

Оглавление

Избранные рекомендации по C++	14
Предисловие	17
Вступление	18
О книге.....	20
Код примеров.....	23
Благодарности	24
Об авторах	26
От издательства	28

ЧАСТЬ I BIKESHEDDING — ЭТО ПЛОХО

Глава 1.1. Р.2. Придерживайтесь стандарта ISO C++	30
Что такое стандарт ISO C++	30
История C++	30
Инкапсуляция вариаций	32
Вариации в окружении времени выполнения	32
Вариации на уровне языка C++ и компилятора	33
Расширения для C++	34
Защита заголовочных файлов	35
Вариации в основных типах	35
Нормативные ограничения	36
Изучение старых способов	37
Обратная совместимость в C++	37
Прямая совместимость и Y2K	38
Следите за последними изменениями в стандарте	39
IsoCpp	39
Конференции	40
Другие источники	40

Глава 1.2. F.51. Если есть выбор, используйте аргументы по умолчанию вместо перегрузки.....	42
Введение.....	42
Доработка ваших абстракций: дополнительные аргументы или перегрузка?	43
Тонкости разрешения перегрузки	45
Вернемся к примеру	47
Однозначная природа аргументов по умолчанию.....	49
Альтернативы перегрузке.....	50
Иногда без перегрузки не обойтись.....	51
Подведем итог	52
Глава 1.3. C.45. Не определяйте конструктор по умолчанию, который просто инициализирует переменные-члены; для этой цели лучше использовать внутриклассовые инициализаторы членов	53
Зачем нужны конструкторы по умолчанию	53
Как инициализируются переменные-члены.....	55
Что может случиться, если поддерживать класс будут два человека	58
Сборная солянка из конструкторов	58
Аргументы по умолчанию могут запутать ситуацию в перегруженных функциях.....	60
Подведем итог	60
Глава 1.4. C.131. Избегайте тривиальных геттеров и сеттеров	62
Архаичная идиома.....	62
Абстракции.....	63
Простая инкапсуляция.....	66
Инварианты класса.....	69
Существительные и глаголы.....	71
Подведем итог	72
Глава 1.5. E5.10. Объявляйте имена по одному в каждом объявлении	73
Позвольте представить.....	73
Обратная совместимость	76
Пишите более ясные объявления.....	77
Структурное связывание	78
Подведем итог	79

Глава 1.6. NR.2. Функции не обязательно должны иметь только один оператор возврата.....	80
Правила меняются	80
Гарантия очистки	83
Идиома RAII	85
Пишите хорошие функции	88
Подведем итог	90

ЧАСТЬ II НЕ НАВРЕДИТЕ СЕБЕ

Глава 2.1. P.11. Инкапсулируйте беспорядочные конструкции, а не разбрасывайте их по всему коду.....	92
Все одним глотком	92
Что означает инкапсулировать запутанную конструкцию	94
Назначение языка и природа абстракции.....	96
Уровни абстракции.....	100
Абстракция путем рефакторинга и проведения линии	101
Подведем итог	102
Глава 2.2. I.23. Минимизируйте число параметров в функциях	103
Сколько они должны получать?	103
Упрощение через абстрагирование	105
Делайте так мало, как возможно, но не меньше	107
Примеры из реальной жизни	109
Подведем итог	111
Глава 2.3. I.26. Если нужен кросс-компилируемый ABI, используйте подмножество в стиле C	112
Создавайте библиотеки.....	112
Что такое ABI.....	114
Сокращайте до абсолютного минимума.....	115
Распространение исключений.....	118
Подведем итог	119
Глава 2.4. C.47. Определяйте и инициализируйте переменные-члены в порядке их объявления.....	121
Подведем итог	131

Глава 2.5. CP.3. Сведите к минимуму явное совместное использование записываемых данных 132

Традиционная модель выполнения..... 132

Подождите, это еще не все..... 134

Предотвращение взаимоблокировок и гонок за данными 137

Отказ от блокировок и мьютексов 140

Подведем итог 143

Глава 2.6. T.120. Используйте метапрограммирование шаблонов, только когда это действительно необходимо 144

std::enable_if => requires..... 152

Подведем итог 156

**ЧАСТЬ III
ПРЕКРАТИТЕ ЭТО ИСПОЛЬЗОВАТЬ**

Глава 3.1. I.11. Никогда не передавайте владение через простой указатель (T*) или ссылку (T&)..... 158

Использование области свободной памяти 158

Производительность интеллектуальных указателей 161

Использование простой семантики ссылок 163

gsl::owner 164

Подведем итог 167

Глава 3.2. I.3. Избегайте синглтонов..... 168

Глобальные объекты — это плохо..... 168

Шаблон проектирования «Синглтон» 169

Фиаско порядка статической инициализации 170

Как скрыть синглтон..... 173

Только один из них должен существовать в каждый момент работы кода 174

Подождите минутку..... 176

Подведем итог 179

Глава 3.3. C.90. Полагайтесь на конструкторы и операторы присваивания вместо memset и memсru 180

