

Вопросы теоретического минимума¹

1. Что определяет класс? Чем отличается класс от объекта?
2. Можно ли объявлять массив объектов? А массив классов?
3. Разрешается ли объявлять указатель на объект? А указатель на класс?
4. Допускается ли передавать объекты в качестве параметров, и какими способами? А возвращать как результат?
5. Как называется использование объекта одного класса в качестве поля другого класса?
6. Является ли структура классом? Чем класс отличается от структуры?
7. Какие ключевые слова в C++ обозначают класс?
8. Объясните принцип инкапсуляции.
9. Что такое композиция?
10. Для чего используются ключевые слова **public** и **private**?
11. Можно ли использовать ключевые слова **public** и **private** в структуре?
12. Существуют ли ограничения на использование **public** и **private** в классе? А в структуре?
13. Обязательно ли делать поля класса приватными?
14. Что такое метод? Как вызывается метод?
15. Может ли метод быть приватный?
16. Как определить метод непосредственно внутри класса? А вне класса? Чем эти определения отличаются?
17. Можно в методах присваивать параметрам значения по умолчанию?
18. Что обозначается ключевым словом **this**?
19. Зачем нужны константные методы? Чем отличается определение константного метода от обычного?
20. Может ли константный метод вызываться для объектов-переменных? А обычный метод — для объектов-констант?
21. Объясните принцип полиморфизма.
22. Сколько места в памяти занимает объект класса? Как это узнать?
23. Каков размер «пустого» объекта?
24. Влияют ли методы на размер объекта?
25. Одинаков ли размер класса и аналогичной структуры?
26. Какие операции нельзя перегружать? Как вы думаете, почему?
27. Можно ли перегружать операции для встроенных типов данных?
28. Можно ли при перегрузке изменить приоритет операции?
29. Можно ли определить новую операцию?
30. Перечислите особенности перегрузки операций как методов класса. Чем отличается перегрузка внешним образом от перегрузки как метода класса?
31. Какой результат должны возвращать операции с присваиванием?
32. Как различаются перегруженная префиксная и постфиксная операции инкремента и декремента?
33. Что означает выражение ***this**? В каких случаях оно используется?

¹Ссылка на оригинал — <http://www.rsdn.ru/forum/cpp/1870577.1.aspx>

34. Какие операции не рекомендуется перегружать как методы класса? Почему?
35. Какие операции разрешается перегружать только как методы класса?
36. Дайте определение дружественной функции. Как объявляется дружественная функция? А как определяется?
37. Дайте определение конструктора. Каково назначение конструктора? Перечислите отличия конструктора от метода.
38. Сколько конструкторов может быть в классе? Допускается ли перегрузка конструкторов? Какие виды конструкторов создаются по умолчанию?
39. Может ли конструктор быть приватным? Какие последствия влечет за собой объявление конструктора приватным?
40. Приведите несколько случаев, когда конструктор вызывается неявно.
41. Как проинициализировать динамическую переменную?
42. Как объявить константу в классе? Можно ли объявить дробную константу?
43. Каким образом разрешается инициализировать константные поля в классе?
44. В каком порядке инициализируются поля в классе? Совпадает ли этот порядок с порядком перечисления инициализаторов в списке инициализации конструктора?
45. Какие конструкции C++ разрешается использовать в списке инициализации в качестве инициализирующих выражений?
46. Какой вид конструктора фактически является конструктором преобразования типов?
47. Для чего нужны функции преобразования? Как объявить такую функцию в классе?
48. Как запретить неявное преобразование типа, выполняемое конструктором инициализации?
49. Какие проблемы могут возникнуть при определении функций преобразования?
50. Для чего служит ключевое слово **explicit**?
51. Влияет ли наличие целочисленных констант-полей на размер класса?
52. Разрешается ли объявлять массив в качестве поля класса. Как присвоить элементам массива начальные значения?
53. Сколько операндов имеет операция индексирования `[]`? Какой вид результата должна возвращать эта операция?
54. Для чего нужны статические поля в классе? Как они определяются?
55. Как объявить в классе и проинициализировать статический константный массив?
56. Что такое выравнивание и от чего оно зависит? Влияет ли выравнивание на размер класса?
57. Дайте определение контейнера.
58. Какие виды встроенных контейнеров в C++ вы знаете?
59. Какие виды доступа к элементам контейнера вам известны?
60. Чем отличается прямой доступ от ассоциативного?
61. Перечислите операции, которые обычно реализуются для последовательного доступа к элементам контейнера.
62. Дайте определение итератора.
63. Можно ли реализовать последовательный доступ без итератора? В чем преимущества реализации последовательного доступа с помощью итератора?
64. Что играет роль итератора для массивов C++?

