Оглавление

От издательства ... 12
Об авторе ... 13
Введение ... 14
Назначение этой книги ... 14
Условные обозначения ... 14
Примеры исходного кода ... 15
Исходный код ... 15
Значение и объект ... 15
Признательности ... 15
Дальнейшая информация ... 16
Циппи ... 16
1. Стандартная библиотека ... 17
Краткий обзор ... 17
История ... 17
Краткий обзор ... 18
Вспомогательные инструменты ... 19
Стандартная библиотека шаблонов ... 19
Библиотеки для работы с числами ... 22
Библиотеки для работы с текстом ... 22
Библиотека для работы с вводом и выводом данных ... 23
Библиотеки для работы с многопоточностью ... 23
Использование библиотек ... 25
Вставка заголовочных файлов ... 25
Импорт стандартной библиотеки ... 25
Использование именных пространств ... 26
Сборка исполняемого файла ... 28
2. Вспомогательные инструменты ... 29
Краткий обзор ... 29
Вспомогательные функции ... 29
std::min, std::max и std::minmax ... 30
std::midpoint и std::lerp ... 31
std::move ... 33
std::forward ... 33
std::to_underlying ... 35
std::swap ... 35
Адаптеры функциональных единиц кода ... 35
std::bind ... 35
std::bind_front (C++20) ... 37
std::bind_back (C++23) ... 37

std::function 37
Пары 38
std::make_pair 39
Кортежи 39
std::make_tuple 40
std::tie и std::ignore 40
Обертки вокруг ссылок 41
std::ref и std::cref 42
Умные указатели 43
std::unique_ptr 44
Специальные удалители 47
std::make_unique 47
std::shared_ptr 47
std::make_shared 48
std::shared_ptr из указателя this 49
std::weak_ptr 49
Циклические ссылки 51
Признаки типов 52
Проверка информации о типе 53
Отношения типов 56
Модификации типов 57
Операции на признаках 58
Отношения между членами 59
Константно-вычисляемый контекст 59
Библиотека для работы со временем 60
Временная точка 61
Продолжительность времени 62
Часы 64
Время дня 65
Календарь 65
Часовой пояс 66
Функциональность библиотеки chrono по вводу-выводу 68
Новые обобщенные типы std::any, std::optional и std::variant 68

3. Интерфейс всех контейнеров 78

Краткий обзор 78
Создание и удаление 79
Размер контейнеров 80
Доступ к элементам контейнера 81
Присваивание и обмен контейнеров местами 83
Сравнение контейнеров 84
Стирание контейнеров 85

4. Контейнеры последовательностей 87

Краткий обзор 87
Массивы 89

Векторы ... 90
Размер в сопоставлении с емкостью ... 91
Очередь с двусторонним доступом ... 93
Списки ... 94
Впереднаправленные списки ... 96

5. Ассоциативные контейнеры ... 99

Краткий обзор ... 99
Проверка на наличие элемента ... 101
Вставка и удаление ... 102
Упорядоченные ассоциативные контейнеры ... 103
Краткий обзор ... 103
Ключи и значения ... 103
Критерий сравнения ... 104
Специальные функции поиска ... 105
Неупорядоченные ассоциативные контейнеры ... 107
Краткий обзор ... 107
Ключи и значения ... 108
Производительность ... 108
Хеш-функция ... 109

6. Адаптеры контейнеров ... 112

Краткий обзор ... 112
Линейные контейнеры ... 112
Стек ... 113
Очередь ... 114
Очередь с приоритетом ... 115
Ассоциативные контейнеры ... 116
std::sorted_unique ... 117

7. Виды на последовательности объектов ... 118

Краткий обзор ... 118
Доступ к смежно расположенным объектам ... 118
Многомерный доступ ... 122

8. Итераторы ... 128

Краткий обзор ... 128
Категории итераторов ... 129
Критерий итератора ... 130
Полезные функции ... 131
Адаптеры итераторов ... 133
Итераторные адаптеры вставки ... 133
Потоковые итераторные адаптеры ... 134

