Тема 1. Основные понятия формальных языков и грамматик

1. Счетное множество допустимых символов языка называется алфавит.
2. В общем случае язык можно определить следующими способами:

1 перечислением всех допустимых цепочек языка;

2 указанием способа порождения цепочек языка (заданием грамматики языка);

3 определением метода распознавания цепочек языка.

3. Описание способа построения предложений некоторого языка называется Грамматика.
4. Сколько классов согласно грамматики Хомского выделяется? 4
5. Какие типы грамматик выделяются согласно иерархии Хомского? грамматики с фразовой структурой;

контекстно-зависимые (КЗ) и неукорачивающиеся грамматики ;

контекстно-свободные (КС) грамматики;

регулярные грамматики

6. К какому типу грамматик относится грамматика с фразовой структурой? Тип 0
7. К какому типу грамматик относится контекстно-свободная грамматика? Тип 2
8. К какому типу грамматик относится регулярная грамматика?

   Тип 3
9. К какому типу грамматик относится контекстно-зависимая грамматика? Тип 1
10. Грамматика - это … описание способа построения предложений некоторого языка.
11. _Язык - это заданный набор символов и правил, устанавливающих способы комбинации этих символов между собой для записи осмысленных текстов.
12. _Алфавит - это счетное множество допустимых символов языка.

13. Алфавит обозначают чаще всего символом V
14. Язык обозначают чаще всего символоВ общем случае язык можно определить следующими способами: м L
15. Грамматику языка обозначают чаще всего символом G
16. Цепочка символов - это произвольная последовательность символов, записанных один за другим.
17. Цепочки символов обозначаются греческими буквами.
18. Количество символов в цепочке называют длина цепочки.
19. Запись символов цепочки в обратном порядке называется обращение…
20. Обращение цепочки обозначается как α^R
21. Итерация цепочки *n* раз обозначается как α^n
22. Цепочка, не содержащая ни одного символа, называется ___пустой____ цепочкой.
23. Пустая цепочка обозначается как λ
24. Множество всех цепочек над алфавитом V без λ обозначается V+
25. Множество всех цепочек над алфавитом V, включая λ, обозначается V*

Язык определяется тремя способами:

1 перечислением всех допустимых цепочек языка;

2 указанием способа порождения цепочек языка (заданием грамматики языка);

3 определением метода распознавания цепочек языка

26. Набор правил, определяющий допустимые конструкции языка - это Синтаксис
27. Раздел языка, определяющий значение предложений языка (смысл для всех допустимых цепочек языка) – это Семантика
28. Совокупность слов (словарный запас) языка - это Лексика
29. Для задания языка программирования необходимо решить три вопроса:

    определить множество допустимых символов **языка**;

определить множество правильных программ **языка**;

задать смысл для каждой правильной программы

30. __Грамматика - это математическая система, определяющая язык, т.е. правила порождения цепочек символов, принадлежащих этому языку.

31. Для записи грамматики чаще всего используют форму Бэкуса-Наура

32. Грамматика задается четырьмя составляющими:

VT – множество терминальных символов, или алфавит терминальных символов;

VN – множество нетерминальных символов, или алфавит нетерминальных символов;

P – множество правил (продукций) грамматики вида $\alpha \rightarrow \beta$, где $\alpha \in (VN \cup VT)+$, $\beta \in (VN \cup VT)^*$;

S – целевой (начальный) символ грамматики, $S \in VN$.

33. Какие формы задания грамматики используются для описания языков?

Форма Бэкуса-Наура

Форма с метасимволами

34. Лексика - это процесс преобразования последовательности символов (например, в компьютерной программе или веб-странице) в последовательность токенов (строк с присвоенным и, таким образом, идентифицированным значением).

Тема 2. Основные принципы построения трансляторов

35. Транслятор - это программа, которая переводит программу на исходном (входном) языке в эквивалентную ей программу на результирующем (выходном) языке.

36. Компиля́тор- это транслятор, который осуществляет перевод исходной программы в эквивалентную ей результирующую (объектную) программу на языке машинных команд или на языке ассемблера.

37. Процесс компиляции состоит из двух основных этапов:

    1.Анализ

    2.Синтез

38. Основными фазами компиляции являются:

    1.Лексический анализ

    2.Синтаксический разбор

    3.Семантический анализ

    4.Подготовка к генерации кода

    5.Генерация кода

39. анализ - это распознавание текста исходной программы, создание и заполнение таблиц для хранения данных, необходимых для дальнейших преобразований.

40. Лексический анализатор - часть компилятора, которая читает литеры программы на исходном языке и строит из них слова (лексемы) на исходном языке.

