

УДК 004.4
ББК 32.973.202
Г84

Гримм Р.

Г84 Параллельное программирование на современном C++ / пер. с англ. В. Ю. Винника. – М.: ДМК Пресс, 2022. – 616 с.: ил.

ISBN 978-5-97060-957-6

Книга во всех подробностях освещает параллельное программирование на современном C++. Особое внимание уделено опасностям и трудностям параллельного программирования (например, гонке данных и мертвой блокировке) и способам борьбы с ними. Приводятся многочисленные примеры кода, позволяющие читателю легко закрепить теорию на практических примерах.

Издание адресовано читателям, которые хотят освоить параллельное программирование на одном из наиболее распространенных языков.

УДК 004.4
ББК 32.973.202

Copyright Concurrency with Modern C++ published by Rainer Grimm. Copyright ©2020 Rainer Grimm

Все права защищены. Любая часть этой книги не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами без письменного разрешения владельцев авторских прав.

ISBN 978-5-97060-957-6 (рус.)

© Rainer Grimm, 2020
© Перевод, оформление, издание,
ДМК Пресс, 2022

Дизайн обложки разработан с использованием ресурса freepik.com.

Содержание

От издательства	17
Введение	18
КРАТКИЙ ОБЗОР	22
1. Параллельное программирование и современный язык C++	23
1.1. Стандарты C++ 11 и C++ 14: закладка фундамента	24
1.1.1. Модели памяти.....	24
1.1.1.1. Атомарные переменные.....	25
1.1.2. Управление потоками	25
1.1.2.1. Классы для поддержки потоков	25
1.1.2.2. Данные в совместном доступе	26
1.1.2.3. Локальные данные потока	27
1.1.2.4. Переменные условия	27
1.1.2.5. Кооперативное прерывание потоков (стандарт C++ 20)	28
1.1.2.6. Семафоры (стандарт C++ 20).....	28
1.1.2.7. Защёлки и барьеры (стандарт C++ 20)	28
1.1.2.8. Задания	28
1.1.2.9. Синхронизированные потоки вывода (стандарт C++ 20).....	29
1.2. Стандарт C++ 17. Параллельные алгоритмы в стандартной библиотеке	29
1.2.1. Политики выполнения.....	30
1.2.2. Новые параллельные алгоритмы	30
1.3. Сопрограммы в стандарте C++ 20.....	30
1.4. Учебные примеры	31
1.4.1. Вычисление суммы элементов вектора	31
1.4.2. Потокобезопасное создание объекта-одиночки.....	31
1.4.3. Поэтапная оптимизация с использованием инструмента CppMem.....	31
1.4.4. Быстрая синхронизация потоков.....	31
1.4.5. Вариации на тему фьючерсов	31
1.4.6. Модификации и обобщения генераторов.....	32
1.4.7. Способы управления заданиями	32
1.5. Будущее языка C++.....	32
1.5.1. Исполнители.....	32
1.5.2. Расширенные фьючерсы	33
1.5.3. Транзакционная память	33
1.5.4. Блоки заданий.....	33
1.5.5. Библиотека для векторных вычислений	34
1.6. Шаблоны и эмпирические правила.....	34

1.6.1. Шаблоны синхронизации	34
1.6.2. Шаблоны параллельной архитектуры	34
1.6.3. Эмпирические правила.....	35
1.7. Структуры данных.....	35
1.8. Сложности параллельного программирования	35
1.9. Библиотека для работы со временем	35
1.10. Обзор инструментального средства SprMet.....	35
1.11. Пояснение некоторых терминов	36

