Таблица соответствия

Наименование раздела дисциплины по (РПД)	Раздел в тестовых вопросах
Технологический цикл разработки программных средств	Раздел 1
Методы проектирования программного обеспечения	Раздел 2
Использование абстракций и спецификаций при разработке программ	Раздел 2
Тестирование и отладка ПО	Раздел 3
Объектно-ориентированное проектирование ИС	Раздел 4
Прототипное проектирование ИС (RAD-технология)	Раздел 4

РАЗДЕЛ 1. Технологический цикл разработки программных средств

1 **Какие программы можно отнести к системному программному обеспечению:**

1. операционные системы;
2. прикладные программы;
3. игровые программы.

2. **Какие программы можно отнести к системному ПО:**

1. драйверы;
2. текстовые редакторы;
3. электронные таблицы;
4. графические редакторы.

3. Специфические особенности ПО как продукта:

1. продажа по ценам ниже себестоимости (лицензирование);
2. низкие материальные затраты при создании программ;
3. возможность создание программ небольшие коллективом или даже одним человеком;
4. разнообразие решаемых задач с помощью программных средств.

4. Какие программы можно отнести к системному ПО:

1. программа расчета заработной платы;
2. электронные таблицы;
3. СУБД (системы управления базами данных).

5. Какие программы нельзя отнести к системному ПО:

1. игровые программы;
2. компиляторы языков программирования;
3. операционные системы;
4. системы управления базами данных.

6. Какие программы можно отнести к прикладному программному обеспечению:

1. электронные таблицы;
2. таблицы решений;
3. СУБД (системы управления базами данных).

7. Какие программы можно отнести к прикладному ПО:

1. программа расчета заработной платы;
2. диспетчер программ;
3. программа «Проводник» (Explorer).

8. Какие программы нельзя отнести к прикладному ПО:

1. компиляторы и (или) интерпретаторы;
2. текстовые и (или) графические редакторы;
3. электронные таблицы.

9. Можно ли отнести операционную систему к программному обеспечению:

1. да;
2. нет.

10. Можно ли отнести операционную систему к прикладному программному обеспечению:

1. да;
2. нет.

11. Специфические особенности ПО как продукта:

1. низкие затраты при дублировании;
2. универсальность;
3. простота эксплуатации;
4. наличие поддержки (сопровождения) со стороны разработчика.

12. Какие программы можно отнести к системному ПО:

1. утилиты;
2. экономические программы;
3. статистические программы;
4. мультимедийные программы.

13. Этап, занимающий наибольшее время, в жизненном цикле программы:

1. сопровождение;
2. проектирование;
3. тестирование;
4. программирование;
5. формулировка требований.

14. Этап, занимающий наибольшее время, при разработке программы:

1. тестирование;
2. сопровождение;
3. проектирование;
4. программирование;
5. формулировка требований.

15. Первый этап в жизненном цикле программы:

1. формулирование требований;
2. анализ требований;
3. проектирование;
4. автономное тестирование;
5. комплексное тестирование.

16. Один из необязательных этапов жизненного цикла программы:

1. оптимизация;
2. проектирование;
3. тестирование;
4. программирование;
5. анализ требований.

17. Самый большой этап в жизненном цикле программы:

1. эксплуатация;
2. изучение предметной области;
3. программирование;
4. тестирование;
5. корректировка ошибок.

18. Какой этап выполняется раньше:

1. отладка;
2. тестирование.

19. Какой этап выполняется раньше:

1. отладка;
2. оптимизация;
3. программирование;
4. тестирование.

20. Что выполняется раньше:

1. компиляция;
2. отладка;
3. компоновка;
4. тестирование.

21. Что выполняется раньше:

1. проектирование;
2. программирование;
3. отладка;
4. тестирование.

22. В стадии разработки программы не входит:

1. автоматизация программирования;
2. постановка задачи;
3. составление спецификаций;
4. эскизный проект;
5. тестирование.

23. Самый важный критерий качества программы:

1. работоспособность;
2. надежность;
3. эффективность;
4. быстродействие;
5. простота эксплуатации.

24. Способы оценки качества:

1. сравнение с аналогами;
2. наличие документации;
3. оптимизация программы;
4. структурирование алгоритма.

25. Существует ли связь между эффективностью и оптимизацией программы:

1. да;
2. нет.

26. Наиболее важный критерий качества:

1. надежность;
2. быстродействие;
3. удобство в эксплуатации;
4. удобный интерфейс;
5. эффективность.

27. Способы оценки надежности:

1. тестирование;
2. сравнение с аналогами;
3. трассировка;
4. оптимизация.

28. Повышает ли качество программ оптимизация:

1. да;
2. нет.

