Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра экономической математики, информатики и статистики (ЭМИС)

дружественные функции. наследование классов. множественное наследование

Отчет по лабораторной работе по дисциплине «Объектно-ориентированное программирование»

Студент группы 549  Баулин С.К.  
«\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2020 г.  
Кандидат физико-математических наук, доцент кафедры ЭМИС \_\_\_\_\_\_\_ Шельмина Е. А. «\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2020 г.

\_\_\_\_\_\_

оценка

Томск 2020

**Лабораторная работа №7**

**Дружественные функции. Наследование классов. Множественное наследование.**

Цель работы: освоить и применить на практике дружественные функции, наследование классов, множественное наследование.

Теоретические сведения

Дружественной функцией класса называется функция, которая, не являясь его компонентом, имеет доступ к его защищённым и собственным компонентам. Для реализации прав друга функция должна быть описана в теле класса со спецификатором friend («друг»).

Функция frnd\_put() описана в классе charlocus как дружественная функция и определена обычным образом как глобальная функция (вне класса, без ука­зания его имени, без операции :: и без спецификатора friend). Как дружес­твенная, она получает доступ к защищённым (protected) дан­ным класса и изменяет значение символа того объекта, адрес которого будет передан ей как значение первого параметра.

Функция может быть дружественной по отношению к нескольким классам:

* class CL2;
* class CL1 friend void ff(CL1,CL2); ...
* class CL2 friend void ff(CL1,CL2); ...
* void ff(...) тело функции.

Наследование

Наследование - это механизм получения нового класса на основе уже существующего. Существующий класс может быть дополнен или изменен для создания нового класса.

Существующие классы называются базовыми, а новые – производными. Производный класс наследует описание базового класса; затем он может быть изменен добавлением новых членов, изменением существующих функций-членов и изменением прав доступа. С помощью наследования может быть создана иерархия классов, которые совместно используют код и интерфейсы.

Наследуемые компоненты не перемещаются в производный класс, а остаются в базовых классах.

В иерархии производный объект наследует разрешенные для наследования компоненты всех базовых объектов (public, protected).

Допускается множественное наследование – возможность для некоторого класса наследовать компоненты нескольких никак не связанных между собой базовых классов. В иерархии классов соглашение относительно доступности компонентов класса следующее:

private – член класса может использоваться только функциями – членами данного класса и функциями – “друзьями” своего класса. В производном классе он недоступен.

protected – то же, что и private, но дополнительно член класса с данным атрибутом доступа может использоваться функциями-членами и функциями – “друзьями” классов, производных от данного.

public – член класса может использоваться любой функцией, которая является членом данного или производного класса, а также к public - членам возможен доступ извне через имя объекта.

Следует иметь в виду, что объявление friend не является атрибутом доступа и не наследуется.

Синтаксис определения производного класса:

class имя\_класса : список\_базовых\_классов

{список\_компонентов\_класса};

В производном классе унаследованные компоненты получают статус доступа private, если новый класс определен с помощью ключевого слова class, и статус public, если с помощью struct.

Явно изменить умалчиваемый статус доступа при наследовании можно с помощью атрибутов доступа – private, protected и public, которые указываются непосредственно перед именами базовых классов.

Конструкторы и деструкторы производных классов

Поскольку конструкторы не наследуются, при создании производного класса наследуемые им данные-члены должны инициализироваться конструктором базового класса. Конструктор базового класса вызывается автоматически и выполняется до конструктора производного класса. Параметры конструктора базового класса указываются в определении конструктора производного класса. Таким образом происходит передача аргументов от конструктора производного класса конструктору базового класса.

Например.

class Basis

{ int a,b;

public:

Basis(int x,int y){a=x;b=y;}

};

class Inherit:public Basis

{int sum;

public:

Inherit(int x,int y, int s):Basis(x,y){sum=s;}

};

Объекты класса конструируются снизу вверх: сначала базовый, потом компоненты-объекты (если они имеются), а потом сам производный класс. Таким образом, объект производного класса содержит в качестве подобъекта объект базового класса.

