Шаблоны функций

Шаблоны функций — это функции, которые служат образцом для создания других подобных функций. Главная идея — создание функций без указания точного типа(ов) некоторых или всех переменных. Для этого мы определяем функцию, указывая **тип параметра шаблона**, который используется вместо любого типа данных. После того, как мы создали функцию с типом параметра шаблона, мы фактически создали «трафарет функции». При вызове шаблона функции, компилятор использует «трафарет» в качестве образца функции, заменяя тип параметра шаблона на фактический тип переменных, передаваемых в функцию!

Объявление параметров шаблона:

- Сначала пишем ключевое слово template, которое сообщает компилятору, что дальше мы будем объявлять параметры шаблона.
- Параметры шаблона функции указываются в угловых скобках (<>). Для создания типов параметров шаблона используются ключевые слова typename и class.
- Затем называем тип параметра шаблона (обычно T).
- Если требуется несколько типов параметров шаблона, то они разделяются запятыми.

Экземпляры шаблонов функций

Функция с фактическими типами данных называется **экземпляром шаблона функции (или «объектом шаблона функции»).** Если вы создадите шаблон функции, но не вызовете его, экземпляры этого шаблона созданы не будут.

Якщо шаблона функція використовується з користувацьким класом, то обов'язково потрібно переконатися, що всі оператори всередині цієї функції перевантажені для користувацького класу.

Шаблоны классов

Создание шаблона класса аналогично созданию шаблона функции.

Шаблоны классов работают точно так же, как и шаблоны функций: компилятор копирует шаблон класса, заменяя типы параметров шаблона класса на фактические (передаваемые) типы данных, а затем компилирует эту копию. Если у вас есть шаблон класса, но вы его не используете, то компилятор не будет его даже компилировать. Пример: std::vector — это шаблон класса.

Процесс написания шаблон класса нельзя разбить на файлы .cpp и .h так как мы это делали при написании обычных классов. Все что мы раньше выносили в файл .cpp теперь придется писать в .h ниже. Есть и другой выход: перенести все из .cpp в файл .inl, а затем подключить .inl в нижнюю часть файла .h. Еще один альтернативный вариант — использовать **подход трех файлов**:

- Определение шаблона класса хранится в заголовочном файле.
- Определения методов шаблона класса хранятся в отдельном файле .cpp.
- Затем добавляем третий файл, который содержит все необходимые нам экземпляры шаблона класса.

Параметр non-type в шаблоне

Параметр non-type в шаблоне — это специальный параметр шаблона, который заменяется не типом данных, а конкретным значением. Этим значением может быть:

- целочисленное значение или перечисление;
- указатель или ссылка на объект класса;
- указатель или ссылка на функцию;
- указатель или ссылка на метод класса;
- std::nullptr_t.

Явная специализация шаблона функции

Специализацию шаблона функции (или «полную/явную специализацию шаблона функции») используется для создания отдельной версии функции для значений определенного типа данных.

Всё просто: записываем экземпляр шаблона функции (если функция является методом класса, то делаем это за пределами класса), указывая нужный нам тип данных.

Явная специализация шаблона класса

Специализация шаблона класса (или **«явная специализация шаблона класса»**) позволяет специализировать шаблон класса для работы с определенным типом данных (или сразу с несколькими типами данных, если есть несколько параметров шаблона).

Специализация шаблона класса рассматривается компилятором как полностью отдельный и независимый класс, хоть и выделяется как обычный шаблон класса. Это означает, что мы можем изменить в классе всё что угодно, включая его реализацию/методы/спецификаторы доступа и т.д.

Чтобы создать специализацию класса, пишем ключевое слово template<>, которое сообщает компилятору, что это шаблон, а пустые угловые скобки означают, что нет никаких параметров. А параметров нет из-за того, что мы заменяем единственный параметр шаблона (T, который отвечает за тип данных) конкретным типом данных. Затем мы пишем имя класса и добавляем к нему <тип_данных>, сообщая компилятору, что будем работать с указаным типом. Все остальные изменения — это просто детали реализации класса.

Частичная специализация шаблона

Частичная специализация шаблона позволяет выполнить специализацию шаблона класса (но не функции!), где некоторые (но не все) параметры шаблона явно определены.

Частичная специализация шаблонов и Указатели

Использование частичной специализации шаблона класса для работы с типами указателей особенно полезно, так как позволяет предусмотреть все возможные варианты использования кода на практике.