29. Существует ли связь между надежностью и быстродействием:

1. нет:
2. да.

30. В каких единицах можно измерить надежность:

1. отказов/час;
2. км/час;
3. Кбайт/сек;
4. операций/сек.

31. В каких единицах можно измерить быстродействие:

1. отказов/час;
2. км/час;
3. Кбайт/сек;
4. операций/сек.

32. Что относится к этапу программирования:

1. написание кода программы;

2. разработка интерфейса;
3. работоспособность;
4. анализ требований.

33. Последовательность этапов программирования:

1. компилирование, компоновка, отладка;
2. компоновка, отладка, компилирование;
3. отладка, компилирование, компоновка;
4. компилирование, отладка, компоновка.

34) Инструментальные средства программирования:

1. компиляторы, интерпретаторы;
2. СУБД (системы управления базами данных);
3. BIOS (базовая система ввода-вывода);
4. ОС (операционные системы).

35. На языке программирования составляется:

1. исходный код;
2. исполняемый код;
3. объектный код;
4. алгоритм.

36. Правила, которым должна следовать программа это:

1. алгоритм;
2. структура;
3. спецификация;
4. состав информации.

37. Можно ли внутри цикла поместить еще один цикл:

1. да;
2. нет.

38. Можно ли внутри условного оператора поместить еще одно условие:

1. да;
2. нет.

39. Можно ли одно большое (длинное) выражение разбить на несколько выражений:

1. да;
2. нет.

40. Если имеется стандартная функция, нужно ли писать собственную:

1. нет;
2. да.

41. Доступ, при котором записи файла читаются в физической последовательности, называется:

1. последовательным;
2. прямым;
3. простым;
4. основным.

42. Доступ, при котором записи файла обрабатываются в произвольной последовательности, называется:

1. прямым;
2. последовательным;
3. простым;
4. основным.

43. Методы программирования (укажите НЕ верный ответ):

1. логическое;
2. структурное;
3. модульное.

44. Что выполняется раньше:

1. разработка алгоритма;
2. выбор языка программирования;
3. написание исходного кода;
4. компиляция.

45. Можно ли переменным присваивать произвольные идентификаторы:

1. да;
2. нет.

46. Найдите НЕ правильное условие для создания имен:

1. имена могут содержать пробелы;
2. длинное имя можно сократить;
3. из имени лучше выбрасывать гласные;
4. можно использовать большие буквы.

47. Какие символы не допускаются в именах переменных:

1. пробелы;
2. цифры;
3. подчеркивание

48. Модно ли использовать имена, которые уже были использованы в другой программе (модуле):

1. да;
2. нет.

49. Можно ли ставить знак подчеркивания в начале имени:

1. да, но не рекомендуется;
2. да, без ограничений;
3. нет.

50. Как называется способ составления имен переменных, когда в начале имени сообщается тип переменной:

1. прямым указанием;
2. венгерской нотацией;
3. структурным программированием;
4. поляризацией.

51. Можно ли писать комментарии в отдельной строке:

1. да;
2. нет.

52. Транслируются ли комментарии:

1. да;
2. нет.

53. Наличие комментариев позволяет:

1. быстрее найти ошибки в программе;
2. быстрее писать программы;
3. быстрее выполнять программы.

54. Наличие комментариев позволяет:

1. легче разобраться в программе;
2. применять сложные структуры;
3. увеличить быстродействие.

55. Наличие комментариев позволяет:

1. улучшить читабельность программы;
2. улучшить эксплуатацию программы;
3. повысить надежность программы.

56. Что определяет выбор языка программирования:

1. область приложения;
2. знание языка;
3. наличие дополнительных библиотек.

57. Возможно ли комбинирование языков программирования в рамках одной задачи:

1. да;
2. нет.

58. Допустимо ли комбинирование языков программирования в рамках одной задачи :

1. да;
2. нет.

59. Для каких задач характерно использование большого количества исходных данных, выполнение операций поиска, группировки:

1. для экономических задач;
2. для системных задач;
3. для инженерных задач.

60. Для каких задач характерен большой объем вычислений, использование сложного математического аппарата:

1. для инженерных задач;
2. для системных задач;
3. для экономических задач.

61. На каком этапе производится выбор языка программирования:

1. проектирование;
2. программирование;
3. отладка;
4. тестирование.

62. Можно ли использовать комбинацию языков программирования в рамках одного проекта:

1. да;
2. нет.

Раздел 2

Методы проектирования программного обеспечения и

Использование абстракций и спецификаций при разработке программ

97. Что такое автоматизация программирования:

1. создание исходного кода программными средствами;
2. создание исходного кода при помощи компилятора;
3. создание исходного кода без разработки алгоритма.