41. Результатом лексического анализа являются
    таблица идентификаторов
    таблица лексем

42. В результате синтаксического разбора получается дерево разбора

43. Таблица лексем не включает …

44. В таблице лексем любая лексема может встречаться в ней любое количество раз

45. В таблице идентификаторов любая лексема может встречаться в ней только один раз

46. Исходными данными для работы транслятора служит текст входной программы

47. Результатом работы компилятора служит Объектный код

48. На этапе синтеза выполняется: Генерация кода

49. Метод хэш-адресации для организации таблицы идентификаторов заключается в помещении каждого элемента таблицы в ячейку, адрес которой возвращает хэш-функция, вычисленная для этого элемента.

50. Метод цепочек для организации таблицы идентификаторов заключается в добавлении к каждому элементу одного поля, в котором может содержаться ссылка на любой элемент таблицы.

51. Фаза, на которой компилятором выполняются предварительные действия, необходимые для синтеза результирующей программы, но не ведущие к порождению текста на выходном языке, называется Подготовка к генерации кода

52. Фаза, непосредственно связанная с порождением текста результирующей программы, называется Генерация кода.

53. Фаза, которая выполняет выделение синтаксических конструкций в тексте исходной программы, обработанном лексическим анализатором, называется Синтаксический разбор

54. Укажите правильные утверждения:
    - Интерпретатор принимает одну инструкцию в качестве входных данных.
    - Компилятор принимает всю программу в качестве входных данных.
    - В отличие от компилятора интерпретатор не порождает результирующую программу (объектный код), а просто выполняет исходную программу.
    - Интерпретатор принимает всю программу в качестве входных данных.

55. В компиляторах линейный анализ называется: лексическим, или сканированием.

56. В компиляторах иерархический анализ называется разбором (parsing), или синтаксическим анализом .

57. В процессе семантического анализа: Проверяется наличие семантических ошибок в исходной программе и накапливается информация о типах для след.стадии - генерации кода.

58. Контекстно-свободные грамматики представляют собой формализацию рекурсивных правил, используемых при Синтаскическом анализе .

59. Укажите правильные утверждения:
    - Лексические конструкции не требуют рекурсии.
    - Построение синтаксических конструкций основано на рекурсии.
    - Дерево разбора описывает синтаксическую структуру поступающей информации.
    - Любой транслятор является компилятором.

60. Какие фазы работы компилятора составляют этап анализа?

    Распознавание текста исходной программы, создание и заполнение таблиц идентификаторов.

61. Какой способ организации таблицы идентификаторов состоит в том, чтобы добавлять элементы в таблицу в порядке их поступления?

    Способ простого линейного списка.

62. Коллизия- это ситуация, когда двум различным элементам из области определения хеш-функции соответствует одно и то же значение этой функции.

Тема 3. Конечные автоматы

63. Конечный автомат M(Q, V, δ, q0, F) называют детерминированным конечным автоматом (ДКА), в котором для каждой последовательности входных символов существует лишь одно состояние, в которое автомат может перейти из текущего.

64. Что такое граф переходов конечного автомата?

    Состояния и дуги ДС

65. Что такое лексема?

это структурная единица языка, которая состоит из элементарных символов языка и не содержит в своем составе других структурных единиц языка

66. Полным алфавитом грамматики $G(VT,VN,P,S)$ называют Целевой символ грамматики…

67. Конечные автоматы делятся на: детерминированные конечные автоматы и недетерминированные конечные автоматы

68. Какие объекты не входят в состав детерминированного конечного автомата?

    (эти входят):

    $A = (X, Q, \delta, q_0, F),$

    X - входной алфавит;

    Q - алфавит состояний автомата;

    δ : Q x X —> Q - функция переходов;

    q_0 - начальное состояние автомата;

    F - множество заключительных состояний автомата.

69. Каким способом можно представить функции переходов?

    состояния и дуги ДС

70. Какие задачи решает конечный автомат? Используется для представления и управления потоком выполнения каких-либо команд. Конечный автомат идеально подходит для реализации искусственного интеллекта в играх, получая аккуратное решение без написания громоздкого и сложного кода.

71. При построении компиляторов чаще всего используют: полностью определеный ДКА.

72. Конечным автоматом называют пятерку вида M(Q, V, δ, q_0, F), где Q - это **конечное** множество состояний **автомата**

73. Конечным автоматом называют пятерку вида M(Q, V, δ, q_0, F), где V - это **конечное** множество допустимых входных символов (алфавит **автомата**)

74. Конечным автоматом называют пятерку вида M(Q, V, δ, q_0, F), где δ - это функция переходов, отображающая декартово произведение множеств **V**×**Q** во множество подмножеств **Q**: **δ**(a,**q**)=R, а **V**, **q** **Q**, R **Q**

75. Конечным автоматом называют пятерку вида M(Q, V, δ, q_0, F), где q_0 - это начальное состояние **автомата**

76. Конечным автоматом называют пятерку вида M(Q, V, δ, q_0, F), где F - это непустое множество **конечных** состояний **автомата**

77. Функция переходов конечного автомата зависит от:

78. Можно ли граф переходов конечного автомата использовать для однозначного определения данного автомата? можно

79. Всегда ли недетерминированный конечный автомат может быть преобразован к детерминированному виду? нет

80. Какое максимальное количество состояний может содержать ДКА после преобразования из недетерминированного КА?

