

# Лекция 6

8 октября

## Множество *Перв*

- 
- 
- 
- 
- Для данной контекстно-свободной грамматики и цепочки  $x$ , состоящей из символов этой грамматики, определим
  - $\text{Перв}(x)$
  - как множество терминальных символов, которые стоят в начале цепочек, выводимых из  $x$ , т.е. являются первыми символами этих цепочек.

## Пример

- 
- 
- 
- 1.  $\langle S \rangle \rightarrow \langle B \rangle b \langle A \rangle$
- 2.  $\langle S \rangle \rightarrow a \langle B \rangle$
- 3.  $\langle A \rangle \rightarrow \langle B \rangle \langle C \rangle$
- 4.  $\langle A \rangle \rightarrow c \langle A \rangle b \langle B \rangle$
- 5.  $\langle B \rangle \rightarrow d \langle C \rangle$
- 6.  $\langle B \rangle \rightarrow \varepsilon$
- 7.  $\langle C \rangle \rightarrow d \langle S \rangle a$
- 8.  $\langle C \rangle \rightarrow \varepsilon$

## Множества $\text{Перв}()$ для правых частей правил

- 
- 
- 
- 1.  $\text{Перв}(<B> b <A>) = \{b, d\}$
- 2.  $\text{Перв}(a <B>) = \{a\}$
- 3.  $\text{Перв}(<B> <C>) = \{d\}$
- 4.  $\text{Перв}(c <A> b <B>) = \{c\}$
- 5.  $\text{Перв}(d <C>) = \{d\}$
- 6.  $\text{Перв}(e) = \{ \}$
- 7.  $\text{Перв}(d <S> a) = \{d\}$
- 8.  $\text{Перв}(e) = \{ \}$
- 
- 
-

## Аннулирующие правила

- 
- 
- $\text{След}(<A>) = \{a, b, \#\}$
- $\text{След}(<B>) = \{a, b, \#\}$
- $\text{След}(<C>) = \{a, b, \#\}$
- 
- 
- Цепочка  $x$ , состоящая из символов грамматики, называется *аннулирующей*, если из неё можно вывести  $\epsilon$ -цепочку, т.е.
- $x \Rightarrow^* \epsilon$
- Правило грамматики называется *аннулирующим* тогда и только тогда, когда его правая часть является аннулирующей цепочкой.
-

## Множества выбора

- Определим множество выбора для правил произвольного вида следующим образом:
- 
- Для данного правила  
 $\langle A \rangle \rightarrow x$
- где  $x$  – цепочка, состоящая из терминалов и нетерминалов, определим
  - $\text{Выбор}(\langle A \rangle \rightarrow x) = \text{Перв}(x)$
  - если  $x$  – не аннулирующая, и  
 $\text{Выбор}(\langle A \rangle \rightarrow x) = \text{Перв}(x) \cup \text{След}(\langle A \rangle)$
  - если  $x$  – аннулирующая цепочка.
  - 
  - Если  $p$  – номер правила  $\langle A \rangle \rightarrow x$ , то можно написать  $\text{Выбор}(p)$  вместо  $\text{Выбор}(\langle A \rangle \rightarrow x)$ .

## Построение множеств выбора для всех правил примера

- Выбор(1) = Перв(<B> b <A>) = {b,d}
- Выбор(2) = Перв(a<B>) = {a}
- Выбор(3) = Перв(<B> <C>) **и** След(<A>) = {d} **и** {a,b,#} = {a,b,d,#}
- Выбор(4) = Перв(c <A> b <B>) = {c}
- Выбор(5) = Перв(d <C>) = {d}
- Выбор(6) = Перв( $\varepsilon$ ) **и** След(<B>) = {} **и** {a,b,#} = {a,b,#}
- Выбор(7) = Перв(d <S> a) = {d}
- Выбор(8) = Перв( $\varepsilon$ ) **и** След(<C>) = {} **и** {a,b,#} = {a,b,#}
- 
- Заметим, что множества выбора для правил с одинаковыми левыми частями не пересекаются: Выбор(1) и Выбор(2), Выбор(3) и Выбор(4), Выбор(5) и Выбор(6), Выбор(7) и Выбор(8).
-

## Определение LL(1)-грамматики

- 
- 
- Контекстно-свободная грамматика называется *LL(1)-грамматикой* тогда и только тогда, когда множества выбора для правил с одинаковыми левыми частями попарно не пересекаются.
- 
- Аббревиатура *LL* означает “левый вход – левый вывод”.
- Цифра 1 в записи наименования грамматики говорит о том, что решение о применении того или иного правила принимается на основании анализа одного (первого) символа текущей входной цепочки.
- 
- Приведённая в параграфе 5.3 процедура построения распознавателя на основе МП-автомата расширяется так, чтобы быть применимой к *LL(1)*-грамматике.
- Правило 5 должно быть заменено на следующие два правила:

# Изменения в процедуре построения МП-автомата

5<sup>a</sup>. Правило грамматики “применяется” всякий раз, когда верхний магазинный символ является его левой частью, а текущий входной символ принадлежит множеству выбора этого правила.