98. В чем сущность автоматизации программирования:

1. создание программы без написания ее текста;
2. получение готовой программы без выполнения компоновки;

3. в отсутствии компиляции.

99. Возможна ли автоматизация программирования:

1. да;
2. нет.

100. Создание исполняемого кода программы без написания исходного кода называется:

1. составлением спецификаций;
2. отладкой;
3. проектированием.
4. автоматизацией программирования;

101. Одно из преимуществ автоматизации программирования:

1. наглядное программирование с визуальным контролем;
2. получение стандартной программы;
3. создание программы с оптимальным кодом.

102. Один из методов автоматизации программирования:

1. структурное программирование;
2. модульное программирование;
3. визуальное программирование;
4. объектно-ориентированное программирование.

103. Влияет ли автоматизация программирования на эффективность программы:

1. нет;
2. да

104. Автоматизация программирования позволяет:

1. повысить надежность программы;

2. сократить время разработки программы;
3. повысить быстродействие программы.

105. Позволяет ли автоматизация программирования всегда создавать эффективные программы:

1. да.
2. нет;

106. Позволяет ли автоматизация программирования всегда создавать надежные программы:

А) нет;

В) да.

107. Недостаток автоматизации программирования;

1. низкое быстродействие;
2. большой размер программы;
3. сложность программы.

108. Возможны ли ошибки при автоматизации программирования:

1. да;
2. нет.

109. Нужно ли выполнять тестирование при автоматизации программирования:

1. да;
2. нет.

110. Выполняется ли процедура компиляции при автоматизации программирования:

1. да;
2. нет.

111. Что легко поддается автоматизации:

1. интерфейс;
2. работа с файлами;
3. сложные логические задачи;
4. алгоритмизация.

112. Относится ли визуальное программирование к средствам автоматизации:

1. да;
2. нет.

113. Нахождение наилучшего варианта из множества возможных:

1. оптимизация;
2. тестирование;
3. автоматизация;
4. отладка;
5. сопровождение.

114. Что такое оптимизация программ:

1. улучшение работы существующей программы;
2. создание удобного интерфейса пользователя;
3. разработка модульной конструкции программы;
4. применение методов объектно-ориентированного программирования.

115. Критерии оптимизации:

1. время выполнения или размер требуемой памяти;
2. размер программы и ее эффективность;
3. независимость модулей;
4. качество программы, ее надежность.

116. Критерии оптимизации:

1. эффективность использования ресурсов;
2. структурирование алгоритма;
3. структурирование программы.

117. Возможна ли оптимизация программ без участия программиста:

1. да;
2. нет.

118. Возможна ли оптимизация циклов:

1. да;
2. нет.

119. В чем заключается оптимизация условных выражений:

1. в изменении порядка следования элементов выражения;
2. в использовании простых логических выражений;
3. в использовании сложных логических выражений;
4. в использовании операций AND, OR и NOT.

120. Оптимизация циклов заключается в:

1. уменьшении количества повторений тела цикла;
2. просмотре задачи с другой стороны;
3. упрощение задачи за счет включения логических операций.

121. Оптимизация программы это:

1. модификация;
2. отладка;
3. повышение сложности программы;
4. уменьшение сложности программы.

122. Критерии оптимизации программы:

1. быстродействие или размер программы;
2. быстродействие и размер программы;
3. надежность или эффективность;
4. надежность и эффективность.

123. Результат оптимизации программы:

1. эффективность;
2. надежность;
3. машино-независимость;
4. мобильность.

124. Сущность оптимизации циклов:

1. сокращение количества повторений выполнения тела цикла;
2. сокращение тела цикла;
3. представление циклов в виде блок-схем;
4. трассировка циклов;
5. поиск ошибок в циклах.

125. В чем сущность модульного программирования:

1. в разбиении программы на отдельные функционально независимые части;
2. в разбиении программы на отдельные равные части;
3. в разбиение программы на процедуры и функции;

126. Можно ли сочетать модульное и структурное программирование:

1. да;
2. нет.

127. Может ли модуль включать несколько процедур или функций:

1. да;
2. нет.

128. Рекомендуемые размеры модулей:

1. небольшие;
2. большие;
3. равные;
4. фиксированной длины.

129. В чем заключается независимость модуля:

1. в написании, отладке и тестировании независимо от остальных модулей;
2. в разработке и написании независимо от других модулей;
3. в независимости от работы основной программы.

130. При модульном программировании желательно, чтобы модуль имел:

1. большой размер;
2. небольшой размер;
3. фиксированный размер;
4. любой размер.

131. Модульное программирование это:

1. разбиение программы на отдельные части;
2. структурирование;
3. использование стандартных процедур и функций.

