# Общая информация

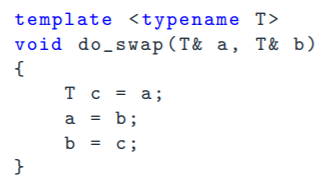
В отличие от языка Java, обобщённые классы в C++ в строгом смысле отсутствуют. Вместо них в C++ используется развитый язык макроопределений, предназначенный для порождения кода во время компиляции программы. Ключевым элментом этого языка является понятие шаблона, который является рецептом для генерации кода класса или функции (метода). Объявление шаблона начинается с ключевого слова template, за которым следует список формальных параметров шаблона:



Тело шаблона представляет собой определение класса или функции, тем самым позволяя объявлять шаблон класса и шаблон функции.

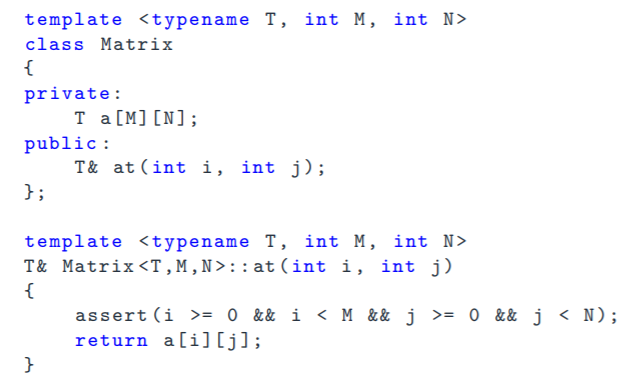
# Типовые формальные параметры шаблона

Формальные параметры шаблона – это идентификаторы, областью видимости которых является тело шаблона. Они могут обозначать типы, значения или, в свою очередь, другие шаблоны. При применении шаблона формальные параметры в его теле заменяются на конкретные типы, значения и шаблоны, и полученный код компилируется. Типовые параметры шаблона обозначают типы: синтаксически имя типового параметра в списке формальных параметров шаблона предваряется ключевым словом typename (или class). Например, шаблон функции:

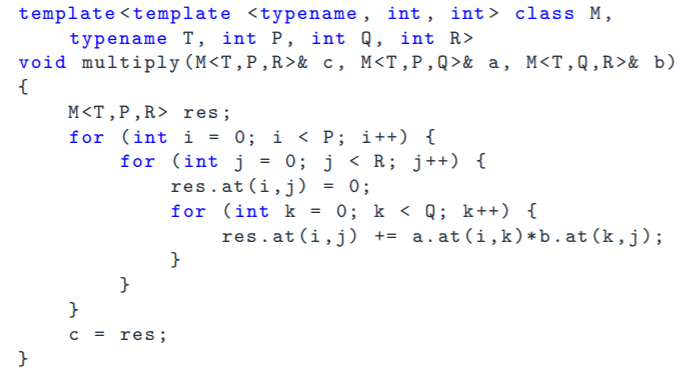


# Нетиповые формальные параметры шаблона

Нетиповые параметры шаблона представляют значения, а не типы. Их синтаксис похож на объявление переменных. Внутри тела шаблона имя такого параметра обозначает константу указанного типа. Например, шаблон для матриц:

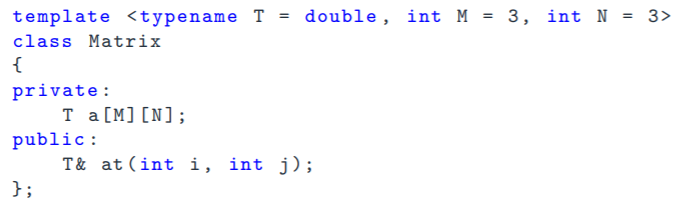


# Шаблонные формальные параметры шаблона

Шаблонные параметры шаблона позволяют параметризовать шаблон другим шаблоном. Синтаксически такой параметр записывается как объявление шаблона без тела. Например, умножение матриц: 