65. Что такое деструктор? Может ли деструктор иметь параметры?
66. Почему для классов-контейнеров деструктор надо писать явным образом?
67. Допускается ли перегрузка деструкторов?
68. Что такое «глубокое копирование» и когда в нем возникает необходимость?
69. Какое копирование осуществляет стандартный конструктор копирования?
70. Чем отличается копирование от присваивания?
71. Объясните, почему в операции присваивания требуется проверка присваивания самому себе?
72. Можно ли в качестве операции индексирования использовать операцию вызова функции `()`? В чем ее преимущества перед операцией `[]`?
73. Почему необходимо писать два определения операции индексирования? Чем они отличаются?
74. Дайте определение вложенного класса.
75. Можно ли класс-итератор реализовать как внешний класс? А как вложенный? В чем отличия этих методов реализации?
76. Может ли объемлющий класс иметь неограниченный доступ к элементам вложенного класса? А вложенный класс — к элементам объемлющего?
77. Ограничена ли глубина вложенности классов?
78. Можно ли определить вложенный класс внешним образом? Зачем это может понадобиться?
79. Каким образом вложенный класс может использовать методы объемлющего класса? А объемлющий — методы вложенного?
80. Что такое «запредельный» элемент, какую роль он играет в контейнерах?
81. Объясните, по каким причинам трудно написать универсальный контейнер, элементы которого могут иметь произвольный тип.
82. Назовите ключевые слова `C++`, которые используются для обработки исключений.
83. Исключение — это:
84. Каким образом исключение генерируется?
85. Каковы функции контролируемого блока?
86. Что обозначается ключевым словом **`catch`**?
87. Какого типа может быть исключение?
88. Сколько параметров разрешается писать в заголовке секции-ловушки?
89. Какими способами разрешается передавать исключение в блок обработки?
90. Объясните, каким образом преодолеть ограничение на передачу единственного параметра в блок обработки.
91. Почему нельзя выполнять преобразования типов исключений при передаче в секцию-ловушку?
92. Напишите конструкцию, которая позволяет перехватить любое исключение.
93. Могут ли контролируемые блоки быть вложенными?
94. Зачем нужен «контролируемый блок-функция» и чем он отличается от обычного контролируемого блока?
95. Перечислите возможные способы выхода из блока обработки.
96. Каким образом исключение «передать дальше»?
97. Сколько секций-ловушек должно быть задано в контролируемом блоке?

98. Что такое «спецификация исключений»?
99. Что происходит, если функция нарушает спецификацию исключений?
100. Учитывается ли спецификация исключений при перегрузке функций?
101. Что такое «иерархия исключений»?
102. Существуют ли стандартные исключения? Назовите два-три типа стандартных исключений.
103. Поясните «взаимоотношение» исключений и деструкторов.
104. Объясните, зачем может понадобиться подмена стандартных функций завершения.
105. Какие виды нестандартных исключений вы знаете?
106. В чем отличие механизма структурной обработки исключений **Windows** от стандартного механизма?
107. Какие две роли выполняет наследование?
108. Какие виды наследования возможны в C++?
109. Чем отличается модификатор доступа **protected** от модификаторов **private** и **public**?
110. Чем открытое наследование отличается от закрытого и защищенного?
111. Какие функции не наследуются?
112. Сформулируйте правила написания конструкторов в производном классе.
113. Каков порядок вызова конструкторов? А деструкторов?
114. Можно ли в производном классе объявлять новые поля? А методы?
115. Если имя нового поля совпадает с именем унаследованного, то каким образом разрешить конфликт имен?
116. Что происходит, если имя метода-наследника совпадает с именем базового метода?
117. Сформулируйте принцип подстановки.
118. Когда выполняется понижающее приведение типов?
119. Объясните, что такое «срезка» или «расщепление».
120. Объясните, зачем нужны виртуальные функции.
121. Что такое связывание?
122. Чем «раннее» связывание отличается от «позднего»?
123. Какие два вида полиморфизма реализованы в C++?
124. Дайте определение полиморфного класса.
125. Может ли виртуальная функция быть дружественной функцией класса?
126. Наследуются ли виртуальные функции?
127. Каковы особенности вызова виртуальных функций в конструкторах и деструкторах?
128. Можно ли сделать виртуальной перегруженную операцию, например, сложение?
129. Может ли конструктор быть виртуальным? А деструктор?
130. Как виртуальные функции влияют на размер класса?
131. Как объявляется «чистая» виртуальная функция?
132. Дайте определение абстрактного класса.
133. Наследуются ли чистые виртуальные функции?