Уничтожаются объекты в обратном порядке: сначала производный, потом его компоненты-объекты, а потом базовый объект.

Таким образом, порядок уничтожения объекта противоположен по отношению к порядку его конструирования.

Задания для самостоятельной работы.

Задание 1. Реализовать дружественные функции для работы с объектами классов.

Даны два массива (классы множества чисел). Написать дружественную функцию, сортирующую эти массивы по возрастанию. Скриншоты с кодом программы и результатом приведены на рисунках 1.1 – 1.4.

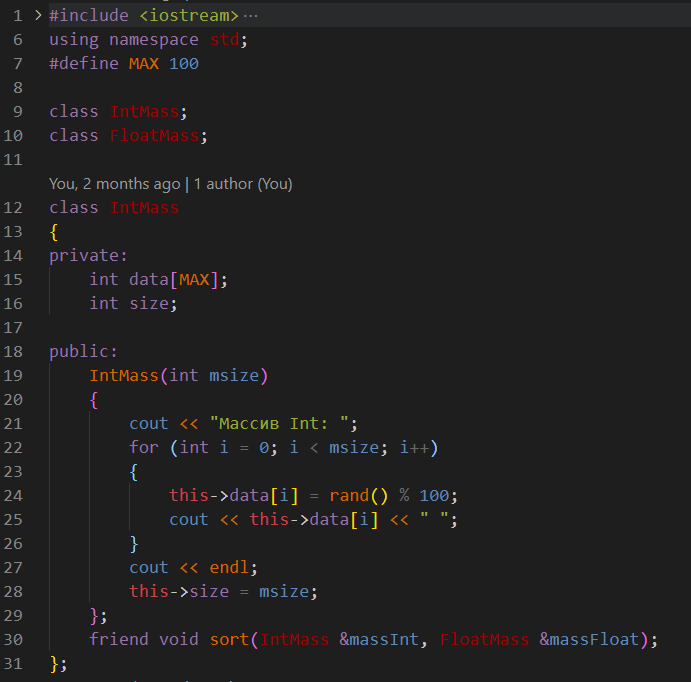
Задание 2. Наследование. Требуется создать базовый класс и определить общие и специфические методы для данного класса. Создать производные классы, в которые добавить свойства и методы. Часть методов переопределить. Создать массив объектов базового класса и заполнить объектами производных классов. Предусмотреть передачу аргументов конструкторам базового класса, использовать виртуальные и перегруженные функции.

Создать базовый класс «Грузоперевозчик» и производные классы «Самолет», «Поезд», «Автомобиль». Определить время и стоимость перевозки для указанных городов и расстояний.

Скриншоты с кодом программы и результатом приведены на рисунках 2.1 – 2.4.

Задание 3. Множественное наследование. Необходимо построить иерархию классов согласно схеме наследования, приведенной в варианте задания. Каждый класс должен содержать инициализирующий конструктор и функцию show для вывода значений. Функция main должна иллюстрировать иерархию наследования.

Скриншоты с заданием, кодом программы и результатом приведены на рисунках 3.1 – 3.5.

