

Оглавление

Предисловие.....	24
Современный C++	24
В курсе событий.....	25
Кому пригодится эта книга.....	25
Готовые файлы с исходным кодом программ.....	25
Три профессиональных компилятора на выбор.....	26
Учебный метод готовых объектов	26
Что такое готовые объекты?.....	27
Бесплатные классы.....	27
Проект Boost.....	27
Примеры кода с готовыми объектами	28
Краткое содержание книги	28
Часть 1. Быстрый старт и базовые возможности C++	29
Часть 2. Массивы, указатели и строки.....	30
Часть 3. Объектно-ориентированное программирование	31
Часть 4. Контейнеры, итераторы и алгоритмы стандартной библиотеки.....	33
Часть 5. Более сложные темы	34
Приложения.....	35
Дополнительные материалы на сайте deitel.com	35
C++ Core Guidelines	36
Библиотека Guidelines Support Library (GSL)	36
Профессиональные статические анализаторы кода.....	37
Интернет-ресурсы для программистов	37
Stack Overflow	37
GitHub.....	38
Docker.....	39
Некоторые ключевые документы и ресурсы по C++	39
Документация.....	40
Блоги	40
Дополнительные ресурсы	41
Сайты, где ответят на ваши вопросы.....	41
Общение с авторами.....	41

Благодарности.....	41
Рецензенты.....	42
Артур О’Двайр	43
GitHub.....	43
Мэтт Годболт и Compiler Explorer.....	43
Дитмар Куль.....	44
Райнер Гримм	44
Брайан Гетц.....	44
Разработчики программ с открытым исходным кодом и блогеры	44
Поисковая система Google.....	45
Grammarly.....	45
Об авторах.....	46
От издательства	47
Перед началом работы.....	48
Условные обозначения	48
Загрузка примеров кода.....	48
Компиляторы, на которых мы тестировали исходный код	49
Не все примеры кода совместимы со всеми тремя компиляторами	49
Установка Visual Studio Community Edition в Windows	50
Установка Xcode в macOS.....	50
Установка актуальной версии GNU C++.....	50
Установка GNU Compiler Collection в Ubuntu Linux, запущенной в подсистеме Windows для Linux	51
Docker и контейнеры Docker	52
Установка Docker	52
Загрузка контейнера Docker, содержащего GNU Compiler Collection.....	52
Загрузка контейнера Docker, содержащего Clang	53
Ответы на ваши вопросы по C++	53
Онлайн-документация по C++	54
О библиотеке форматирования текста {fmt}	54
Средства статического анализа кода.....	54
Глава 1. Введение и тест-драйв популярных бесплатных компиляторов C++.....	56
1.1. Введение	57
1.2. Тест-драйв: компиляция приложения, написанного на C++20	58
1.2.1. Компиляция и запуск приложения в Windows с помощью Visual Studio 2022.....	59

1.2.2. Компиляция и запуск приложения в macOS с помощью Xcode	63
1.2.3. Компиляция и запуск приложения в Linux с помощью g++	67
1.2.4. Компиляция и запуск приложения в контейнере Docker с помощью g++	70
1.2.5. Компиляция и запуск приложения в контейнере Docker с помощью clang++	71
1.3. Закон Мура, многоядерные процессоры и многопоточное программирование	73
1.4. Что такое объектно-ориентированное программирование?	75
1.5. Итоги	78
Глава 2. Азы программирования на C++	79
2.1. Введение	80
2.2. Первая программа на C++: вывод текста на экран	80
2.3. Модификация нашей первой программы на C++	85
2.4. Вторая программа на C++: сложение целых чисел	86
2.5. Арифметические операторы	90
2.6. Принятие решений: операторы равенства и сравнения	92
2.7. Готовые объекты: создание и использование объектов класса string стандартной библиотеки	96
2.8. Итоги	100
Глава 3. Управляющие инструкции (часть 1)	101
3.1. Введение	102
3.2. Управляющие структуры	102
3.2.1. Последовательность	103
3.2.2. Инструкции выбора	104
3.2.3. Инструкции цикла	104
3.2.4. Комбинации управляющих инструкций	105
3.3. Инструкция одиночного выбора if	106
3.4. Инструкция двойного выбора if..else	107
3.4.1. Вложенные инструкции if..else	108
3.4.2. Блоки	109
3.4.3. Тернарный условный оператор ?:	110
3.5. Инструкция цикла while	110
3.6. Цикл со счетчиком	111
3.6.1. Реализация цикла со счетчиком	112
3.6.2. Целочисленное деление с усечением	113
3.7. Цикл с контрольным значением	114
3.7.1. Реализация цикла с контрольным значением	114
3.7.2. Явное и неявное преобразование основных типов данных	117
3.7.3. Форматирование чисел с плавающей точкой	118

