Программирование на языке C++
Лекция 7

Множественное наследование

Александр Смаль

Множественное наследование

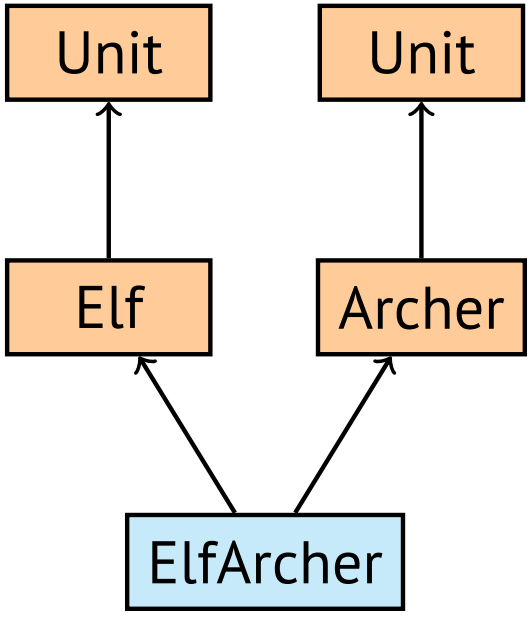
Множественное наследование (multiple inheritance) — возможность наследовать сразу несколько классов.

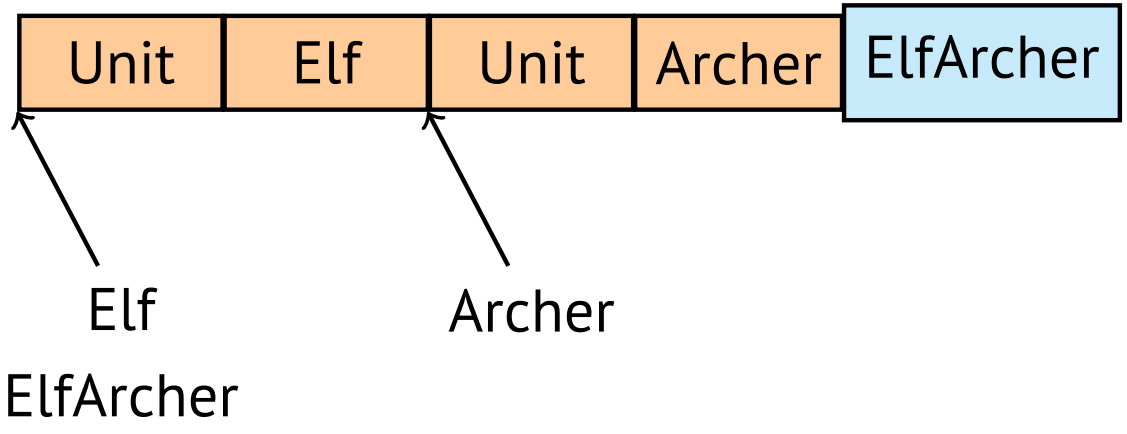
```
struct Unit {
    Unit(unitid id, int hp): id_(id), hp_(hp) {}
    virtual unitid id() const { return id_; }
    virtual int    hp() const { return hp_; }
private:
    unitid id_;
    int    hp_;
};

struct Elf:    Unit { ... };
struct Archer: Unit { ... };

struct ElfArcher: Elf, Archer {
    unitid id() const { return Elf::id(); }
    int    hp() const { return Elf::hp(); }
};
```

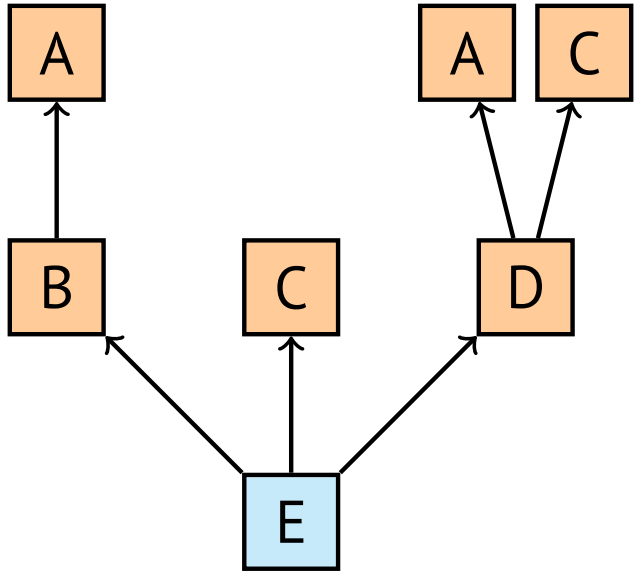
Представление в памяти





Важно: указатели при приведении могут смещаться.

