1. Язык C++ считается низкоуровневым или высокоуровневым?
2. Какие отличия между указателем и ссылкой?
3. Что такое rvlaue и lvalue переменные?
4. Что такое rvalue и lvalue ссылки?
5. Как связаны lvalue и rvalue ссылки и переменные?
6. Что такое ООП?
7. Что подразумевает абстракция с точки зрения ООП?
8. Что такое инкапсуляция?
9. Что такое наследование?
10. Что такое полиморфизм?
11. Какие существуют виды полиморфизма?
12. Для чего и на какие файлы производится разбиение программы на C++?
13. Что такое union в C++, когда оно может быть применимо?
14. Что такое перечисление в C++ и особенности его использования?
15. Что такое указатель на функцию и как он может быть использован?
16. Какие способы группировки данных в C++ вам известны?
17. Для чего предназначены структуры?
18. Где может быть определена структура или класс?
19. Допустимо ли использование указателей/ссылок/массивов структур?
20. Какие существуют способы передачи параметров в функцию?
21. Для чего предназначены классы, в чем их отличие от структур?
22. Что такое выравнивание памяти в структурах и классах?
23. Что такое инвариант класса?
24. В чем отличие функций от методов?
25. В каких случаях используются значения по умолчанию в функциях?
26. Что такое публичный интерфейс?
27. Какие существуют модификаторы доступа, для чего они используются?
28. Что такое геттеры и сеттеры?
29. Где применяется неявный указатель this?
30. Для чего используется ключевое слово const?
31. Что такое константные ссылки/указатели, указатели/ссылки на константу?
32. В чем отличие синтаксической и логической константности методов?
33. Для чего используется ключевое слово default?
34. Для чего используется ключевое слово mutable?
35. Какие методы в классе генерируются компилятором?
36. Что такое конструктор?
37. В каких случаях используется перегрузка конструкторов?
38. Какую цель может преследовать создание приватного конструктора?
39. Каким образом и в какой последовательности происходит инициализация полей объекта?
40. Для чего используется ключевое слово explicit?
41. В чем заключается предназначение конструктора по умолчанию?
42. Что такое деструктор, для чего он используется?
43. Каков порядок вызова деструкторов при разрушении объекта?
44. В какой момент вызывается деструктор объекта?
45. Каково время жизни объекта?
46. Зачем нужен виртуальный деструктор?
47. Как осуществляется работа с динамической памятью в C/C++?
48. В чем различие delete и delete[]?
49. Что подразумевается под идиомой RAII?
50. Перечислите основные подходы к обработке ошибок.

51. Для чего предназначен механизм обработки исключительных ситуаций?
52. Что такое исключение?
53. Какие типы данных допустимы для использования в качестве объектов exception?
54. Как происходит возбуждение исключения?
55. Кто отвечает за обработку возникших исключительных ситуаций?
56. Какие 3 уровня гарантии в механизме исключений?
57. Что такое раскрутка стека?
58. Где и для чего используется спецификатор throw?
59. Где и для чего используется спецификатор noexcept?
60. К чему приводит вызов throw без аргументов?
61. Что такое exception-safe операция?
62. Что такое делегирующие конструкторы?
63. Что вы можете сказать о генерации исключений в конструкторе/деструкторе?
64. Что такое ассоциация?
65. Что такое композиция и агрегация, чем они отличаются?
66. Время жизни агрегируемого объекта меньше времени жизни агрегата?
67. Какие классы называются дружественными, для каких целей используется это отношение?
68. В каком случае можно говорить об отношение «реализация»?
69. Как представлены объекты в памяти при использовании механизма наследования?
70. Какие существуют типы наследования, чем они различаются?
71. Наследуются ли конструкторы и деструкторы?
72. Наследуются ли приватные поля базового класса?
73. Что такое полиморфный класс?
74. Что такое виртуальная функция?
75. Как осуществить вызов базовой реализации функции при её переопределении в дочернем классе?
76. Как связаны виртуальные функции и полиморфизм?
77. Что такое переопределение функций?
78. Работает ли переопределение для приватных функций?
79. Что такое таблица виртуальных функций?
80. Как себя ведут виртуальные функции в конструкторе и деструкторе?
81. В каких случаях допустимо приведение указателей/ссылок на дочерний класс к базовому?
82. Что такое чистая виртуальная функция?
83. Какой класс называется абстрактным?
84. Как в C++ реализуются интерфейсы?
85. Что такое перегрузка функций?
86. Как ведет себя перегрузка при наследовании?
87. Опишите процесс выбора функции среди перегруженных.
88. Чем отличаются механизмы раннего и позднего связывания?
89. Что такое множественное наследование?
90. Что такое ромбовидное наследование?
91. Какой существует механизм разрешения проблемы ромбовидного наследования в C++?
92. Как представлены объекты в памяти при виртуальном наследовании?
93. Как реализовано приведение типов в Си?
94. Что такое статическое приведение типов?
95. Что такое динамическое приведение типов?
96. Что такое константное приведение типов?
97. Как работает преобразование в Си-стиле на языке C++?