Рисунок 1.1 – Скриншот кода программы

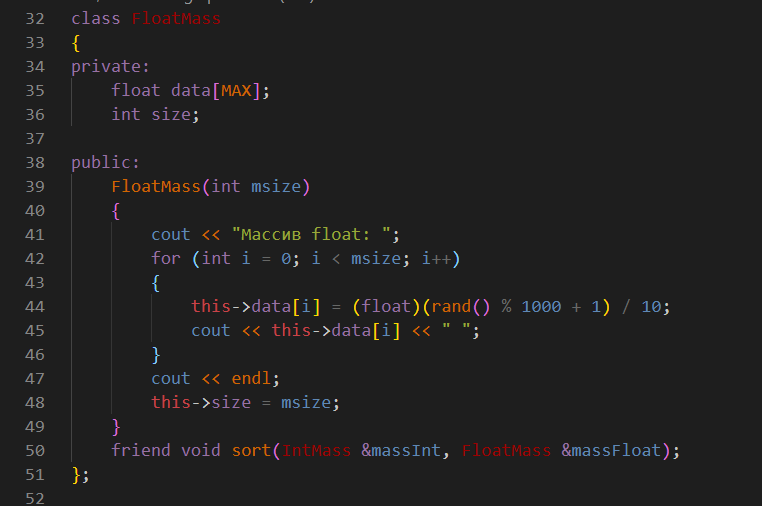
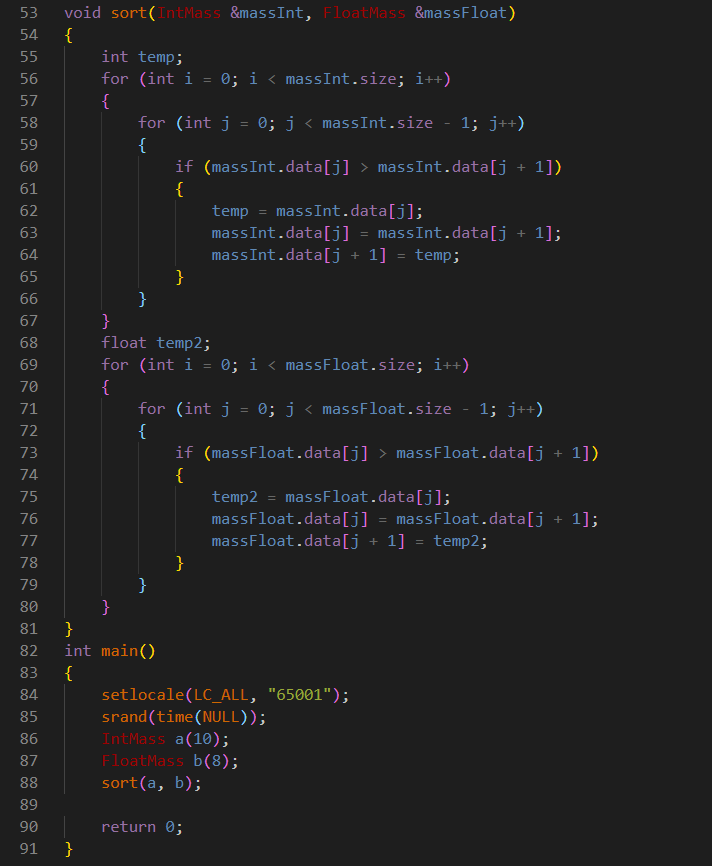
Рисунок 1.2 – скриншот кода программы