134. Можно ли объявить деструктор чисто виртуальным?
135. Чем отличается чистый виртуальный деструктор от чистой виртуальной функции?
136. Зачем требуется определение чистого виртуального деструктора?
137. Наследуется ли определение чистой виртуальной функции?
138. Приведите классификацию целей наследования.
139. Объясните разницу наследования интерфейса от наследования реализации.
140. Назовите причины, требующие разделения программ на части.
141. Дайте определение термина «единица трансляции»?
142. Чем отличается файл с исходным текстом от единицы трансляции?
143. Существуют ли в C++ конструкции, позволяющие идентифицировать отдельный модуль?
144. Какие способы сборки программы вы можете назвать?
145. Что такое «объектный модуль»? Программа, которая «собирает» объектные модули в программу, называется?
146. В чем заключается отличие аргумента «файл» от <файл> в директиве #include?
147. Что такое ODR?
148. Объясните, что такое «страж» включения и зачем он нужен.
149. Является ли интерфейс класса его определением?
150. Сколько определений класса может быть в единице трансляции?
151. Сколько определений класса может быть в многофайловой программе?
152. Чем отличаются стандартные заголовки <string>, <string.h> и <cstring>?
153. Объясните суть идиомы Pimpl.
154. Что такое делегирование и как его можно использовать для повышения степени инкапсуляции?
155. Каким образом глобальную переменную, определенную в одной единице трансляции, сделать доступной в другой единице трансляции? А константу?
156. Можно ли использовать слово extern при объявлении функций?
157. Как локализовать объявление функции в файле?
158. Чем отличается «внешнее» связывание от «внутреннего» связывания?
159. Что такое «спецификации компоновки»?
160. Какие объекты обладают внутренним связыванием по умолчанию?
161. Какие области видимости имен вы знаете?
162. Для чего используются пространства имен?
163. Чем отличаются именованные и неименованные пространства имен?
164. Могут ли пространства имен быть вложенными?
165. Для чего применяются алиасы пространства имен?
166. Как сделать члены пространства имен доступными в нескольких (в пределе — во всех) файлах программного проекта?
167. Объясните разницу между статической и динамической инициализацией.
168. В чем состоит проблема инициализации глобальных статических переменных?

