

Оглавление

Предисловие.....	16
Благодарности.....	18
О книге.....	20
Структура издания	20
Для кого предназначена эта книга	21
Порядок чтения.....	21
Условные обозначения и загрузка кода	22
Требования к программным средствам	22
Форум, посвященный книге	22
Об авторе	24
Об иллюстрации на обложке	25
От издательства	26
Глава 1. Здравствуй, мир конкурентности в C++!	27
1.1. Что такое конкурентность	28
1.1.1. Конкурентность в компьютерных системах.....	28
1.1.2. Подходы к конкурентности.....	31
1.1.3. Сравнение конкурентности и параллелизма.....	33
1.2. Зачем используется конкурентность.....	34
1.2.1. Конкурентность для разделения неотложных задач	34
1.2.2. Конкурентность для повышения производительности: параллелизм задач и данных	35
1.2.3. Когда не нужна конкурентность	36
1.3. Конкурентность и многопоточность в C++	37
1.3.1. История поддержки многопоточности в C++.....	38
1.3.2. Поддержка конкурентности в стандарте C++11	39

1.3.3. Расширение поддержки конкурентности и параллелизма в C++14 и C++17.....	39
1.3.4. Эффективность, обеспеченная библиотекой потоков C++	40
1.3.5. Средства, ориентированные на использование конкретной платформы	41
1.4. Приступаем к практической работе	41
1.4.1. Здравствуй, мир конкурентности.....	42
Резюме	43
Глава 2. Управление потоками	44
2.1. Основы управления потоками	45
2.1.1. Запуск потока	45
2.1.2. Ожидание завершения потока	48
2.1.3. Ожидание в исключительных обстоятельствах.....	49
2.1.4. Запуск потоков в фоновом режиме	51
2.2. Передача аргументов функции потока	53
2.3. Передача права владения потоком.....	55
2.4. Выбор количества потоков в ходе выполнения программы.....	60
2.5. Идентификация потоков.....	63
Резюме	64
Глава 3. Совместное использование данных несколькими потоками.....	65
3.1. Проблемы совместного использования данных несколькими потоками.....	66
3.1.1. Состояние гонки	68
3.1.2. Пути обхода проблемных состояний гонок	69
3.2. Защита совместно используемых данных с применением мьютексов	70
3.2.1. Использование мьютексов в C++	70
3.2.2. Структуризация кода для защиты совместно используемых данных	72
3.2.3. Обнаружение состояния гонки, присущего интерфейсам	74
3.2.4. Взаимная блокировка: проблема и решение	81
3.2.5. Дополнительные рекомендации по обходу взаимных блокировок	84
3.2.6. Гибкая блокировка с применением <code>std::unique_lock</code>	91
3.2.7. Передача владения мьютексом между областями видимости	92
3.2.8. Блокировка с соответствующей степенью детализации.....	94
3.3. Альтернативные средства защиты совместно используемых данных.....	97
3.3.1. Защита совместно используемых данных во время инициализации.....	97
3.3.2. Защита редко обновляемых структур данных.....	101
3.3.3. Рекурсивная блокировка	103
Резюме	104

Глава 4. Синхронизация конкурентных операций	105
4.1. Ожидание наступления события или создания другого условия	106
4.1.1. Ожидание выполнения условий с применением условных переменных	107
4.1.2. Создание потокобезопасной очереди с условными переменными.....	110
4.2. Ожидание единичных событий с помощью фьючерсов	115
4.2.1. Возвращение значений из фоновых задач.....	116
4.2.2. Связывание задачи с фьючерсом	119
4.2.3. Создание промисов <code>std::promise</code>	121
4.2.4. Сохранение исключения на будущее.....	124
4.2.5. Ожидание сразу в нескольких потоках.....	125
4.3. Ожидание с ограничением по времени	128
4.3.1. Часы	128
4.3.2. Продолжительность	130
4.3.3. Моменты времени	131
4.3.4. Функции, принимающие сроки ожидания	133
4.4. Применение синхронизации операций для упрощения кода	135
4.4.1. Функциональное программирование с применением фьючерсов.....	135
4.4.2. Синхронизация операций путем передачи сообщений.....	141
4.4.3. Конкурентность, организованная в стиле продолжений с применением <code>Concurrency TS</code>	146
4.4.4. Выстраиваем продолжения в цепочку	148
4.4.5. Ожидание готовности более чем одного фьючерса.....	151
4.4.6. Ожидание первого фьючерса в наборе с <code>when_any</code>	153
4.4.7. Защелки и барьеры в <code>Concurrency TS</code>	156
4.4.8. Базовый тип защелки <code>std::experimental::latch</code>	156
4.4.9. Основной барьер <code>std::experimental::barrier</code>	157
4.4.10. Барьер <code>std::experimental::flex_barrier</code> — более гибкий соратник барьера <code>std::experimental::barrier</code>	159
Резюме	161
Глава 5. Модель памяти C++ и операции над атомарными типами	162
5.1. Основы модели памяти.....	163
5.1.1. Объекты и их размещение в памяти.....	163
5.1.2. Объекты, области памяти и конкурентность.....	165
5.1.3. Очередность внесения изменений.....	166
5.2. Атомарные операции и типы в C++.....	166
5.2.1. Стандартные атомарные типы.....	167
5.2.2. Операции над <code>std::atomic_flag</code>	171