- Чтобы применить правило вида  
 $\langle A \rangle \rightarrow b x$
- где  $b$  – терминал, а  $x$  – произвольная цепочка, используется переход  
**Заменить( $x$ ), Сдвиг**
- Если правило имеет вид  
 $\langle A \rangle \rightarrow x$
- где  $x$  – цепочка, начинающаяся с нетерминала, используется переход  
**Заменить( $x$ ), Держать**
- Если правило имеет вид  
 $\langle A \rangle \rightarrow b$
- используется переход  
**Вытолкнуть, Сдвиг**
- Если правило имеет вид  
 $\langle A \rangle \rightarrow \epsilon$
- используется переход  
**Вытолкнуть, Держать**
- 5<sup>b</sup>. Если имеется  $\epsilon$ -правило с нетерминалом  $\langle A \rangle$  в левой части, и элемент, соответствующий верхнему магазинному  $\langle A \rangle$  и входному символу  $b$ , не был создан по правилу 5<sup>a</sup>, то можно либо применить это правило, либо отвергнуть цепочку.

## Управляющая таблица МП-автомата для примера

Магазинные символы	a	b	c	d	⊥
<S>	#2	#1	Отвергнуть	#1	Отвергнуть
<A>	#3	#3	#4	#3	#3
<B>	#6	#6	Отвергнуть	#5	#6
<C>	#8	#8	Отвергнуть	#7	#8
a	Вытолк Сдвиг	Отвергнуть	Отвергнуть	Отвергнуть	Отвергнуть
b	Отвергнуть	Вытолк Сдвиг	Отвергнуть	Отвергнуть	Отвергнуть
∇	Отвергнуть	Отвергнуть	Отвергнуть	Отвергнуть	Допустить

Начальное содержимое магазина  $<S>\nabla$

## Элементы таблицы

- #1 Заменить(*B*) *b* <*A*>, Держать
- #2 Заменить(*B*), Сдвиг
- #3 Заменить(*B*)<*C*>, Держать
- #4 Заменить(<*A*> *b* <*B*>), Сдвиг
- #5 Заменить(<*C*>), Сдвиг
- #6 Вытолк, Держать
- #7 Заменить(<*S*> *a*), Сдвиг
- #8 Вытолк, Держать