Рисунок 1.3 – Скриншот кода программы

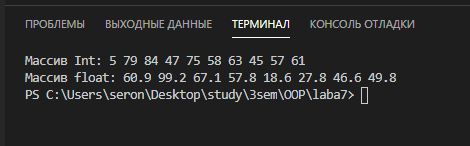
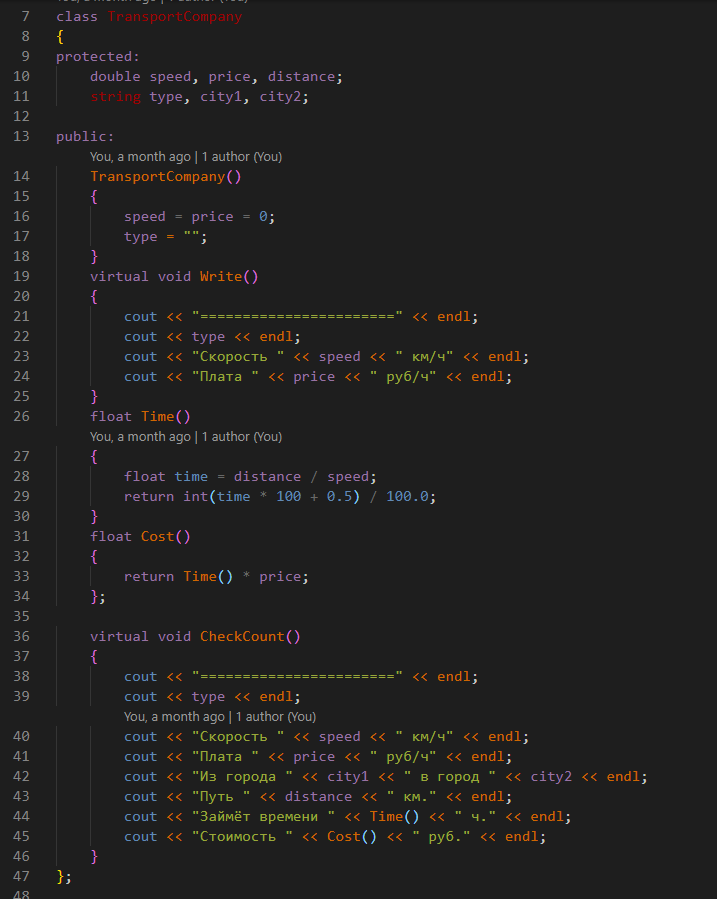
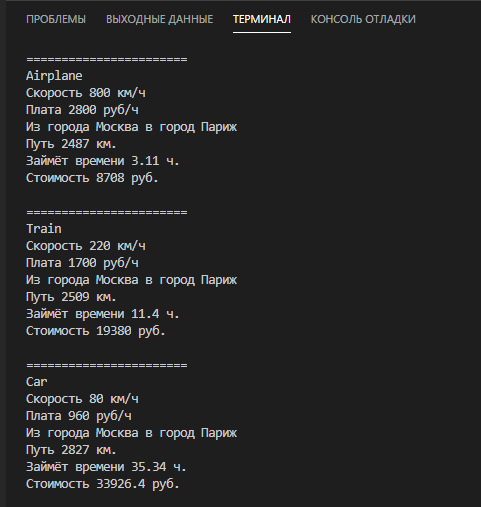
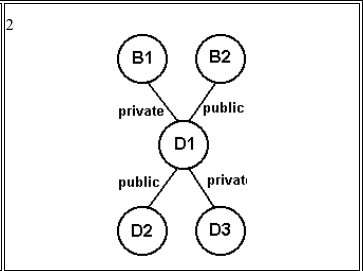
Рисунок 1.4 – Скриншот результата работы программы

