Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«Пермский национальный исследовательский политехнический университет» (ПНИПУ)

Электротехнический факультет

Кафедра информационных технологий и автоматизированных систем

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

**Лабораторная работа по классам № 4**

**по дисциплине**

**«Основы алгоритмизации и программирования»**

**семестр** 2

Выполнил студент гр. ИВТ-21-1б

Балдин Алексей Игоревич

Проверил:

Яруллин Денис Владимирович

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

(оценка) (подпись)

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

(дата)

г. Пермь - 2022

**Цель**

Создание иерархии классов с использованием простого наследования.

**Постановка задачи**

* + 1. Определить пользовательский класс.
    2. Определить в классе следующие конструкторы: без параметров, с параметрами, копирования. Определить в классе деструктор
    3. Определить в классе компоненты-функции для просмотра и установки полей данных (селекторы и модификаторы).
    4. Перегрузить операцию присваивания.
    5. Перегрузить операции ввода и вывода объектов с помощью потоков.
    6. Определить производный класс.
    7. Написать программу, в которой продемонстрировать создание объектов и работу всех перегруженных операций.
    8. Реализовать функции, получающие и возвращающие объект базового класса. Продемонстрировать принцип подстановки.

**Исходные данные**

Вариант 5:

Базовый класс:

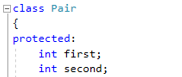
ПАРА\_ЧИСЕЛ (PAIR)

Первое число (first) - int Второе число (second) – int

Определить методы изменения полей и операцию сложения пар (a, b)+(c,d)=(a+b,c+d) Создать производный класс ДЕНЕЖНАЯ\_СУММА(MONEY), с полями Рубли и Копейки. Переопределить операцию сложения и определить операции вычитания и деления денежных сумм.

Сначала создаём тип данных «class» под именем Pair. Поля классов first и second c типом данных (int), находятся в private. В модификаторе доступа – public, создаем конструкторы по умолчанию, c параметрами, деструктор, также перегружаю оператор (+), который как раз-таки складывает поля first и second, еще методы (геттеры и сеттеры) – т.е. установка значений в private поля. Тут же в public прописываем дружественность операторов (>>) и (<<).

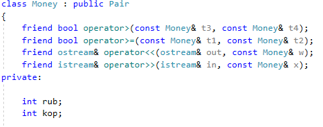
Здесь, создаю класс Pair – в нем наши поля под модификатором доступа – protected, это значит эти поля могут наследоваться в другие классы:



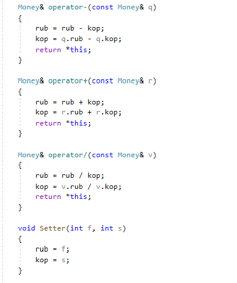
Далее в коде создаются конструкторы, деструкторы. Тут создаю дружественные прототипы перегруженных операторов ввода и вывода:



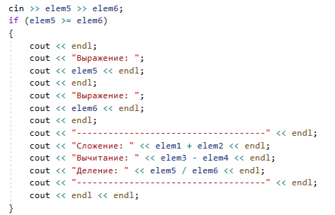
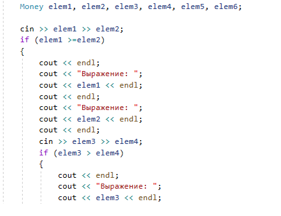
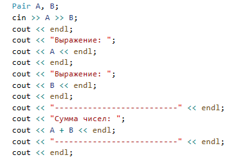
Далее, создаю еще один класс, который будет наследовать класс Pair, а это значит можно будет пользоваться полями класса Pair в классе Money. Также создаю прототипы дружественных операторов ввода и вывода, и прототипы перегруженных операторов «больше» и «больше равно». Также присутствуют поля класса в private секции : rub, kop.