# Значения параметров шаблона по умолчанию

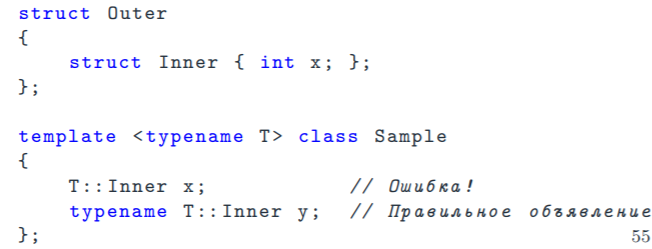
Формальные параметры шаблона могут иметь значения по умолчанию. Синтаксически это оформляется путём добавления после имени параметра знака «=», за которым следует значение:



Если формальный параметр шаблона имеет значение по умолчанию, то все следующие за ним параметры также должны иметь значения по умолчанию. Значения по умолчанию используются в случае, если при применении (инстанциации) шаблона соответствующие фактические параметры не указаны.

# Зависимые идентификаторы в шаблоне

Зависимый идентификатор – это любое имя внутри определения шаблона, которое зависит от формальных параметров шаблона. Смысл зависимых идентификаторов становится ясен компилятору только при инстанциации шаблона, когда известны его фактические параметры. Поэтому по умолчанию компилятор считает, что зависимые идентификаторы обозначают поля и методы. Чтобы указать, что некоторый зависимый идентификатор обозначает тип, перед ним нужно указывать ключевое слово typename.



# Инстанциация шаблонов

Инстанциация шаблона – это порождение кода по шаблону и списку фактических параметров. Инстанциация осуществляется при первом использовании конструкции

имя\_шаблона < список фактических параметров >

Например, в момент, когда компилятор обрабатывает объявления переменной

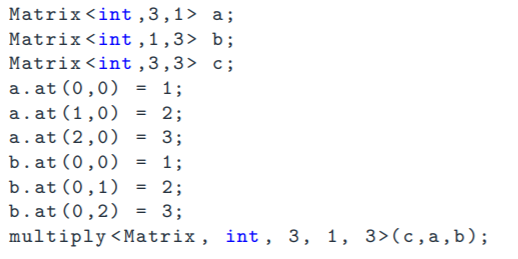


он порождает код, получаемый путём подстановки в тело шаблона Matrix фактических параметров int, 4 и 5. Отметим, что если последний фактический параметр шаблона – типовый, и к тому же представляет собой инстанциацию другого шаблона, то нужно ставить пробел между заключительнми угловыми скобками. Например,



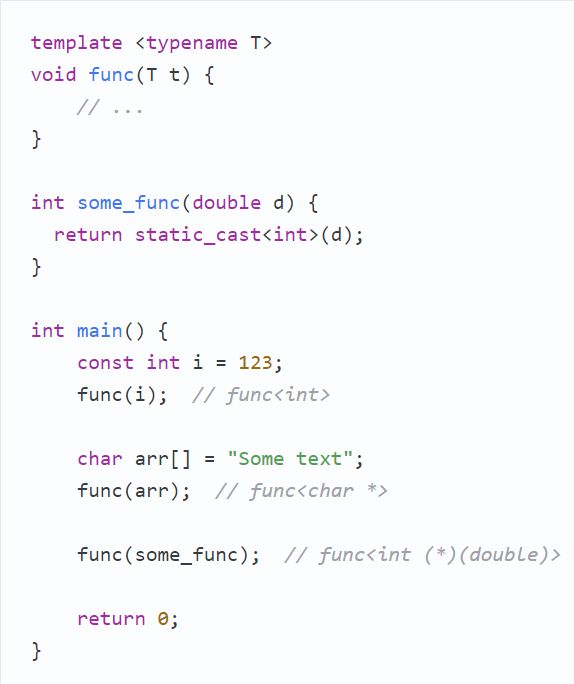
# Выведение фактических параметров при инстанциации шаблона функции

Если параметры шаблона функции используются в списке формальных параметров этой функции, то компилятор может вывести значения фактических параметров шаблона самостоятельно. Например, рассмотрим следующей фрагмент кода:



Автоматическое выведение фактических параметров шаблона функции позволяет упростить вызов функции multiply:

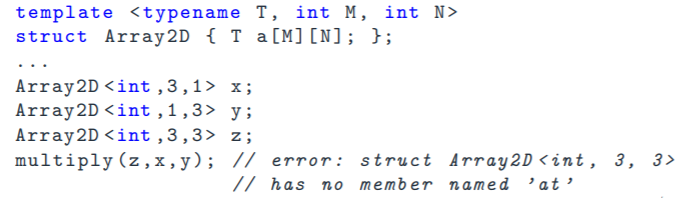


Короче, если кто не понял, что произошло. Компилятор C++ настолько преисполнился в своем познании, что он по аргументам, которые вы ему подсунули, сам поймет, как юзать шаблон. Ну, или говоря умнее, компилятор может самостоятельно вывести аргументы шаблона функции, если это можно сделать однозначно на основании имеющейся информации. При выведении типов параметров шаблона применяются стандартные преобразования как при вызове обычной функции (отбрасывается *const* у литеральных типов, массивы сводятся к указателям, ссылки на функции приводятся к указателям на функции и т.д.). Вот еще пример

Но это мы рассматриваем шаблоны функций. Для шаблонов класса подобное можно делать с 17-ого стандарта. Кому интересно, вот ссылка: <https://habr.com/ru/post/461963/>

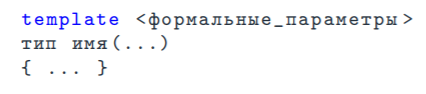
# Требования к фактическим параметрам шаблонов

Тело шаблона накладывает требования на его фактические параметры. Например, в теле шаблона функции multiply подразумевается, что значения типа M – это объекты, имеющие метод at, который возвращает ссылку на значение, к которому могут быть применены операции сложения и умножения. Если фактические параметры шаблона не удовлетворяют требованиям, инстанциация шаблона с этими фактическими параметрами приведёт к ошибке времени компиляции. Например,

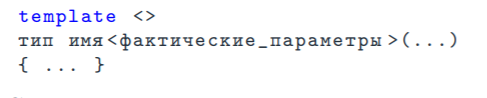


# Специализация шаблона функции

Специализация шаблона – это разработка отдельной версии порождаемого шаблоном кода для конкретного набора фактических параметров. Пусть имеется шаблон функции вида

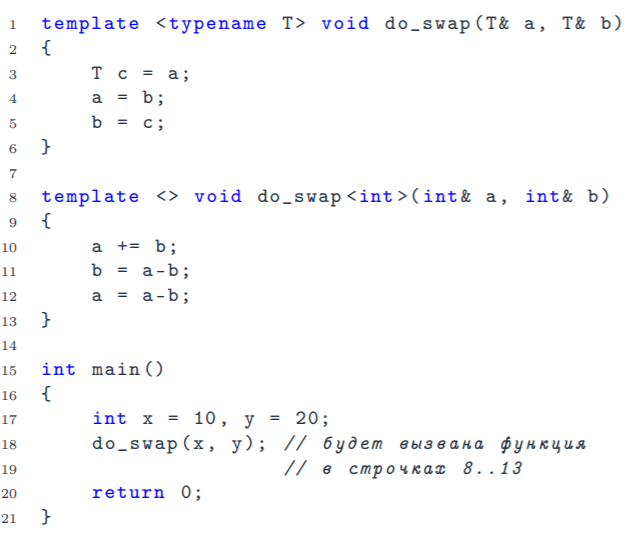


Специализированная версия функции для конкретного набора фактических параметров записывается как

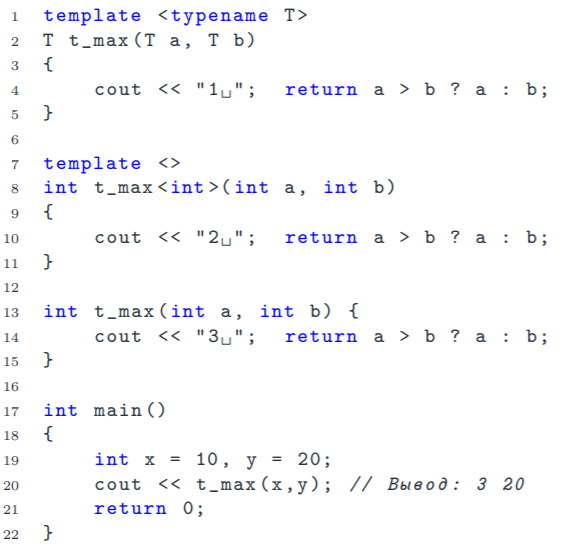


Следует иметь в виду, что специализация шаблона функции не даёт особенных преимуществ перед обычной перегрузкой функций.