132. Можно ли использовать оператор GO TO в модульном программах:

1. можно;
2. нельзя.

133. Разрешается ли использование циклов при модульном программировании:

1. да;
2. нет.

134. Разрешается ли использование условных операторов при модульном программировании:

1. да;
2. нет.

135. Сократится ли размер программы, если ее написать в виде набора модулей:

1. нет;
2. да.

136. Достоинство модульного программирования:

1. создание программы по частям в произвольном порядке;
2. не требует компоновки;
3. всегда дает эффективные программы;
4. снижает количество ошибок.

137. Недостаток модульного программирования:

1. увеличивает трудоемкость программирования;
2. усложняет процедуру комплексного тестирования;
3. снижает быстродействие программы;
4. не позволяет выполнять оптимизацию программы.

138. Достоинство модульного программирования:

1. возможность приступить к тестированию до завершения написания всей программы;
2. не требует комплексного тестирования;
3. уменьшает размер программы;

4. повышает надежность программы.

139. Допустимо ли использование оператора GO TO при структурном программировании:

1. нет;
2. да.

140. Можно ли сочетать структурное программирование с модульным:

1. можно;
2. нельзя;
3. только в особых случаях.

141. Любую ли программу можно привести к структурированному виду:

1. любую;
2. не все;
3. нельзя.

142. Можно ли использовать оператор GO TO в структурированных программах:

1. можно;
2. нельзя;
3. только в особых случаях.

143. Возможно, ли преобразовать неструктурированную программу к структурному виду:

1. да;
2. нет.

144. Возможно ли программирование без оператора GO TO:

1. да;
2. нет.

145. При структурном программировании задача выполняется:

1. поэтапным разбиением на более легкие задачи;
2. без участия программиста;
3. объединением отдельных модулей программы.

146. Разрешается ли использование оператора GO TO при структурном программировании:

1. нет;
2. да;
3. иногда.

147. Разрешается ли использование циклов при структурном программировании:

1. да;
2. нет.

148. Разрешается ли использование оператора IF при структурном программировании:

1. да;
2. нет.

149. Программирование без GO TO применяется. при:

a. структурном программировании;
b. модульном программировании;
c. объектно-ориентированном программировании;
d. все ответы верные.

150. Достоинство структурного программирования:

1. можно приступить к комплексному тестированию на раннем этапе разработки;
2. можно приступить к автономному тестированию на раннем этапе разработки;
3. нет необходимости выполнять тестирование;
4. можно пренебречь отладкой.

151. Достоинство структурного программирования:

1. облегчает работу над большими и сложными проектами;
2. повышает быстродействие программы;
3. снижает затраты на программирование.

152. Недостаток структурного программирования:

1. увеличивает размер программы;
2. снижает эффективность;
3. уменьшает количество ошибок;
4. не требует отладки.

153. Повышает ли читабельность программ структурное кодирование:

1. да;
2. нет.

Раздел 3. Тестирование и отладка ПО

63. Для решения экономических задач характерно применение:

1. СУБД (систем управления базами данных);
2. языков высокого уровня;
3. языков низкого уровня;
4. применение сложных математических расчетов.

64. Для решения инженерных задач характерно применение:

1. САПР (систем автоматизированного проектирования);
2. СУБД (систем управления базами данных);
3. ОС (операционных систем).

65. Причины синтаксических ошибок:

1. плохое знание языка программирования;
2. ошибки в исходных данных;
3. ошибки, допущенные на более ранних этапах;
4. неправильное применение процедуры тестирования.

66. Когда можно обнаружить синтаксические ошибки:

1. при компиляции;
2. при отладке;
3. при тестировании;
4. на этапе проектирования;
5. при эксплуатации.

67. Ошибки компоновки заключаются в том, что:

1. указано внешнее имя, но не объявлено;
2. неправильно использовано зарезервированное слово;
3. составлено неверное выражение;
4. указан неверный тип переменной.

68. Могут ли проявиться ошибки при изменении условий эксплуатации:

1. да;
2. нет.

69. Могут ли проявиться ошибки при изменении в предметной области:

1. да;
2. нет.

70. Возможно ли программирование с защитой от ошибок:

1. да;

2. нет.

71. Есть ли недостатки программирования с защитой от ошибок:

1. да;
2. нет.

72. Защитное программирование это:

1. встраивание в программу отладочных средств;
2. создание задач защищенных от копирования;
3. разделение доступа в программе;
4. использование паролей;
5. оформление авторских прав на программу.

73. Вид ошибки с неправильным написанием служебных слов (операторов):

1. синтаксическая;
2. семантическая;
3. логическая;
4. символьная.