5.2.3. Операции над <code>std::atomic<bool></code>	173
5.2.4. Операции над <code>std::atomic<T*></code> : арифметика указателей	176
5.2.5. Операции над стандартными атомарными целочисленными типами.....	177
5.2.6. Шаблон первичного класса <code>std::atomic<></code>	178
5.2.7. Свободные функции для атомарных операций	180
5.3. Синхронизация операций и принудительное упорядочение.....	183
5.3.1. Отношение «синхронизируется с».....	184
5.3.2. Отношение «происходит до»	185
5.3.3. Упорядочение доступа к памяти для атомарных операций.....	187
5.3.4. Последовательности освобождений и отношения «синхронизируется с».....	207
5.3.5. Барьеры.....	209
5.3.6. Упорядочение неатомарных операций с помощью атомарных операций	211
5.3.7. Упорядочение неатомарных операций	212
Резюме	216
Глава 6. Разработка конкурентных структур данных с блокировками	217
6.1. Что означает разработка структур данных для конкурентного доступа.....	218
6.1.1. Рекомендации по разработке структур данных для конкурентного доступа.....	219
6.2. Конкурентные структуры данных с блокировками	220
6.2.1. Потокбезопасный стек, использующий блокировки	221
6.2.2. Потокбезопасная очередь с использованием блокировок и условных переменных.....	224
6.2.3. Потокбезопасная очередь с использованием подробной детализации блокировок и условных переменных.....	228
6.3. Разработка более сложных структур данных с использованием блокировок	240
6.3.1. Создание потокбезопасной поисковой таблицы с использованием блокировок	240
6.3.2. Создание потокбезопасного списка с использованием блокировок.....	246
Резюме	250
Глава 7. Разработка конкурентных структур данных без блокировок.....	251
7.1. Определения и выводы	252
7.1.1. Типы структур данных, не подвергаемых блокировкам	252
7.1.2. Структуры данных, свободные от блокировок	254
7.1.3. Структуры данных, свободные от ожиданий.....	254
7.1.4. Все за и против создания структур данных, свободных от блокировок	255
7.2. Примеры структур данных, свободных от блокировок	256
7.2.1. Создание потокбезопасного стека без блокировок	257

7.2.2. Устранение утечек: управление памятью в структурах данных, свободных от блокировок	261
7.2.3. Определение узлов, не подлежащих утилизации, с помощью указателей опасности	266
7.2.4. Определение используемых узлов путем подсчета ссылок	274
7.2.5. Применение модели памяти к свободному от блокировок стеку	281
7.2.6. Создание потокобезопасной очереди без блокировок	285
7.3. Рекомендации по созданию структур данных без блокировок	298
7.3.1. Для создания прототипа используйте <code>std::memory_order_seq_cst</code>	298
7.3.2. Воспользуйтесь схемой утилизации памяти, свободной от блокировок	299
7.3.3. Остерегайтесь проблемы ABA	299
7.3.4. Выявляйте циклы активного ожидания и организуйте помощь другому потоку	300
Резюме	301
Глава 8. Разработка конкурентного кода	302
8.1. Способы распределения работы между потоками	303
8.1.1. Распределение данных между потоками до начала обработки	304
8.1.2. Рекурсивное распределение данных	305
8.1.3. Распределение работы по типам задач	309
8.2. Факторы, влияющие на производительность конкурентного кода	312
8.2.1. А сколько у нас процессоров?	313
8.2.2. Конкуренция при обращении к данным и перебрасывание данных в кэш-памяти процессоров	314
8.2.3. Ложное совместное использование памяти	317
8.2.4. Насколько близко расположены по отношению друг к другу ваши данные?	318
8.2.5. Переоценка вычислительных возможностей и чрезмерное количество переключений задач	319
8.3. Разработка структур данных для высокопроизводительных многопоточных приложений	320
8.3.1. Распределение элементов массива для сложных операций	320
8.3.2. Схемы доступа к данным в других структурах данных	322
8.4. Дополнительные факторы, учитываемые при разработке конкурентных программ	324
8.4.1. Безопасность исключений в параллельных алгоритмах	325
8.4.2. Масштабируемость и закон Амдала	332
8.4.3. Компенсация потерь на ожидание за счет применения нескольких потоков	333
8.4.4. Повышение отзывчивости за счет конкурентности	334