3.8. Вложенные управляющие инструкции	119
3.8.1. Постановка задачи	119
3.8.2. Реализация программы	120
3.8.3. Предотвращение сужающих преобразований с помощью скобочной инициализации	122
3.9. Составные операторы присваивания	123
3.10. Операторы инкремента и декремента.....	123
3.11. Несовместимость основных типов данных на разных платформах.....	127
3.12. Готовые объекты: работа с целыми числами произвольной величины.....	127
3.13. Форматирование текста функцией <code>format</code>	133
3.14. Итоги	135
Глава 4. Управляющие инструкции (часть 2)	137
4.1. Введение	138
4.2. Как работает цикл со счетчиком	138
4.3. Инструкция цикла <code>for</code>	139
4.4. Инкремент и декремент счетчика в цикле <code>for</code>	142
4.5. Программа для вычисления суммы четных чисел	143
4.6. Программа для расчета сложных процентов.....	144
4.7. Инструкция цикла <code>do...while</code>	148
4.8. Инструкция множественного выбора <code>switch</code>	150
4.9. Инструкции выбора с инициализаторами.....	156
4.10. Инструкции <code>break</code> и <code>continue</code>	157
4.11. Логические операторы	159
4.11.1. Оператор <code>&&</code> (логическое И).....	159
4.11.2. Оператор <code> </code> (логическое ИЛИ)	160
4.11.3. Вычисление по короткой схеме	161
4.11.4. Оператор <code>!</code> (логическое НЕ)	162
4.11.5. Программное создание таблиц истинности для логических операторов.....	162
4.12. Типичные ошибки при работе с операторами равенства (<code>==</code>) и присваивания (<code>=</code>)	164
4.13. Готовые объекты: создание и чтение ZIP-файлов.....	166
4.14. Форматирование текста с указанием ширины полей и точности.....	171
4.15. Итоги	174
Глава 5. Функции и шаблоны функций	175
5.1. Введение	176
5.2. Функции и классы — основные компоненты программ.....	176
5.3. Функции математической библиотеки	177
5.4. Определения функций и прототипы функций.....	180

5.5. Порядок вычисления аргументов функции.....	183
5.6. Дополнительные сведения о прототипах функций и принудительном приведении аргументов	184
5.6.1. Сигнатуры функций и прототипы функций	184
5.6.2. Принудительное приведение аргументов.....	184
5.6.3. Правила продвижения аргументов и неявные преобразования.....	185
5.7. Заголовки стандартной библиотеки C++	187
5.8. Генерация случайных чисел.....	190
5.8.1. Бросаем игровой кубик	191
5.8.2. Бросаем кубик 60 миллионов раз.....	192
5.8.3. Инициализация генератора случайных чисел	194
5.8.4. Инициализация генератора случайных чисел с помощью <code>random_device</code>	195
5.9. Программа «Игра в кости» и знакомство с перечислениями	196
5.10. Правила областей видимости	202
5.11. Встроенные функции.....	207
5.12. Ссылки и ссылочные параметры.....	208
5.13. Аргументы по умолчанию.....	212
5.14. Оператор разрешения области видимости.....	214
5.15. Перегрузка функций.....	214
5.16. Шаблоны функций.....	218
5.17. Рекурсия.....	221
5.18. Пример рекурсивных вычислений: числа Фибоначчи.....	224
5.19. Рекурсия или итерация?	228
5.20. <code>Lnfylun Lhqtomh Wjtz Qarcv: Qjwazkrplm xzz Xndmwwqhlz</code>	230
5.21. Итоги	234
Глава 6. Массивы, векторы, библиотека <code>ranges</code> и функциональное программирование	235
6.1. Введение	236
6.2. Массивы.....	237
6.3. Объявление массива.....	237
6.4. Инициализация массива в цикле.....	238
6.5. Инициализация массива со списком инициализаторов.....	241
6.6. Цикл <code>for</code> для диапазона значений (C++11) и цикл <code>for</code> для диапазона значений с инициализатором (C++20).....	242
6.7. Заполнение массива и спецификатор <code>constexpr</code>	245
6.8. Суммирование элементов массива.....	247
6.9. Визуализация данных массива на простой диаграмме	248
6.10. Использование элементов массива в качестве счетчиков	250

