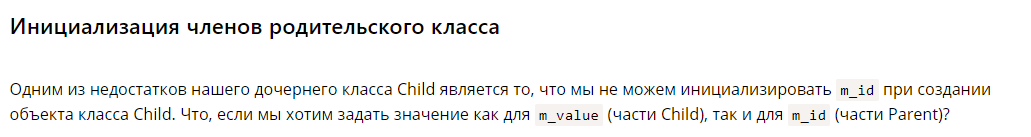




Рассмотрим, что происходит при создании объектов дочернего класса. Как мы уже говорили, класс Child состоит из двух частей: часть Parent и часть Child. Когда C++ создает объекты дочерних классов, то он делает это поэтапно. Сначала создается самый верхний класс иерархии (тот, который родитель). Затем создается дочерний класс, который идет следующим по порядку, и так до тех пор, пока не будет создан последний класс (тот, который находится в самом низу иерархии).

Поэтому, при создании объекта класса Child, сначала создается часть Parent класса Child (с использованием конструктора по умолчанию класса Parent) и после того, как с частью Parent покончено, создается вторая часть Child (с использованием конструктора по умолчанию класса Child).

В этом есть смысл, так как по логике вещей ребенок не может существовать без родителей. Это также способствует безопасности и эффективности выполнения кода: дочерний класс часто использует переменные-члены и методы родителя, но родительский класс ничего не знает о своем дочернем классе. Первоначальная инициализация родительского класса гарантирует, что его переменные-члены и методы будут проинициализированы до момента использования их дочерним классом. Язык C++ выполняет построение дочерних классов поэтапно, начиная с верхнего класса иерархии и заканчивая нижним классом иерархии. По мере построения каждого класса для выполнения инициализации вызывается соответствующий конструктор соответствующего класса.



 C++ запрещает дочерним классам инициализировать наследуемые переменные-члены родительского класса в списке инициализации своего конструктора. Другими словами, значение переменной может быть задано только в списке инициализации конструктора, принадлежащего тому же классу, что и переменная-член.

Почему C++ так делает? Ответ связан с [**константными**](https://ravesli.com/urok-37-simvolnye-konstanty-const-constexpr/) переменными и [**ссылками**](https://ravesli.com/urok-88-ssylki/). Подумайте, что произошло бы, если бы m\_id был const. Поскольку константы должны быть инициализированы значениями при создании, то конструктор родительского класса должен установить это значение при создании переменной-члена. В то же время конструктор дочернего класса выполняется только после выполнения конструкторов родительского класса. Каждый дочерний класс имел бы тогда возможность инициализировать эту переменную, потенциально изменяя её значение! Ограничивая инициализацию переменных конструктором класса, к которому принадлежат эти переменные, язык C++ гарантирует, что все переменные будут инициализированы только один раз.

