

Содержание

Об авторе	11
Введение	15
Терминология и соглашения	16
Замечания и предложения	20
От редакции	20
Ждем ваших отзывов!	21
Глава 1. Вывод типов	23
1.1. Вывод типа шаблона	23
1.2. Вывод типа <code>auto</code>	31
1.3. Знакомство с <code>decltype</code>	36
1.4. Как просмотреть выведенные типы	42
Глава 2. Объявление <code>auto</code>	49
2.1. Предпочитайте <code>auto</code> явному объявлению типа	49
2.2. Если <code>auto</code> выводит нежелательный тип, используйте явно типизированный инициализатор	54
Глава 3. Переход к современному C++	61
3.1. Различие между <code>{ }</code> и <code>()</code> при создании объектов	61
3.2. Предпочитайте <code>nullptr</code> значениям <code>0</code> и <code>NULL</code>	69
3.3. Предпочитайте объявление псевдонимов применению <code>typedef</code>	73
3.4. Предпочитайте перечисления с областью видимости перечислениям без таковой	78
3.5. Предпочитайте удаленные функции закрытым неопределенным	84
3.6. Объявляйте перекрывающиеся функции как <code>override</code>	88
3.7. Предпочитайте итераторы <code>const_iterator</code> итераторам <code>iterator</code>	95
3.8. Если функции не генерируют исключений, объявляйте их как <code>noexcept</code>	98
3.9. Используйте, где это возможно, <code>constexpr</code>	105
3.10. Делайте константные функции-члены безопасными в смысле потоков	111
3.11. Генерация специальных функций-членов	116

Глава 4. Интеллектуальные указатели	125
4.1. Используйте <code>std::unique_ptr</code> для управления ресурсами путем исключительного владения	126
4.2. Используйте <code>std::shared_ptr</code> для управления ресурсами путем совместного владения	133
4.3. Используйте <code>std::weak_ptr</code> для <code>std::shared_ptr</code> -подобных указателей, которые могут быть висячими	142
4.4. Предпочитайте использование <code>std::make_unique</code> и <code>std::make_shared</code> непосредственному использованию оператора <code>new</code>	146
4.5. При использовании идиомы указателя на реализацию определяйте специальные функции-члены в файле реализации	155
Глава 5. Rvalue-ссылки, семантика перемещений и прямая передача	165
5.1. Азы <code>std::move</code> и <code>std::forward</code>	166
5.2. Отличие универсальных ссылок от rvalue-ссылок	171
5.3. Используйте <code>std::move</code> для rvalue-ссылок, а <code>std::forward</code> — для универсальных ссылок	176
5.4. Избегайте перегрузок для универсальных ссылок	184
5.5. Знакомство с альтернативами перегрузки для универсальных ссылок	190
Отказ от перегрузки	190
Передача <code>const T&</code>	190
Передача по значению	190
Диспетчеризация дескрипторов	191
Ограничения шаблонов, получающих универсальные ссылки	194
Компромиссы	200
5.6. Свертывание ссылок	202
5.7. Считайте, что перемещающие операции отсутствуют, дороги или не используются	208
5.8. Познакомьтесь с случаями некорректной работы прямой передачи	211
Инициализаторы в фигурных скобках	213
0 и <code>NULL</code> в качестве нулевых указателей	214
Целочисленные члены-данные <code>static const</code> и <code>constexpr</code> без определений	214
Имена перегруженных функций и имена шаблонов	216
Битовые поля	217
Резюме	219
Глава 6. Лямбда-выражения	221
6.1. Избегайте режимов захвата по умолчанию	222
6.2. Используйте инициализирующий захват для перемещения объектов в замыкания	229

6.3. Используйте параметры <code>decltype</code> для <code>auto&&</code> для передачи с помощью <code>std::forward</code>	234
6.4. Предпочитайте лямбда-выражения применению <code>std::bind</code>	237
Глава 7. Параллельные вычисления	245
7.1. Предпочитайте программирование на основе задач программированию на основе потоков	245
7.2. Если важна асинхронность, указывайте <code>std::launch::async</code>	249
7.3. Делайте <code>std::thread</code> неподключаемым на всех путях выполнения	254
7.4. Помните о разном поведении деструкторов дескрипторов потоков	260
7.5. Применяйте фьючерсы <code>void</code> для одноразовых сообщений о событиях	265
7.6. Используйте <code>std::atomic</code> для параллельности, <code>volatile</code> — для особой памяти	272
Глава 8. Тонкости	281
8.1. Рассмотрите передачу по значению для копируемых параметров, которые легко перемещаются и всегда копируются	281
8.2. Рассмотрите применение размещения вместо вставки	291
Предметный указатель	301