ПАРАЛЛЕЛЬНОЕ ПРОГРАММИРОВАНИЕ

В ПОДРОБНОСТЯХ	37
-----------------------------	-----------

2. Модель памяти

2.1. Начальное представление о модели памяти	38
2.1.1. Что такое область памяти?	39
2.1.2. Что происходит, когда два потока обращаются к одной области памяти	39
2.2. Модель памяти как контракт	40
2.2.1. Основы.....	42
2.2.2. Трудности.....	42
2.3. Атомарные переменные	43
2.3.1. Отличие сильной модели памяти от слабой.....	44
2.3.1.1. Сильная модель памяти	44
2.3.1.2. Слабая модель памяти	46
2.3.2. Атомарный флаг.....	47
2.3.2.1. Циклическая блокировка	48
2.3.2.2. Сравнение циклической блокировки с мьютексом	50
2.3.2.3. Синхронизация потоков.....	53
2.3.3. Шаблон <code>std::atomic</code>	54
2.3.3.1. Фундаментальный атомарный интерфейс	55
2.3.3.2. Атомарные типы с плавающей точкой в стандарте C++ 20.....	66
2.3.3.3. Атомарный тип указателя	67
2.3.3.4. Атомарные целочисленные типы	67
2.3.3.5. Псевдонимы типов	70
2.3.4. Функции-члены атомарных типов	71
2.3.5. Свободные функции над атомарными типами	73
2.3.5.1. Особенности типа <code>std::shared_ptr</code> (до стандарта C++ 20).....	74
2.3.6. Шаблон класса <code>std::atomic_ref</code> в стандарте C++ 20	76
2.3.6.1. Мотивация	76
2.3.6.2. Специализации шаблона <code>std::atomic_ref</code>	80
2.3.6.3. Полный список атомарных операций.....	82
2.4. Синхронизация и порядок доступа к памяти.....	83
2.4.1. Шесть вариантов модели памяти в языке C++	83
2.4.1.1. Виды атомарных операций.....	84
2.4.1.2. Ограничения на синхронизацию и порядок доступа	85
2.4.2. Последовательно-согласованное выполнение	86

2.4.3. Семантика захвата и освобождения	88
2.4.3.1. Транзитивность	90
2.4.3.2. Типичное недоразумение	93
2.4.3.3. Последовательность освобождений	97
2.4.4. Модель памяти <code>std::memory_order_consume</code>	99
2.4.4.1. Порядок захвата и освобождения	100
2.4.4.2. Порядок освобождения и потребления	101
2.4.4.3. Различие порядков «освобождение-захват» и «освобождение-потребление»	102
2.4.4.4. Зависимости данных в модели <code>std::memory_order_consume</code>	102
2.4.5. Ослабленная семантика	104
2.4.5.1. Отсутствие ограничений на синхронизацию и порядок операций	104
2.5. Барьеры	106
2.5.1. Барьер <code>std::atomic_thread_fence</code>	106
2.5.1.1. Что такое барьеры памяти	106
2.5.1.2. Три барьера	107
2.5.1.3. Барьеры захвата и освобождения	109
2.5.1.4. Синхронизация с использованием атомарных переменных и барьеров	111
2.5.2. Барьер <code>std::atomic_signal_fence</code>	116
3. Управление потоками	117
3.1. Базовые потоки: класс <code>std::thread</code>	117
3.1.1. Создание потока	118
3.1.2. Время жизни потоков	119
3.1.2.1. Функции <code>join</code> и <code>detach</code>	120
3.1.3. Передача аргументов при создании потока	122
3.1.3.1. Передача по значению и по ссылке	122
3.1.4. Перечень функций-членов	125
3.2. Усовершенствованные потоки: класс <code>std::jthread</code> (стандарт C++ 20)	129
3.2.1. Автоматическое присоединение к потоку	129
3.2.2. Прерывание по запросу в классе <code>std::jthread</code>	131
3.3. Данные в совместном доступе	133
3.3.1. Мьютексы	134
3.3.1.1. Затруднения с мьютексами	138
3.3.2. Блокировщики	141
3.3.2.1. Тип <code>std::lock_guard</code>	141
3.3.2.2. Тип <code>std::scoped_lock</code>	142
3.3.2.3. Тип <code>std::unique_lock</code>	143
3.3.2.4. Блокировщик <code>std::shared_lock</code>	144
3.3.3. Функция <code>std::lock</code>	148
3.3.4. Потокобезопасная инициализация	151
3.3.4.1. Константные выражения	151
3.3.4.2. Функция <code>std::call_once</code> и флаг <code>std::once_flag</code>	152
3.3.4.3. Локальные статические переменные	156
3.4. Данные с потоковой длительностью хранения	157