74. Вид ошибки с неправильным использованием служебных слов (операторов):

1. семантическая;
2. синтаксическая;
3. логическая;
4. символьная.

75. Ошибки при написании программы бывают:

1. синтаксические;
2. орфографические;
3. лексические;

4. фонетические;
5. морфологические.

76. Процедура поиска ошибки, когда известно, что она есть это:

1. отладка;
2. тестирование;
3. компоновка;
4. транзакция;
5. трансляция.

77. Программа для просмотра значений переменных при выполнении программы:

1. отладчик;
2. компилятор;
3. интерпретатор;
4. трассировка;
5. тестирование.

78. Отладка – это:

1. процедура поиска ошибок, когда известно, что ошибка есть;
2. определение списка параметров;
3. правило вызова процедур (функций);
4. составление блок-схемы алгоритма.

79. Когда программист может проследить последовательность выполнения команд программы:

1. при трассировке;
2. при тестировании;
3. при компиляции;
4. при выполнении программы;
5. при компоновке.

80. На каком этапе создания программы могут появиться синтаксические ошибки:

1. программирование;
2. проектирование;
3. анализ требований;
4. тестирование.

81. Когда приступают к тестированию программы:

1. когда программа уже закончена;
2. после постановки задачи;
3. на этапе программирования;
4. на этапе проектирования;
5. после составления спецификаций,

82. Тестирование бывает:

1. автономное;
2. инструментальное;
3. визуальное;
4. алгоритмическое.

83. Тестирование бывает:

1. комплексное;
2. инструментальное;
3. визуальное;
4. алгоритмическое.

84. Существует ли различие между отладкой и тестированием:

1. да;
2. нет.

85. При комплексном тестировании проверяются:

1. согласованность работы отдельных частей программы;
2. правильность работы отдельных частей программы;
3. быстродействие программы;
4. эффективность программы.

86. Чему нужно уделять больше времени, чтобы получить хорошую программу:

1. тестированию;
2. программированию;
3. отладке;
4. проектированию.

87. Процесс исполнения программы с целью обнаружения ошибок:

1. тестирование;
2. кодирование;
3. сопровождение;
4. проектирование.

88. Автономное тестирование это:

1. тестирование отдельных частей программы;
2. инструментальное средство отладки;
3. составление блок-схем;
4. пошаговая проверка выполнения программы.

89. Трассировка это:

1. проверка пошагового выполнения программы;
2. тестирование исходного кода;
3. отладка модуля;

4. составление блок-схемы алгоритма.

90. Локализация ошибки:

1. определение места возникновения ошибки;
2. определение причин ошибки;
3. обнаружение причин ошибки;
4. исправление ошибки.

91. Назначение тестирования:

1. повышение надежности программы;
2. обнаружение ошибок;
3. повышение эффективности программы;
4. улучшение эксплуатационных характеристик;
5. приведение программы к структурированному виду.

92. Назначение отладки:

1. поиск причин существующих ошибок;
2. поиск возможных ошибок;
3. составление спецификаций;
4. разработка алгоритма.

93. Инструментальные средства отладки (НЕ правильный ответ):

1. компиляторы;
2. отладчики;
3. трассировка.

94. Отладка программ это:

1. локализация и исправление ошибок;
2. алгоритмизация программирования;

3. компиляция и компоновка.

95. Что выполняется раньше, автономная или комплексная отладка:

1. автономная;
2. комплексная.

96. Что выполняется раньше, отладка или тестирование:

1. отладка;
2. тестирование.

Раздел 4
Объектно-ориентированное проектирование ИС и Прототипное проектирование ИС (RAD-технология)

154. Разрешается ли использование циклов при объектно-ориентированном программировании:

1. да;
2. нет.

155. Разрешается ли использование оператора IF при объектно-ориентированном программировании:

1. да;
2. нет.

156. Предусматривает ли объектно-ориентированное программирование использование стандартных процедур и функций:

1. да;
2. нет.

157. Можно ли сочетать объектно-ориентированное и структурное программирование

1. можно;
2. нельзя.

158) Можно ли сочетать объектно-ориентированное и модульное программирование:

1. можно;
2. нельзя.

159. Что такое объект, в объектно-ориентированное программировании:

1. тип данных;
2. структура данных;
3. событие;
4. обработка событий;
5. использование стандартных процедур.

160. Инкапсуляция это:

1. определение новых типов данных;
2. определение новых структур данных;
3. объединение переменных, процедур и функций в одно целое;
4. разделение переменных, процедур и функций;
5. применение стандартных процедур и функций.

161 Наследование это:

1. передача свойств экземплярам;
2. передача свойств предкам;

3. передача свойств потомкам;
4. передача событий потомкам.