6.11. Использование массива для подсчета результатов опроса	251
6.12. Сортировка массива и поиск в массиве.....	252
6.13. Многомерные массивы.....	255
6.14. Основы функционального программирования	259
6.14.1. «Что» и «как»	260
6.14.2. Передача функций в качестве аргументов другим функциям, основы лямбда-выражений.....	261
6.14.3. Фильтрация, преобразование и свертка с библиотекой ranges (C++20)	263
6.15. Готовые объекты: шаблон класса vector стандартной библиотеки C++	268
6.16. Итоги	275
Глава 7. Указатели в современном C++ (фактор риска).....	277
7.1. Введение	278
7.2. Объявление и инициализация указателей.....	280
7.2.1. Объявление указателя.....	280
7.2.2. Инициализация указателя.....	280
7.2.3. Нулевые указатели до C++11	281
7.3. Операторы для работы с указателями	281
7.3.1. Оператор взятия адреса (&).....	281
7.3.2. Оператор косвенного обращения (*)	282
7.3.3. Использование операторов взятия адреса (&) и косвенного обращения (*) ...	283
7.4. Передача аргументов по указателю	284
7.5. Традиционные массивы в стиле C.....	286
7.5.1. Объявление традиционного массива и доступ к его элементам.....	286
7.5.2. Инициализация традиционного массива	289
7.5.3. Передача традиционного массива функции	289
7.5.4. Объявление параметров функции, обрабатывающей традиционный массив....	289
7.5.5. Пример с функциями sort, begin и end стандартной библиотеки (C++11).....	290
7.5.6. Недостатки традиционных массивов.....	290
7.6. Функция to_array (C++20), преобразующая традиционные массивы в std::array ...	291
7.7. Использование const с указателями и данными, на которые они указывают.....	293
7.7.1. Неконстантный указатель на неконстантные данные.....	293
7.7.2. Неконстантный указатель на константные данные	293
7.7.3. Константный указатель на неконстантные данные	294
7.7.4. Константный указатель на константные данные.....	295
7.8. Оператор sizeof.....	296
7.9. Выражения и арифметические операции с указателями	299
7.9.1. Прибавление и вычитание целых чисел	300
7.9.2. Вычитание одного указателя из другого.....	301

7.9.3. Присваивание указателя.....	301
7.9.4. Невозможность разыменования указателя void*	302
7.9.5. Сравнение указателей	302
7.10. Готовые объекты: доступ к контейнерам через span (C++20)	302
7.11. Строки на основе указателей	309
7.11.1. Аргументы командной строки.....	310
7.11.2. Еще одна возможность функции to_array (C++20)	312
7.12. Другие темы, связанные с указателями	313
7.13. Итоги	314
Глава 8. Строки и их представления, текстовые файлы, CSV-файлы и регулярные выражения	315
8.1. Введение	316
8.2. Присваивание и конкатенация строк.....	317
8.3. Сравнение строк	319
8.4. Создание подстроки	321
8.5. Обмен значениями двух строк.....	321
8.6. Получение характеристик строки	322
8.7. Поиск подстрок и символов в строке	325
8.8. Замена и удаление символов в строке	327
8.9. Вставка символов в строку	329
8.10. Преобразования между числами и строками (C++11).....	330
8.11. Представления string_view (C++17)	332
8.12. Файлы и потоки.....	335
8.13. Создание файла с последовательным доступом	336
8.14. Чтение данных из файла с последовательным доступом.....	340
8.15. Чтение и запись текста в кавычках (C++14).....	342
8.16. Обновление данных в файле с последовательным доступом.....	344
8.17. Обработка строковых потоков	344
8.18. Необработанные строковые литералы	347
8.19. Готовые объекты: чтение и анализ CSV-файла с данными о катастрофе «Титаника»	348
8.19.1. Чтение CSV-файла с помощью библиотеки rapidcsv.....	349
8.19.2. Работа с набором данных «Титаник».....	351
8.20. Готовые объекты: основы регулярных выражений	359
8.20.1. Сравнение целых строк с шаблонами.....	361
8.20.2. Замена подстрок.....	365
8.20.3. Поиск в тексте по шаблонам	366
8.21. Итоги	369