3.5. Переменные условия	160
3.5.1. Использование предиката в функции ожидания	163
3.5.2. Утерянные и ложные пробуждения	164
3.5.3. Процедура ожидания	165
3.6. Кооперативное прерывание потоков (стандарт C++ 20)	166
3.6.1. Класс <code>std::stop_source</code>	167
3.6.2. Класс <code>std::stop_token</code>	168
3.6.3. Класс <code>std::stop_callback</code>	169
3.6.4. Общий механизм послылки сигналов	172
3.6.5. Особенности класса <code>std::jthread</code>	175
3.6.6. Новые перегрузки функции <code>wait</code> в классе <code>std::condition_variable_any</code>	175
3.7. Семафоры (стандарт C++ 20)	178
3.8. Зашёлки и барьеры (стандарт C++ 20)	182
3.8.1. Класс <code>std::latch</code>	182
3.8.2. Класс <code>std::barrier</code>	187
3.9. Асинхронные задания	190
3.9.1. Отличие заданий от потоков	191
3.9.2. Функция <code>std::async</code>	192
3.9.2.1. Политика запуска	193
3.9.2.2. Запустить и забыть	195
3.9.2.3. Параллельное вычисление скалярного произведения	196
3.9.3. Тип <code>std::packaged_task</code>	198
3.9.4. Типы <code>std::promise</code> и <code>std::future</code>	203
3.9.4.1. Тип <code>std::promise</code>	205
3.9.4.2. Тип <code>std::future</code>	205
3.9.5. Тип <code>std::shared_future</code>	207
3.9.6. Обработка исключений в асинхронных заданиях	211
3.9.7. Оповещения	214
3.10. Синхронизированные потоки вывода (стандарт C++ 20)	216
3.11. Краткие итоги	223
4. Параллельные алгоритмы в стандартной библиотеке	225
4.1. Политики выполнения	226
4.1.1. Параллельное и векторизованное выполнение	227
4.1.1.1. Код без оптимизации	228
4.1.1.2. Максимальная оптимизация	228
4.1.2. Обработка исключений	228
4.1.3. Опасность гонок данных и мёртвых блокировок	230
4.2. Алгоритмы стандартной библиотеки	231
4.3. Новые параллельные алгоритмы	232
4.3.1. Новые перегрузки	237
4.3.2. Наследие функционального программирования	237
4.4. Поддержка в различных компиляторах	239
4.4.1. Компилятор Microsoft Visual Compiler	239
4.4.2. Компилятор GCC	240
4.4.3. Будущие реализации параллельных стандартных алгоритмов	240

4.5. Вопросы производительности	241
4.5.1. Компилятор Microsoft Visual Compiler	243
4.5.2. Компилятор GCC	244
4.6. Краткие итоги	244

5. Сопрограммы в стандарте C++ 20

5.1. Функция-генератор	247
5.2. Особенности сопрограмм	249
5.2.1. Типичные сценарии использования	249
5.2.2. Разновидности сопрограмм	249
5.2.3. Требования к сопрограммам	250
5.2.4. Преобразование функции в сопрограмму	250
5.2.4.1. Ограничения	251
5.3. Концептуальная модель	251
5.3.1. Объект-обещание	252
5.3.2. Дескриптор сопрограммы	252
5.3.3. Кадр сопрограммы	254
5.4. Ожидание отложенного вычисления	254
5.4.1. Прообраз ожидания	254
5.4.2. Общие требования к контроллерам ожидания	255
5.4.3. Стандартные контроллеры ожидания	255
5.4.4. Функция <code>initial_suspend</code>	256
5.4.5. Функция <code>final_suspend</code>	256
5.4.6. Получение контроллера ожидания	257
5.5. Процесс функционирования сопрограммы	258
5.5.1. Управление обещанием	258
5.5.2. Управление ожиданием	259
5.6. Оператор <code>co_return</code> и жадный фьючерс	261
5.7. Оператор <code>co_yield</code> и бесконечный поток данных	263
5.8. Оператор <code>co_await</code>	266
5.8.1. Запуск задания по запросу	266
5.9. Синхронизация потоков	268
5.10. Краткие итоги	273