    Если недетерминированный автомат имеет n состояний, то эквивалентный детерминированный автомат, который мы только что описали, может, вообще говоря, иметь $2n$ состояний, по числу подмножеств исходного множества состояний.

81. Какие из следующих утверждений являются истинными?

- Если язык задан КА, то он может быть задан КС-грамматикой.
- Для любого КА можно построить эквивалентный ему ДКА.
- Если язык задан КА, то для него можно построить регулярное выражение.
- Если язык задан КС-грамматикой, то для него можно построить регулярное выражение.

82. Распознавателем для регулярных языков является конечные автоматы …

83. Конечный автомат решает следующие задачи: Абстрактный синтез; Абстрактный анализ; Структурный синтез; Структурный анализ;

84. Какой тип языков является самым распространенным и широко используемым в области вычислительных систем, являясь при этом самым простым? Регулярные **языки**

85. Текущее состояние конечного автомата характеризуется: значением такой переменной, которая вместе с заданным значением входной переменной позволяет определить выходную переменную в данный тактовый момент и **состояние** в следующий тактовый момент

86. Распознаватели на основе односторонних недетерминированных автоматов без внешней памяти (конечные автоматы) используются для:

    лексического анализа текста исходной программы, т.е. для выделения в нем простейших конструкций языка.

87. Конечный автомат называют полностью определенным детерминированным конечным автоматом, если в нем нет дуг с меткой и из любого состояния по любому входному символу возможен переход в точности в одно состояние …

Тема 4. Лексические анализаторы

88. Таблица идентификаторов содержит: всю необходимую компилятору информацию об элементе, может пополняться по мере работы компилятора

89. В основном в языках программирования символ (Открытая круглая скобка

90. В основном в языках программирования символ) Закрытая круглая скобка

91. К какому типу лексем можно отнести логическую операцию and? Ключевое слово

92. В каком порядке располагаются лексемы в таблице лексем?

    в том же **порядке**, что и в исходной программе (**порядок лексем в** ней не меняется), а **в таблице** идентификаторов **лексемы располагаются** в любом **порядке** так, чтобы обеспечить удобство поиска.

93. Возможны два варианта взаимодействия лексического и синтаксического анализаторов: последовательный; параллельный

94. Что не относится к лексемам языков программирования?

    относить: идентификаторы, ключевые слова, литералы, операторы и разделители.(НЕ ОТНОСИТЬ ВСЕ ОСТАЛЬНОЕ)

95. Первую фазу компилятора представляет лексический анализатор

96. Что поступает на вход лексического анализатора? текст исходной программы

97. Укажите правильные утверждения.???

98. Регулярные языки замкнуты относительно следующих операций: объединение, пересечение, дополнение, разность, обращение, итерация, конкатенация, гомоморфизм, обратный гомоморфизм

99. Укажите три способа задания регулярных языков.

    регулярные (праволинейные и леволинейные) грамматики;
    конечные автоматы (КА) ;

    регулярные множества.

100. Лексический анализатор не должен помещать в таблицу идентификаторов:

101. Что является результатом работы лексического анализатора? перечень всех найденных в тексте исходной программы лексем с учетом характеристик каждой лексемы

102. Основная задача лексического анализатора - это разбить входной текст, состоящий из последовательности одиночных символов, на последовательность слов, или лексем…

103. Что такое определение границ лексем? это выделение тех строк в общем потоке входных символов, для которых надо выполнять распознавание

104. Лексический анализ - это проверка цепочек в соответствии с заданной грамматикой на основе регулярных языков.

105. В результате работы лексического анализатора на выходе получают: перечень всех найденных в тексте исходной программы лексем

106. Алгоритм работы простейшего сканера обязательно включает:

просматривается входной поток символов программы на исход-

ном языке до обнаружения очередного символа, ограничивающего лек-

сему;

для выбранной части входного потока выполняется функция распознавания лексемы;

при успешном распознавании информация о выделенной лек-

семе заносится в таблицу лексем, и алгоритм возвращается к первому

этапу;

при неуспешном распознавании выдается сообщение об ошиб-

ке, а дальнейшие действия зависят от реализации сканера - либо его выполнение прекращается, либо делается попытка распознать следую-

щую лексему (идет возврат к первому этапу алгоритма).

Работа программы-сканера продолжается до тех пор, пока не будут просмотрены все символы программы на исходном языке из входного потока.