Глава 9. Пользовательские классы	370
9.1. Введение	371
9.2. Тест-драйв объекта пользовательского класса Account.....	371
9.3. Класс Account с элементом данных, set-функцией и get-функцией	373
9.3.1. Определение класса	373
9.3.2. Спецификаторы доступа private и public.....	376
9.4. Конструктор класса Account	377
9.5. Разработка программ с set-функциями и get-функциями.....	381
9.6. Добавление «денежной» переменной в класс Account	383
9.7. Отделение интерфейса от реализации на примере класса Time.....	386
9.7.1. Интерфейс класса	388
9.7.2. Отделение интерфейса класса от реализации класса.....	388
9.7.3. Определение класса	389
9.7.4. Функции класса Time.....	390
9.7.5. Включение заголовка класса в файл исходного кода.....	391
9.7.6. Оператор разрешения области видимости (::).....	391
9.7.7. Функция setTime и исключения.....	392
9.7.8. Функции to24HourString и to12HourString.....	392
9.7.9. Неявное встраивание функций.....	393
9.7.10. Функции класса в сравнении с глобальными функциями	393
9.7.11. Использование класса Time	393
9.7.12. Размер объекта.....	395
9.8. Процесс компиляции и компоновки программы.....	395
9.9. Область видимости класса и доступ к элементам класса.....	397
9.10. Функции доступа и служебные функции	398
9.11. Конструкторы с аргументами по умолчанию.....	398
9.11.1. Усовершенствованный класс Time	398
9.11.2. Перегрузка конструкторов и делегирование конструкторов (C++11).....	404
9.12. Деструкторы	405
9.13. Вызов конструкторов и деструкторов	405
9.14. Коварная ловушка: возврат ссылки или указателя на скрытый элемент данных...	409
9.15. Оператор присваивания по умолчанию.....	412
9.16. Константные объекты и константные функции класса.....	414
9.17. Композиция: объекты как элементы классов.....	416
9.18. Дружественные функции и классы	421
9.19. Указатель this.....	423
9.19.1. Неявное и явное использование указателя this для доступа к элементам данных объекта	424
9.19.2. Использование указателя this для каскадного вызова функций	425

9.20. Статические элементы класса: данные и функции, общие для всех объектов класса.....	429
9.21. Агрегаты.....	434
9.21.1. Инициализация агрегата.....	435
9.21.2. Назначенные инициализаторы (C++20).....	435
9.22. Готовые объекты: сериализация в формате JSON.....	436
9.22.1. Сериализация вектора объектов, содержащего открытые данные.....	438
9.22.2. Сериализация вектора объектов, содержащего скрытые данные.....	442
9.23. Итоги.....	444
Глава 10. ООП: наследование и динамический полиморфизм.....	446
10.1. Введение.....	447
10.2. Базовые и производные классы.....	450
10.2.1. Иерархия класса CommunityMember.....	450
10.2.2. Иерархия класса Share и открытое наследование.....	451
10.3. Взаимосвязь между базовым и производным классами.....	453
10.3.1. Объявление и использование класса SalariedEmployee.....	453
10.3.2. Создание иерархии наследования: SalariedEmployee и SalariedCommissionEmployee.....	456
10.4. Конструкторы и деструкторы в производных классах.....	462
10.5. Принцип динамического полиморфизма на примере объектов в видеоигре.....	463
10.6. Отношения между объектами в иерархии наследования.....	464
10.6.1. Вызов функций базового класса из объектов производного класса.....	465
10.6.2. Направление указателя производного класса на объект базового класса.....	468
10.6.3. Вызовы функций производного класса через указатель базового класса.....	469
10.7. Виртуальные функции и виртуальные деструкторы.....	470
10.7.1. Зачем нужны виртуальные функции.....	470
10.7.2. Объявление виртуальной функции.....	471
10.7.3. Вызов виртуальной функции.....	471
10.7.4. Виртуальные функции в иерархии SalariedEmployee.....	472
10.7.5. Виртуальные деструкторы.....	476
10.7.6. Финальные функции и классы.....	476
10.8. Абстрактные классы и чистые виртуальные функции.....	477
10.8.1. Чистые виртуальные функции.....	477
10.8.2. Драйверы устройств: полиморфизм в операционных системах.....	478
10.9. Полиморфизм в программе расчета зарплаты.....	479
10.9.1. Создание абстрактного базового класса Employee.....	480
10.9.2. Создание конкретного производного класса SalariedEmployee.....	482
10.9.3. Создание конкретного производного класса CommissionEmployee.....	485
10.9.4. Демонстрация полиморфных операций с объектами.....	487

