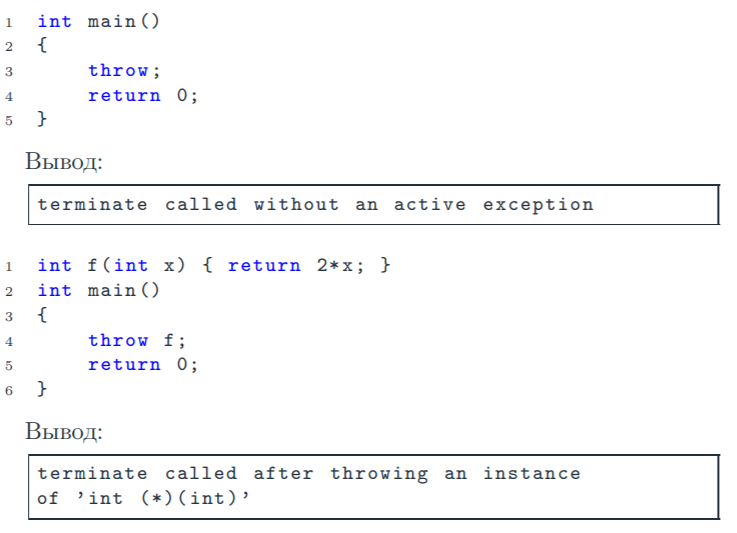
# Оператор throw

В отличие от языка Java, в качестве исключения в C++ может выступать любое значение (объект, строка, значение базового типа и т.п.). Для порождения исключения предназначен оператор throw, который может вызываться как с указанием значения, описывающего исключительную ситуацию, так и без его указания:



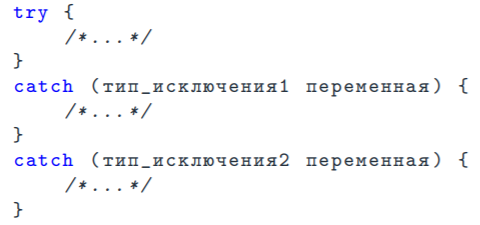
Механизм исключений в C++ не предусматривает запоминание информации о стеке вызовов в момент порождения исключения. Поэтому сообщение, выводимое при аварийном завершении программы по причине необработанного исключения, менее информативно.

Пример

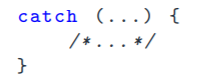


# Синтаксис try- и catch-блоков

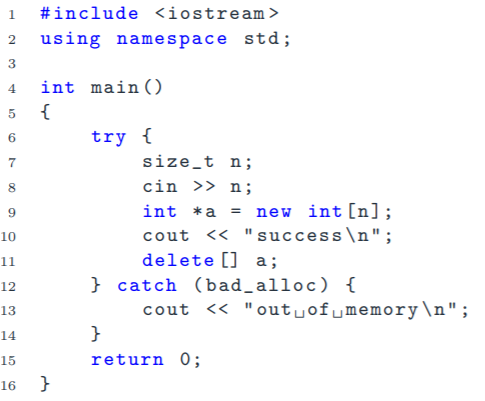
Как и в языке Java, участки кода, в которых ожидается возникновение исключительной ситуации, обрамляются try блоками. С try-блоками связаны один или несколько catch блоков, осуществляющих перехват исключений по типу:



Catch-блок, который может перехватить любое исключение, записывается как



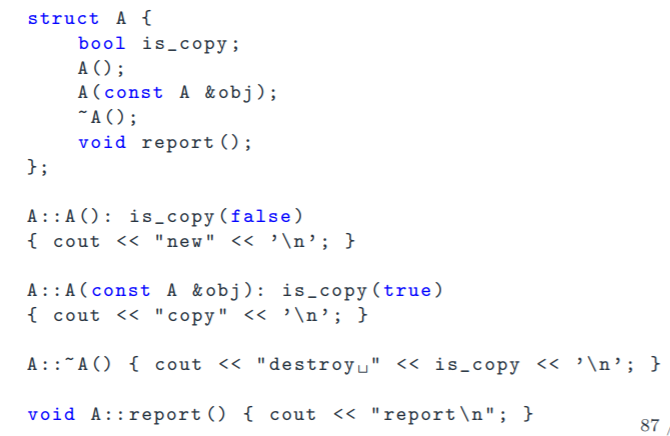
Пример: перехват исключения std::bad\_alloc



Исключение std::bad\_alloc порождается операцией new при невозможности выделить блок динамической памяти.