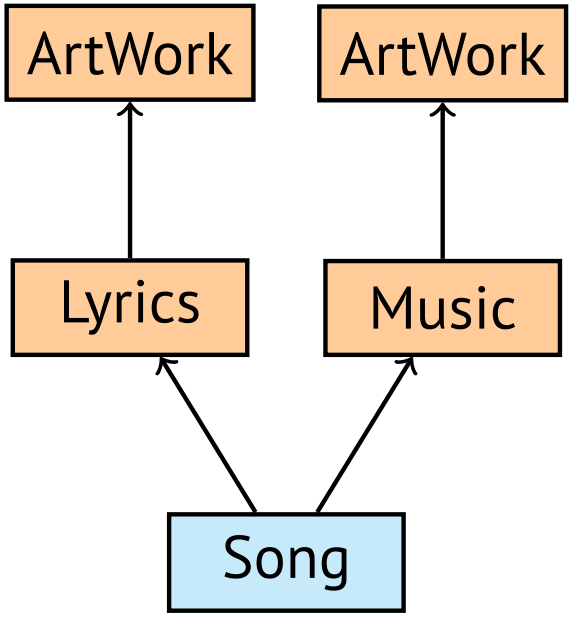
Создание и удаление объекта

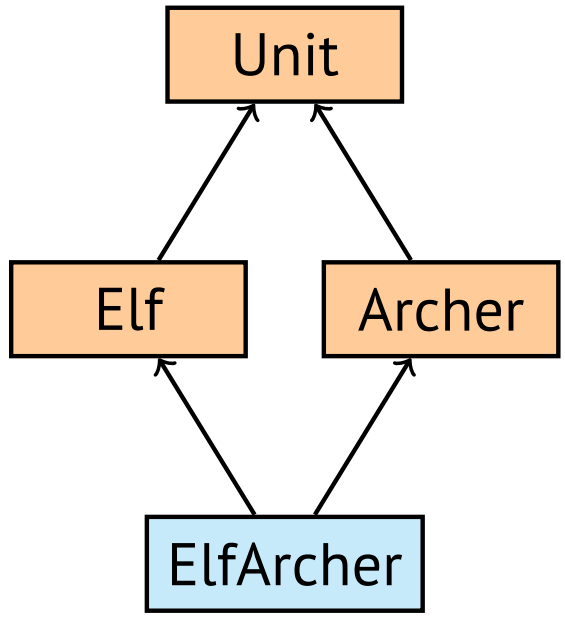


Порядок вызова конструкторов: `A, B, C, A, C, D, E`.
Деструкторы вызываются в обратном порядке.
Проблемы:

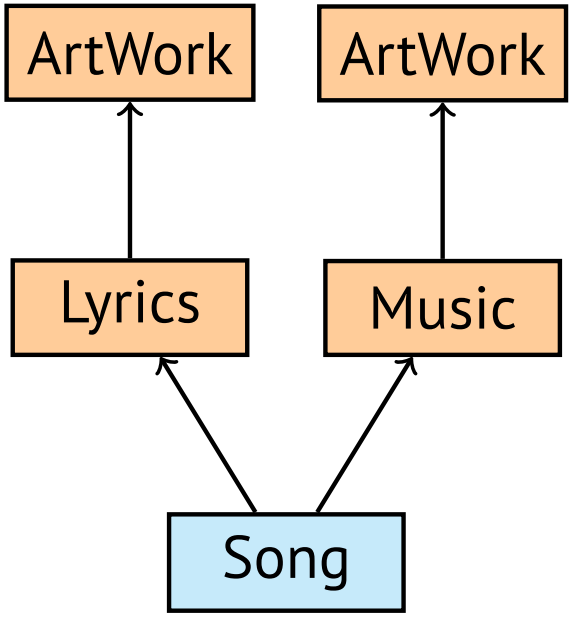
1. Дублирование A и C.
2. Недоступность первого C.