169. Какие элементы класса можно объявлять статическими?
170. Можно ли объявить в классе статическую константу? А константный статический массив?
171. А какие статические поля можно инициализировать непосредственно в классе?
172. Как определяются статические поля? В какой момент работы программы выполняется инициализация статических полей?
173. Сколько места в классе занимают статические поля ?
174. Чем отличается статический метод от обычного?
175. Какие методы класса не могут быть статическими?
176. Какие применения статических полей вы можете привести? А каким образом применяются статические методы?
177. Приведите структуру и принцип действия паттерна Singleton.
178. Для чего предназначены шаблоны?
179. Какие виды шаблонов в C++ вы знаете?
180. Объясните термин «инстанцирование шаблона».
181. В чем разница между определением и объявлением шаблона?
182. Объясните назначение ключевого слова `typename`.
183. Какие виды параметров разрешается задавать в шаблоне класса? А в шаблоне функции?
184. Можно ли параметрам шаблона присваивать значения по умолчанию?
185. Может ли параметром шаблона быть другой шаблон? Каковы особенности объявления параметра-шаблона?
186. Что такое специализация шаблона? Объясните разницу между полной и частичной специализацией.
187. Разрешается ли специализировать шаблон функции?
188. Может ли класс-шаблон быть вложенным в другой класс-шаблон? А в обычный класс?
189. Можно ли объявить в классе шаблонный метод? А шаблонный конструктор?
190. Можно ли перегружать функцию-шаблон?
191. Какие параметры функции-шаблона выводятся автоматически?
192. Может ли шаблон класса быть наследником обычного класса? А обычный класс от шаблона?
193. Объясните, что такое класс свойств (класс трактовок).
194. Каким образом можно использовать возможность наследования обычного класса от шаблона?
195. Может ли шаблонный конструктор быть конструктором по умолчанию?
196. Для чего применяются директивы явного инстанцирования?
197. Объясните, в чем состоят проблемы, возникающие при разделении шаблонного класса на интерфейс и реализацию?
198. Что такое «модель явного инстанцирования» и как она работает?
199. Может ли шаблонный класс иметь «друзей»?
200. Какие проблемы возникают при объявлении дружественной функции для класса-шаблона?
201. Разрешается ли определять в классе-шаблоне статические поля? А статические методы?
202. Что такое «инициализация нулем»?

203. Что является единицей памяти в C++? Какие требования к размеру единицы памяти прописаны в стандарте C++?
204. В каких единицах выдает результат операция sizeof? Какие типы данных имеют размер 1?
205. Какие три вида памяти входят в модель памяти C++?
206. Сколько видов динамической памяти обеспечивает C++?
207. Какие функции для работы с динамической памятью достались C++ по наследству от C? В какую библиотеку они включены?
208. Какие функции выделяют память, и с помощью каких функций память освобождается?
209. Какое важное отличие имеет функция calloc() от функции malloc()?
210. Какие действия выполняют функции выделения памяти, если память не может быть выделена?
211. Зависит ли объем выделенной памяти от типа указателя? Влияет ли выравнивание на объем выделяемой динамической памяти?
212. Можно ли с помощью функции realloc() уменьшить объем выделенной памяти?
213. Что произойдет, если функции free() передать в качестве аргумента нулевой указатель?
214. В чем главное отличие объектно-ориентированного механизма new/delete от механизма malloc()/free()?
215. Сколько существует форм new/delete? В чем их отличие?
216. Какие типы являются POD-типами? Чем отличается работа механизма new/delete с POD-объектами и nonPOD-объектами?
217. Какие функции выполняет обработчик new?
218. Можно ли реализовать собственный обработчик new и «прицепить» его к механизму new/delete?
219. В чем главное отличие объединения от других видов классов C++?
220. Может ли объединение участвовать в иерархии наследования?
221. Разрешается ли определять для объединения конструкторы и деструктор? А виртуальные функции?
222. В чем похожи и чем отличаются объединение и размещающий new?
223. Объясните, почему при использовании размещающего new нужно явным образом вызывать деструктор?
224. Зачем нужны интеллектуальные указатели?
225. Что такое «стратегия владения»? Сколько стратегий владения вы знаете?
226. Какой интеллектуальный указатель реализован в стандартной библиотеке STL, и какую стратегию владения он реализует?
227. Объясните, в чем преимущества и недостатки интеллектуальных указателей со счетчиком ссылок.
228. Разрешается ли перегружать new и delete и какими способами?
229. Опишите схему функции, перегружающей глобальную функцию new.
230. Отличается ли реализация перегруженной функции new[]() для массивов от реализации «обычной» функции new()?
231. Как вы думаете, почему функции new/delete, перегружаемые для класса, являются статическими?
232. Зачем при перегрузке new/delete для класса нужно проверять размер запрашиваемой памяти?
233. Объясните, чем определяется «динамичность» контейнеров?
234. Что такое «стратегия распределения памяти», и какие стратегии выделения памяти вы знаете?