107. Конечный автомат для реализации лексического анализатора должен быть иметь не только входной язык, но и выходной. Он **должен** не только уметь распознать правильную лексему на входе, но и породить связанную с ней последовательность символов на выходе …

108. Какие из следующих утверждений являются истинными:
- Если язык задан регулярным множеством, то он может быть задан праволинейной грамматикой.
- Если язык задан КА, то он может быть задан КС-грамматикой.
- Если язык задан КА, то для него можно построить регулярное выражение.
- Если язык задан КС-грамматикой, то для него можно построить регулярное выражение.

109. Эффективным методом поиска идентификаторов в упорядоченном списке из N элементов является: бинарный или логарифмический **поиск**

110. Сколько шагов используется при построении таблицы идентификаторов по методу бинарного дерева? 4 шага

111. Метод хеш-адресации имеет такие недостатки как: неэффективное использование объема памяти под таблицу идентификаторов и необходимость соответствующего разумного выбора хэш-функции

112. Чем различаются таблица лексем и таблица идентификаторов? В отличие от **таблицы идентификаторов** в неё попадают не только **идентификаторы**, но и **лексемы** других типов – ключевые слова, константы, знаки операций и разделители. Кроме того, для **таблицы лексем** важен порядок следования **лексем** – они обязательно располагаются в ней в том же порядке, что и в исходной программе.

113. После обнаружения лексемы лексический анализатор должен выполнить следующие действия:
1. Поместить новую лексему в таблицу лексем.
2. Проверить наличие лексем в таблице идентификаторов.
3. Добавить лексему в таблицу идентификаторов, если она отсутствует.
4. Выдать пользователю сообщение об ошибках.
5. Прервать при необходимости процесс компиляции.

   1 -> 2 -> 3 -> 5 -> 4

114. Укажите правильные утверждения.
- Лексемами языков программирования являются ключевые слова, идентификаторы, константы, знаки операций и разделители.
- Лексический анализатор представляет собой первую фазу компилятора.
- В результате работы лексического анализатора на выходе получают таблицу лексем и таблицу идентификаторов.
- Лексический анализатор является обязательной частью компилятора.

115. Лексический анализатор строится на основе формализованного описания лексики и синтаксиса языка

116. Количество записей в таблице идентификаторов не может быть меньше, чем элементов в исходной программе

117. Для написания лексем используют

Тема 5. Основные принципы работы синтаксических анализаторов

118. Дерево вывода - это граф, который соответствует некоторой цепочке вывода.

119. Листьями дерева вывода являются вершины, обозначенные терминальными символами грамматики

120. Способ построения дерева вывода «сверху вниз» от целевого символа к листьям называется деревом классификации

121. Способ построения дерева вывода «снизу вверх» от листьев к целевому символу называется регрессионным деревом

122. В результате работы синтаксического анализатора формируется является синтаксическое строение предложения, представленное либо в виде дерева зависимостей

123. На вход синтаксического анализатора могут подаваться следующие данные: последовательность лексем и таблицы, например, таблица внешних представлений, которые являются выходом лексического анализатора.

124. Для контекстно-свободных языков распознователями являются:
частный случай формальной грамматики (тип 2 по иерархии Хомского), у которой левые части всех продукций являются одиночными нетерминалами (объектами, обозначающими какую-либо сущность языка (например: формула, арифметическое выражение, команда и не имеющими конкретного символьного значения). Смысл термина «контекстно-свободная» заключается в том, что есть возможность применить продукцию к нетерминалу, причём независимо от контекста этого нетерминала (в отличие от общего случая неограниченной грамматики Хомского).

125. Условием достижения недетерминированным МП-автоматом конечного состояния является:
Недетерминированный магазинный автомат есть формальная система M = (Q, Σ, Γ, δ, q0, Z0, F), где Q — конечное множество состояний; Σ — конечный входной алфавит; Γ — конечный магазинный алфавит; δ — отображение типа $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to 2^{Q \times \Gamma^*}$

представляющее конечное

управление автомата; q0∈Q — начальное состояние; Z0∈ Γ — начальный символ магазина, который в самом начале является единственным содержимым магазина; F ⊆ Q — множество конечных состояний.

126. Результатом работы синтаксического анализатора является:
дерево разбора

127. Синтаксический анализатор - это Парсер, или синтаксический анализатор, — часть программы, преобразующей входные данные в структурированный формат. Парсер выполняет синтаксический анализ текста

128. Какую задачу не решает синтаксический анализатор?

наверное генерацию кода, составление таблицы лексем

129. Какие из следующих утверждений справедливы: ???

130. Если для каждой цепочки символов языка L, заданной грамматикой G, можно построить единственный левосторонний и единственный правосторонний граф или вывод, то грамматика называется однозначной

131. Результатом работы синтаксического анализатора на основе КС-грамматики входного языка является дерево вывода

132. Синтаксические конструкции языков программирования, как правило, могут быть описаны с помощью …КС-грамматик

133. Какие из следующих утверждений справедливы: ???