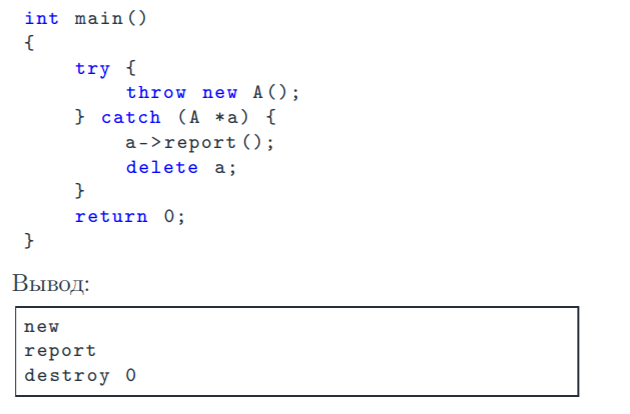
# Жизненный цикл объектов-исключений

Исследуем жизненный цикл объектов исключений с помощью класса A, который протоколирует факты создания, копирования и уничтожения своих экземпляров:

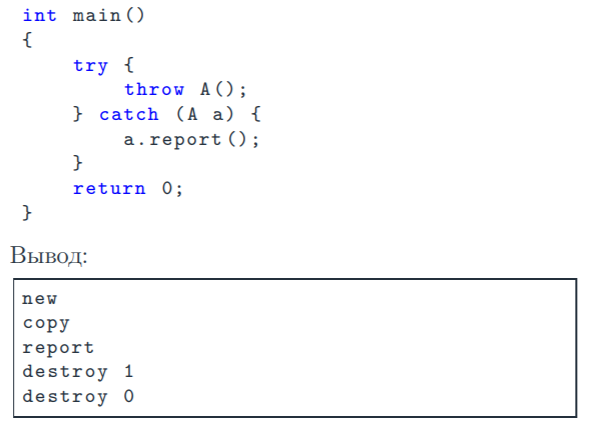


Пример 1: жизненный цикл объектов-исключений

Порождение объекта-исключения в куче и перехват, при котором он передаётся в catch-блок по указателю. В конце catch-блока необходимо явное уничтожение объекта.

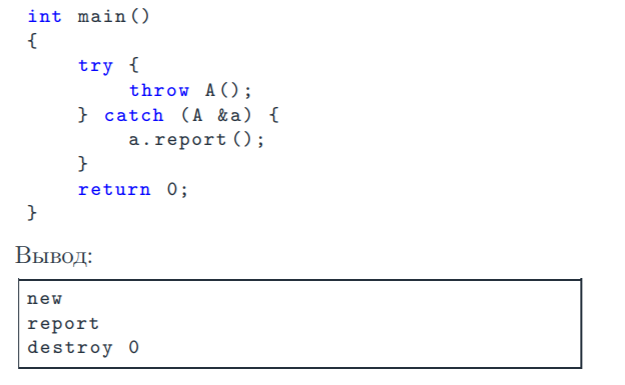


Пример 2: жизненный цикл объектов-исключений

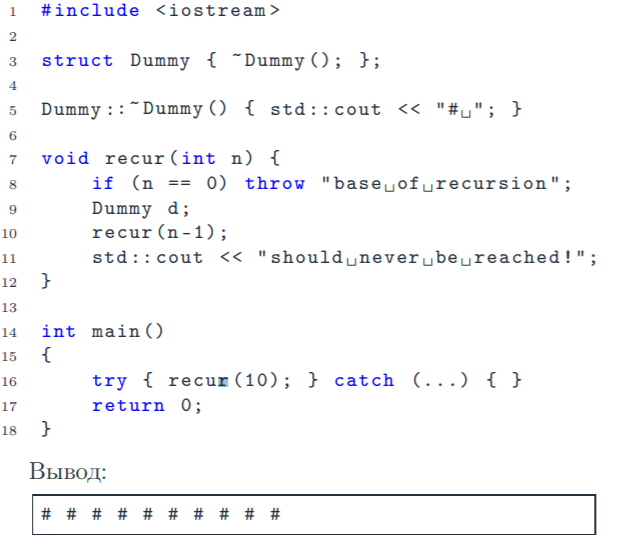


Пример 3: жизненный цикл объектов-исключений

Порождение временного объекта-исключения и перехват, при котором он передаётся в catch-блок по ссылке, т.е. без создания копии.