162 Полиморфизм это:

1. изменение поведения потомков, имеющих общих предков;
2. передача свойств по наследству;
3. изменение поведения потомков на разные события;
4. изменение поведения экземпляров, имеющих общих предков;

163 Три "кита" объектно-ориентированного метода программирования:

1. предки, родители, потомки;
2. полиморфизм, инкапсуляция, наследование;
3. свойства, события, методы;
4. визуальные, не визуальные компоненты и запросы.

164 Какое утверждение верно:

1. предки наследуют свойства родителей;
2. родители наследуют свойства потомков;
3. потомки не могут иметь общих предков;
4. потомки наследуют свойства родителей.

165. Может ли дочерний элемент иметь двух родителей:

1. да;
2. нет;
3. только для визуальных элементов;
4. если их свойства совпадают.

166 Могут ли два визуальных компонента иметь общего предка:

1. да;

2. нет;
3. если их свойства совпадают;
4. если их методы совпадают.

167. Есть ли различие между объектом и экземпляром:

1. да;
2. нет;
3. если у них общий предок.

168. Есть ли различие в поведении объекта и экземпляра того же типа:

1. да;
2. если у них есть общий предок;
3. нет;
4. если у них нет общего предков.

169. Изменение свойств, приводит к изменению поведения экземпляра:

1. нет;
2. только для визуальных;
3. только НЕ для визуальных ;
4. да .

170. Можно ли свойствам присваивать значения:

1. да (всегда);
2. не всегда;
3. нет.

171. Можно ли переопределять методы:

1. да;
2. нет.

172. Можно ли переопределять свойства:

1. да;
2. нет.

173. Могут ли два различных объекта реагировать на событие по-разному:

1. да;
2. нет.

174. Могут ли два экземпляра одного объекта реагировать на событие по-разному:

1. да;
2. нет.

175. Какой методикой проектирования пользуются при структурном программировании:

1. сверху вниз;
2. снизу-вверх.

176. Какой этап проектирования может быть исключен:

1. эскизный проект;
2. технический проект;
3. рабочий проект.

177. Какие этапы проектирования можно объединять:

1. технический и рабочий;
2. эскизный и рабочий;
3. технический и эскизный.

178. Модульное программирование применимо при:

1. проектировании сверху вниз;
2. проектирование снизу-вверх;

179. Процесс преобразования постановки задачи в план алгоритмического или вычислительного решения это:

1. проектирование;
2. анализ требований;
3. программирование;
4. тестирование.

180. Составление спецификаций это:

1. формализация задачи;
2. эскизный проект;
3. поиск алгоритма;
4. отладка.

181. Этап разработки программы, на котором дается характеристика области применения программы:

1. техническое задание;
2. эскизный проект;
3. технический проект;
4. внедрение;
5. рабочий проект.

182. Укажите правильную последовательность создания программы:

1. формулирование задачи, анализ требований, проектирование, программирование;
2. анализ требований, проектирование, программирование, тестирование, отладка;
3. анализ требований, программирование, проектирование, тестирование;
4. анализ требований, проектирование, программирование, модификация, трассировка;
5. формулирование задачи, анализ требований, программирование, проектирование, отладка.

183. Уточнение структуры входных и выходных данных, разработка алгоритмов, определение элементов интерфейса входят в:

1. технический проект;
2. рабочий проект;
3. эскизный проект.

184. Несуществующий метод проектирования:

1. алгоритмическое;
2. нисходящее;
3. восходящее.

185. Метод проектирования:

1. нисходящее;
2. алгоритмическое;
3. логическое;
4. использование языков программирования;
5. составление блок-схем.

186. Нисходящее проектирование это:

1. последовательное уточнение (детализация);
2. составление блок-схем;
3. разделение программы на отдельные участи (блоки);
4. трассировка.

187. Признаки нисходящего программирования:

1. последовательная детализация;
2. наличие оптимизации;
3. наличие тестирования;
4. автоматизация программирования.

188. Какой методикой пользуются при структурном программировании:

1. сверху вниз;
2. снизу-вверх.

189. Проектирование сверху вниз это:

1. последовательное разбиение общих задач на более мелкие;
2. составление из отдельных модулей большой программы.

190. Проектирование снизу-вверх это:

a.i.1. составление из отдельных модулей большой программы;

a.i.2. последовательное разбиение общих задач на более мелкие.

191. Модульное программирование применимо при:

1. проектировании сверху вниз;
2. проектирование снизу-вверх;
3. и в том, и другом случае;
4. ни в коем случае.

192. Какой методикой проектирования пользуются при структурном программировании:

1. сверху вниз;
2. снизу-вверх.

