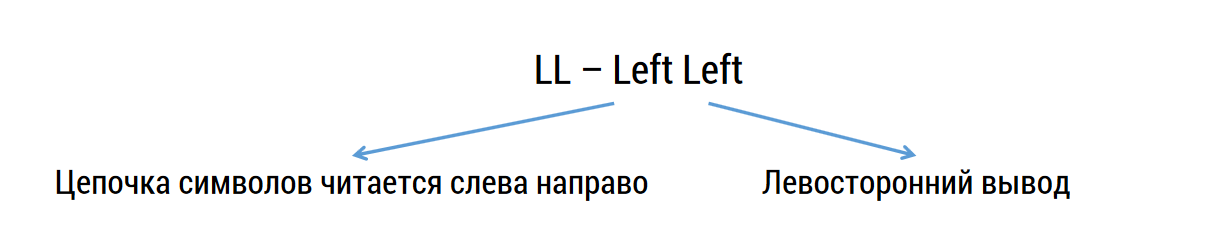
**Методический материал для лабораторной работы № 9 .** Для LL(1) анализатора построить управляющую таблицу M.

**Определение LL(k) грамматики.**Грамматика обладает свойством LL(k), k > 0, если на каждом шаге вывода для однозначного выбора очередной альтернативы достаточно знать символ на верхушке стека и рассмотреть  
первые k символов от текущего положения считывающей головки во входной цепочке символов (использование для выбора одной из множества альтернатив не одного, а нескольких символов входной цепочки).   


* Алгоритм разбора входных цепочек для LL(k)-грамматик –  
  k-предсказывающий алгоритм
* Всякая LL(k)-грамматика для любого k > 0 является однозначной
* Любая грамматика, допускающая разбор по методу рекурсивного спуска, является LL(1)-грамматикой (но не наоборот!)
* Никакая леворекурсивная грамматика не может быть LL(k)-грамматикой

**FIRST и FOLLOW**

**FIRST:** (𝑽𝑻 ∪ 𝑽𝑵)\* →

**FOLLOW:** 𝑽𝑵→

**Множество FIRST(A)** - множество терминальных символов, которыми начинаются цепочки, выводимые из А в грамматике G(VT,VN,P,S), т.е. 𝐹𝐼𝑅𝑆𝑇(𝐴) = {𝑎 ∈ 𝑽𝑻 𝐴 → 𝑎**β**, 𝐴 ∈ 𝑽𝑵, **β** ∈ (𝑽𝑻 ∪ 𝑽𝑵)\*}

**FIRST(α)={a | αaβ} {**ε| **α** ε  **}**

**𝐹𝐼𝑅𝑆𝑇(𝐴)= FIRST(α)**

**Множество FOLLOW(A)** - множество терминальных символов, которые следуют за цепочками, выводимыми за А в грамматике G(VT,VN,P,S), т.е. 𝐹𝑂𝐿𝐿𝑂𝑊(𝐴) = { 𝑎 ∈ 𝑽𝑻 | 𝑆 →\* 𝑆 →\* α𝐴β, β → 𝑎γ, 𝐴 ∈ 𝑽𝑵, α, β, γ ∈ (𝑽𝑻 ∪ 𝑽𝑵)∗} {⊥ | 𝑆 →\* α𝐴}

**Алгоритм формирования массива 𝐹𝐼𝑅𝑆𝑇[] для нетерминалов грамматики**

**𝐹𝐼𝑅𝑆𝑇[ ]**

While ( change)

for A→α G , 𝐴 ∈ 𝑽𝑵

**𝐹𝐼𝑅𝑆𝑇[ А] = 𝐹𝐼𝑅𝑆𝑇[**α**]**

**Алгоритм формирования массива FOLLOW[] для нетерминалов грамматики**

**FOLLOW[]**

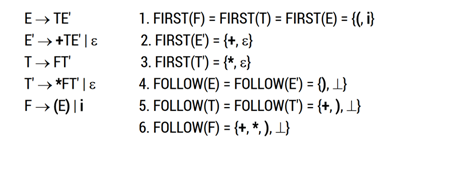
While ( change)

for A→α G, 𝐴 ∈ 𝑽𝑵

for В: **𝛂=βВγ**

**FOLLOW[B]= 𝐹𝐼𝑅𝑆𝑇[γ]** ε ( **FOLLOW[A] if** ε **𝐹𝐼𝑅𝑆𝑇[γ]**)

**Пример**



**Пример**

E→ E+Т

E→ Т

Т→ Т\*F

T→F

F→(E)

F→ n

|  |  |  |
| --- | --- | --- |
|  | **FIRST** | **FOLLOW** |
| **E** | (, n | ⊥ , +, ) |
| **T** | (, n | ⊥ , +, \*, ) |
| **F** | (, n | ⊥ , +, \*, ) |

