагрузка.

Методы:

* конструктор без параметров (по умолчанию);
* конструктор с параметрами;
* конструктор копирования;
* деструктор;
* метод, осуществляющий ввод значений полей класса с клавиатуры;
* метод, осуществляющий вывод значений полей класса на экран.

В определение конструкторов и деструктора поместите идентифицирующую их работу информацию.

Например:

cout <<“Вызван конструктор без параметров”;

### Вариант 5

Задание: Простое наследование.

Разработайте класс NOTE1 – производный от класса NOTE. Элементы класса:

* с кем встреча.

Методы:

* конструктор без параметров (по умолчанию);
* конструктор с параметрами;
* конструктор копирования;
* деструктор;
* метод, осуществляющий ввод значений полей класса с клавиатуры;
* метод, осуществляющий вывод значений полей класса на экран.

В определение конструкторов и деструктора поместите идентифицирующую их работу информацию.

Например:

cout << “Вызван конструктор без параметров”;

### Вариант 6

Задание: Простое наследование.

Разработайте класс PRICE1 – производный от класса PRICE. Элементы класса:

* количеств оставшихся единиц товара.

Методы:

* конструктор без параметров (по умолчанию);
* конструктор с параметрами;
* конструктор копирования;
* деструктор;
* метод, осуществляющий ввод значений полей класса с клавиатуры;
* метод, осуществляющий вывод значений полей класса на экран.

В определение конструкторов и деструктора поместите идентифицирующую их работу информацию.

Например:

cout << “Вызван конструктор без параметров”;

### Вариант 7

Задание: Простое наследование.

Разработайте класс AVTO1 – производный от класса AVTO. Элементы класса:

* год выпуска.

Методы:

* конструктор без параметров (по умолчанию);
* конструктор с параметрами;
* конструктор копирования;
* деструктор;
* метод, осуществляющий ввод значений полей класса с клавиатуры;
* метод, осуществляющий вывод значений полей класса на экран.

В определение конструкторов и деструктора поместите идентифицирующую их работу информацию.

Например:

cout << “Вызван конструктор без параметров”;

### Вариант 8

Задание: Простое наследование.

Разработайте класс ZNAK1 – производный от класса ZNAK. Элементы класса:

* телефон.

Методы:

* конструктор без параметров (по умолчанию);
* конструктор с параметрами;
* конструктор копирования;
* деструктор;
* метод, осуществляющий ввод значений полей класса с клавиатуры;
* метод, осуществляющий вывод значений полей класса на экран.

В определение конструкторов и деструктора поместите идентифицирующую их работу информацию.

Например:

cout << “Вызван конструктор без параметров”;

### Вариант 9

Задание: Простое наследование.

Разработайте класс BILET– производный от класса MARSH.

* количество мест;
* стоимость билета.

Методы:

* конструктор без параметров (по умолчанию);
* конструктор с параметрами;
* конструктор копирования;
* деструктор;
* метод, осуществляющий ввод значений полей класса с клавиатуры;
* метод, осуществляющий вывод значений полей класса на экран.

В определение конструкторов и деструктора поместите идентифицирующую их работу информацию.

Например:

cout << “Вызван конструктор без параметров”;

### Вариант 10

Задание: Простое наследование.

Разработайте класс СLIENT – производный от класса PERSON. Элементы класса:

* количество остановок в отеле.

Методы:

* конструктор без параметров (по умолчанию);
* конструктор с параметрами;
* конструктор копирования;
* деструктор;
* метод, осуществляющий ввод значений полей класса с клавиатуры;
* метод, осуществляющий вывод значений полей класса на экран.

В определение конструкторов и деструктора поместите идентифицирующую их работу информацию.

Например:

cout << “Вызван конструктор без параметров”;

### Вариант 11

Задание: Простое наследование.

Разработайте класс TOVAR1 – производный от класса TOVAR. Элементы класса:

* цена при реализации;
* остаток товара.

Методы:

* конструктор без параметров (по умолчанию);
* конструктор с параметрами;
* конструктор копирования;
* деструктор;
* метод, осуществляющий ввод значений полей класса с клавиатуры;
* метод, осуществляющий вывод значений полей класса на экран.

В определение конструкторов и деструктора поместите идентифицирующую их работу информацию. Например:

cout << “Вызван конструктор без параметров”;

### Вариант 12

Задание: Простое наследование.

Разработайте класс DATA – производный от класса BOOK.

* дата, когда выдана книга.

Методы:

* конструктор без параметров (по умолчанию);
* конструктор с параметрами;
* конструктор копирования;
* деструктор;
* метод, осуществляющий ввод значений полей класса с клавиатуры;
* метод, осуществляющий вывод значений полей класса на экран.

В определение конструкторов и деструктора поместите идентифицирующую их работу информацию.

Например:

cout << “Вызван конструктор без параметров”;

### Вариант 13

Задание: Простое наследование.

Разработайте класс ABITURIENT1 – производный от класса ABITURIENT. Элементы класса:

* средний балл.

Методы:

* конструктор без параметров (по умолчанию);
* конструктор с параметрами;
* конструктор копирования;
* деструктор;
* метод, осуществляющий ввод значений полей класса с клавиатуры;
* метод, осуществляющий вывод значений полей класса на экран.

В определение конструкторов и деструктора поместите идентифицирующую их работу информацию.

Например:

cout << “Вызван конструктор без параметров”;

### Вариант 14

Задание: Простое наследование.

Разработайте класс ORDER1 – производный от класса ORDER. Элементы класса:

* банк-отправитель.

Методы:

* конструктор без параметров (по умолчанию);
* конструктор с параметрами;
* конструктор копирования;
* деструктор;
* метод, осуществляющий ввод значений полей класса с клавиатуры;
* метод, осуществляющий вывод значений полей класса на экран.

В определение конструкторов и деструктора поместите идентифицирующую их работу информацию.

Например:

cout << “Вызван конструктор без параметров”;

### Вариант 15

Задание: Простое наследование. Разработайте класс SCHOOL– производный от класса CITY. Элементы класса:

* количество выпускников.

Методы:

* конструктор без параметров (по умолчанию);
* конструктор с параметрами;
* конструктор копирования;
* деструктор;
* метод, осуществляющий ввод значений полей класса с клавиатуры;
* метод, осуществляющий вывод значений полей класса на экран.

В определение конструкторов и деструктора поместите идентифицирующую их работу информацию.

Например:

cout << “Вызван конструктор без параметров”;

## КОНТРОЛНЫЕ ВОПРОСЫ

1. Перечислите принципы ООП, поясните суть принципа простого наследования.
2. Поясните, чем отличаются различные типы наследования: public, protected, private.
3. Запишите синтаксис описания производного класса.
4. Приведите порядок вызова конструкторов и деструкторов при наследовании.
5. Поясните понятие переопределения функции.