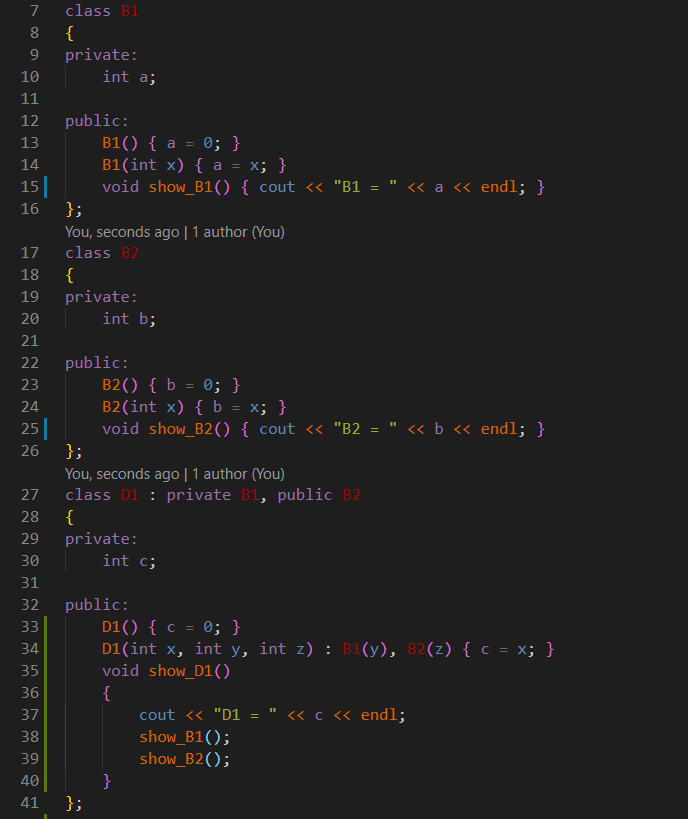
Рисунок 2.1 – Скриншот кода программы

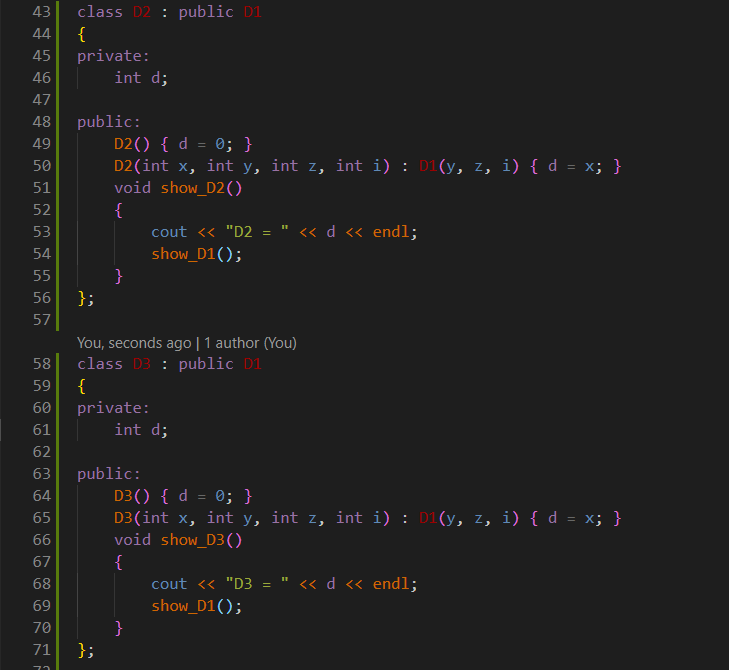
Рисунок 2.2 – Скриншот кода программы

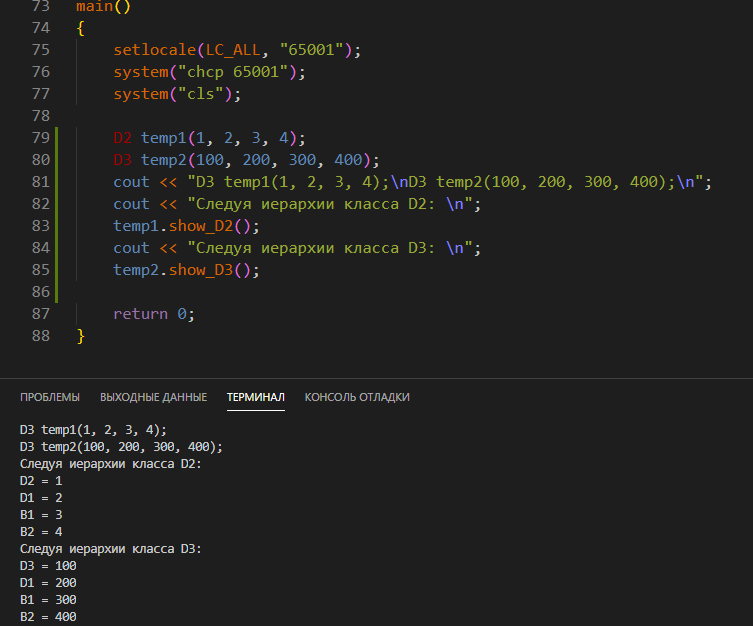
Рисунок 2.3 – Скриншот кода программы

Рисунок 2.4 – Скриншот результата работы программы

Рисунок 3.1 – Скриншот задания 3

Рисунок 3.2 – Скриншот кода программы

Рисунок 3.3 – Скриншот кода программы

Рисунок 3.4 – Скриншот результата работы программы

Вывод: освоены и применены на практике дружественные функции, наследование классов, множественное наследование.