В погоне за максимальной производительностью..... 180

Ужасные накладные расходы конструкторов..... 181

Самый простой класс..... 183

О чем говорит стандарт.....	185
А как же метасру?	188
Никогда не позволяйте себе недооценивать компилятор.....	189
Подведем итог	191
Глава 3.4. ES.50. Не приводите переменные с квалификатором const к неконстантному типу	192
Работа с большим количеством данных.....	193
Брандмауэр const	195
Реализация двойного интерфейса	196
Кэширование и отложенные вычисления	198
Два вида const.....	199
Сюрпризы const	201
Подведем итог	202
Глава 3.5. E.28. При обработке ошибок избегайте глобальных состояний (например, errno).....	204
Обрабатывать ошибки сложно	204
Язык C и errno.....	204
Коды возврата.....	206
Исключения	207
<system_error>.....	208
Boost.Outcome	209
Почему обрабатывать ошибки так сложно	210
Свет в конце туннеля	212
Подведем итог	214
Глава 3.6. SF.7. Не используйте using namespace в глобальной области видимости в заголовочном файле	215
Не делайте этого	215
Неоднозначность	216
Использование using.....	217
Куда попадают символы	219
Еще более коварная проблема.....	222
Решение проблемы операторов разрешения области видимости	223
Искушение и расплата	225
Подведем итог	226

ЧАСТЬ IV ИСПОЛЬЗУЙТЕ НОВУЮ ОСОБЕННОСТЬ ПРАВИЛЬНО

Глава 4.1.	F.21. Для возврата нескольких выходных значений используйте структуры или кортежи.....	228
	Форма сигнатуры функции	228
	Документирование и аннотирование	230
	Теперь можно вернуть объект	231
	Можно также вернуть кортеж	234
	Передача и возврат по неконстантной ссылке	237
	Подведем итог	240
Глава 4.2.	Enum.3. Старайтесь использовать классы-перечисления вместо простых перечислений.....	241
	Константы.....	241
	Перечисления с заданной областью видимости.....	244
	Базовый тип	246
	Неявное преобразование	247
	Подведем итог	249
Глава 4.3.	ES.5. Минимизируйте области видимости.....	250
	Природа области видимости	250
	Область видимости блока	251
	Область видимости пространства имен.....	253
	Область видимости класса.....	256
	Область видимости параметров функции.....	258
	Область видимости перечисления	259
	Область действия параметра шаблона.....	260
	Область видимости как контекст	261
	Подведем итог	262
Глава 4.4.	Con.5. Используйте <code>constexpr</code> для определения значений, которые можно вычислить на этапе компиляции	263
	От <code>const</code> к <code>constexpr</code>	263
	C++ по умолчанию.....	265
	Использование <code>constexpr</code>	267
	<code>inline</code>	271
	<code>constexpr</code>	272

constinit	273
Подведем итог	275
Глава 4.5. Т.1. Используйте шаблоны для повышения уровня абстрактности кода	276
Повышение уровня абстракции	278
Шаблоны функций и абстракция	280
Шаблоны классов и абстракция	283
Выбор имени — сложная задача	285
Подведем итог	286
Глава 4.6. Т.10. Задавайте концепции для всех аргументов шаблона	287
Как мы здесь оказались?	287
Ограничение параметров	290
Как абстрагировать свои концепции	293
Разложение на составляющие через концепции	296
Подведем итог	297

ЧАСТЬ V ПИШИТЕ ХОРОШИЙ КОД ПО УМОЛЧАНИЮ

Глава 5.1. P.4. В идеале программа должна быть статически типобезопасной	300
Безопасность типов — это средство защиты в C++	300
Объединения	302
Приведение	304
Целые без знака	307
Буферы и размеры	310
Подведем итог	311
Глава 5.2. P.10. Неизменяемые данные предпочтительнее изменяемых	312
Неправильные значения по умолчанию	312
const в объявлениях функций	315
Подведем итог	319
Глава 5.3. I.30. Инкапсулируйте нарушения правил	320
Скрытие неприглядных вещей	320
Поддержание видимости, что все в порядке	322
Подведем итог	327

Глава 5.4. ES.22. Не объявляйте переменные, пока не получите значения для их инициализации	329
Важность выражений и операторов	329
Объявление в стиле C.....	330
Объявление с последующей инициализацией	332
Максимальное откладывание объявления.....	333
Локализация контекстно зависимой функциональности	335
Устранение состояния.....	337
Подведем итог	339
Глава 5.5. Per.7. При проектировании учитывайте возможность последующей оптимизации	340
Максимальная частота кадров.....	340
Работа вдалеке от железа	342
Оптимизация через абстракцию.....	346
Подведем итог	349
Глава 5.6. E.6. Используйте идиому RAII для предотвращения утечек памяти.....	350
Детерминированное уничтожение.....	350
Утечка файлов	353
Почему это так важно	356
Все это выглядит чересчур сложным: будущие возможности	358
Где все это получить.....	361
Заключение	364
Послесловие.....	366