**G принадлежит классу LL(1) тогда и только тогда, когда для любых двух различных правил А → α | β выполняется:**

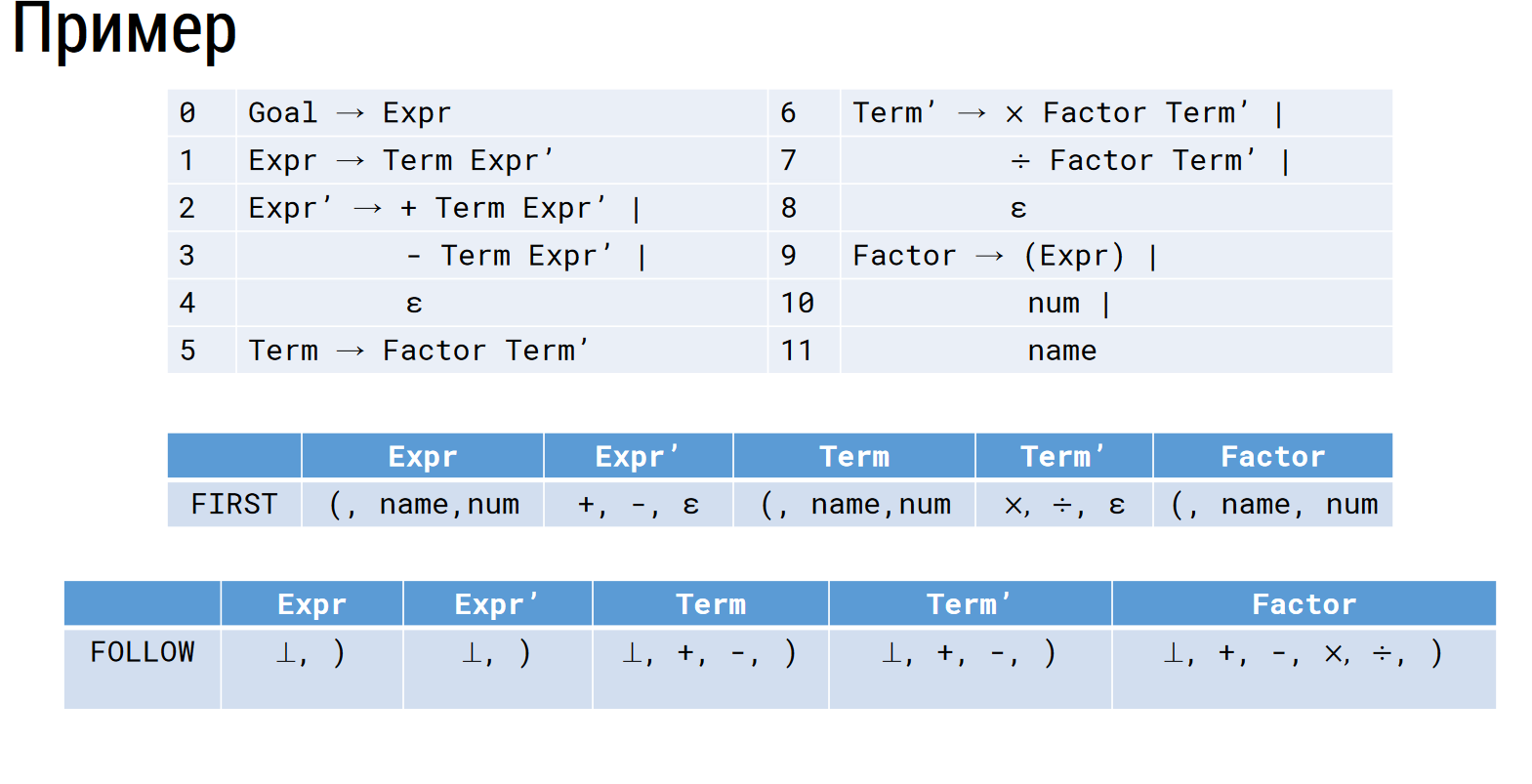
1. Не существует такого терминала **а**, для которого и **α**, и **β** порождают строку, начинающуюся с **а**.  
2. Пустую строку может порождать не более чем одно из правил α или β.  
3. Если β ⇒\* ε, то α не порождает ни одну строку, начинающуюся с терминала из FOLLOW(А). Аналогично, если α ⇒\* ε, то β не порождает ни одну строку, начинающуюся с терминала из FOLLOW(А).

**Таблица предиктивного синтаксического анализа или управляющая таблица.**

𝑇[𝐴, 𝑎] – двумерный массив  
𝐴 – нетерминал  
𝑎 – терминал или символ, маркирующий конец входного потока

**Алгоритм построения управляющей таблицы**

Вход: грамматика G.  
Выход: таблица синтаксического анализа T.  
Метод: для каждого правила грамматики А → α выполняем следующие действия.  
1. Для каждого терминала **а** из FIRST(A) добавляем А → α в ячейку 𝑇 [А, 𝑎].  
2. Если ε ∈ FIRST(A), то для каждого терминала 𝑏 из FOLLOW(А) добавляем А → α в 𝑇[А, 𝑏]. Если ε ∈ FIRST(A) и ⊥ ∈ FOLLOW(А), то добавляем А → α также и в T[A, ⊥].  
Если после выполнения этих действий ячейка 𝑇[А, а] осталась без правила, устанавливаем ее значение равным error (это значение обычно представляется пустой записью таблицы).



Управляющая таблица построенная по грамматике