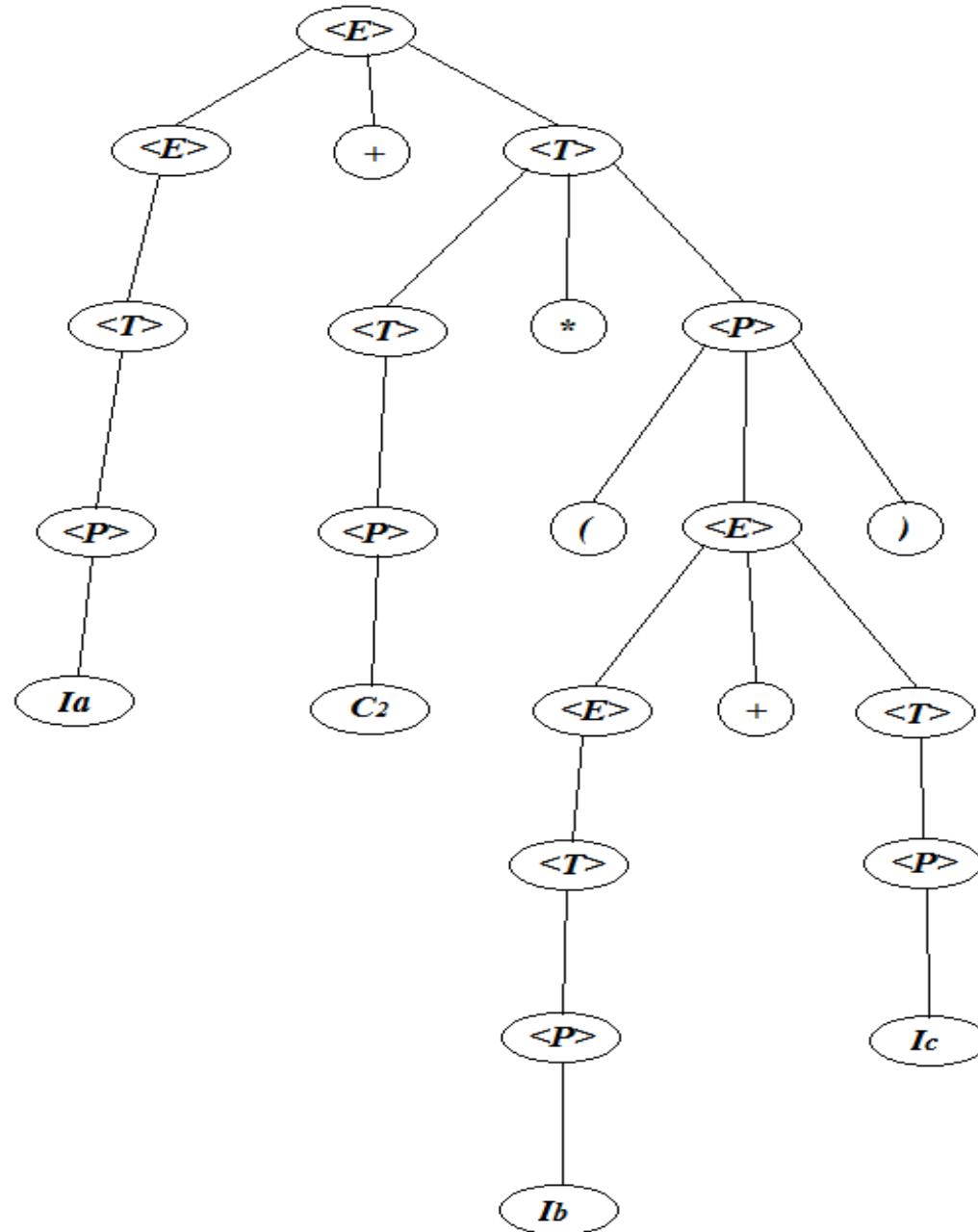
## Пример

- Пусть дана следующая грамматика:
- 
- 1.  $\langle E \rangle \rightarrow \langle E \rangle + \langle T \rangle$
- 2.  $\langle E \rangle \rightarrow \langle T \rangle$
- 3.  $\langle T \rangle \rightarrow \langle T \rangle * \langle P \rangle$
- 4.  $\langle T \rangle \rightarrow \langle P \rangle$
- 5.  $\langle P \rangle \rightarrow (\langle E \rangle)$
- 6.  $\langle P \rangle \rightarrow I$
- 7.  $\langle P \rangle \rightarrow C$
- 
- где  $I$  и  $C$  есть терминалы, обозначающие идентификатор и константу соответственно.