9. Вызываемые единицы кода ... 136

Краткий обзор ... 136
Функции ... 137

Функциональные объекты 137
Предопределенные функциональные объекты 138
Лямбда-функции 139
10. Стандартные алгоритмы 140
Краткий обзор 140
Общепринятые правила 141
Итераторы как связующее звено 143
Последовательное, параллельное либо параллельно-векторизованное исполнение 143
Политики исполнения 143
Параллельные версии стандартных алгоритмов 145
for_each 146
Немодифицирующие стандартные алгоритмы 148
Поиск элементов 148
Подсчет элементов 149
Проверка условий на диапазонах 150
Сравнение диапазонов 151
Поиск диапазонов внутри диапазонов 153
Модифицирующие стандартные алгоритмы 155
Копирование элементов и диапазонов 155
Замена элементов и диапазонов 156
Удаление элементов и диапазонов 157
Заполнение и создание диапазонов 159
Перемещение диапазонов 160
Обмен диапазонами 161
Преобразование диапазонов 161
Инверсия диапазонов 162
Поворот диапазонов 163
Сдвиг диапазонов 164
Произвольная перетасовка диапазонов 165
Удаление дубликатов 166
Разбиение диапазонов 167
Сортировка 169
Двоичный поиск 171
Операции слияния 172
Кучи 176
Минимум и максимум 178
Перестановки 180
Численные алгоритмы 181
Новые параллельные алгоритмы в стандарте C++17 183
Неинициализированная память 187
11. Диапазоны 189
Краткий обзор 189
Диапазон 190

std::ranges 190
Сторожок 191
Вид 192
Адаптеры диапазонов 192
std::generator 195
Работа прямо на контейнерах 195
Композиция функций 197
Ленивое оценивание 198
Алгоритмы из std в сопоставлении с алгоритмами из std::ranges 199

12. Числа 201

Краткий обзор 201
Случайные числа 201
Генератор случайных чисел 201
Распределение случайных чисел 202
Числовые функции, унаследованные у языка C 204
Математические константы 206

13. Строковые объекты 208

Краткий обзор 208
Создание и удаление 210
Конвертация между последовательностями символов C++ и C 211
Размер в сопоставлении с емкостью 212
Сравнение 214
Конкатенация строковых объектов 214
Доступ к элементам 215
Ввод и вывод данных 216
Поиск 217
Проверка на наличие подпоследовательности символов 220
Проверка на наличие префикса или суффикса 220
Проверка на наличие содержащейся подпоследовательности символов 220
Модифицирующие операции 221
Числовые конвертации 223

14. Виды на строковые объекты 226

Краткий обзор 226
Создание и инициализация 227
Немодифицирующие операции 228
Модифицирующие операции 228

15. Регулярные выражения 231

Краткий обзор 231
Типы символов 232
Объекты регулярного выражения 233
Результат поиска match_results 234
std::sub_match 236

Совпадение 237
Поиск 238
Замена 239
Форматирование 241
Повторный поиск 242
std::regex_iterator 242
std::regex_token_iterator 243

16. Потоки ввода и вывода 245

Краткий обзор 245
Иерархия 246
Функции ввода и вывода 247
Извлечение данных (ввод) 248
Форматированное извлечение 248
Неформатированное извлечение 249
Вставка данных (вывод) 250
Конкретизатор формата 250
Потоки данных 253
Строковые потоки 253
Файловые потоки 255
Произвольный доступ 257
Состояние потока данных 258
Пользовательские типы данных 260

17. Форматирование 262

Краткий обзор 262
Форматирующие функции 262
std::vformat, std::vformat_to и std::make_format_args 263
std::print и std::println 263
Синтаксис 264
Конкретизация формата 265
Символы заполнения и выравнивание текста 265
Знак, ширина и прецизионность чисел 265
Типы данных 266
Пользовательский форматизатор 266
Форматирование контейнера std::vector 267

18. Файловая система 269

Краткий обзор 269
Классы 271
Манипулирование разрешениями на доступ к файлу 272
Функции-нечлены 273
Чтение и установка времени последней записи файла 274
Информация о пространстве в файловой системе 275
Типы файлов 276
Получение типа файла 277

