Краткое введение в теорию грамматик.

Синтаксис языков программирования принято описывать с помощью специальной математической нотации, которая называется *бесконтекстной грамматикой* или формами Бэкуса-Наура (БНФ).

Бесконтекстная грамматика(далее *грамматика*) характеризуется четырьмя параметрами: G={Vt, Vn, S, P}.
Vt – конечный алфавит терминальных символов (*терминалов*).
В теории компиляции терминалы грамматики - это токены лексического анализа.
Vn – конечный алфавит нетеминальных символов (*нетерминалов*). Содержательную интерпретацию понятия нетерминал мы рассмотром чуть позже.
S – выделенный нетерминал, который называется *стартовым* символом грамматики.
P – конечный набор *продукций* вида A->a1…aN. Здесь A – нетерминал, он называется *левой* частью продукции, a1…aN – цепочка из терминалов и нетерминалов, она называется *правой* частью продукции.
Продукции одного и того же нетерминала обычно записывают совместно A->beta1 | beta2 | … | betaK . В этом случае betaJ называют альтернативами продукций нетерминала A.

Как работают грамматики при описании языков? Продукция рассматривается как переписывающее правило, замещающее левую часть правой.
Говорят, что цепочка alpha порождает цепочку beta за один шаг (alpha->beta), если alpha=gamma1 A gamma2, beta=gamma1 a1…aN gamma2 и A->a1…aN – продукция.
Отношение -> обобщается отношением «порождает за один или несколько шагов» (alpha=>beta), если существует последовательность цепочек gamma0, gamma1, … gammaM , такая что alpha=gamma0->gamma1->…->gammaM=beta.

Особый интерес представляют порождения из стартового символа грамматики S=>alpha. В этом случае цепочка alpha называется *сентенциальной формой*

грамматики. Сентенциальная форма, состоящая только из терминалов, называется предложением.

***Язык* L(G), порождаемый грамматикой – это множество всех предложений.**

Наряду с порождениями из стартового символа S, мы можем рассматривать порождения из других нетерминалов.

L(A) – это множество всех терминальных цепочек, порожденных из A. Такое множество называется *синтаксическим классом* грамматики. Нетерминал можно интерпретировать просто как символическое обозначение синтаксического класса.

В языках программирования широко используются такие синтаксические классы, как выражения. Выражения могут разделяться на подклассы: арифметические, логические и т.п. Каждый синтаксический класс имеет определенную структуру, отличающую его от других классов.

В общем случае грамматика может содержать продукции вида A->eps, здесь eps обозначает пустую цепочку. Если применяется такая продукция, следующая сентенциальная форма будет на один символ короче предшесвующей. Поэтому грамматика, содержащая eps-продукции называется *укорачивающей*. Если eps-продукций нет, то грамматика *неукорачивающая*.

Грамматики G1 и G2 называются *эквивалентными*, если они порождают один и тот же язык: L(G1)=L(G2).

Грамматики G1 и G2 называются *почти эквивалентными*, если L(G1)=L(G2) U {eps}.

Для любой укорачивающей грамматики можно построить почти эквивалентную неукорачивающую грамматику.

Когда мы рассматривали порождение сентенциальных форм, мы не уточняли к какому именно нетерминалу применяется продукция. Если на каждом шаге порождения продукция применяется к самому левому нетерминалу, то порождение называется *левым*, а сентециальные формы – *левосентенциальными*.

Аналогично определются *правые* порождения и *правосентенциальные* формы.

Существует очень наглядное графическое представление порождений, из которого удалена информация о порядке замещения нетерминалов, - *дерево разбора*.

Узлы дерева помечаются термиалами и нетерминалами. Корень дерева помечается стартовым символом S. Из каждого узла, помеченного нетерминалом A выпускается росток, узлы которого помечены символами правой части продукции A->a1…aN. Крону дерева формируют узлы, помеченные только терминалами.

Грамматика, в которой хотя бы для одного предложения имеется два дерева разбора, называется *неоднозначной*, в противном случае грамматика *однозначная*. Не существует общего агоритма, позволяющего для неоднозначной грамматики построить эквивалентную однозначную. Эта задача алгоритмически неразрешима. Вместе с тем для языков программирования неоднозначность грамматики часто удается устранить путем несложных эквивалентных преобразований.

Примеры грамматик.

1. **Язык, порождаемый грамматикой, формируют цепочки, составленные из атомов и сбалансированного набора скобок.**
   Алфавит термиалов:{ ( ) a }
   Продукции:
   S -> L #1
   L -> H E) #2
   H -> (#3 | H E #4
   E -> a #5 | L #6

S –стартовый символ.
Другие нетерминалы обозначают синтаксические классы:
L – список,
H – заголовок списка,
E – элемент списка.

Терминал a обозначает атом.

Вслед за правой частью продукции записан ее уникальный номер.

Продукция #2 порождает «хвост» списка, включая в него последний элемент.

Продукция #3 порождает простейший заголовок списка – символ (.

Продукция #4 «накачивает» список элементами.

Продукции #5 и #6 «утверждают», что элемент списка – это атом или список.

Именно продукция #6 демонстрирует мощь и превосходство аппарата грамматик над конечными автоматами. Язык, порождаемый без применения продукции #6, можно описать с помощью конечного автомата, но в случае применения этой продукции конечные автоматы бессильны.

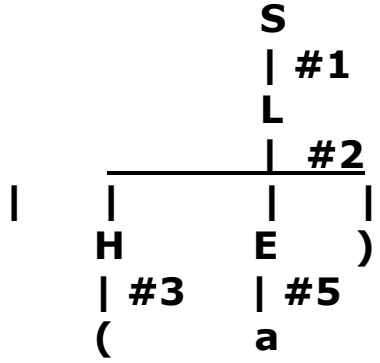
Заметьте. что предложение языка обязательно содержит хотя бы один атом, а значит цепочка () языку не принадлежит.

Правое порождениепредложения (a):

```
   #1  #2      #5       #3
S -> L -> H E ) -> H a ) -> ( a )
```

Продукции замещают самый правый нетерминал сентенциальной формы. **S**

Дерево разбора:



Самостоятельно постройте дерево для ((a a)a).

2. Палиндром – это слово или фраза, которая слева направо прочитывается так же как справа налево.
Слова: ротор, кабак.
Фразы:
Леша на полке клопа нашел
Аргентина манит негра
Ты моден и недомыт
Он в аду давно

Разработайте неукорачивающую грамматику, порождающую язык палиндромов над алфавитом {0 1}.
Примените метод математической индукции. В качестве базы возьмите цепочки 0, 1, 00, 11. Постройте дерево разбра предложения 10100101.

Учебные тренажеры системы конструирования компиляторов получают описание грамматики в форме текстового файла(.txt). Например, файл g0.txt содержит данные:

```
# $g0
   (    )    a
#
   S -> L #1
   L -> H E ) #2
   H -> ( #3 |
        H E #4
   E -> a #5 |
        L #6
```

В начале текста между символами # записан алфавит терминалов грамматики. За ним следует перечень продукций. Первой записана продукция стартового символа.