193. В чем заключается иерархический подход в решении задачи:

1. в последовательном разбиении задачи на более мелкие составные части;
2. в выделении основных и второстепенных элементов;
3. в возможности параллельного выполнения отдельных частей задачи.

194. Какой метод проектирования соответствует иерархическому подходу в решении задачи:

1. нисходящее (сверху вниз);

2. восходящее (снизу-вверх).

195. В каких единицах измеряются затраты на проектирование:

a.i.1. в человеко-днях;

a.i.2. в долларах;

a.i.3. в тенге;

a.i.4. в килобайтах.

196. Зависит ли трудоемкость разработки от сложности алгоритма:

1. да;

2. нет.

197. Зависит ли трудоемкость разработки от количества программистов:

1. да;

2. нет.

198. Зависит ли трудоемкость разработки от языка или системы программирования:

1. да;

2. нет.

199. Зависит ли трудоемкость разработки от количества обрабатываемой информации:

1. да;

2. нет.

200. Зависит ли трудоемкость разработки от вида информации:

1. да;

2. нет.

201. Если вы приобрели программу законным путем, являетесь ли вы собственником программы:

1. нет;
2. да.

202. Если вы приобрели программы законным путем, имеете ли вы право вносить в нее изменения:

1. нет;
2. да

203. Если вы приобрели программы законным путем, имеете ли вы право продать ее:

1. да;
2. нет.

204. Кому принадлежит право собственности на ПО:

1. разработчику;
2. продавцу;
3. покупателю.

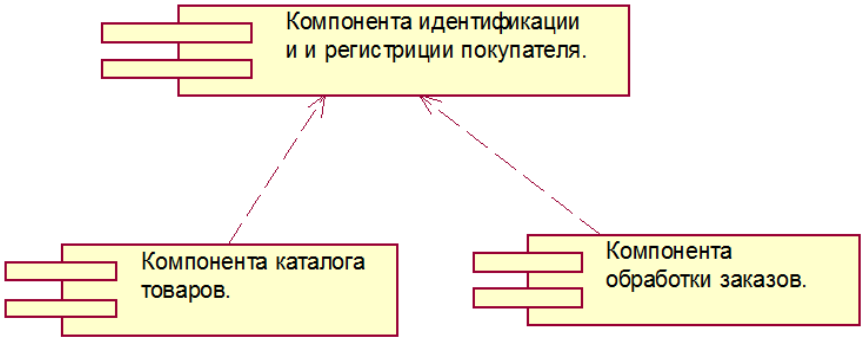
205. Кому принадлежит авторское право на ПО:

1. разработчику;
2. продавцу;
3. покупателю.

206. Что охраняется законом:

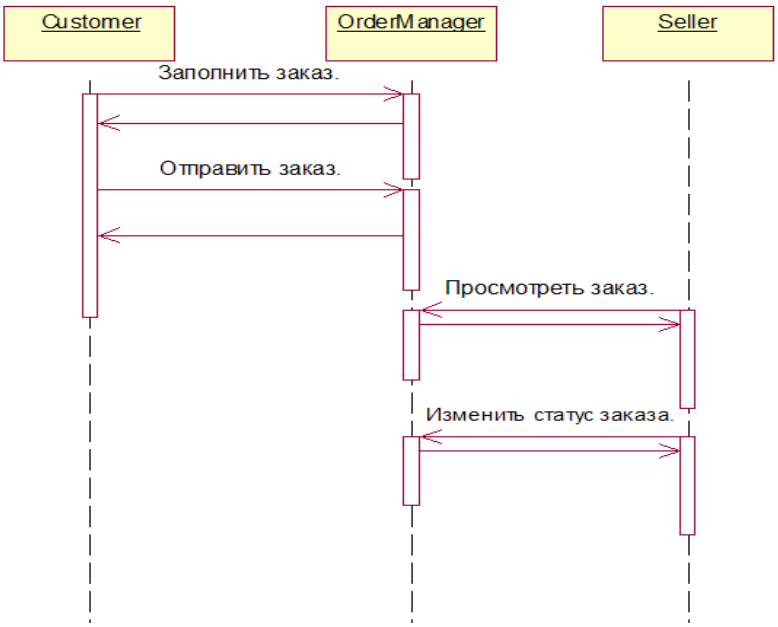
1. структура базы данных;
2. содержание базы данных