19. Многопоточность 279
Краткий обзор 279
Модель памяти 279
Атомарные типы данных 280
`std::atomic_flag` 280
std::atomic 281
Фундаментальный интерфейс атомарных типов 281
Пользовательские атомарные типы std::atomic<user-defined type> 282
Все атомарные операции 285
Потоки инструкций 287
std::thread 287
Создание 287
Время жизни 288
Аргументы 289
Операции 291
Автоматическое присоединение 292
Сигнал остановки 293
std::stop_token, std::stop_source и std::stop_callback 294
Коллективные переменные 296
Гонка за данными 296
Мьютексы 298
Взаимная бокировка 300
Блокировочные замки 302
Потокобезопасная инициализация 305
Потоково-локальные данные 306
Условные переменные 307
Семафоры 310
Координационные типы 312
std::latch 312
std::barrier 314
Задания 315
Потоки инструкций в сопоставлении с заданиями 315
Синхронизация 321

20. Сопрограммы 323
Краткий обзор 323
co_yield 324
co_await 324
Типы, допускающие ожидание 325
Бесконечный поток данных посредством co_yield 325

Предметный указатель 329

От издательства

Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге – что понравилось или, может быть, не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв на нашем веб-сайте www.dmkpress.com, зайдя на страницу книги и оставив комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу dmkpress@gmail.com; при этом укажите название книги в теме письма.

Если вы являетесь экспертом в какой-либо области и заинтересованы в написании новой книги, заполните форму на нашем веб-сайте по адресу http://dmkpress.com/authors/publish_book/ или напишите в издательство по адресу dmkpress@gmail.com.

Список опечаток

Хотя мы приняли все возможные меры для того, чтобы обеспечить высокое качество наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг, мы будем очень благодарны, если вы сообщите о ней главному редактору по адресу dmkpress@gmail.com. Сделав это, вы избавите других читателей от недопонимания и поможете нам улучшить последующие издания этой книги.

Нарушение авторских прав

Пиратство в интернете по-прежнему остается насущной проблемой. Издательство «ДМК Пресс» очень серьезно относится к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в интернете с незаконной публикацией какой-либо из наших книг, пожалуйста, пришлите нам ссылку на интернет-ресурс, чтобы мы могли применить санкции.

Ссылку на подозрительные материалы можно прислать по адресу электронной почты dmkpress@gmail.com.

Мы высоко ценим любую помощь по защите наших авторов, благодаря которой мы можем предоставлять вам качественные материалы.

Об авторе

Райнер Гримм с 1999 года работает архитектором программного обеспечения, руководителем группы и преподавателем. В 2002 году он организовывал встречи стажеров компании по повышению квалификации. С 2002 года проводит обучающие курсы. Его первые учебные курсы были посвящены проприетарному управленческому программному обеспечению, но вскоре он начал преподавать языки программирования Python и C++. В свободное время любит писать статьи о языках C++, Python и Haskell. Райнеру также нравится выступать на конференциях. Еженедельно публикует посты в своем английском блоге Modernes Cpp[1], а также в немецком блоге[2], размещенном на Heise Developer.

С 2016 года является независимым преподавателем и проводит семинары по современным языкам C++ и Python. Опубликовал несколько книг на разных языках о современном C++ и, в частности, о конкурентности. В силу своей профессии всегда находится в поиске наилучшей методики преподавания современного языка C++.

[1] См. https://www.modernescpp.com/.

[2] См. https://www.grimm-jaud.de/index.php/blog.

Введение

Назначение этой книги

Книга «Стандартная библиотека C++ в примерах и пояснениях» представляет собой сжатый справочник по стандартной библиотеке текущего стандарта C++23 языка программирования C++, формально именуемого «Международным стандартом ISO/IEC 14882:2024(E) – Язык программирования C++»[1]. Стандарт C++23 содержит свыше 2100 страниц и подчиняется более крупному стандарту C++20. По сравнению с ним, стандарты C++23 и C++17 не отличаются особой величиной или малостью, а стандарт C++14 является небольшим дополнением к стандарту C++11. Стандарт C++11 содержит более 1300 страниц и был опубликован в 2011 году. Это произошло через 13 лет после выхода первого и единственного стандарта языка C++, C++98. Разумеется, еще есть стандарт C++03, который был опубликован в 2003 году. Но C++03 считается исправленным релизом.

Цель этого краткого справочника состоит в том, чтобы предоставить сжатое руководство по стандартной библиотеке C++. В книге я исхожу из того, что вы знакомы с языком C++, и в этом случае вы получите максимальную пользу от данного справочника. Если же язык C++ для вас в новинку, то рекомендуется начать с учебника по ядру языка C++. После освоения такого учебника вы будете во всеоружии сделать следующий большой шаг, прочитав эту книгу. В целях облегчения вашей работы в книге приводятся многочисленные короткие фрагменты исходного кода, которые призваны связывать теорию и практику.