6. Учебные примеры

6.1. Вычисление суммы элементов вектора	274
6.1.1. Суммирование элементов вектора в одном потоке	274
6.1.1.1. Суммирование в цикле по диапазону	275
6.1.1.2. Суммирование алгоритмом <code>std::accumulate</code>	276
6.1.1.3. Использование блокировщика	277
6.1.1.4. Использование атомарной переменной	278
6.1.1.5. Сводные данные по однопоточным алгоритмам	280
6.1.2. Многопоточное суммирование с общей переменной	281
6.1.2.1. Использование блокировщика	281
6.1.2.2. Использование атомарной переменной	283
6.1.2.3. Использование атомарной переменной с функцией <code>fetch_add</code>	285

6.1.2.4. Использование ослабленной семантики	286
6.1.2.5. Сводные данные по алгоритмам с общей переменной.....	287
6.1.3. Раздельное суммирование в потоках	287
6.1.3.1. Использование локальной переменной	287
6.1.3.2. Использование переменных с потоковым временем жизни	292
6.1.3.3. Использование асинхронных заданий	294
6.1.3.4. Сводные данные	296
6.1.4. Суммирование вектора: подведение итогов.....	297
6.1.4.1. Однопоточные алгоритмы	297
6.1.4.2. Многопоточные алгоритмы с общей переменной.....	297
6.1.4.3. Многопоточные алгоритмы с локальными переменными.....	297
6.2. Потокобезопасное создание объекта-одиночки	299
6.2.1. Шаблон «Блокировка с двойной проверкой»	300
6.2.2. Измерение производительности.....	301
6.2.3. Потокобезопасный вариант реализации Мейерса	304
6.2.4. Реализации на основе блокировщика	305
6.2.5. Реализация на основе функции <code>std::call_once</code>	307
6.2.6. Решение на основе атомарных переменных.....	308
6.2.6.1. Семантика последовательной согласованности.....	308
6.2.6.2. Семантика захвата и освобождения.....	310
6.2.7. Сводные данные.....	312
6.3. Поэтапная оптимизация с использованием инструмента CppMem	312
6.3.1. Неатомарные переменные.....	314
6.3.1.1. Анализ программы.....	315
6.3.2. Анализ программы с блокировкой	320
6.3.3. Атомарные переменные с последовательной согласованностью	321
6.3.3.1. Анализ программы инструментом CppMem.....	322
6.3.3.2. Последовательность операций.....	326
6.3.4. Атомарные переменные с семантикой захвата и освобождения	327
6.3.4.1. Анализ программы инструментом CppMem.....	329
6.3.5. Смесь атомарных и неатомарных переменных.....	331
6.3.5.1. Анализ программы инструментом CppMem.....	332
6.3.6. Атомарные переменные с ослабленной семантикой.....	333
6.3.6.1. Анализ инструментом CppMem	334
6.3.7. Итоги	335
6.4. Быстрая синхронизация потоков	335
6.4.1. Переменные условия.....	336
6.4.2. Решение на основе атомарного флага	338
6.4.2.1. Решение с двумя флагами	338
6.4.2.2. Решение с одним атомарным флагом.....	340
6.4.3. Решение на основе атомарной логической переменной	341
6.4.4. Реализация на семафорах.....	343
6.4.5. Сравнительный анализ	345
6.5. Вариации на тему фьючерсов	345
6.5.1. Ленивый фьючерс.....	348
6.5.2. Выполнение сопрограммы в отдельном потоке	351
6.6. Модификации и обобщения генераторов	355