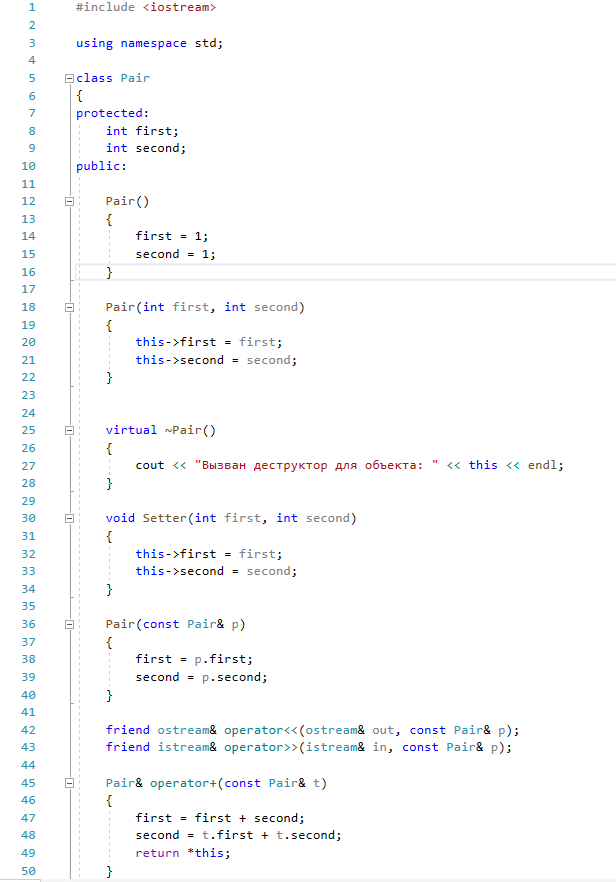
Перегружаю оператор (-), оператор (+), деления, и также присутствует метод (Setter) – для установки значения полей private.

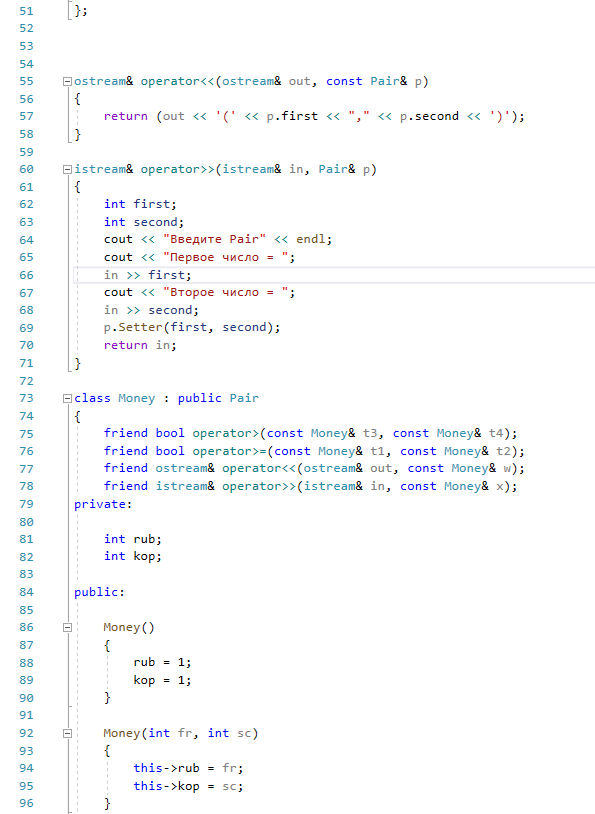


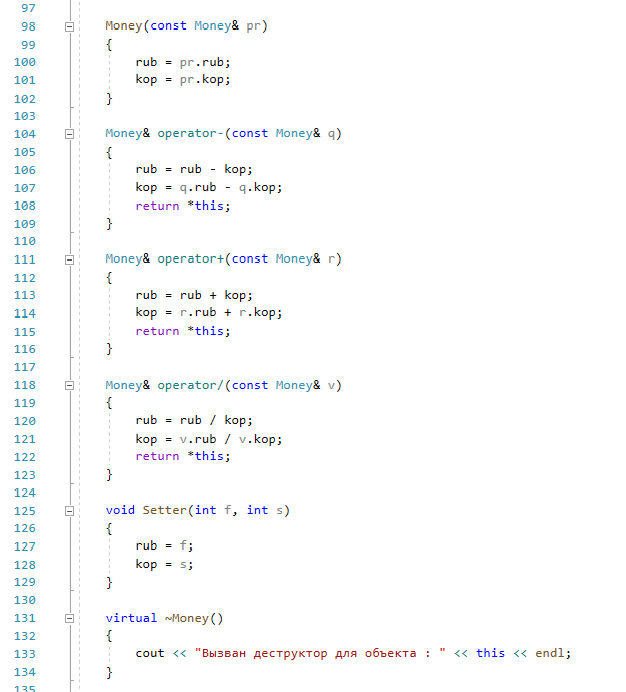
В main() создаю поля типа Pair, вывожу поп условию.