Условные обозначения

В книге используется всего несколько типографских условных обозначений.

Особые шрифты

курсивный шрифт

используется в случаях, когда нужно выделить что-то существенное.

`моноширинный жирный шрифт`

используется в исходном коде, инструкциях, ключевых словах и именах типов, переменных, функций и классов.

Особые значки

Используются для выделения уникальной информации, подсказок и предупреждений.

---

[1] См. https://www.iso.org/standards.html.

Заголовок информации

Текст информации.

Заголовок подсказки

Описание подсказки.

Заголовок предупреждения

Описание предупреждения.

Примеры исходного кода

Я не очень люблю использовать директивы и объявления, потому что они скрывают именное пространство библиотеки. В книге они используются таким образом, чтобы происхождение всегда можно было выводить из директивы (`using namespace std;`) или объявления *using* (`using std::cout;`). Тем не менее ввиду ограниченного объема страницы их приходилось время от времени использовать.

Во фрагментах исходного кода показаны только заголовочные файлы избранной функциональности. Для булевых величин показывается `true` либо `false`, а манипулятор ввода-вывода `std::boolalpha` не используется. Если ваш компилятор поддерживает модульную организацию стандартной библиотеки по стандарту C++23, то заголовки можно заменить инструкцией `import std`.

Исходный код

Ради лаконичности в этой книге приводятся только короткие фрагменты исходного кода. Название всей программы находится в первой строке фрагмента.

Значение и объект

Экземпляры фундаментальных типов данных в книге называются значениями. Такое правило было унаследовано языком C++ у языка C. Экземпляры более сложных типов или классов, которые нередко состоят из фундаментальных типов, называются объектами. Объектами обычно являются экземпляры пользовательских типов, классов или контейнеров.

Признательности

Прежде всего хотел бы поблагодарить лектора издательства O'Reilly Александру Фоллениус за немецкую версию книги «C++ Standardbibliothek – kurz & gut»[1]. Данная версия является прародителем книги, которую вы держите в руках. Неоценимую работу по корректуре книги оказали Карстен Анерт, Гунтрам

[1] См. http://shop.oreilly.com/product/9783955619688.do.

Берти, Дмитрий Ганюшин, Свен Йоханнсен, Торстен Робицки, Барт Вандевостейн и Феликс Винтер. Всем им огромное спасибо.

В своем англоязычном блоге www.ModernesCpp.com[1] я разместил запрос на перевод книги на английский язык и в результате получил гораздо больше откликов, чем ожидал. Особая благодарность всем вам, включая моего сына Мариуса, который вычитывал ее первым.

Вот имена в алфавитном порядке: Махеш Аттарде, Рик Ауде, Пит Барроу, Майкл Бен-Дэвид, Дэйв Бернс, Альваро Фернандес, Джульетта Гримм, Джордж Хааке, Клэр Макрэй, Арне Мерц, Ян Рив, Джейсон Тернер, Барт Вандевоэстин, Иван Вергилиев и Анджей Варжинский.

Дальнейшая информация

Идею книги довольно легко перефразировать следующим образом: это «все, что должен знать каждый профессиональный программист на C++ о стандартной библиотеке C++». Такой замысел книги является причиной того, что многие вопросы остались без ответа, поэтому в начале каждой новой темы даются ссылки на подробную информацию. Ссылка будет перенаправлять к отличному онлайновому ресурсу en.cppreference.com[2,3].

Циппи

А теперь позвольте представить вам Циппи. Циппи будет сопровождать вас по всей книге. Надеюсь, она вам понравится.

Меня зовут Циппи.
Я – любопытная, умная и,
конечно же, женственная!

[1] См. http://www.modernescpp.com/index.php/do-you-wan-t-to-proofread-a-book.

[2] См. http://en.cppreference.com/w/.