6.6.1. Модификации программы	358
6.6.1.1. Если сопрограмму не пробуждать	358
6.6.1.2. Сопрограмма не приостанавливается на старте	359
6.6.1.3. Сопрограмма не приостанавливается при выдаче значения.....	360
6.6.2. Обобщение	361
6.7. Способы управления заданиями	364
6.7.1. Функционирование контроллера ожидания	364
6.7.2. Автоматическое возобновление работы	367
6.7.3. Автоматическое пробуждение сопрограммы в отдельном потоке	370
6.8. Краткие итоги	373
7. Будущее языка C++	374
7.1 Исполнители	374
7.1.1. Долгий путь исполнителя	375
7.1.2. Что такое исполнитель	376
7.1.2.1. Свойства исполнителя	376
7.1.3. Первые примеры	377
7.1.3.1. Использование исполнителя.....	377
7.1.3.2. Получение исполнителя	378
7.1.4. Цели разработки исполнителей.....	379
7.1.5. Терминология	380
7.1.6. Функции выполнения.....	381
7.1.6.1. Единичная кардинальность	382
7.1.6.2. Множественная кардинальность.....	382
7.1.6.3. Проверка требований к исполнителю	382
7.1.7. Простой пример использования	383
7.2. Расширенные фьючерсы	386
7.2.1. Техническая спецификация	386
7.2.1.1. Обновлённое понятие фьючерса.....	386
7.2.1.2. Средства асинхронного выполнения.....	388
7.2.1.3. Создание новых фьючерсов	388
7.2.2. Унифицированные фьючерсы.....	391
7.2.2.1. Недостатки фьючерсов	391
7.2.2.2. Пять новых концептов	394
7.2.2.3. Направления дальнейшей работы	395
7.3. Транзакционная память.....	396
7.3.1. Требования ACI(D).....	396
7.3.2. Синхронизированные и атомарные блоки	397
7.3.2.1. Синхронизированные блоки.....	397
7.3.2.2. Атомарные блоки	400
7.3.3. Транзакционно-безопасный и транзакционно-небезопасный код.....	401
7.4. Блоки заданий.....	401
7.4.1. Разветвление и слияние.....	402
7.4.2. Две функции для создания блоков заданий.....	403
7.4.3. Интерфейс	404
7.4.4. Планировщик заданий	404
7.5. Библиотека для векторных вычислений.....	405

7.5.1. Векторные типы данных.....	406
7.5.2. Интерфейс векторизированных данных.....	406
7.5.2.1. Вспомогательные типы-признаки	406
7.5.2.2. Выражения над значениями векторного типа.....	407
7.5.2.3. Приведение типов	407
7.5.2.4. Алгоритмы над векторизированными значениями	407
7.5.2.5. Свёртка по операции.....	408
7.5.2.6. Свёртка с маской.....	408
7.5.2.7. Классы свойств	408
7.6. Итоги	409

8. Шаблоны и эмпирические правила 410

8.1. История понятия.....	410
8.2. Неоценимая польза шаблонов	412
8.3. Шаблоны или эмпирические правила	413
8.4. Антишаблоны	413
8.5. Итоги	414