134. МП-автомат определяют в виде R(Q,V,Z,δ,q_0,z_0,F), где Q - это …
входной алфавит

135. МП-автомат определяют в виде R(Q,V,Z,δ,q_0,z_0,F), где V - это …
алфавит стека, т. е. символы, которые можно добавлять в стек

136. МП-автомат определяют в виде R(Q,V,Z,δ,q_0,z_0,F), где Z - это …

множество состояний автомата

137. МП-автомат определяют в виде R(Q,V,Z,δ,q_0,z_0,F), где δ - это …

функция переходов

138. МП-автомат определяют в виде R(Q,V,Z,δ,q_0,z_0,F), где q_0 - это …
начальное состояние автомата;

139. МП-автомат определяют в виде $R(Q,V,Z,\delta,q_0,z_0,F)$, где z_0 - это … единственный символ, находящийся в стеке при начале работы автомата

140. МП-автомат определяют в виде $R(Q,V,Z,\delta,q_0,z_0,F)$, где F - это множество принимающих состояний

141. Распознавателем для КС-языков является МП-автоматы …

142. МП-автомат в отличие от обычного КА имеет стек …

143. Конфигурация МП-автомата определяется тремя параметрами:

144. Отличие обычного МП-автомата от расширенного МП-автомата: в магазине можно заменять не только верхний символ, но и непустую цепочку символов, расположенную в верхних ячейках магазина, так, что первый ее символ есть верхний символ магазина, второй символ — символ, расположенный под верхним, и т.д.

145. МП-автомат называется детерминированным_, если из каждой его конфигурации возможно не более одного перехода в следующую конфигурацию.

146. МП-автомат называется недетерминированным, если при одной и той же конфигурации возможен более чем один переход.

147. Можно ли полностью устранить рекурсию из правил грамматики, записанных в форме Бэкуса-Наура? **полностью** исключить **рекурсию из** выводов **грамматики** невозможно. **Можно** избавиться только от одного вида **рекурсии** – левого или правого, то есть преобразовать исходную **грамматику** G к одному из видов: нелеворекурсивному (избавиться от левой **рекурсии**) или неправорекурсивному (избавиться от правой **рекурсии**)

Тема 6. Синтаксические распознаватели на основе грамматик предшествования

148. Матрица предшествования служит для определения распознавателя языка

149. Распознаватель на основе алгоритма «сдвиг-свертка» называют восходящим распознавателем

150. Процесс порождения предложения языка на основе правил, определяющих язык грамматики, называется Выводом

151. Последовательность $\alpha \Rightarrow \gamma_1 \Rightarrow \gamma_2 \Rightarrow \cdots \Rightarrow \gamma_n = \beta$ называется сдвиг-свертка

152. Если на основе цепочки β нельзя больше делать ни одного шага вывода, то он называется законченным (или конечным)

153. Дерево вывода - это граф, который соответствует некоторой цепочке вывода.

154. Листьями дерева вывода являются вершины, обозначенные терминальными символами грамматики

155. Если для каждой цепочки символов языка L, заданной грамматикой G, можно построить единственный левосторонний и единственный правосторонний граф или вывод, то грамматика называется однозначной

156. Принцип организации распознавателя на основе грамматики предшествования исходит из того, что для каждой упорядоченной пары символов устанавливаются отношения предшествования: Предшествует основе, составляет основу, следует за основой.

157. Отношение предшествования $B_i <\cdot B_{i+1}$, где символ B_{i+1} - крайний левый символ некоторой основы, называют Предшествует основе

158. Отношение предшествования $B_i \cdot > B_{i+1}$, где символ B_i - крайний правый символ некоторой основы, называют Следует за основой

159. Отношение предшествования $B_i =\cdot B_{i+1}$, где символы B_i и B_{i+1} принадлежат одной основе, называют Составляет основу

160. Укажите последовательность шагов в алгоритме «сдвиг-свертка» для грамматики операторного предшествования: порядок верный

1. Поместить в верхушку стека символ начала цепочки, считывающий указатель - в начало входной цепочки символов.
2. Сравнить с помощью отношения предшествования терминальный символ, ближайший к вершине стека (левый символ отношения), с текущим символом входной цепочки (правый символ отношения).