235. Рассмотрите следующую стратегию распределения памяти: память выделяется для нескольких элементов блоками фиксированной длины, но блоки связываются в список. Для какого вида контейнера можно использовать такую стратегию?
236. Какие операции можно перегрузить для доступа к элементам двумерного массива?
237. В чем заключаются сложности использования операции индексирования [] для доступа к элементам двумерного массива?
238. Каковы способы реализации операций с контейнерами?
239. Какую конструкцию можно назвать «обобщенный алгоритм»?
240. Каким образом объявить указатель на метод?
241. Объясните разницу между указателем на функцию и указателем на метод.
242. Каким образом получить адрес метода?
243. Можно ли указателю на функцию присваивать адрес метода?
244. Какие операции определены в C++ для косвенного вызова метода через указатель?
245. Что такое «функтор»? Приведите пример функционального класса.
246. Какими способами функтор вызывается?
247. Можно ли использовать наследование при разработке функторов?
248. Разрешается ли операцию вызова функции () определять как виртуальный метод? А как статический?
249. В чем преимущества функторов перед указателями на функции?
250. Объясните, зачем нужны адаптеры функторов? Какие виды адаптеров вы знаете?
251. Как используются классы свойств при разработке функторов?
252. Объясните, что такое «композиция» и приведите примеры?
253. Объясните, чем отличается множественное наследование от простого?
254. Приведите структуру и принцип действия паттерна Adapter.
255. Сформулируйте основную проблему множественного наследования.
256. Выполняется ли принцип подстановки при открытом множественном наследовании?
257. Что такое виртуальное наследование? Каковы его преимущества и недостатки по сравнению с обычным наследованием?
258. Может ли виртуальное наследование быть одиночным?
259. Влияет ли виртуальное наследование на размер класса?
260. Объясните, каким образом с помощью виртуального наследования можно вообще запретить наследование.
261. Какие средства C++ составляют RTTI?
262. Объясните разницу между повышающим, понижающим и перекрестным приведением.
263. Какими свойствами должен обладать класс, чтобы с ним работал механизм RTTI?
264. В чем приведение указателей отличается от приведения ссылок?
265. Какие исключения связаны с механизмом RTTI?
266. Что такое «поток» — дайте определение.
267. Как классифицируются потоки, реализованные в библиотеках ввода/вывода C++?
268. Что такое буферизация и зачем она нужна?

269. Какие библиотеки ввода/вывода реализованы в C++ и чем они отличаются?
270. Перечислите стандартные потоки и объясните их назначение.
271. Зачем нужен процесс форматирования и когда он выполняется?
272. Что такое «форматная строка», и в каких функциях она используется?
273. Объясните назначение элементов спецификатора формата.
274. Сколько спецификаторов формата может быть в форматной строке?
275. Какой из элементов спецификатора формата не является умалчиваемым?
276. Перечислите несколько известных вам обозначений типов в спецификаторе формата, и укажите их назначение.
277. Сколько модификаторов типа вы знаете, и какую роль модификатор типа играет в спецификаторе формата?
278. С помощью какого флага можно выровнять выводимое значение влево? А каким образом вывести ведущие нули?
279. Какое действие оказывают на выводимую строку ширина, точность и флаги в спецификаторе формата?
280. Для чего в спецификаторе формата может использоваться символ звездочка («*»)? Чем отличается действие этого символа при вводе и при выводе?
281. Каковы особенности ввода строк?
282. Каким образом ограничить набор вводимых символов при вводе?
283. Что является главной проблемой при использовании форматного ввода/вывода из библиотеки `<cstdio>`?
284. Объясните, для чего нужны строковые потоки. Почему строковые потоки — всегда форматируемые?
285. С помощью каких функций выполняется работа со строковыми потоками?
286. Можно ли использовать тип `string` (и каким образом) со строковыми потоками?
287. Объясните, в чем заключается различие между текстовым и двоичным файлом.
288. Объясните, что означает «открыть» файл и «закрыть» файл?
289. Каким образом внешний файл связывается с потоком?
290. Можно ли один и тот же поток связать с разными файлами? А один и тот же файл с разными потоками?
291. Перечислите режимы открытия файла. Чем отличается режим `"r"` от режима `"a"`?
292. Какую роль в режиме открытия играет знак плюс («+»)?
293. В каких случаях необходимо следить за ситуацией «конец файла»? Каким способом это делается?
294. Можно ли текстовый файл открыть как двоичный? А двоичный — как текстовый?
295. Какие функции ввода/вывода используются для обмена с текстовыми файлами?
296. Перечислите функции ввода/вывода для работы с двоичными файлами.
297. Какие функции реализованы в библиотеке `<cstdio>` для обеспечения прямого доступа к записям двоичного файла? Можно ли их использовать для работы с текстовыми файлами?
298. Объясните назначение функции `fseek()`.
299. Чем отличается функция `ftell()` от функции `fgetpos()`?
300. Объясните, что означает «перенаправление» потока? Какие потоки можно перенаправлять и куда?
301. Каким образом перенаправление ввода можно использовать для ввода строк с пробелами?
302. В чем преимущества объектно-ориентированной библиотеки по сравнению с процедурной?