[3] Как вариант https://cplusplus.com/. – *Прим. перев.*

1. Стандартная библиотека

Краткий обзор

Стандартная библиотека C++ представляет собой набор предопределенных типов, классов и функций, которые обеспечивают главные функциональности программ на C++, упрощают решение общих задач программирования и повышают эффективность разработки программного обеспечения. Она состоит из многочисленных компонентов, включая библиотеку ввода-вывода, библиотеку контейнеров, библиотеку стандартных алгоритмов, библиотеку по работе со строковыми объектами, библиотеку по работе с числами и библиотеку вспомогательных инструментов, где содержатся дополнительные вспомогательные средства, такие как умные указатели и функциональные объекты.

Эта глава служит двум целям. Она призвана дать краткое изложение ее функциональных возможностей и первое представление о том, как их использовать.

История

У языка программирования C++ и, соответственно, у его стандартной библиотеки долгая история. C++ появился в 1980-х годах прошлого тысячелетия и эволюционно пока что остановился на 2023 году. Любой, кто знаком с разработкой программного обеспечения, знает о той скорости, с которой развивается наша область. Так что 40 лет – это очень большой срок. Возможно, вас не очень удивит тот факт, что первые компоненты C++, такие как потоки ввода-вывода данных, были сконструированы с другим мышлением, нежели в современной стандартной библиотеке шаблонов (STL, от англ. Standard Template Library). Язык C++ начинался как объектно ориентированный, включал в свой состав обобщенное программирование посредством стандартной библиотеки шаблонов, а ныне перенял многие идеи функционального программирования. Эта эволюция развития программного обеспечения за последние 40 лет, которую можно наблюдать в стандартной библиотеке C++, также отражает и то, как решаются задачи программной реализации.

Первая стандартная библиотека стандарта C++98 была выпущена в 1998 году и состояла из трех компонентов. Это были упомянутые ранее потоки ввода-вывода данных, в основном для работы с файлами, библиотека для работы со строковыми объектами и стандартная библиотека шаблонов (STL). Стандартная библиотека шаблонов способствует прозрачному применению алгоритмов на контейнерах.

Хронология языка C++

История продолжается в 2005 году стандартом ISO/IEC TR 19768 под общим названием «Технический отчет №1» (TR1). Расширение библиотеки C++ по стандарту ISO/IEC TR 19768 не было официальным, но почти все компоненты вошли в состав стандарта C++11, например библиотеки для работы с регулярными выражениями, умными указателями, хеш-таблицами, случайными числами и временем, основанные на библиотеках boost (http://www.boost.org/).

В качестве дополнения к стандартизации TR1 стандарт C++11 получил один новый компонент: библиотеку для работы с многопоточностью.

Стандарт C++14 был лишь незначительным обновлением стандарта C++11, поэтому в рамках C++14 было добавлено лишь несколько улучшений к существующим библиотекам по работе с умными указателями, кортежами, признаками типов и многопоточностью.

В стандарт C++17 вводятся библиотеки для работы с файловой системой и два новых типа данных: `std::any` и `std::optional`.

В стандарте C++20 появляются четыре выдающиеся функциональные возможности: концепты, диапазоны, сопрограммы и модули. Помимо этой большой четверки, в C++20 есть и другие жемчужины: оператор трехпутного сравнения, библиотека для работы с форматированием и типы данных, связанные с конкурентностью, – семафоры, защелки и барьеры.

В стандарте C++23 была улучшена большая четверка стандарта C++20: расширена функциональность диапазонов, появился генератор сопрограмм `std::generator`, а также модульная стандартная библиотека C++.

Краткий обзор

Поскольку в языке C++ существует большое число библиотек, зачастую довольно трудно найти библиотеку, которая удобно подходила бы для каждого варианта использования.

Вспомогательные инструменты

Вспомогательные инструменты (утилиты) – это, по сути дела, общецелевые библиотеки, которые могут применяться во многих контекстах.

Примерами вспомогательных инструментов являются функции вычисления минимума или максимума значений, срединной точки двух значений, а также перестановки значений местами или их перемещения. Благодаря безопасному сравнению целых чисел неявного приведения типов не происходит.

Другими вспомогательными инструментами являются полиморфная функциональная обертка `std::function`, вспомогательные функции `std::bind` и `std:bind_front`. С помощью функций `std::bind` или `std::bind_front` можно легко создавать новые функции из существующих. Для их связывания с переменной и последующего вызова необходима полиморфная функциональная обертка `std::function`.

