

# Содержание

---

<b>Предисловие</b>	9
<b>Вопросы организации и стратегии</b>	13
0. Не мелочитесь, или Что не следует стандартизировать	14
1. Компилируйте без замечаний при максимальном уровне предупреждений	16
2. Используйте автоматические системы сборки программ	19
3. Используйте систему контроля версий	20
4. Одна голова хорошо, а две — лучше	21
<b>Стиль проектирования</b>	23
5. Один объект — одна задача	24
6. Главное — корректность, простота и ясность	25
7. Кодирование с учетом масштабируемости	27
8. Не оптимизируйте преждевременно	29
9. Не pessимизируйте преждевременно	31
10. Минимизируйте глобальные и совместно используемые данные	32
11. Скрытие информации	33
12. Кодирование параллельных вычислений	34
13. Ресурсы должны быть во владении объектов	37
<b>Стиль кодирования</b>	39
14. Предпочитайте ошибки компиляции и компоновки ошибкам времени выполнения	40
15. Активно используйте const	42
16. Избегайте макросов	44
17. Избегайте магических чисел	46
18. Объявляйте переменные как можно локальнее	47
19. Всегда инициализируйте переменные	48
20. Избегайте длинных функций и глубокой вложенности	50
21. Избегайте зависимостей инициализаций между единицами компиляции	52
22. Минимизируйте зависимости определений и избегайте циклических зависимостей	53
23. Делайте заголовочные файлы самодостаточными	55
24. Используйте только внутреннюю, но не внешнюю защиту директивы #include	56
<b>Функции и операторы</b>	57
25. Передача параметров по значению, (интеллектуальному) указателю или ссылке	58
26. Сохраняйте естественную семантику перегруженных операторов	59
27. Отдавайте предпочтение каноническим формам арифметических операторов и операторов присваивания	60
28. Предпочитайте канонический вид ++ и --, и вызов префиксных операторов	62
29. Используйте перегрузку, чтобы избежать неявного преобразования типов	64
30. Избегайте перегрузки &&,    и , (запятой)	65
31. Не пишите код, который зависит от порядка вычислений аргументов функции	67
<b>Проектирование классов и наследование</b>	69
32. Ясно представляйте, какой вид класса вы создаете	70
33. Предпочитайте минимальные классы монолитным	72
34. Предпочитайте композицию наследованию	73

35. Избегайте наследования от классов, которые не спроектированы для этой цели	75
36. Предпочитайте предоставление абстрактных интерфейсов	77
37. Открытое наследование означает заменимость. Наследовать надо не для повторного использования, а чтобы быть повторно использованным	79
38. Практикуйте безопасное перекрытие	81
39. Виртуальные функции стоит делать неоткрытыми, а открытые — не виртуальными	83
40. Избегайте возможностей неявного преобразования типов	85
41. Делайте данные-члены закрытыми (кроме случая агрегатов в стиле структур C)	87
42. Не допускайте вмешательства во внутренние дела	89
43. Разумно пользуйтесь идиомой Pimpl	91
44. Предпочитайте функции, которые не являются ни членами, ни друзьями	94
45. new и delete всегда должны разрабатываться вместе	95
46. При наличии пользовательского new следует предоставлять все стандартные типы этого оператора	97
<b>Конструкторы, деструкторы и копирование</b>	99
47. Определяйте и инициализируйте переменные-члены в одном порядке	100
48. В конструкторах предпочитайте инициализацию присваиванию	101
49. Избегайте вызовов виртуальных функций в конструкторах и деструкторах	102
50. Делайте деструкторы базовых классов открытыми и виртуальными либо защищенными и не виртуальными	104
51. Деструкторы, функции освобождения ресурсов и обмена не ошибаются	106
52. Копируйте и ликвидируйте согласованно	108
53. Явно разрешайте или запрещайте копирование	109
54. Избегайте срезки. Подумайте об использовании в базовом классе клонирования вместо копирования	110
55. Предпочитайте канонический вид присваивания	113
56. Обеспечьте бессбойную функцию обмена	114
<b>Пространства имен и модули</b>	117
57. Храните типы и их свободный интерфейс в одном пространстве имен	118
58. Храните типы и функции в разных пространствах имен, если только они не предназначены для совместной работы	120
59. Не используйте using для пространств имен в заголовочных файлах или перед директивой #include	122
60. Избегайте выделения и освобождения памяти в разных модулях	125
61. Не определяйте в заголовочном файле объекты со связыванием	126
62. Не позволяйте исключениям пересекать границы модулей	128
63. Используйте достаточно переносимые типы в интерфейсах модулей	130
<b>Шаблоны и обобщенность</b>	133
64. Разумно сочетайте статический и динамический полиморфизм	134
65. Выполняйте настройку явно и преднамеренно	136
66. Не специализируйте шаблоны функций	140
67. Пишите максимально обобщенный код	142
<b>Обработка ошибок и исключения</b>	143
68. Широко применяйте assert для документирования внутренних допущений и инвариантов	144
69. Определите разумную стратегию обработки ошибок и строго ей следуйте	146
70. Отличайте ошибки от ситуаций, не являющихся ошибками	148

71. Проектируйте и пишите безопасный в отношении ошибок код	151
72. Для уведомления об ошибках следует использовать исключения	154
73. Генерируйте исключения по значению, перехватывайте — по ссылке	158
74. Уведомляйте об ошибках, обрабатывайте и преобразовывайте их там, где следует	159
75. Избегайте спецификаций исключений	160
<b>STL: контейнеры</b>	<b>163</b>
76. По умолчанию используйте vector. В противном случае выбирайте контейнер, соответствующий задаче	164
77. Вместо массивов используйте vector и string	166
78. Используйте vector (и string::c_str) для обмена данными с API на других языках	167
79. Храните в контейнерах только значения или интеллектуальные указатели	168
80. Предпочитайте push_back другим способом расширения последовательности	169
81. Предпочитайте операции с диапазонами операциям с отдельными элементами	170
82. Используйте подходящие идиомы для реального уменьшения емкости контейнера и удаления элементов	171
<b>STL: алгоритмы</b>	<b>173</b>
83. Используйте отлаченную реализацию STL	174
84. Предпочитайте вызовы алгоритмов самостоятельно разрабатываемым циклам	176
85. Пользуйтесь правильным алгоритмом поиска	179
86. Пользуйтесь правильным алгоритмом сортировки	180
87. Делайте предикаты чистыми функциями	182
88. В качестве аргументов алгоритмов и компараторов лучше использовать функциональные объекты, а не функции	184
89. Корректно пишите функциональные объекты	186
<b>Безопасность типов</b>	<b>187</b>
90. Избегайте явного выбора типов — используйте полиморфизм	188
91. Работайте с типами, а не с представлениями	190
92. Избегайте reinterpret_cast	192
93. Избегайте применения static_cast к указателям	193
94. Избегайте преобразований, отменяющих const	194
95. Не используйте преобразование типов в стиле C	195
96. Не применяйте memstr или memstr к не-POD типам	197
97. Не используйте объединения для преобразований	198
98. Не используйте неизвестные аргументы (троеточия)	199
99. Не используйте недействительные объекты и небезопасные функции	200
100. Не рассматривайте массивы полиморфно	201
<b>Список литературы</b>	<b>202</b>
<b>Резюме из резюме</b>	<b>206</b>
<b>Предметный указатель</b>	<b>220</b>