303. В каких состояниях может находиться поток? Каким образом отслеживается состояние «конец потока»?
304. Какие объектно-ориентированные потоки связаны со стандартными потоками?
305. Чем отличаются объектно-ориентированные строковые потоки от процедурных строковых потоков?
306. Каким образом строковые потоки можно использовать для ограничения ширины поля ввода? А можно ли с той же целью использовать строковые потоки `<cstdio>`?
307. Сравните средства форматирования объектно-ориентированной и процедурной библиотеки.
308. Каким образом ввести строку типа `string` с пробелами?
309. Каково назначение флагов форматирования? Какие средства реализованы в библиотеке для работы с флагами форматирования?
310. Что такое «манипулятор»? В чем преимущества манипуляторов перед флагами форматирования?
311. Как связываются файлы с потоками в объектно-ориентированной библиотеке?
312. Можно ли файлы, записанные функциями библиотеки `<cstdio>`, прочитать объектно-ориентированными средствами? А наоборот?
313. Перечислите режимы открытия объектно-ориентированных файловых потоков. каким образом комбинируются режимы открытия файловых потоков?
314. Обязательно ли закрывать файл, связанный с объектно-ориентированным файловым потоком? А открывать?
315. Каким образом открыть файловый поток для чтения и записи одновременно?
316. Как открыть файловый поток для дозаписи?
317. Можно ли вывести значение переменной в двоичном виде и как это сделать?
318. Разрешается ли наследовать от классов библиотеки ввода/вывода?
319. Каким образом можно перенаправить объектно-ориентированный поток?
320. Как используется буфер потока для копирования потока?
321. Какими операциями выполняется форматированный ввод/вывод в файловые потоки? А неформатированный?
322. Реализованы ли в объектно-ориентированной библиотеке средства прямого доступа к файловым потокам? Сравните их с аналогичными средствами библиотеки `<cstdio>`.
323. С какими объектно-ориентированными потоками разрешается, и с какими не разрешается использовать средства прямого доступа?
324. Покажите, каким образом можно выполнить перегрузку операций ввода/вывода для нового типа данных.
325. Как выполняется обработка ошибок ввода/вывода в объектно-ориентированной библиотеке?
326. Какое стандартное исключение генерируется при ошибках ввода/вывода? Обязательно ли оно генерируется?
327. Чем стандартные широкие потоки отличаются от узких?
328. Что такое — «локаль», и каково ее назначение?
329. Как установить русский шрифт при выводе в консольное окно?
330. Чем отличается ли ввод/вывод широких файловых потоков от узких?
331. Перечислите все последовательные контейнеры стандартной библиотеки. Чем они отличаются друг от друга?
332. Перечислите адаптеры последовательных контейнеров и дайте их подробную характеристику.
333. Почему для адаптеров-очередей нельзя использовать вектор в качестве базового?
334. Чем простая очередь `queue` отличается от приоритетной очереди `priority_queue`?