8.5. Разработка конкурентного кода на практике	336
8.5.1. Параллельная реализация <code>std::for_each</code>	337
8.5.2. Параллельная реализация <code>std::find</code>	339
8.5.3. Параллельная реализация <code>std::partial_sum</code>	345
Резюме	354
Глава 9. Усовершенствованное управление потоками	355
9.1. Пулы потоков	356
9.1.1. Простейший пул потоков	356
9.1.2. Ожидание завершения задач, переданных пулу потоков.....	358
9.1.3. Задачи, ожидающие завершения других задач	362
9.1.4. Предотвращение конкуренции при обращении к очереди работ	365
9.1.5. Перехват работы.....	367
9.2. Прерывание потоков	371
9.2.1. Запуск и прерывание другого потока	371
9.2.2. Обнаружение того, что поток был прерван	373
9.2.3. Прерывание ожидания на условной переменной.....	374
9.2.4. Прерывание ожидания на <code>std::condition_variable_any</code>	377
9.2.5. Прерывание других блокирующих вызовов	379
9.2.6. Обработка прерываний	380
9.2.7. Прерывание фоновых задач при выходе из приложения.....	381
Резюме	382
Глава 10. Алгоритмы параллельных вычислений.....	383
10.1. Перевод стандартных библиотечных алгоритмов в режим параллельных вычислений	383
10.2. Политики выполнения	384
10.2.1. Общие последствия от задания политики выполнения	385
10.2.2. <code>std::execution::sequenced_policy</code>	386
10.2.3. <code>std::execution::parallel_policy</code>	387
10.2.4. <code>std::execution::parallel_unsequenced_policy</code>	388
10.3. Параллельные алгоритмы из стандартной библиотеки C++	388
10.3.1. Примеры использования параллельных алгоритмов.....	391
10.3.2. Подсчет посещений	393
Резюме	395
Глава 11. Тестирование и отладка многопоточных приложений.....	396
11.1. Типы ошибок, связанных с конкурентностью	397
11.1.1. Нежелательная блокировка	397
11.1.2. Состояния гонки	398

11.2. Приемы обнаружения ошибок, связанных с конкурентностью	399
11.2.1. Просмотр кода с целью поиска возможных ошибок.....	400
11.2.2. Обнаружение ошибок, связанных с конкурентностью, путем тестирования	402
11.2.3. Разработка кода с прицелом на удобство тестирования	404
11.2.4. Приемы тестирования многопоточного кода	406
11.2.5. Структурирование многопоточного тестового кода	408
11.2.6. Тестирование производительности многопоточного кода.....	411
Резюме	412

Приложения

Приложение А. Краткий справочник по некоторым функциям языка C++11	414
А.1. Ссылки на r-значения	414
А.1.1. Семантика перемещений.....	415
А.1.2. Ссылки на r-значения и шаблоны функций	418
А.2. Удаленные функции	419
А.3. Функции по умолчанию	420
А.4. constexpr-функции.....	424
А.4.1. constexpr и типы, определенные пользователем	425
А.4.2. constexpr-объекты.....	428
А.4.3. Требования к constexpr-функциям.....	428
А.4.4. constexpr и шаблоны.....	430
А.5. Лямбда-функции	430
А.5.1. Лямбда-функции, ссылающиеся на локальные переменные	432
А.6. Вариативные шаблоны	436
А.6.1. Расширение пакета параметров	437
А.7. Автоматическое выведение типа переменной.....	440
А.8. Локальные переменные потока	441
А.9. Выведение аргументов шаблона класса.....	442
Резюме	443
Приложение Б. Краткое сравнение библиотек для написания конкурентных программ.....	444
Приложение В. Среда передачи сообщений и полный пример программы управления банкоматом	446
Приложение Г. Справочник по C++ Thread Library.....	463
Г.1. Заголовок <chrono>	463
Г.1.1. Шаблон класса std::chrono::duration.....	464
Г.1.2. Шаблон класса std::chrono::time_point.....	474