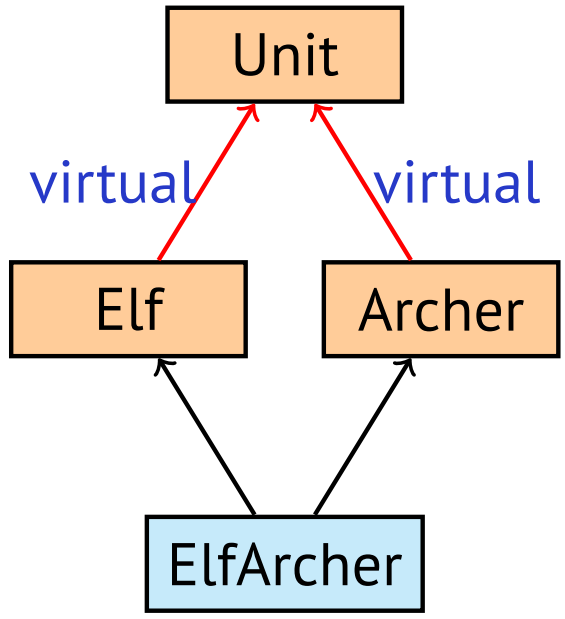
Виртуальное наследование





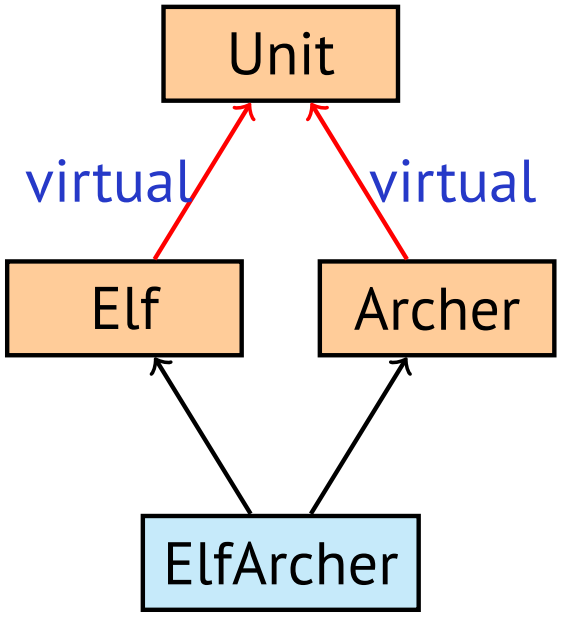
Виртуальное наследование

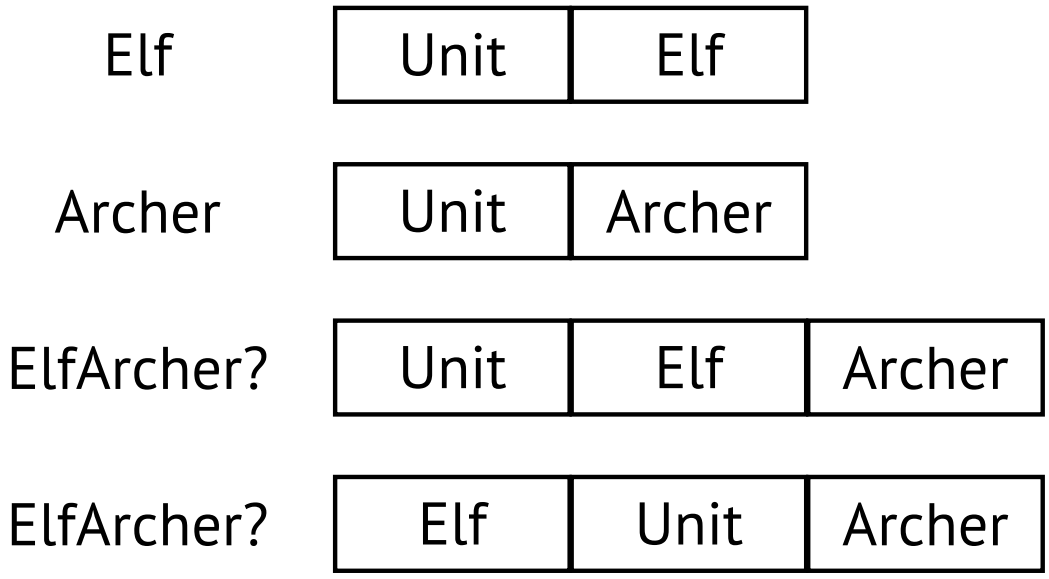




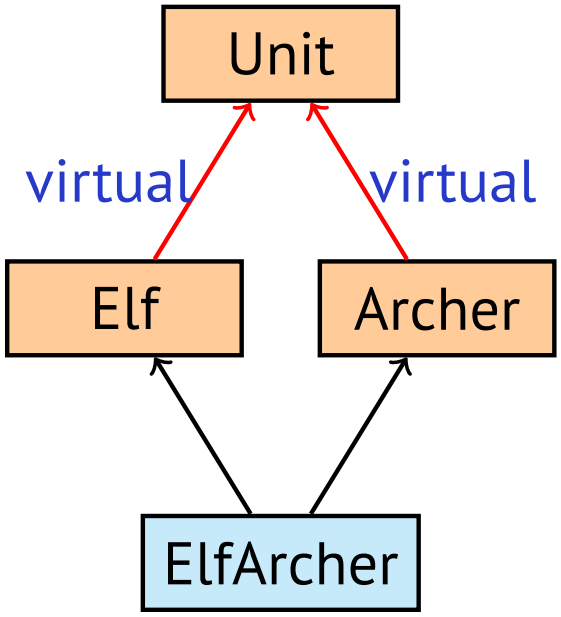

```
struct Unit {};
struct Elf:  virtual Unit {};
struct Archer: virtual Unit {};
struct ElfArcher: Elf, Archer {};
```

Как устроено расположение в памяти?