335. Каким требованиям должны удовлетворять элементы контейнера?
336. Могут ли быть указатели элементами контейнера? А итераторы?
337. Почему нельзя использовать в качестве элементов контейнера стандартный интеллектуальный указатель `auto_ptr`?
338. Зачем в контейнере `list` реализованы собственные методы сортировки поиска и слияния? Можно ли пользоваться соответствующими стандартными алгоритмами при обработке списка?
339. Перечислите типовые виды конструкторов, с помощью которых можно создавать последовательный контейнер.
340. Можно ли инициализировать контейнер элементами встроеного массива? А элементами другого контейнера? Какими способами это можно сделать?
341. Почему конструктор инициализации, параметрами которого являются итераторы, сделан шаблонным во всех контейнерах?
342. Какие методы реализованы в контейнере-векторе для доступа к элементам?
343. Отличается ли функция `at()` доступа по индексу от перегруженной операции индексирования и чем?
344. Перечислите методы контейнера `deque`, относящиеся к определению размеров контейнера.
345. Чем метод `size()` отличается от метода `capacity()`? А в чем отличие этих методов от метода `max_size()`?
346. Перечислите методы контейнера `list`, предназначенные для вставки удаления и замены элементов. Отличаются ли эти методы от соответствующих методов вектора и дека?
347. Каким образом выполняются операции сравнения контейнеров?
348. Разрешается ли изменять элемент ассоциативного контейнера, доступный в данный момент по итератору?
349. Какие контейнеры называются ассоциативными и почему?
350. Чем контейнер `map` отличается от контейнера `multimap`?
351. Объясните, почему в ассоциативных контейнерах нельзя изменять элемент, доступный в данный момент по итератору.
352. По каким причинам в контейнере-множестве не реализованы типовые операции объединения, пересечения, разности и другие?
353. Как используется структура-пара в ассоциативных контейнерах?
354. Объясните, что такое «критерий сортировки», и каким требованиям он должен удовлетворять? Какой критерий сортировки принят по умолчанию?
355. Какими преимуществами обладает функция `make_pair()` по сравнению с конструктором `pair()`?
356. Почему в контейнерах-отображениях операция индексирования перегружена, а в контейнерах-множествах — нет?
357. Какие гарантии безопасности обеспечивают контейнеры стандартной библиотеки?
358. Что такое «транзакционная гарантия безопасности» и чем она отличается от базовой?
359. На какие 4 класса по надежности можно разделить все операции с контейнерами?
360. Что такое «распределитель памяти» и зачем он нужен?
361. Чем отличается битовый вектор `bitset` от битового вектора `vector<bool>`?
362. Дайте определение итератора.
363. Что такое «начальный» итератор и «конечный» итератор? Какие методы, связанные с итераторами, обязательно включает каждый контейнер?

- 364. Чем константный итератор отличается от неконстантного?
- 365. Объясните, что такое «недействительный» итератор. В каких случаях итераторы становятся недействительными?
- 366. Какие категории итераторов вы знаете? Какие операции обязательно реализуются для всех категорий итераторов?
- 367. К какому виду итераторов можно отнести встроенный указатель и почему?
- 368. Какие вспомогательные функции для итераторов вы знаете? В каких случаях оправдано их применение?
- 369. Какие адаптеры итераторов реализованы в библиотеке?
- 370. Объясните, почему итераторы реализованы как вложенные классы в контейнерах.
- 371. Чем отличаются итераторы вставки от обычных итераторов?
- 372. Каким образом используются потоковые итераторы?
- 373. Какие стандартные функторы реализованы в библиотеке STL? Каково их основное назначение?
- 374. Для чего нужны адаптеры функторов `bind1st()` и `bind2nd()`?
- 375. Как применяются адаптеры-отрицатели?
- 376. Почему алгоритмы `remove()` не удаляют элементы из контейнеров? Как реально удалить элементы из контейнера?
- 377. Чем отличается стабильная сортировка от обычной?
- 378. Какую функцию выполняет алгоритмы `unique()`?
- 379. Могут ли стандартные алгоритмы работать со строками?
- 380. Нужно ли сортировать ассоциативные контейнеры?
- 381. Можно ли алгоритмы для работы с множествами применять для последовательных контейнеров? При каких условиях?
- 382. Какие алгоритмы предназначены для заполнения контейнера значениями? С какими контейнерами они могут работать?
- 383. Каким образом заполнить с помощью алгоритма `generate()` последовательный контейнер, не имеющий ни одного элемента?
- 384. Перечислите алгоритмы, предназначенные для операций с каждым элементом контейнера.
- 385. Можно ли с помощью алгоритма `for_each()` изменить элементы контейнера?