3. Если имеет место отношение <· или =·, то необходимо выполнить сдвиг (перенос текущего символа из входной цепочки в стек и сдвиг считывающего указателя на один шаг вправо) и вернуться к шагу 2. Иначе перейти к шагу 4.
4. Если имеет место отношение ·>, то произвести свертку. Терминальные и нетерминальные символы, составляющие основу, надо удалить из стека, а затем выбрать из грамматики правило, имеющее правую часть, совпадающую с основой, и поместить в стек левую часть выбранного правила. Если правило, совпадающее с основой, найти не удалось, то необходимо прервать выполнение алгоритма и сообщить об ошибке, иначе, если разбор не закончен, вернуться к шагу 2.
5. Если не установлено ни одно отношение предшествования между текущим символом входной цепочки и самым верхним терминальным символом в стеке, то надо прервать выполнение алгоритма и сообщить об ошибке.

161. Укажите последовательность шагов в алгоритме «сдвиг-свертка» для грамматики простого предшествования: порядок верный

1. Поместить в верхушку стека символ начала цепочки, считывающий указатель - в начало входной цепочки символов.
2. Сравнить с помощью отношения предшествования символ, находящийся на вершине стека (левый символ отношения), с текущим символом входной цепочки (правый символ отношения).
3. Если имеет место отношение <· или =·, то произвести сдвиг (перенос текущего символа из входной цепочки в стек и сдвиг считывающего указателя на один шаг вправо) и вернуться к шагу 2. Иначе перейти к шагу 4.
4. Если имеет место отношение ·>, то произвести свертку. Для этого надо найти на вершине стека все символы, связанные отношением =·, удалить эти символы из стека. Затем выбрать из грамматики правило, имеющее правую часть, совпадающую с основой, и поместить в стек левую часть выбранного правила. Если правило, совпадающее с основой, найти не удалось, то необходимо прервать выполнение алгоритма и сообщить об ошибке, иначе, если разбор не закончен, вернуться к шагу 2.
5. Если не установлено ни одно отношение предшествования между текущим символом входной цепочки и символом на вершине стека, то надо прервать выполнение алгоритма и сообщить об ошибке.

162. Наиболее простыми и распространенными являются два типа грамматик предшествования: простого и операторного предшествования

Тема 7. Общие принципы генерации кода

163. Генерация объектного кода выполняется компилятором: перевод внутреннего представления исходной программы в цепочку символов выходного языка

164. Перевод компилятором внутреннего представления исходной программы в цепочку символов выходного языка - это …

    Генерация объектного кода.

165. Генерация объектного кода порождает результирующую программу на:

    Языке ассемблера или машинном коде.

166. Генерация объектного кода выполняется на основе …

    Результатов синтаксического анализа.

167. Генерация объектного кода выполняется только после …

    как выполнен синтаксический анализ программы и все необходимые действия по подготовке к генерации кода.

168. Укажите правильные утверждения.

???

169. Тетрады представляют собой запись операций в форме …

1. * (B, C, T1) …из четырех составляющих: операция, два операнда и результат.

2. + (Tl, D, T2) многоадресный код с явно именуемым результатом (тетрады)

3. * (B, 10, T3) /* Это всё примеры, все правильные, НЕ выбор варианта */

4. – (T2, T3, T4)

5. := (T4, 0, A)

170. Триады представляют собой запись операций в форме …

1. * (B, C) …из трех составляющих: операция и два операнда.

2. + (^1, D) многоадресный код с неявно именуемым результатом (триады)

3. * (B, 10) /* Аналогично предыдущему */

4. – (^2, ^3)

5. := (A, ^4)

171. Генерация объектного кода должна …

выполнять действия, связанные с преобразованием сложных синтаксических структур в линейные цепочки.

172. Особенностью триад является то, что один или оба операнда могут быть ссылками на другую триаду в том случае, если …

в качестве операнда данной триады выступает результат выполнения другой триады.

173. Выражение A:=B*C+D, записанное в виде триад будет иметь вид:

1: * (B, C)

2: + (^1, D)

3: := (A, ^2)

174. Порядок вычисления триад …

операция, заданная триадой, выполняется над операндами, а если в качестве одного из операндов (или обоих операндов) выступает ссылка на другую триаду, то берется результат вычисления той триады.

Или же, если имелось ввиду другое, в том же порядке, что и арифметические операции: скобки, умнож/делен, плюс/минус, в конце присваивание.

175. Если какой-либо операнд в триаде отсутствует, то …

(например, если триада представляет собой унарную операцию), то он может быть опущен или заменен пустым операндом (в зависимости от принятой формы записи и ее реализации).

176. Триады не обладают следующим преимуществом:

ОБЛАДАЮТ: (В ответе выберете то, чего здесь нет)

• являются линейной последовательностью операций, в отличие от синтаксического дерева, и потому проще преобразуются в результирующий код;

• занимают меньше памяти, чем тетрады, дают больше возможностей по оптимизации программы, чем обратная польская запись;

• явно отражают взаимосвязь операций между собой, что делает их применение удобным, особенно при оптимизации внутреннего представления программы;

• промежуточные результаты вычисления триад могут храниться в регистрах процессора, что удобно при распределении регистров и выполнении машинно-зависимой оптимизации;

• по форме представления находятся ближе к двухадресным машинным командам, чем другие формы внутреннего представления программ, а именно эти команды более всего распространены в наборах команд большинства современных компьютеров.