Как устроено расположение в памяти?



На самом деле.

Elf	Unit	Elf	
Archer	Unit	Archer	
ElfArcher?	Unit	Elf	Archer
ElfArcher?	Elf	Unit	Archer
Elf	Elf	Unit	
Archer	Archer	Unit	
ElfArcher	Elf	Archer	Unit

Доступ через таблицу виртуальных методов

```
struct Unit {
    unitid id;
};
struct Elf  : virtual Unit { };
struct Archer : virtual Unit { };
struct ElfArcher : Elf, Archer { };
```

Доступ через таблицу виртуальных методов

```
struct Unit {
    unitid id;
};
struct Elf  : virtual Unit { };
struct Archer : virtual Unit { };
struct ElfArcher : Elf, Archer { };
```

Рассмотрим такой код:

```
    Elf * e = (rand() % 2)? new Elf() : new ElfArcher();
    unitid id = e->id; // (*)
```

Доступ через таблицу виртуальных методов

```
struct Unit {
    unitid id;
};
struct Elf  : virtual Unit { };
struct Archer : virtual Unit { };
struct ElfArcher : Elf, Archer { };
```

Рассмотрим такой код:

```
    Elf * e = (rand() % 2)? new Elf() : new ElfArcher();
    unitid id = e->id; // (*)
```

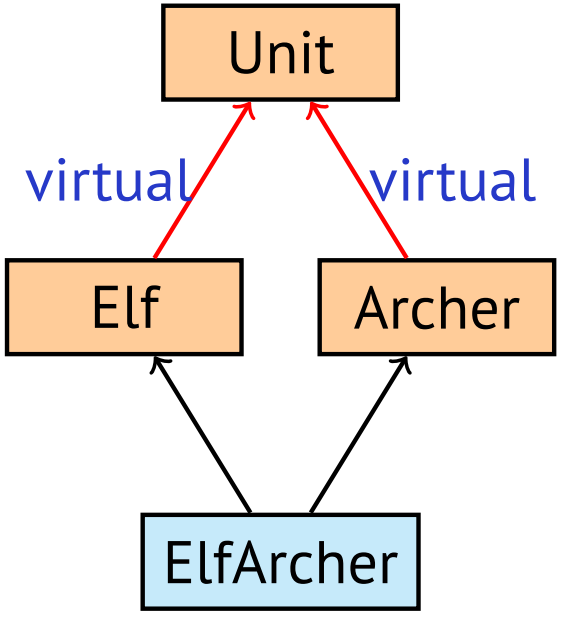
Строка (*) будет преобразована в строку

```
    unitid id = e->__getUnitPtr__()->id;
```

где `__getUnitPtr__()` — это служебный виртуальный метод.

Кто вызывает конструктор базового класса?

```
struct Unit {
    Unit(unitid id, int health_points);
};
struct Elf: virtual Unit {
    explicit Elf(unitid id)
        : Unit(id, 100) {}
};
struct Archer: virtual Unit {
    explicit Archer(unitid id)
        : Unit(id, 120) {}
};
struct ElfArcher: Elf, Archer {
    explicit ElfArcher(unitid id)
        : Unit(id, 150)
        , Elf(id)
        , Archer(id) {}
};
```



Заключение

- Не используйте множественное наследование для наследования реализации.

Заключение

- Не используйте множественное наследование для наследования реализации.
- Используйте концепцию интерфейсов (классы без реализаций и членов данных).

Заключение

- Не используйте множественное наследование для наследования реализации.
- Используйте концепцию интерфейсов (классы без реализаций и членов данных).
- Помните о неприятностях, связанных с множественным наследованием.

Заключение

- Не используйте множественное наследование для наследования реализации.
- Используйте концепцию интерфейсов (классы без реализаций и членов данных).
- Помните о неприятностях, связанных с множественным наследованием.
- Хорошо подумайте перед тем, как использовать виртуальное наследование.

Заключение

- Не используйте множественное наследование для наследования реализации.
- Используйте концепцию интерфейсов (классы без реализаций и членов данных).
- Помните о неприятностях, связанных с множественным наследованием.
- Хорошо подумайте перед тем, как использовать виртуальное наследование.
- Помните о неприятностях, связанных с виртуальным наследованием.