10.10. Динамический полиморфизм, виртуальные функции и динамическое связывание «под капотом» компилятора	490
10.11. Идиома неvirtуального интерфейса.....	493
10.12. Наследование интерфейса вместо наследования реализации.....	501
10.12.1. Переосмысление иерархии класса Employee: интерфейс CompensationModel.....	503
10.12.2. Класс Employee.....	503
10.12.3. Реализации CompensationModel.....	505
10.12.4. Тест-драйв новой иерархии.....	508
10.12.5. Преимущества внедрения зависимостей	509
10.13. Полиморфизм на основе шаблона класса std::variant и функции std::visit	509
10.14. Множественное наследование	516
10.14.1. Ромбовидное наследование	521
10.14.2. Устранение конфликта подбъектов с помощью виртуального наследования базового класса	523
10.15. Защищенные элементы класса.....	525
10.16. Открытое, защищенное и скрытое наследование.....	527
10.17. Другие методы динамического полиморфизма и статический полиморфизм	528
10.17.1. Другие методы динамического полиморфизма.....	529
10.17.2. Методы статического полиморфизма	530
10.17.3. Другие концепции полиморфизма.....	532
10.18. Итоги.....	533
Глава 11. Перегрузка операторов, семантика копирования и перемещения, умные указатели	535
11.1. Введение	536
11.2. Перегруженные операторы класса string стандартной библиотеки.....	538
11.3. Основные принципы перегрузки операторов.....	543
11.3.1. Перегрузка операторов не происходит автоматически	543
11.3.2. Операторы, которые не могут быть перегружены.....	544
11.3.3. Операторы, которые не нужно перегружать	544
11.3.4. Правила и ограничения перегрузки операторов.....	544
11.4. Управление динамической памятью с помощью операторов new и delete.....	545
11.5. Современные методы управления динамической памятью в C++: RAII и умные указатели	548
11.5.1. Умные указатели	549
11.5.2. Умный указатель unique_ptr.....	549
11.5.3. Владение объектами при работе с unique_ptr.....	551
11.5.4. Указатели unique_ptr и традиционные массивы	551

11.6. Создание класса <code>myArray</code> с перегрузкой операторов.....	552
11.6.1. Специальные функции класса	554
11.6.2. Работа с классом <code>myArray</code>	554
11.6.3. Определение класса <code>myArray</code>	565
11.6.4. Конструктор, задающий размер объекта <code>myArray</code>	566
11.6.5. Скобочная инициализация в конструкторе (C++11).....	567
11.6.6. Копирующий конструктор и оператор копирующего присваивания	568
11.6.7. Перемещающий конструктор и оператор перемещающего присваивания	572
11.6.8. Деструктор	576
11.6.9. Функции <code>toString</code> и <code>size</code>	577
11.6.10. Перегрузка операторов равенства и неравенства.....	577
11.6.11. Перегрузка оператора индекса.....	579
11.6.12. Перегрузка унарного конвертирующего оператора <code>bool</code>	581
11.6.13. Перегрузка оператора префиксного инкремента	581
11.6.14. Перегрузка оператора постфиксного инкремента	582
11.6.15. Перегрузка оператора сложения с присваиванием	583
11.6.16. Перегрузка бинарных операторов извлечения из потока и вставки в поток.....	584
11.6.17. Дружественная функция <code>swap</code>	587
11.7. Оператор трехстороннего сравнения (C++20).....	587
11.8. Преобразования между типами	591
11.9. Конструкторы и конвертирующие операторы со спецификатором <code>explicit</code>	592
11.10. Перегрузка оператора вызова функции.....	595
11.11. Итоги.....	595
Глава 12. Исключения и обзор контрактного программирования.....	597
12.1. Введение.....	598
12.2. Поток управления при обработке исключения.....	602
12.2.1. Определение класса исключений для представления типа проблемы, которая может возникнуть.....	602
12.2.2. Демонстрация обработки исключений	603
12.2.3. Включение кода в блок <code>try</code>	605
12.2.4. Определение обработчика <code>catch</code> для <code>DivideByZeroException</code>	605
12.2.5. Модель завершения обработки исключений	606
12.2.6. Поток управления при вводе ненулевого знаменателя.....	607
12.2.7. Поток управления при вводе нулевого знаменателя	607
12.3. Гарантии безопасности исключений и <code>noexcept</code>	608
12.4. Повторная генерация исключений	609
12.5. Размотка стека и необработанные исключения	611