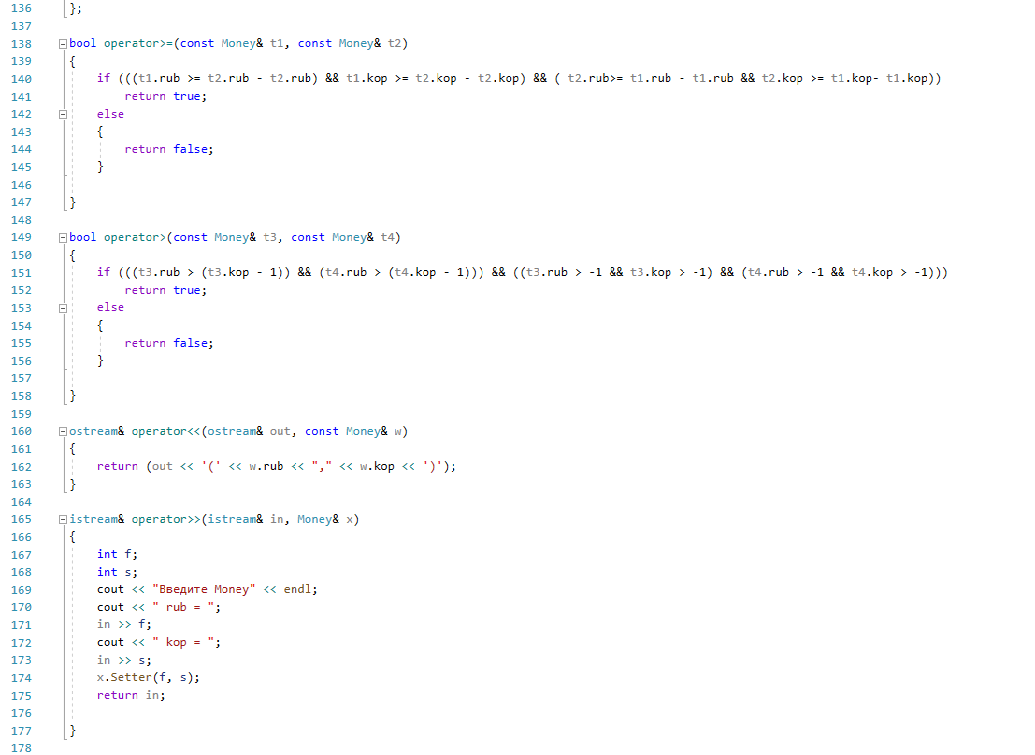


Код программы:

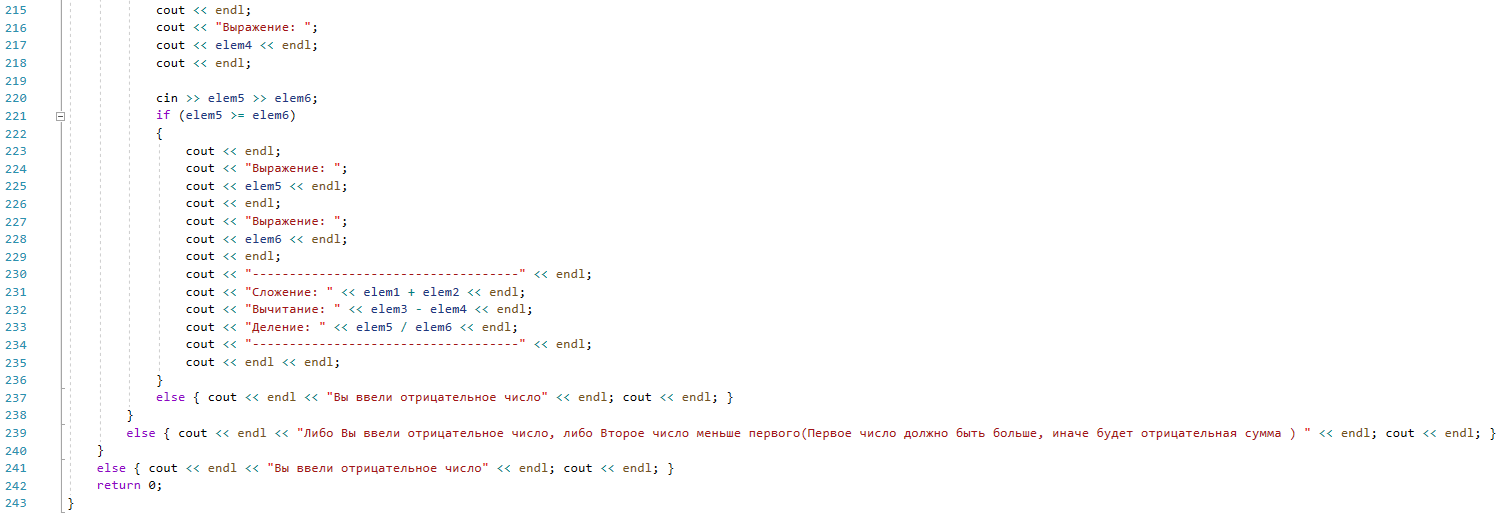












Результат программы:

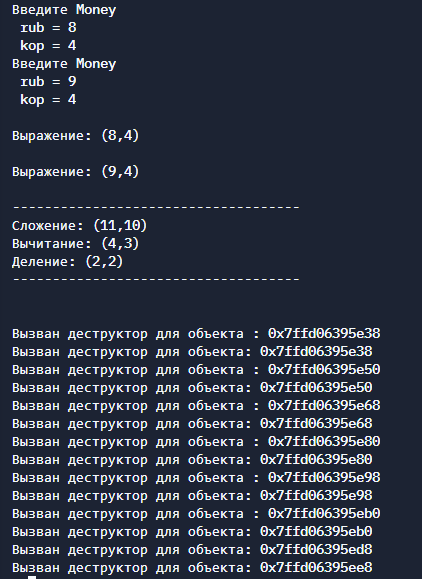
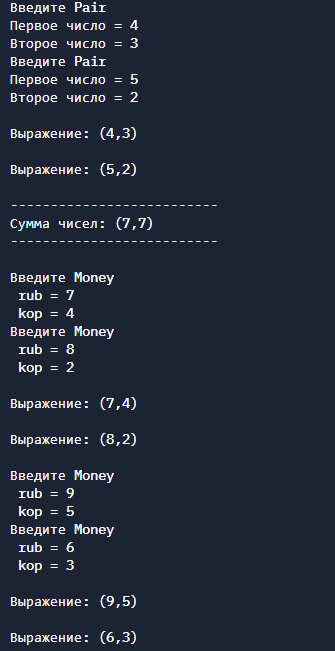


Диаграмма к задаче:

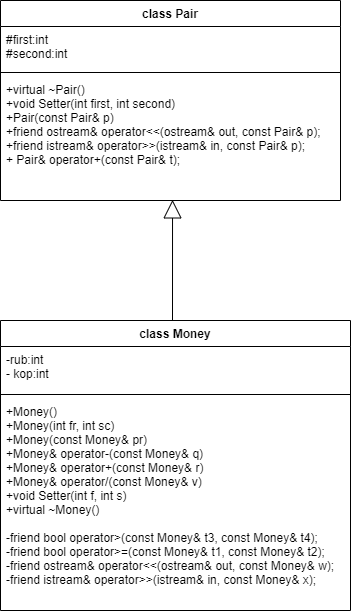


Рисунок 1

**Контрольные вопросы:**

* 1. Для чего используется механизм наследования?
  2. Каким образом наследуются компоненты класса, описанные со спецификатором public?
  3. Каким образом наследуются компоненты класса, описанные со спецификатором private?
  4. Каким образом наследуются компоненты класса, описанные со спецификатором protected?
  5. Каким образом описывается производный класс?
  6. Наследуются ли конструкторы?
  7. Наследуются ли деструкторы?
  8. В каком порядке конструируются объекты производных классов?
  9. В каком порядке уничтожаются объекты производных классов?
  10. Что представляют собой виртуальные функции и механизм позднего связывания?
  11. Могут ли быть виртуальными конструкторы? Деструкторы?
  12. Наследуется ли спецификатор virtual?
  13. Какое отношение устанавливает между классами открытое наследование?
  14. Какое отношение устанавливает между классами закрытое наследование?
  15. В чем заключается принцип подстановки?
  16. Имеется иерархия классов:

**Ответы на вопросы:**

1. Наследование — это механизм создания нового класса на основе уже существующего. При этом к существующему классу могут быть добавлены новые элементы (данные и функции), либо существующие функции могут быть изменены. Основное назначение механизма наследования — повторное использование кодов, так как большинство используемых типов данных являются вариантами друг друга, и писать для каждого свой класс нецелесообразно.
2. Модификатор наследования public не изменяет уровня доступа. Производный класс наследует все компоненты своего базового класса, но может непосредственно использовать только те из них, которые определены с атрибутами public и protected.
3. Private трансформирует компоненты базового класса с атрибутами доступа public и protected в компоненты private производного класса, в то время как private компоненты становятся недоступны в производном классе.
4. Спецификатор protected можно использовать не только для придания членам класса статуса "защищенности", но и для наследования базового класса. Если базовый класс наследуется как защищенный, все его открытые и закрытые члены становятся защищенными членами производного класса.
5. class имя класса: список\_базовых\_классов {список\_компонентов\_класса};
6. Конструкторы **не** **наследуются**, поэтому производный класс должен иметь собственные конструкторы. Порядок вызова конструкторов определяется приведенными далее правилами: Если в конструкторе производного класса явный вызов конструктора базового класса отсутствует, автоматически вызывается конструктор базового класса без параметров.
7. Деструкторы **не** **наследуется**, поэтому даже при отсутствии в производном классе деструктора, он не передается из базового, а формируется компилятором как умалчиваемый.
8. Объекты класса конструируются снизу вверх: сначала базовый, потом компоненты объекты (если они имеются), а потом сам производный класс. Т.Е. объект производного класса содержит в качестве подобъекта объект базового класса.
9. Уничтожаются объекты в обратном порядке: сначала производный, потом его компоненты-объекты, а потом базовый объект. Как мы знаем, объект уничтожается при завершении программы или при выходе из области действия определения объектов и эти действия выполняет деструктор.
10. Связывание — это сопоставление вызова функции с вызовом. В C++ все функции по умолчанию имеют раннее связывание, то есть компилятор и компоновщик решают, какая именно функция должна быть вызвана, до запуска программы. Виртуальные функции имеют позднее связывание, то есть при вызове функции нужное тело выбирается на этапе выполнения программы.
11. Не может быть виртуальных конструкторов, но виртуальный деструктор позволяет уничтожить объект с помощью указателя базового класса, вызывая производные деструкторы соответствующим образом. Например, на самом деле не запускайте этот код.
12. Виртуальность наследуется. После того как функция определена как виртуальная, ее повторное определение в производном классе (с тем же самым прототипом) создает в этом классе новую виртуальную функцию, причем спецификатор virtual может не использоваться. Конструкторы не могут быть виртуальными, в отличие от деструкторов.
13. Открытое наследование устанавливает между классами отношение: При открытом наследовании все, что характеризует объекты класса-предка, является справедливым и для объектов класса-наследника.
14. Закрытое наследование называется отношением LIKEA, или наследованием реализации. Закрытое и защищенное наследование не создает иерархии типов. Поэтому, если класс colorsquare будет закрытым наследником класса rectangle, преобразования указателей невозможны. В основном такое наследование используется для повторного использования кода. Практически закрытое наследование не используется.
15. Функции, использующие указатели или ссылки на базовые классы, должны иметь возможность использовать объекты производных классов, не зная об этом.
16. Иерархия классов позволяет определять новые классы на основе уже имеющихся. Имеющиеся классы обычно называют базовыми(иногда порождающими), а новые классы, формируемые на основе базовых, – производными (порожденными, классами-потомками или наследниками). Производные классы «получают наследство» – данные и методы своих базовых классов, и могут пополняться собственными компонентами (данными и собственными методами).

Вопросы 17,18,19,20 – продемонстрированы в решении лаб.р.