207 Введите название диаграммы латинскими буквами



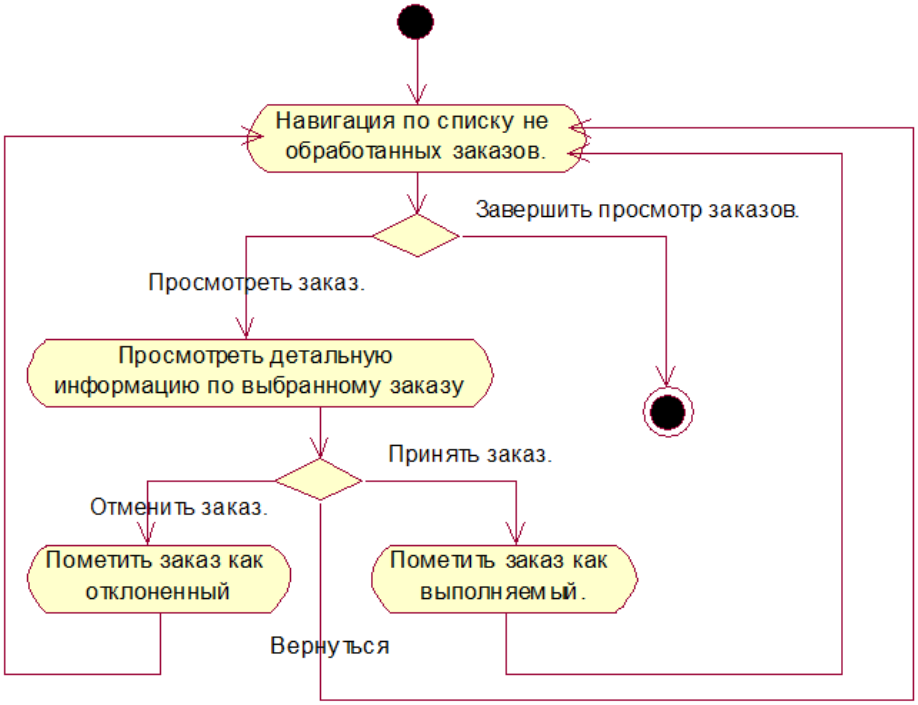
Component diagrams

208 Введите название диаграммы латинскими буквами



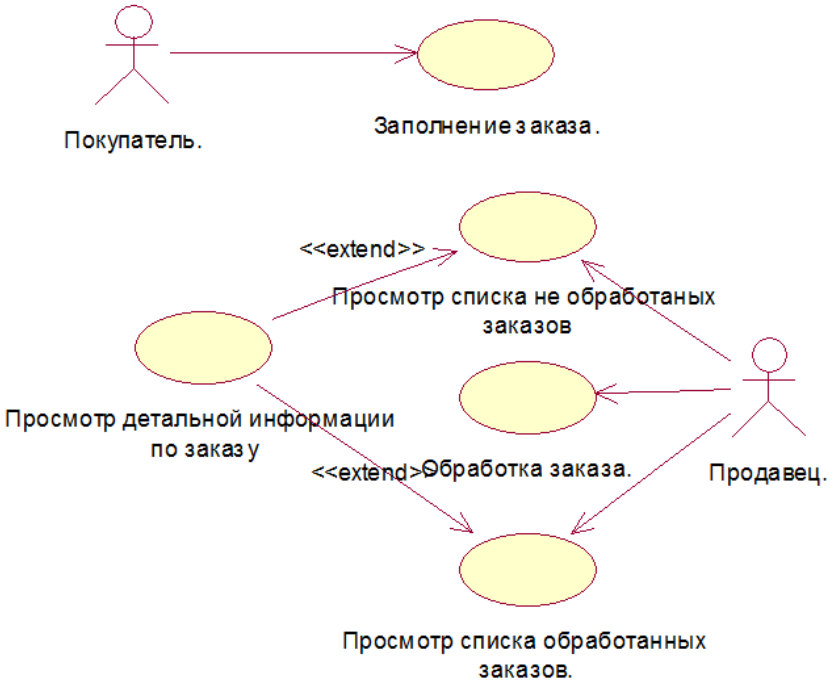
Sequence diagrams

209 Введите название диаграммы латинскими буквами



Activity diagrams

210 Введите название диаграммы латинскими буквами



Use case diagrams

211 UML позволяет задавать следующие аспекты:

1. Диаграммы вариантов использования (use case diagrams)
2. Диаграммы классов (class diagrams)
3. Диаграммы поведения

212 К диаграммам поведения отноятся:

1. Диаграммы состояний (statechart diagrams)
2. Диаграммы действий (activity diagrams)
3. Диаграммы взаимодействия (interaction diagrams)

213 Выберете существующие в проектировании диаграммы

1. Диаграммы последовательностей(sequence diagrams)
2. Диаграммы взаимодействий(collaboration diagrams)
3. Диаграммы реализации (implementation diagrams)
4. Диаграммы компонент (component diagram)
5. Диаграммы развертывания (deployment diagram)