12.6. Когда надо обрабатывать исключения	613
12.6.1. Макрос <code>assert</code>	616
12.6.2. Немедленное завершение	616
12.7. Конструкторы, деструкторы и обработка исключений	617
12.7.1. Генерация исключений в конструкторах	617
12.7.2. Перехват исключений в конструкторах с помощью функциональных блоков <code>try</code>	618
12.7.3. Исключения, деструкторы и <code>noexcept(false)</code>	620
12.8. Обработка исключений оператора <code>new</code>	621
12.8.1. Генерация исключения <code>bad_alloc</code> оператора <code>new</code>	622
12.8.2. Оператор <code>new</code> , возвращающий <code>nullptr</code> при сбое	623
12.8.3. Обработка исключений оператора <code>new</code> с помощью функции <code>set_new_handler</code>	624
12.9. Иерархия исключений стандартной библиотеки	625
12.10. Альтернатива блоку <code>finally</code> в языке C++: RAII («получение ресурсов есть инициализация»)	628
12.11. Библиотеки, поддерживающие и исключения, и коды ошибок	629
12.12. Журнал ошибок	630
12.13. Основы контрактного программирования	631
12.14. Итоги	640
Глава 13. Контейнеры и итераторы стандартной библиотеки	642
13.1. Введение	643
13.2. Основы контейнеров	645
13.2.1. Общие вложенные типы в последовательных и ассоциативных контейнерах	647
13.2.2. Общие функции контейнеров	648
13.2.3. Требования к элементам контейнера	651
13.3. Работа с итераторами	652
13.3.1. Использование <code>istream_iterator</code> для ввода и <code>ostream_iterator</code> для вывода	652
13.3.2. Категории итераторов	654
13.3.3. Поддерживаемые контейнеры	655
13.3.4. Предопределенные типы итераторов	656
13.3.5. Операторы для работы с итераторами	656
13.4. Несколько слов об алгоритмах для обработки контейнеров	658
13.5. Последовательные контейнеры	658
13.6. Последовательный контейнер <code>vector</code>	659
13.6.1. Использование векторов и итераторов	659
13.6.2. Функции для обработки элементов вектора	664

13.7. Последовательный контейнер list	668
13.8. Последовательный контейнер deque.....	673
13.9. Ассоциативные контейнеры.....	675
13.9.1. Ассоциативный контейнер multiset.....	676
13.9.2. Ассоциативный контейнер set.....	680
13.9.3. Ассоциативный контейнер multimap.....	682
13.9.4. Ассоциативный контейнер map	685
13.10. Адаптеры контейнеров.....	686
13.10.1. Адаптер stack.....	687
13.10.2. Адаптер queue	689
13.10.3. Адаптер priority_queue	690
13.11. Псевдоконтейнер bitset.....	692
13.12. Основы нотации «О-большое»	694
13.13. Основы хеш-таблиц	697
13.14. Итоги.....	700

Глава 14. Алгоритмы стандартной библиотеки, диапазоны и представления C++20	701
14.1. Введение.....	702
14.2. Требования алгоритмов: концепты (C++20).....	704
14.3. Лямбда-выражения и алгоритмы	706
14.4. Алгоритмы.....	710
14.4.1. Алгоритмы fill, fill_n, generate и generate_n.....	710
14.4.2. Алгоритмы equal, mismatch и lexicographical_compare.....	713
14.4.3. Алгоритмы remove, remove_if, remove_copy и remove_copy_if	716
14.4.4. Алгоритмы replace, replace_if, replace_copy и replace_copy_if.....	721
14.4.5. Алгоритмы перемешивания, подсчетов, поиска минимального и максимального элементов	723
14.4.6. Алгоритмы поиска и сортировки	727
14.4.7. Алгоритмы swap, iter_swap и swap_ranges.....	731
14.4.8. Алгоритмы copy_backward, merge, unique, reverse, copy_if и copy_n	734
14.4.9. Алгоритмы inplace_merge, unique_copy и reverse_copy	737
14.4.10. Операции со множествами.....	739
14.4.11. Алгоритмы lower_bound, upper_bound и equal_range	742
14.4.12. Алгоритмы min, max и minmax.....	744
14.4.13. Алгоритмы gcd, lcm, iota, reduce и partial_sum (заголовок <numeric>).....	745
14.4.14. Пирамидальная сортировка кучи и приоритетные очереди	749
14.5. Функциональные объекты (функторы).....	754