9. Шаблоны синхронизации..... 415

9.1. Управление общим доступом.....	415
9.1.1. Копирование значения	416
9.1.1.1. Гонка данных при передаче по ссылке	416
9.1.1.2. Проблемы со временем жизни объектов, передаваемых по ссылке	419
9.1.1.3. Материал для дальнейшего изучения	421
9.1.2. Потокковая область хранения.....	421
9.1.2.1. Материал для дальнейшего изучения	422
9.1.3. Использование фьючерсов.....	422
9.1.3.1. Материал для дальнейшего изучения	423
9.2. Управление изменяемым состоянием.....	423
9.2.1. Локальные блокировщики	424
9.2.1.1. Материал для дальнейшего изучения	426
9.2.2. Параметризованные блокировщики	426
9.2.2.1. Шаблон «Стратегия».....	426
9.2.2.2. Реализация параметризованных блокировщиков	428
9.2.2.3. Материал для дальнейшего изучения	434
9.2.3. Потокобезопасный интерфейс	434
9.2.3.1. Тонкости потокобезопасных интерфейсов	437
9.2.3.2. Материал для дальнейшего изучения	440
9.2.4. Охраняемая приостановка	440
9.2.4.1. Принцип вталкивания и принцип втягивания.....	441
9.2.4.2. Ограниченное и неограниченное ожидания	442
9.2.4.3. Оповещение одного или всех ожидающих потоков	443
9.2.4.4. Материал для дальнейшего изучения	446
9.3. Краткие итоги.....	446

10. Шаблоны параллельной архитектуры	447
10.1. Активный объект.....	448
10.1.1. Компоненты шаблона	448
10.1.2. Преимущества и недостатки активных объектов.....	450
10.1.3. Реализация.....	451
10.1.3.1. Материал для дальнейшего изучения.....	457
10.2. Объект-монитор.....	457
10.2.1. Требования	458
10.2.2. Компоненты.....	458
10.2.3. Принцип действия монитора	459
10.2.3.1. Преимущества и недостатки мониторов.....	459
10.2.3.2. Реализация монитора.....	460
10.2.3.3. Материал для дальнейшего изучения.....	464
10.3. Полусинхронная архитектура	464
10.3.1. Преимущества и недостатки.....	466
10.3.2. Шаблон «Реактор»	466
10.3.2.1. Требования	466
10.3.2.2. Решение	467
10.3.2.3. Компоненты	467
10.3.2.4. Преимущества и недостатки	468
10.3.3. Проактор	469
10.3.3.1. Требования	469
10.3.3.2. Решение	469
10.3.3.3. Компоненты	470
10.3.3.4. Преимущества и недостатки	471
10.3.4. Материал для дальнейшего изучения	472
10.4. Краткие итоги.....	472
11. Эмпирические правила	473
11.1. Общие правила	473
11.1.1. Рецензирование кода	473
11.1.2. Сведение к минимуму совместного доступа к изменяемым данным	475
11.1.3. Минимизация ожидания	477
11.1.4. Предпочтительное использование неизменяемых данных	478
11.1.4.1. Пользовательские типы данных и константы этапа компиляции.....	479
11.1.5. Использование чистых функций.....	481
11.1.6. Отыскание правильных абстракций.....	482
11.1.7. Использование статических анализаторов кода.....	483
11.1.8. Использование динамических анализаторов	483
11.2. Работа с потоками.....	484
11.2.1. Общие вопросы многопоточного программирования	484
11.2.1.1. Создание как можно меньшего числа потоков.....	484
11.2.1.2. Использование заданий вместо потоков.....	487
11.2.1.3. Особая осторожность при отсоединении потока	488

11.2.1.4. Предпочтительность потоков с автоматическим присоединением.....	488
11.2.2. Управление доступом к данным.....	489
11.2.2.1. Передача данных по значению	489
11.2.2.2. Использование умного указателя для совместного владения данными	489
11.2.2.3. Сокращение времени блокировки.....	492
11.2.2.4. Обёртывание мьютекса в блокировщик	493
11.2.2.5. Предпочтительный захват одного мьютекса	493
11.2.2.6. Необходимость давать блокировщикам имена	494
11.2.2.7. Атомарный захват нескольких мьютексов	495
11.2.2.8. Не вызывать неизвестный код под блокировкой	496
11.2.3. Переменные условия	497
11.2.3.1. Обязательное использование предиката.....	497
11.2.3.2. Замена переменных условия обещаниями и фьючерсами.....	499
11.2.4. Обещания и фьючерсы	500
11.2.4.1. Предпочтительность асинхронных заданий.....	500
11.3. Модель памяти	500
11.3.1. Недопустимость volatile-переменных для синхронизации	501
11.3.1.1. Совет избегать неблокирующего программирования.....	501
11.3.2. Использование шаблонов неблокирующего программирования....	501
11.3.3. Использование гарантий, предоставляемых языком	501
11.3.4. Не нужно изобретать велосипед	502
11.3.4.1. Библиотека Boost.Lockfree	502
11.3.4.2. Библиотека CDS	502
11.4. Краткие итоги.....	503