С помощью вспомогательного шаблонного класса `std::pair` и его обобщения `std::tuple` можно создавать разнородные пары и кортежи произвольной длины.

На практике довольно удобны ссылочные функциональные обертки `std::ref` и `std::cref`. С их помощью можно создавать ссылочную обертку для переменной, которая для `std::cref` является константой.

Конечно же, главной изюминкой вспомогательных инструментов в языке C++ являются умные указатели. Они позволяют автоматически и в явной форме управлять памятью. Используя уникальный умный указатель `std::unique_ptr`, можно моделировать концепцию явного владения, а с помощью коллективного умного указателя `std::shared_ptr` – моделировать коллективное владение. Коллективный умный указатель `std::shared_ptr` задействует подсчет ссылок, беря на себя заботы о своем ресурсе. Третий вариант, слабый указатель `std::weak_ptr`, помогает избавляться от циклических зависимостей между коллективными указателями `std::shared_ptr`. Циклические ссылки являются классической проблемой подсчета ссылок.

Библиотека `type_traits` для работы с признаками (чертами) типов позволяет проверять, сравнивать и манипулировать информацией о типах во время компиляции.

Библиотека `chrono` для работы со временем является импортируемым дополнением к новым способностям C++ в отношении многопоточности. Но она также довольно удобна для измерения производительности и включает поддержку календаря и часовых поясов.

Благодаря обобщенным типам `std::any`, `std::optional` и `std::variant` в стандарте C++17 появляются три специальных типа данных, которые могут иметь любое значение, опциональное значение либо вариант значений.

Стандартная библиотека шаблонов

Стандартная библиотека шаблонов (STL) в языке C++ – это мощная библиотека, предлагающая алгоритмы и структуры данных, которые облегчают прозрачное применение алгоритмов на контейнерах. Она предоставляет набор

распространенных алгоритмов, контейнеров, итераторов и функциональных объектов, служащих для повышения эффективности и простоты программирования на C++.

Три компонента стандартной библиотеки шаблонов

Если смотреть на стандартную библиотеку шаблонов с высоты птичьего полета, то она состоит из трех компонентов. Это контейнеры, алгоритмы и итераторы. При этом алгоритмы работают на контейнерах, а итераторы связывают их между собой. Контейнеры предъявляют лишь минимальные требования к своим элементам. Такая абстракция обобщенного программирования позволяет комбинировать алгоритмы и контейнеры уникальным образом.

В стандартной библиотеке C++ имеется богатая коллекция контейнеров, включая контейнеры последовательностей и ассоциативные контейнеры. Ассоциативные контейнеры классифицируются на упорядоченные и неупорядоченные.

Каждый контейнер последовательности имеет уникальную область применения. Тем не менее в 95 % случаев использования контейнер `std::vector` является верным выбором. Он может динамически корректировать свой размер, автоматически управляет памятью и обеспечивает отличную производительность. Напротив, контейнер `std::array` представляет собой единственный контейнер последовательностей, который не может корректировать свой размер во время исполнения. Он оптимизирован под минимальные накладные расходы на память и издержки производительности. Контейнер `std::vector` отличается тем, что помещает новые элементы в конец контейнера. С другой стороны, для размещения элемента в начале следует использовать контейнер `std::deque`. Два дополнительных контейнера – контейнер `std::list` как двусвязный список и контейнер `std::forward_list` как односвязный список – оптимизированы под высокопроизводительные операции в произвольных позициях в контейнере.

Ассоциативные контейнеры являются контейнерами пар ключ-значение. Они предоставляют свои значения по соответствующему ключу. Типичным примером использования ассоциативного контейнера является телефонный справочник, в котором ключ «фамилия» используется, чтобы извлекать значение «номер телефона». В языке C++ есть восемь ассоциативных контейнеров. С одной стороны, это ассоциативные контейнеры с упорядоченными ключами: `std::set`, `std::map`, `std::multiset` и `std::multimap`. С другой, неупорядоченные ассоциативные контейнеры: `std::unordered_set`, `std::unordered_map`, `std::unordered_multiset` и `std::unordered_multimap`.

Прежде всего, говоря об упорядоченных ассоциативных контейнерах, разница между контейнерами `std::set` и `std::map` заключается в том, что у первого нет ассоциированного значения. Разница между контейнерами `std::map`