14.6. Проекция	759
14.7. Представления (C++20) и функциональное программирование	763
14.7.1. Адаптеры диапазонов	763
14.7.2. Работа с адаптерами диапазонов и представлениями	765
14.8. Основы параллельных алгоритмов	770
14.9. Краткое описание алгоритмов стандартной библиотеки	772
14.10. Диапазоны в C++23	776
14.11. Итоги	777
Глава 15. Шаблоны, концепты (C++20) и метапрограммирование	779
15.1. Введение	780
15.2. Пользовательские шаблоны классов и статический полиморфизм	783
15.3. Расширения функциональности шаблонов, введенные в C++20	789
15.3.1. Сокращенные шаблоны функций (C++20)	789
15.3.2. Шаблоны лямбда-выражений (C++20)	791
15.4. Концепты (C++20): первый взгляд	791
15.4.1. Неограниченный шаблон функции multiply	793
15.4.2. Ограниченный шаблон функции с условием requires (C++20)	796
15.4.3. Концепты, предопределенные в C++20	799
15.5. Признаки типов	801
15.6. Концепты: детальное рассмотрение	806
15.6.1. Создание пользовательского концепта	807
15.6.2. Использование концепта	808
15.6.3. Концепты в сокращенных шаблонах функций	809
15.6.4. Перегрузка на основе концептов	810
15.6.5. Выражения requires	813
15.6.6. Демонстрационные концепты	817
15.6.7. Технологии, применявшиеся до появления концептов: SFINAE и отправка тегов	818
15.7. Тест-драйв концептов с помощью объявления static_assert	819
15.8. Создание пользовательского алгоритма	822
15.9. Создание пользовательского контейнера и пользовательских итераторов	824
15.9.1. Шаблон класса ConstIterator	827
15.9.2. Шаблон класса Iterator	830
15.9.3. Шаблон класса MyArray	832
15.9.4. Дедуктивная инструкция для скобочной инициализации объекта myArray	836
15.9.5. Тест-драйв класса myArray и его пользовательских итераторов с алгоритмами std::ranges	838

15.10. Аргументы по умолчанию для параметров-типов шаблона	842
15.11. Шаблоны переменных	842
15.12. Вариативные шаблоны и выражения свертки.....	843
15.12.1. Кортежи: вариативный шаблон класса tuple	843
15.12.2. Вариативные шаблоны функций и основы выражений свертки (C++17).....	847
15.12.3. Типы выражений свертки	851
15.12.4. Применение операторов в унарных выражениях свертки.....	852
15.12.5. Применение операторов в бинарных выражениях свертки	854
15.12.6. Оператор «запятая» для повторения операций	856
15.12.7. Ограничение элементов пакета параметров одним типом.....	857
15.13. Метапрограммирование на основе шаблонов.....	859
15.13.1. Шаблоны C++ обладают полнотой по Тьюрингу	860
15.13.2. Вычисления на этапе компиляции.....	861
15.13.3. Условная компиляция в метапрограммировании на основе шаблонов и инструкции constexpr if	866
15.13.4. Метафункции типа	869
15.14. Итоги.....	873
Глава 16. Модули (C++20): технология разработки больших программ	875
16.1. Введение.....	876
16.2. Компиляция и компоновка до C++20	878
16.3. Преимущества и предназначение модулей	880
16.4. Пример перехода на модули: заголовочные единицы трансляции	881
16.5. Модули могут уменьшать размеры единиц трансляции и время компиляции	884
16.6. Пример: создание и использование модуля.....	886
16.6.1. Объявление модульной единицы.....	887
16.6.2. Экспорт объявления	889
16.6.3. Экспорт группы объявлений	890
16.6.4. Экспорт пространства имен	890
16.6.5. Экспорт элемента пространства имен	891
16.6.6. Импорт модуля для его использования в экспорте объявлений	891
16.6.7. Пример: попытка доступа к неэкспортированному элементу модуля.....	893
16.7. Глобальный фрагмент модуля.....	896
16.8. Отделение интерфейса от реализации	896
16.8.1. Пример: единицы реализации модуля	897
16.8.2. Пример: класс на основе модулей.....	900
16.8.3. Фрагмент :private	903