СТРУКТУРЫ ДАННЫХ..... 504

12. Структуры данных с блокировками	505
12.1. Общие соображения	505
12.1.1. Стратегии блокировки	506
12.1.2. Гранулярность интерфейса	508
12.1.3. Типовые сценарии использования	510
12.1.3.1. Производительность в ОС Linux	516
12.1.3.2. Производительность в ОС Windows.....	517
12.1.4. Избегание прорех	517
12.1.5. Конкуренция потоков	520
12.1.5.1. Суммирование в один поток без синхронизации	520
12.1.5.2. Суммирование в один поток с синхронизацией	522
12.1.5.3. Анализ результатов измерений	523
12.1.6. Масштабируемость.....	523
12.1.7. Инварианты	525
12.1.8. Исключения	528
12.2. Потокобезопасный стек	528
12.2.1. Упрощённая реализация	529

12.2.2. Полная реализация.....	531
12.3. Потокбезопасная очередь.....	535
12.3.1. Блокировка очереди целиком.....	536
12.3.2. Раздельная блокировка концов очереди.....	538
12.3.2.1. Некорректная реализация	538
12.3.2.2. Простая реализация очереди.....	539
12.3.2.3. Очередь с фиктивным элементом	542
12.3.2.4. Окончательная реализация	544
12.3.2.5. Ожидание значения из очереди.....	547
12.4. Краткие итоги	550

ДОПОЛНИТЕЛЬНЫЕ СВЕДЕНИЯ 551

13. Сложности параллельного программирования 552

13.1. Проблема АВА.....	552
13.1.1. Наглядное объяснение	552
13.1.2. Некритические случаи эффекта АВА.....	553
13.1.3. Неблокирующая структура данных.....	554
13.1.4. Эффект АВА в действии	554
13.1.5. Исправление эффекта АВА.....	555
13.1.5.1. Ссылка на помеченное состояние.....	555
13.1.5.2. Сборка мусора	555
13.1.5.3. Списки опасных указателей	555
13.1.5.4. Механизм чтения-копирования-модификации.....	556
13.2. Тонкости блокировок	556
13.3. Нарушение инварианта программы	558
13.4. Гонка данных	560
13.5. Мёртвые блокировки.....	561
13.6. Неявные связи между данными.....	563
13.7. Проблемы со временем жизни объектов	566
13.8. Перемещение потоков	567
13.9. Состояние гонки.....	569

14. Библиотека для работы со временем 570

14.1. Взаимосвязь моментов, промежутков времени и часов.....	570
14.2. Моменты времени	571
14.2.1. Перевод моментов времени в календарный формат.....	572
14.2.2. Выход за пределы допустимого диапазона часов.....	573
14.3. Промежутки времени.....	575
14.3.1. Вычисления с промежутками времени	577
14.4. Типы часов	579
14.4.1. Точность и монотонность часов	579
14.4.2. Нахождение точки отсчёта часов	582
14.5. Приостановка и ограниченное ожидание	584
14.5.1. Соглашения об именовании.....	584
14.5.2. Стратегии ожидания	585

15. Обзор инструментального средства СppMet	591
15.1. Упрощённое введение.....	591
15.1.1. Выбор модели.....	592
15.1.2. Выбор программы	592
15.1.2.1. Отображаемые отношения	593
15.1.2.2. Параметры отображения.....	593
15.1.2.3. Предикаты модели	594
15.1.3. Примеры программ.....	594
15.1.3.1. Примеры из статьи.....	594
15.1.3.2. Другие категории примеров.....	595
16. Глоссарий	598
Предметный указатель	606