177. Какая форма внутреннего представления отсутствует?

Известны следующие формы внутреннего представления программ:

1. связочные списочные структуры, представляющие синтаксические деревья;
2. многоадресный код с явно именуемым результатом (тетрады);
3. многоадресный код с неявно именуемым результатом (триады);
4. обратная (постфиксная) польская запись операций;
5. ассемблерный код или машинные команды.

178. Триады представляют собой …

линейную последовательность

179. Укажите последовательность этапов выполнения генерации результирующего кода:

1. Компилятор выделает синтаксическую конструкцию из текста исходной программы.
2. Производит фрагмент результирующего кода.
3. Помещает его в текст результирующей программы.
4. Переходит к следующей синтаксической конструкции.

1 -> 2 -> 3 -> 4

180. Машинно-независимые формы внутреннего представления программы:

Синтаксические деревья, триады, тетрады.

181. Машинно-зависимые формы внутреннего представления программы:

Ассемблерный код и машинные команды.

182. Преимущество триад перед тетрадами состоит в следующем:

Триады требуют меньше памяти для своего представления

Триады ближе к двухадресным машинным командам, чем тетрады, а именно эти команды более всего распространены в наборах команд большинства современных компьютеров.

Тема 8. Синтаксически управляемый перевод

183. Идея СУ-перевода заключается в том, что …

синтаксис и семантика языка взаимосвязаны.

184. Для построения результирующего кода различных синтаксических конструкций входного языка используется …

метод СУ-перевода.

185. Является ли процесс оптимизации результирующего кода обязательным?

Нет

186. Идея _____ основана на том, что синтаксис и семантика языка взаимосвязаны. СУ-перевода

187. В общем случае схемы СУ-перевода могут предусматривать выполнение следующих действий:
    1. Помещение в выходной поток данных машинных кодов или команд ассемблера, представляющих собой результат работы (выход) компилятора;
    2. Выдача пользователю сообщений об обнаруженных ошибках и предупреждениях (которые должны помещаться в выходной поток, отличный от потока, используемого для команд результирующей программы);
    3. Порождение и выполнение команд, указывающих, что некоторые действия должны быть произведены самим компилятором (например, операции, выполняемые над данными, размещенными в таблице идентификаторов).

188. СУ-перевод - это основной метод порождения кода результирующей программы на основании результатов _____.

синтаксического разбора.

189. Обобщенная схема синтаксически управляемого перевода имеет вид Tr=(N, T, П, Г, R, S), где N - это …

N - конечное множество нетерминальных символов.

190. Обобщенная схема синтаксически управляемого перевода имеет вид Tr=(N, T, П, Г, R, S), где T - это …

T - конечный входной алфавит.

191. Обобщенная схема синтаксически управляемого перевода имеет вид Tr=(N, T, П, Г, R, S), где П - это …

П - конечный выходной алфавит.

192. Обобщенная схема синтаксически управляемого перевода имеет вид Tr=(N, T, П, Г, R, S), где Г - это …

Г - конечное множество символов перевода вида A_i, где $A \in N$ и i - целое число.

193. Обобщенная схема синтаксически управляемого перевода имеет вид Tr=(N, T, П, Г, R, S), где R - это …

R - конечное множество правил перевода вида:

$A \rightarrow u, A_1 = v_1, ..., A_m = v_m$.

194. Обобщенная схема синтаксически управляемого перевода имеет вид Tr=(N, T, П, Г, R, S), где S - это …

S - начальный символ, выделенный нетерминал из N.

Тема 9. Принципы оптимизации кода

195. Оптимизация кода может выполняться для следующих типовых синтаксических конструкций:
- линейных участков программы;
- логических выражений;
- циклов;
- вызовов процедур и функций;
- других конструкции входного языка.

196. Машинно-независимые методы оптимизации:
позволяют существенно оптимизировать поток управления программы, а также исключить из нее все виды избыточных вычислений (общие подвыражения, инвариантные вычисления в цикле, частично-избыточные вычисления), большую часть мертвого и недостижимого кода.

197. Машинно-зависимые методы оптимизации:

Машинно-зависимые методы оптимизации ориентированы на конкретную архитектуру целевой вычислительной системы, на которой будет выполняться результирующая программа. Как правило, каждый компилятор ориентирован на одну определенную архитектуру целевой вычислительной системы.

198. Можно ли построить компилятор, исключив фазу оптимизации кода?

Да

199. Эффективность объектного кода, построенного компилятором, зависит от:

200. Свертка объектного кода – это выполнение во время компиляции тех операций исходной программы, для которых значения операндов уже известны

201. Оптимизация программы выполняется на этапах:

подготовки к генерации и непосредственно при генерации объектного кода.