16.9. Разделы модуля	904
16.9.1. Пример: разделы интерфейса модуля	904
16.9.2. Разделы реализации модуля	908
16.9.3. Пример: «подмодули» и разделы	908
16.10. Дополнительные примеры работы с модулями	913
16.10.1. Пример: импорт стандартной библиотеки C++ в виде модулей	914
16.10.2. Пример: запрет циклических зависимостей	916
16.10.3. Пример: инструкции <code>import</code> не наследуются	917
16.10.4. Пример: видимость и доступность модулей	918
16.11. Перенос кода в модули	920
16.12. Будущее модулей и инструментов для работы с ними	921
16.13. Итоги	922
Приложение: видео про модули	924
Приложение: статьи про модули	925
Приложение: глоссарий по модулям	928
Глава 17. Параллельные алгоритмы и конкурентность: высокоуровневый подход	931
17.1. Введение	932
17.2. Параллельные алгоритмы стандартной библиотеки (C++17)	935
17.2.1. Пример: сравнение быстродействия последовательного и параллельного алгоритмов сортировки	936
17.2.2. Когда нужны параллельные алгоритмы	940
17.2.3. Политики выполнения	941
17.2.4. Пример: сравнение быстродействия параллельных и векторных операций	941
17.2.5. Дополнительные сведения о параллельных алгоритмах	944
17.3. Многопоточное программирование	945
17.3.1. Состояния потока и жизненный цикл потока	945
17.3.2. Тупики и неопределенно долгие отсрочки	948
17.4. Запуск задач с помощью <code>std::jthread</code>	951
17.4.1. Определение задачи для выполнения в потоке	951
17.4.2. Выполнение задачи в потоке <code>jthread</code>	953
17.4.3. Преимущества класса <code>jthread</code> перед классом <code>thread</code>	956
17.5. Отношения между производителем и потребителем: первый эксперимент	957
17.6. Синхронизация доступа производителя и потребителя к общим изменяемым данным	965
17.6.1. Класс <code>SynchronizedBuffer</code> : мьютексы, блокировки и условные переменные	967
17.6.2. Тестирование класса <code>SynchronizedBuffer</code>	975

17.7. Производитель и потребитель: минимизация ожидания с помощью кольцевого буфера	979
17.8. Читатели и писатели	989
17.9. Кооперативное прерывание потоков класса <code>jthread</code>	991
17.10. Запуск задач с помощью шаблона функции <code>std::async</code>	994
17.11. Однократная потокобезопасная инициализация	1002
17.12. Основы атомарных типов	1004
17.13. Координация потоков с помощью защелок и барьеров (C++20)	1008
17.13.1. Защелки класса <code>std::latch</code> (C++20)	1009
17.13.2. Барьеры класса <code>std::barrier</code> (C++20)	1012
17.14. Семафоры (C++20)	1016
17.15. C++23: взгляд в будущее конкурентности	1020
17.15.1. Параллельные алгоритмы <code>std::ranges</code>	1020
17.15.2. Конкурентные контейнеры	1020
17.15.3. Другие материалы, связанные с конкурентностью	1021
17.16. Итоги	1021
Глава 18. Корутины (C++20)	1024
18.1. Введение	1025
18.2. Библиотеки поддержки корутин	1026
18.3. Установка библиотек <code>concurrency</code> и <code>generator</code>	1028
18.4. Создание корутины-генератора с помощью библиотеки <code>generator</code>	1029
18.5. Запуск задач с помощью библиотеки <code>concurrency</code>	1033
18.6. Создание корутины, содержащей <code>co_await</code> и <code>co_return</code>	1039
18.7. Низкоуровневые концепции корутин	1048
18.8. Развитие корутин в C++23	1051
18.9. Итоги	1051
Приложение А. Приоритеты и группировка операторов	1052
Приложение Б. Набор символов	1055