5.3. Виртуальное наследование как решение проблемы ромбовидного наследования. Особенности размещения объектов в памяти при виртуальном наследовании. Почему при наличии виртуальных предков размер объекта увеличивается? Поведение компилятора в случае комбинации виртуального и обычного наследования одного и того же предка. Особенности видимости и доступности родительских методов в этих случаях.

Чтобы решить проблему ромбовидного наследования воспользуемся виртуальным наследованием. Оно создает только одну бабушку.

```
struct Granny {
    int g;
};

struct Mother: public virtual Granny {
    int m;
};

struct Father: public virtual Granny {
    int f;
};

struct Son: Mother, Father {
    int s;
};
```

Посмотрим на то как объект Son лежит в памяти

1. Без виртуального наследования: [g][m][g][f][s] (20 байт)
   Mother Father
2. С виртуальным наследованием: [m_ptr][m]....[f_ptr][f][s][g].... (40 байт)

   m_ptr, f_ptr - указатели на некоторое место в памяти, где лежит список указателей на виртуальных родителей (то есть на бабушку). Четыре точки - это пропуск в четыре байта, который возникает из-за выравнивания (адрес 8 байтовых чисел должен быть кратен восьми)

Из рассуждений выше видно, почему размер объекта увеличился (несмотря на то, что одна 4-байтовая бабушка исчезла добавилось 2 восьмибайтовых указателя)

Уберем у одного из родителей виртуальное наследование (допустим у Father). Теперь Son лежит в памяти так: [m_ptr][m][g][f][s][g] (одно g от папиной невиртуальной бабушки и одно от маминой виртуальной). Таким образом, проблему ромбовидного наследования таким способом не решить: обращение к g будет неоднозначным.

Замечание: В данных примерах не важно какое наследование (public/protected/private), так как проверка доступа происходит после выбора того к чему обращаемся, то есть неоднозначность никуда не денется.