Пример: специализация **шаблона функции**



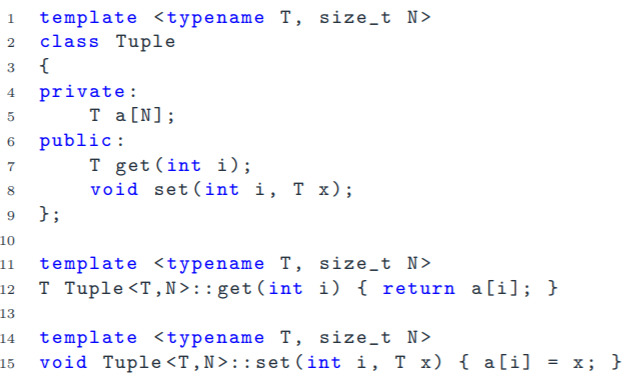
Пример: перегрузка имеет больший приоритет, чем специализация

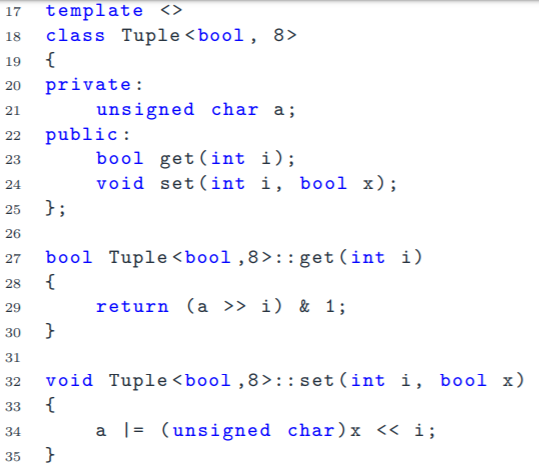


# Cпециализация шаблона класса

Специализированные версии шаблонов классов создаются аналогично специализации шаблонов функций. Например, рассмотрим шаблон класса, представляющего кортежи фиксированной длины

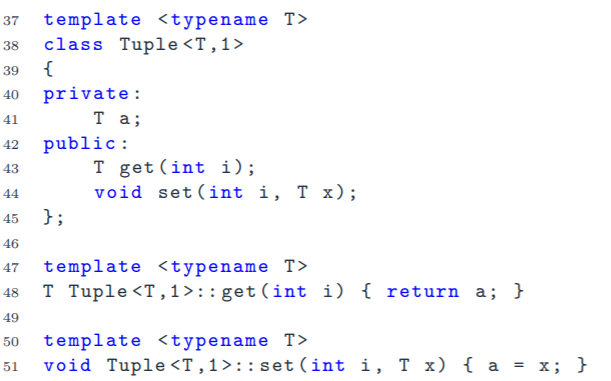
Пример: специализация шаблона класса





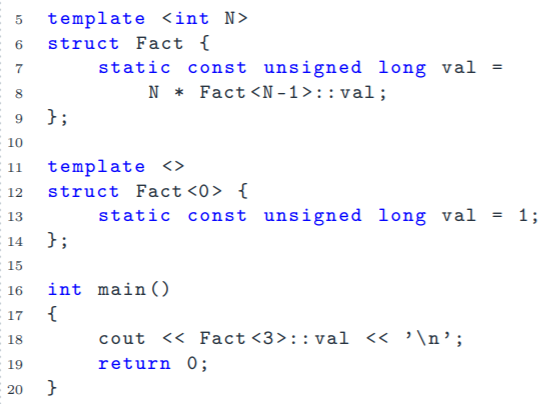
# Частичная специализация шаблона класса

При частичной специализации фиксируется только часть фактических параметров шаблона. Например



# Вычисления во время компиляции

Специализация шаблонов позволяет заставить компилятор выполнить некоторые вычисления во время компиляции. Например, можно попросить компилятор вычислить факториал числа:



Или

