Полиморфизм

Конструктор копии, виртуальные методы, виртуальный деструктор, абстрактные классы

Конструкторы

Обычный конструктор:

`MyClass(/* аргументы */)` - может быть иметь любую сигнатуру кроме тех, что ниже

Конструктор копии:

`MyClass(const MyClass&)` - имеет определенную сигнатуру

Конструктор перемещения (с C++11):

`MyClass(MyClass&&)` - имеет определенную сигнатуру

К коду!

Конструкторы и деструкторы. Порядок вызова при наследовании

Когда мы создаем объект дочернего класса сперва вызывается конструктор самого базированного базового класса в этой иерархии классов. Затем конструктор дочернего класса самого базированного класса, затем его дочерний класс и т.д.

Когда уничтожается объект дочернего класса сперва вызывается его деструктор, потом деструктор класса которого он наследует, далее деструктор класса повыше и т.д. до тех пор, пока не дойдет до деструктора самого базированного в этой иерархии класса

Выше речь идет об конструкторе по умолчанию. Если нужно, чтобы дочерний класс вызывал конкретный конструктор родительского класса, то нужно его явно указать с помощью следующего синтаксиса: `Derrived() : Base("some", "arguments");`

К КОДУ!

Виртуальные методы

Виртуальный метод - **метод базового класса, который предполагается, что будет переопределен в дочернем.**

Нужен для реализации полиморфизма времени выполнения (то есть нормального полиморфизма). Напишем такой метод и переопределим в дочернем, как и следует делать.

К КОДУ!

Виртуальный деструктор

Конструкторы и деструкторы не наследуются, но дочерние классы пользуются ими для того, чтобы построить на их основе свою реализацию, или для очистки памяти класса выше по иерархии.

ЗОЛОТОЕ ПРАВИЛО : если ваш класс может быть унаследован (и этому может быть веская причина), деструктор должен быть виртуальным.

Почему? **К КОДУ!**