30.09.23-Аминев-Эссе-Что такое темплейты в C++

Шаблоны — это фрагменты обобщённого кода, в котором некоторые типы или константы вынесены в параметры. Шаблонными могут быть функции, структуры (классы) и даже переменные. Компилятор превращает использование шаблона в конкретный код, подставляя в него нужные параметры на этапе компиляции. Шаблоны позволяют писать общий код, пригодный для использования с разными типами данных.

Стандартная библиотека C++ построена на шаблонах. Раньше её даже называли Standard Template Library (STL, стандартная библиотека шаблонов). Её контейнеры и итераторы являются шаблонными классами, а алгоритмы — шаблонными функциями. Примеры шаблонных конструкций из стандартной библиотеки нам уже встречались: это, например, контейнер vector.

Перегрузка функций

Количество и типы аргументов функции должны быть известны заранее, на этапе компиляции. Но в языке C++ можно создавать функции с одним и тем же именем, но разным набором или типами аргументов и с разными телами. Такие функции называются перегруженными. Рассмотрим, например, семейство перегруженных функций для печати переменной на экран.

Компилятор, сравнивая разные версии функции друг с другом, смотрит на их имена и набор типов аргументов. При этом имена аргументов ни на что не влияют. Также нельзя перегружать функции по типу возвращаемого значения. Действительно, возвращаемое значение может просто игнорироваться в месте вызова, и компилятор не сможет определить, какая версия функции имеется в виду.

Шаблонные функции

Рассмотрим классический пример. Предположим, у нас есть функция, вычисляющая максимум целых чисел:

Она определена для аргументов типа int. Однако, если применить её к аргументам типа double, результат получится неожиданным. А её применение к строкам или векторам вообще не скомпилируется:

В вызове Max(3.14159, 2.71828) аргументы будут преобразованы к типу int, то есть получится Max(3, 2). Вызов Max(word1, word2) не скомпилируется, так как строки нельзя привести к типу int. Чтобы эти вызовы корректно заработали, надо определить перегруженные версии функции Max:

Выписывать похожие друг на друга версии функций утомительно. Кроме того, такие функции не смогут работать с новыми, неизвестными нам заранее типами. Шаблоны позволяют описать такую функцию один раз, вынеся тип в параметры:

Шаблон начинается с шапки template. Далее в угловых скобках перечисляются формальные имена параметров. В нашем случае параметр один — это тип T (от слова type). Вместо ключевого слова typename в этом месте допускается использовать слово class. А вместо имени T можно было бы использовать любой другой идентификатор.

Так как мы не знаем, будет ли тип T встроенным или сложным, то на всякий случай передаём аргументы в функцию по константной ссылке, чтобы избежать лишнего копирования.

В нашей шаблонной функции Max используется оператор >. Он определён для обычных чисел, строк и векторов (если, конечно, для элементов вектора тоже определён этот оператор). Но если попробовать применить наш шаблон к типу, не поддерживающему оператор >, то произойдёт ошибка компиляции:

Вывод шаблонных параметров

Конкретные версии шаблонной функции Max для нужных типов получаются подстановкой шаблонных аргументов в угловые скобки. Так, Max<int> — это версия нашей функции для типа int, а Max<std::string> — версия для строк. Важно понимать, что, несмотря на общий шаблон, это разные функции, которые просто порождаются компилятором по образцу.

Вызвать шаблонную функцию можно было бы так:

Однако параметры шаблона в угловых скобках можно не писать: компилятор попытается сам угадать эти параметры по типу аргументов:

В случае неоднозначностей, например в вызове Max(3.14159, 2), компилятор не сможет автоматически вывести параметр, и ему придётся подсказать тип: Max<double>(3.14159, 2).

Перегрузка шаблонных функций

Шаблонные функции тоже можно перегружать. Пусть, например, мы хотим вычислять максимум двух векторов, но при этом сравнивать векторы сначала по размеру, а затем уже лексикографически. Стандартное сравнение векторов через оператор > не будет учитывать размер. Поэтому напишем отдельную перегрузку для векторов:

Разрешение неоднозначностей

Когда компилятор видит вызов функции, ему нужно правильно определить, в каком пространстве имён её искать, какую из перегруженных версий выбрать, а в случае шаблонной функции — как вывести параметры шаблона. Для шаблонных функций после выбора перегруженной версии возможен ещё выбор из вариантов полной специализации шаблона. Общие правила поиска нужной функции достаточно сложны, и мы не будем их здесь приводить полностью. Однако в случаях, которые мы будем рассматривать в этом учебнике, выбор нужной функции будет интуитивно понятен.

Шаблонные структуры

Структуры и классы также могут быть описаны в общем виде и параметризованы типами или константами времени компиляции. Типичный пример шаблонной структуры — std::pair. Определим по аналогии свою структуру Triple с тремя шаблонными типами:

Здесь так же, как и в случае функций, компилятор генерирует по образцу две никак не связанные друг с другом структуры Triple<int, int, int> и Triple<std::string, std::string, int>.

В следующих параграфах мы будем подробно рассматривать шаблонные классы, в которых могут быть шаблонные функции-члены.

Информацию прочитал и скопировал отсюда: https://academy.yandex.ru/handbook/cpp/article/templates