Г.1.3. Класс <code>std::chrono::system_clock</code>	477
Г.1.4. Класс <code>std::chrono::steady_clock</code>	479
Г.1.5. Псевдоним типа <code>std::chrono::high_resolution_clock</code>	481
Г.2. Заголовок <code><condition_variable></code>	481
Г.2.1. Класс <code>std::condition_variable</code>	481
Г.2.2. Класс <code>std::condition_variable_any</code>	490
Г.3. Заголовок <code><atomic></code>	499
Г.3.1. Псевдонимы типа <code>std::atomic_xxx</code>	501
Г.3.2. Макросы <code>ATOMIC_xxx_LOCK_FREE</code>	501
Г.3.3. Макрос <code>ATOMIC_VAR_INIT</code>	502
Г.3.4. Перечисление <code>std::memory_order</code>	502
Г.3.5. Функция <code>std::atomic_thread_fence</code>	503
Г.3.6. Функция <code>std::atomic_signal_fence</code>	504
Г.3.7. Класс <code>std::atomic_flag</code>	504
Г.3.8. Шаблон класса <code>std::atomic</code>	508
Г.3.9. Специализации шаблона <code>std::atomic</code>	520
Г.3.10. Специализации <code>std::atomic<integral-type></code>	521
Г.4. Заголовок <code><future></code>	539
Г.4.1. Шаблон класса <code>std::future</code>	540
Г.4.2. Шаблон класса <code>std::shared_future</code>	546
Г.4.3. Шаблон класса <code>std::packaged_task</code>	552
Г.4.4. Шаблон класса <code>std::promise</code>	558
Г.4.5. Шаблон функции <code>std::async</code>	565
Г.5. Заголовок <code><mutex></code>	566
Г.5.1. Класс <code>std::mutex</code>	567
Г.5.2. Класс <code>std::recursive_mutex</code>	570
Г.5.3. Класс <code>std::timed_mutex</code>	572
Г.5.4. Класс <code>std::recursive_timed_mutex</code>	577
Г.5.5. Класс <code>std::shared_mutex</code>	582
Г.5.6. Класс <code>std::shared_timed_mutex</code>	586
Г.5.7. Шаблон класса <code>std::lock_guard</code>	594
Г.5.8. Шаблон класса <code>std::scoped_lock</code>	596
Г.5.9. Шаблон класса <code>std::unique_lock</code>	598
Г.5.10. Шаблон класса <code>std::shared_lock</code>	608
Г.5.11. Шаблон функции <code>std::lock</code>	619
Г.5.12. Шаблон функции <code>std::try_lock</code>	620
Г.5.13. Класс <code>std::once_flag</code>	621
Г.5.14. Шаблон функции <code>std::call_once</code>	621

Г.6. Заголовок <code><ratio></code>	622
Г.6.1. Шаблон класса <code>std::ratio</code>	623
Г.6.2. Псевдоним шаблона <code>std::ratio_add</code>	624
Г.6.3. Псевдоним шаблона <code>std::ratio_subtract</code>	624
Г.6.4. Псевдоним шаблона <code>std::ratio_multiply</code>	625
Г.6.5. Псевдоним шаблона <code>std::ratio_divide</code>	626
Г.6.6. Шаблон класса <code>std::ratio_equal</code>	626
Г.6.7. Шаблон класса <code>std::ratio_not_equal</code>	627
Г.6.8. Шаблон класса <code>std::ratio_less</code>	627
Г.6.9. Шаблон класса <code>std::ratio_greater</code>	628
Г.6.10. Шаблон класса <code>std::ratio_less_equal</code>	628
Г.6.11. Шаблон класса <code>std::ratio_greater_equal</code>	628
Г.7. Заголовок <code><thread></code>	628
Г.7.1. Класс <code>std::thread</code>	629
Г.7.2. Пространство имен <code>this_thread</code>	639