202. В качестве показателей эффективности результирующей программы можно использовать два критерия:
- объем памяти
- скорость выполнения

203. Алгоритм свертки для линейного участка программы работает со специальной таблицей, которая содержит пары (<переменная>,<константа>)

204. Что такое оптимизация программы?
модификация системы для улучшения её эффективности

205. Исключение избыточных вычислений (лишних операций) заключается в нахождении и удалении из объектного кода операций, которые повторно обрабатывают одни и те же операнды.

206. Что такое линейный участок программы?
выполняемая по порядку последовательность операций, имеющая один вход и один выход

Тема 10. Принципы функционирования систем программирования

207. Для задания языка программирования необходимо решить следующие задачи:
определить множество допустимых символов языка;

определить множество правильных программ языка;

задать смысл для каждой правильной программы

208. Первый компилятор был разработан для языка программирования “Ассемблер” в 1952 году.

209. На сегодняшнем этапе развития языков программирования наиболее используемыми являются:
- PHP
- C++
- C#
- JavaScript
- Pascal
- Fortran

210. На сегодняшнем этапе развития языков программирования наиболее используемыми являются:
- Java
- Pascal
- Algol
- Fortran

- PHP

211. _____ - это текстовая подстановка, в ходе выполнения которой каждый идентификатор определенного вида заменяется на цепочку символов.
    - макрокоманда
    - определение
    - функция
    - процедура

212. Процесс выполнения макрокоманды называется макрогенерацией.

213. Цепочка символов, получаемая в результате выполнения макрокоманды называется макрорасширением.

214. Процесс замены обнаруженных макрокоманд на соответствующие строки символов называется макроподстановкой.

215. При применении макрокоманд для указания, какие идентификаторы на какие строки необходимо заменять, используют _____.
макроопределения

216. Язык Си является …
    - компилируемым
    - интерпретируемым
    - транслируемым
    - компилируемо-интерпретируемым

217. Язык Pascal является …
    - компилируемым
    - интерпретируемым
    - транслируемым
    - компилируемо-интерпретируемым

218. Язык Java является …
    - компилируемым
    - интерпретируемым
    - транслируемым
    - компилируемо-интерпретируемым

219. Язык C# является …
    - компилируемым
    - интерпретируемым
    - транслируемым
    - компилируемо-интерпретируемым

220. Язык PHP является …
    - компилируемым

- интерпретируемым
- транслируемым
- компилируемо-интерпретируемым

221. Язык JavaScript является …
- компилируемым
- интерпретируемым
- транслируемым
- компилируемо-интерпретируемым

222. Язык Python является …
- компилируемым
- интерпретируемым
- транслируемым
- компилируемо-интерпретируемым

223. Язык Scala является …
- компилируемым
- интерпретируемым
- транслируемым
- компилируемо-интерпретируемым

224. Язык Ruby является …
- компилируемым
- интерпретируемым
- транслируемым
- компилируемо-интерпретируемым

225. Отметьте компиляторы языка Си.
- Intel C++ compiler
- MinGW
- Active Oberon
- Smalltalk

226. Отметьте компиляторы языка Си.
- C++ Builder
- GNU Compiler Collection
- Groovy
- PowerBuilder

227. Отметьте компиляторы языка Java.
- GCJ
- OpenJDK
- Kaffe
- PowerBuilder

228. Отметьте компиляторы языка Java.
- Oracle JDK
- Android JDK
- PowerBuilder
- Smalltalk

229. Отметьте интерпретаторы (среда выполнения) Java.
- J9
- Dalvik
- ART
- Android JDK
- Oracle JDK

230. Отметьте интерпретаторы (среда выполнения) Java.
- Kaffe
- OpenJDK
- HotSpot
- CACAO

231. Отметьте интерпретаторы (среда выполнения) C#.
- Mono
- Portable.NET
- DotGNU
- CACAO

232. Отметьте интегрированные среды программирования на языке C#.
- Visual Studio
- MonoDevelop
- IntelliJ IDEA
- NetBeans
- Code::Blocks

233. Отметьте интегрированные среды программирования на языке Java.
- Visual Studio
- Eclipse
- IntelliJ IDEA
- NetBeans
- Code::Blocks

234. Отметьте интегрированные среды программирования на языке JavaScript.
- Visual Studio
- Eclipse
- IntelliJ IDEA
- NetBeans
- Code::Blocks

235. Отметьте интегрированные среды программирования на языке PHP.

- Visual Studio
- Eclipse
- IntelliJ IDEA
- NetBeans
- Code::Blocks

236. Отметьте интегрированные среды программирования на языке Python.

- Visual Studio
- Eclipse
- IntelliJ IDEA
- NetBeans
- Code::Blocks