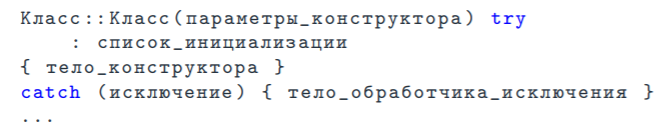


Пример: уничтожение автоматических объектов в процессе передачи исключения в catch-блок



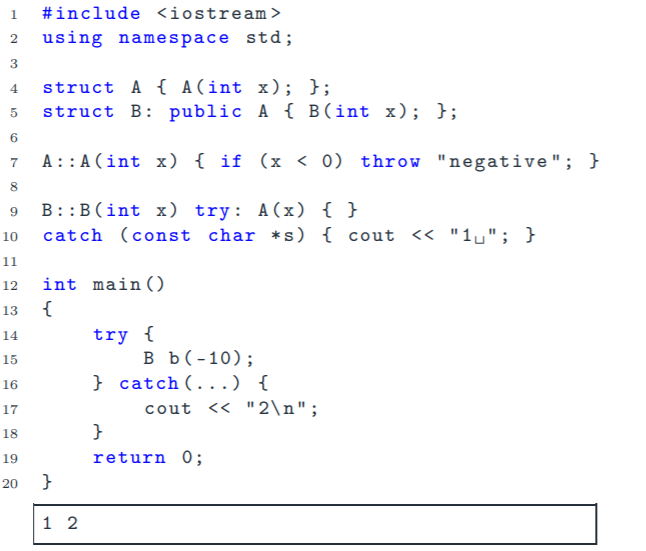
# Перехват исключений в инициализаторах конструкторов

Напомним, что конструкторы в C++ могут иметь списки инициализаторов, в которых вызываются конструкторы объектов, расположенных в полях, и конструкторы базовых классов. Так как конструкторы полей и базовых классов могут порождать исключения, нужно уметь эти исключения перехватывать, например, для протоколирования неудачного создания объектов в логах. Для этой цели служат специальные try-блоки, обрамляющие тело конструктора, которые синтаксически выглядят как

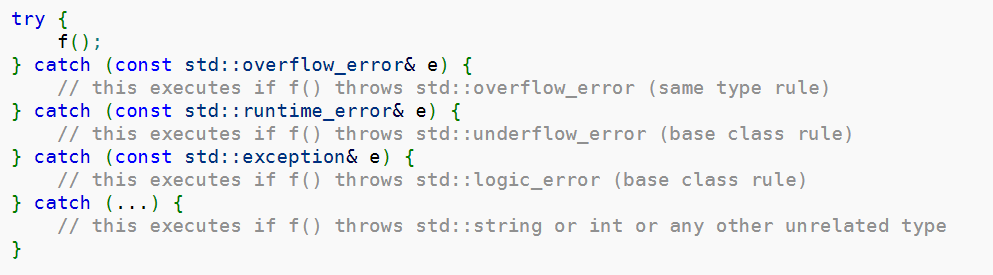


Отметим, что если catch-блок не оканчивается порождением другого исключения, старое исключение не уничтожается и передаётся дальше.

Пример: перехват исключений в инициализаторах конструкторов



# Какие бывают перехватчики?



Более подробно смотрите тут

<http://www.codenet.ru/progr/cpp/Try-Catch-Throw.php> <https://en.cppreference.com/w/cpp/language/try_catch>

# Спецификатор throw в заголовках функций

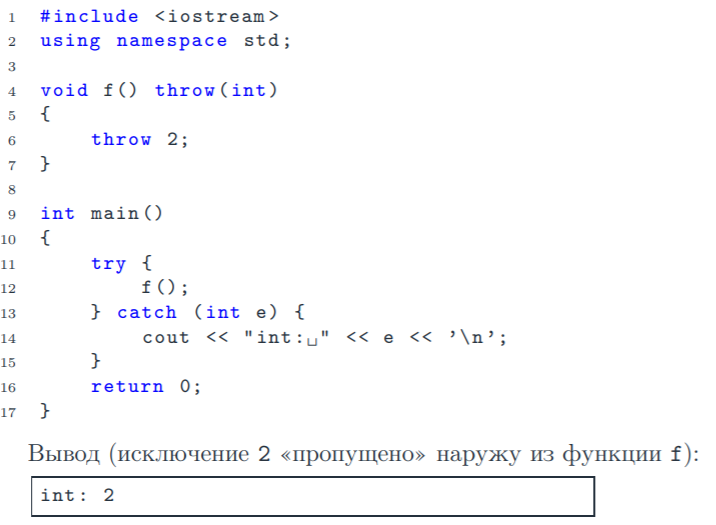
Как и в языке Java, в C++ можно перечислить типы исключений, которые может порождать вызов функции. Для этого после списка формальных параметров следует разместить спецификатор throw:



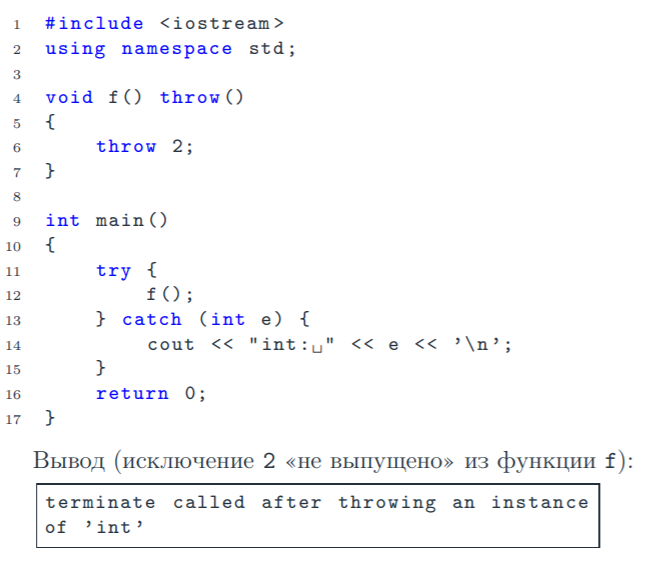
Семантика этого спецификатора несколько отличается от семантики соответствующей конструкции языка Java, а именно:

* Вызов функции, для которой разрешённые исключения не указаны, может порождать любые исключения;
* Вызов функции со спецификатором throw() не порождает никаких исключений;
* Компилятор не следит за тем, какие исключения порождаются внутри функции: вместо этого неразрешённые исключения просто не могут выйти за пределы функции и приводят к аварийному завершению программы.

Пример 1: спецификатор throw

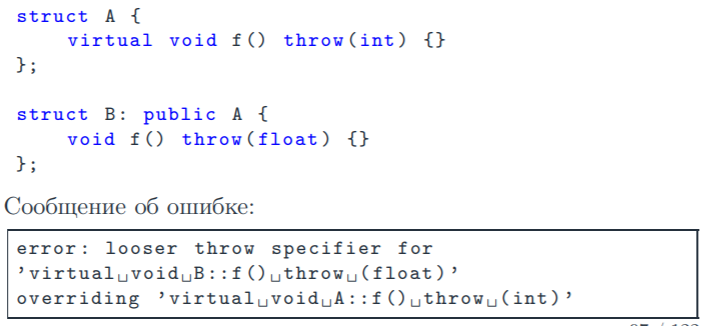


Пример 2: спецификатор throw



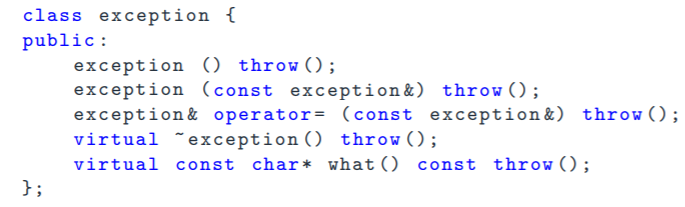
# Спецификатор throw и переопределение виртуальных методов

По очевидным соображениям, связанным с поддержкой полиморфизма, переопределение виртуальных методов не должно ослаблять ограничения, накладываемые спецификатором throw, т.е. переопределённому методу не разрешается порождать исключения, не порождаемые методом базового класса.



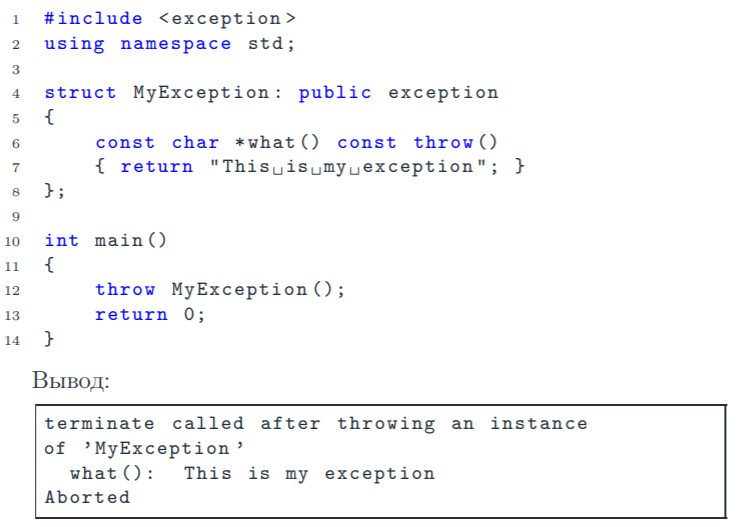
# Создание классов исключений

Рекомендуемым способом создания классов исключений в языке C++ является наследование их от библиотечного класса exception (его объявление расположено в заголовочном файле ):



В производном от exception классе нужно переопределить виртуальный метод what, возвращающий ASCIIZ-строку с описанием исключительной ситуации.

Пример: создание классов исключений



# Разматывание стека

<https://overcoder.net/q/21383/что-такое-разматывание-стека>

# 

# Нелокальный переход к какой-то там оболочке. Речь, вероятнее всего, про setjmp

<https://ru.wikipedia.org/wiki/Setjmp.h>