98. Что такое умные указатели?
99. Опишите принцип работы boost::scoped_ptr.
100. Опишите принцип работы std::auto_ptr.
101. Опишите принцип работы std::shared_ptr.
102. Опишите принцип работы std::weak_ptr.
103. В чем особенности работы умных указателей с массивами?
104. Какие группы операторов в C++ вам известны?
105. Что такое перегрузка операторов, для чего она используется?
106. Для каких типов допустима перегрузка операторов?
107. Где может быть объявлена перегрузка оператора?
108. Какие особенности у перегрузки операторов инкремента и декремента?
109. Как ведут себя операторы с особым порядком вычисления при перегрузке?
110. Наследует ли производный класс перегруженные операторы?
111. Что такое функторы?
112. Что такое ключевое слово using и его связь с перегрузкой методов?
113. Что такое ключевое слово using и его связь с модификаторами доступа?
114. Как защитить объект от копирования?
115. Для чего предназначен механизм RTTI, как его использовать?
116. Какие особенности использования dynamic_cast, его отличия от static_cast?
117. Что такое шаблоны классов?
118. Что такое шаблоны функций?
119. Как осуществляется вывод аргументов шаблона?
120. Что может быть параметром шаблона?
121. Что такое специализация шаблонов?
122. Что такое нешаблонная база?
123. Что такое странно рекурсивный шаблон?
124. Какая разница между typedef и using?
125. Что такое ключевое слово decltype?
126. Что такое принципы SOLID?
127. Что такое принцип единственной ответственности?
128. Что такое принцип открытости/закрытости?
129. Что такое принцип подстановки Барбары Лисков?
130. Что такое принцип разделения интерфейса?
131. Что такое принцип инверсии зависимостей?
132. Что такое move-семантика и как она реализуется в C++?
133. Что такое шаблон проектирования?

Паттерны:

- Порождающие
    1. Одиночка / Singleton
    2. Фабричный метод / Factory Method
    3. Абстрактная фабрика / Abstract Factory
    4. Прототип / Prototype
    5. Строитель / Builder
- Структурные
    1. Фасад / Facade
    2. Легковес / Flyweight
    3. Компоновщик / Composite
    4. Мост / Bridge
    5. Заместитель / Proxy
    6. Декоратор / Decorator
    7. Адаптер / Adapter
- Поведенческие
    1. Цепочка обязанностей / Chain of Responsibility
    2. Команда / Command
    3. Посредник / Mediator
    4. Наблюдатель / Observer
    5. Шаблонный метод / Template Method
    6. Стратегия / Strategy
    7. Посетитель / Visitor
    8. Состояние / State
    9. Снимок / Memento (классическая реализация)
    10. Снимок / Memento (реализация с пустым промежуточным интерфейсом)
    11. Снимок / Memento (реализация с повышенной защитой)