## Вывод выражения

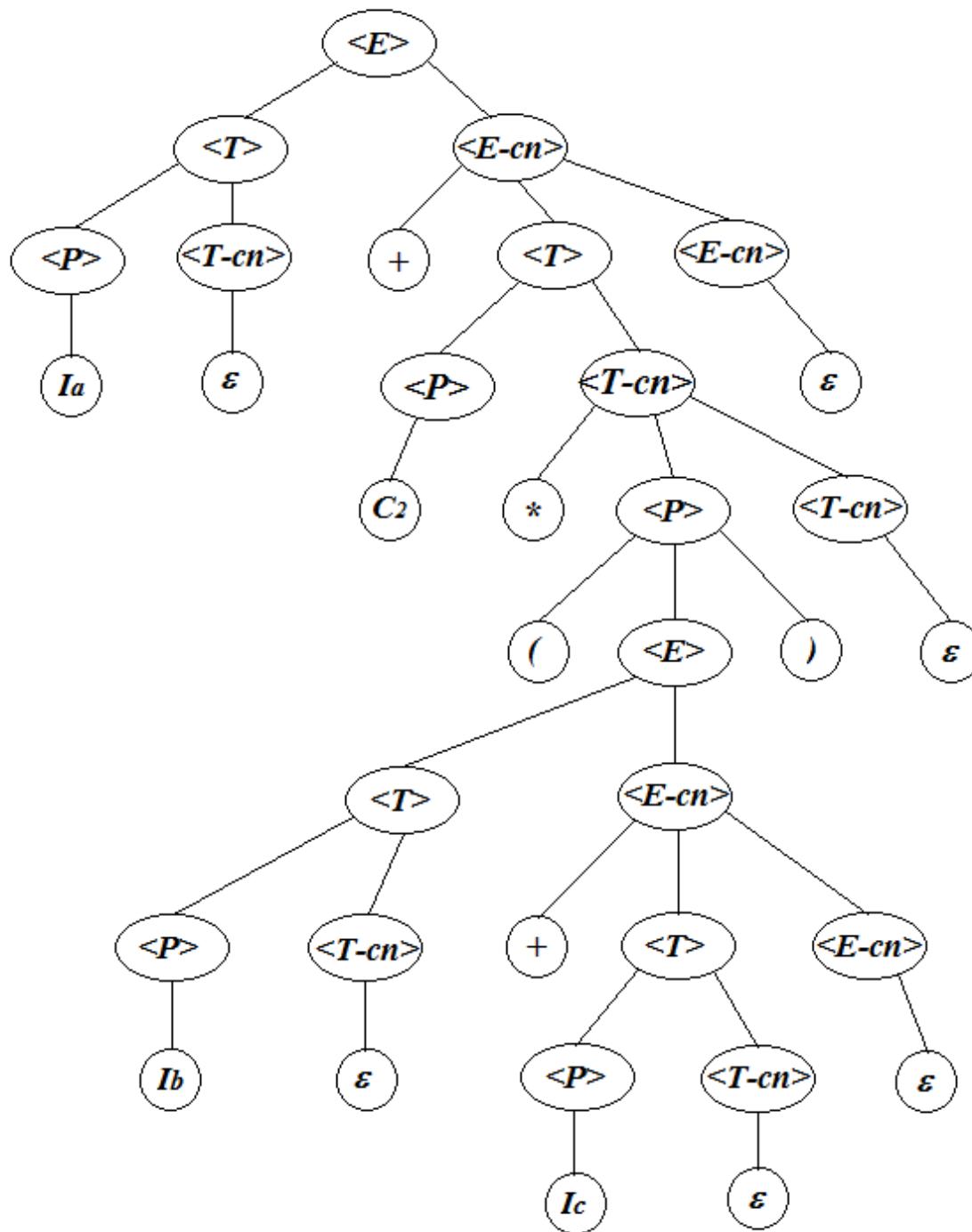
- Приведённая грамматика описывает арифметические выражения с двумя операциями: сложением и умножением.  
Например, вывод выражения  $a+2*(b+c)$  выглядит следующим образом:
  - 
  - $\langle E \rangle \Rightarrow \langle E \rangle + \langle T \rangle \Rightarrow \langle E \rangle + \langle T \rangle * \langle P \rangle \Rightarrow \langle E \rangle + \langle T \rangle * (\langle E \rangle) \Rightarrow$
  - $\Rightarrow \langle E \rangle + \langle T \rangle * (\langle E \rangle + \langle T \rangle) \Rightarrow \langle E \rangle + \langle T \rangle * (\langle E \rangle + \langle P \rangle) \Rightarrow$
  - $\Rightarrow \langle E \rangle + \langle T \rangle * (\langle E \rangle + I_c) \Rightarrow \langle E \rangle + \langle T \rangle * (\langle T \rangle + I_c) \Rightarrow$
  - $\Rightarrow \langle E \rangle + \langle T \rangle * (\langle P \rangle + I_c) \Rightarrow \langle E \rangle + \langle T \rangle * (I_b + I_c) \Rightarrow \langle E \rangle + \langle P \rangle * (I_b + I_c) \Rightarrow$
  - $\Rightarrow \langle E \rangle + C_2 * (I_b + I_c) \Rightarrow \langle T \rangle + C_2 * (I_b + I_c) \Rightarrow \langle P \rangle + C_2 * (I_b + I_c) \Rightarrow$
  - $\Rightarrow I_a + C_2 * (I_b + I_c)$
  -
- Если взять значения лексем, то получаем наше выражение  $a+2*(b+c)$ .
- Более наглядно этот вывод представляется с помощью следующего синтаксического дерева:

# Синтаксическое дерево



## Другой пример

- 
- Рассмотрим теперь следующую грамматику:
- 
- 1.  $\langle E \rangle \rightarrow \langle T \rangle \langle E\text{-список} \rangle$
- 2.  $\langle E\text{-список} \rangle \rightarrow + \langle T \rangle \langle E\text{-список} \rangle$
- 3.  $\langle E\text{-список} \rangle \rightarrow \varepsilon$
- 4.  $\langle T \rangle \rightarrow \langle P \rangle \langle T\text{-список} \rangle$
- 5.  $\langle T\text{-список} \rangle \rightarrow * \langle P \rangle \langle T\text{-список} \rangle$
- 6.  $\langle T\text{-список} \rangle \rightarrow \varepsilon$
- 7.  $\langle P \rangle \rightarrow (\langle E \rangle)$
- 8.  $\langle P \rangle \rightarrow I$
- 9.  $\langle P \rangle \rightarrow C$
- 
- Дерево вывода для того же самого выражения  $a+2*(b+c)$  имеет теперь вид:



## Множества выбора

- Эта грамматика является **LL(1)**-грамматикой. Это можно проверить, построив множества выбора для всех правил:
- 
- $\text{Выбор}(1) = \text{Перв}(\langle T \rangle \langle E\text{-список} \rangle) = \{I, C, \{\}$
- $\text{Выбор}(2) = \text{Перв}(+ \langle T \rangle \langle E\text{-список} \rangle) = \{+\}$
- $\text{Выбор}(3) = \text{След}(\langle E\text{-список} \rangle) = \{\}, \wedge\}$
- $\text{Выбор}(4) = \text{Перв}(\langle P \rangle \langle T\text{-список} \rangle) = \{I, C, \{\}$
- $\text{Выбор}(5) = \text{Перв}(* \langle P \rangle \langle T\text{-список} \rangle) = \{*\}$
- $\text{Выбор}(6) = \text{След}(\langle T\text{-список} \rangle) = \{+, \}, \wedge\}$
- $\text{Выбор}(7) = \text{Перв}((\langle E \rangle)) = \{\{\}$
- $\text{Выбор}(8) = \text{Перв}(I) = \{I\}$
- $\text{Выбор}(9) = \text{Перв}(C) = \{C\}$
- 
- Множества выбора для правил с одинаковыми левыми частями попарно не пересекаются (правила 2 и 3, правила 5 и 6, правила 7, 8 и 9).

# Управляющая таблица

Магазинные символы	I	C	+	*	(	)	$\perp$
<E>	#1	#1	Отверг	Отверг	#1	Отверг	Отверг
<T>	#4	#4	Отверг	Отверг	#4	Отверг	Отверг
<E-сп>	Отверг	Отверг	#2	Отверг	Отверг	#3	#3
<P>	#8	#9	Отверг	Отверг	#7	Отверг	Отверг
<T-сп>	Отверг	Отверг	#6	#5	Отверг	#6	#6
)	Отверг	Отверг	Отверг	Отверг	Отверг	Вытолк Сдвиг	Отверг
$\nabla$	Отверг	Отверг	Отверг	Отверг	Отверг	Отверг	Допустить

Начальное содержимое магазина <E> $\nabla$

## Элементы таблицы

- #1 Заменить( $T$ )  $E$ -список), Держать
- #2 Заменить( $T$ )  $E$ -список), Сдвиг
- #3 Вытолкнуть, Держать
- #4 Заменить( $P$ )  $T$ -список), Держать
- #5 Заменить( $P$ )  $T$ -список), Сдвиг
- #6 Вытолкнуть, Держать
- #7 Заменить( $E$ )), Сдвиг
- #8 Вытолкнуть, Сдвиг
- #9 Вытолкнуть, Сдвиг

# Пример разбора выражения $a+2*b$ , т.е. $I_a+C_2*I_b$

Текущая входная цепочка	Содержимое магазина
$I + C * I \perp$	$\langle E \rangle \nabla$
$I + C * I \perp$	$\langle T \rangle \langle E\text{-сп} \rangle \nabla$
$I + C * I \perp$	$\langle P \rangle \langle T\text{-сп} \rangle \langle E\text{-сп} \rangle \nabla$
$+ C * I \perp$	$\langle T\text{-сп} \rangle \langle E\text{-сп} \rangle \nabla$
$+ C * I \perp$	$\langle E\text{-сп} \rangle \nabla$
$C * I \perp$	$\langle T \rangle \langle E\text{-сп} \rangle \nabla$
$C * I \perp$	$\langle P \rangle \langle T\text{-сп} \rangle \langle E\text{-сп} \rangle \nabla$
$* I \perp$	$\langle T\text{-сп} \rangle \langle E\text{-сп} \rangle \nabla$
$I \perp$	$\langle P \rangle \langle T\text{-сп} \rangle \langle E\text{-сп} \rangle \nabla$
$\perp$	$\langle T\text{-сп} \rangle \langle E\text{-сп} \rangle \nabla$
$\perp$	$\langle E\text{-сп} \rangle \nabla$
$\perp$	$\nabla$
Допустить	

## **6. ТРАНСЛИРУЮЩИЕ ГРАММАТИКИ**

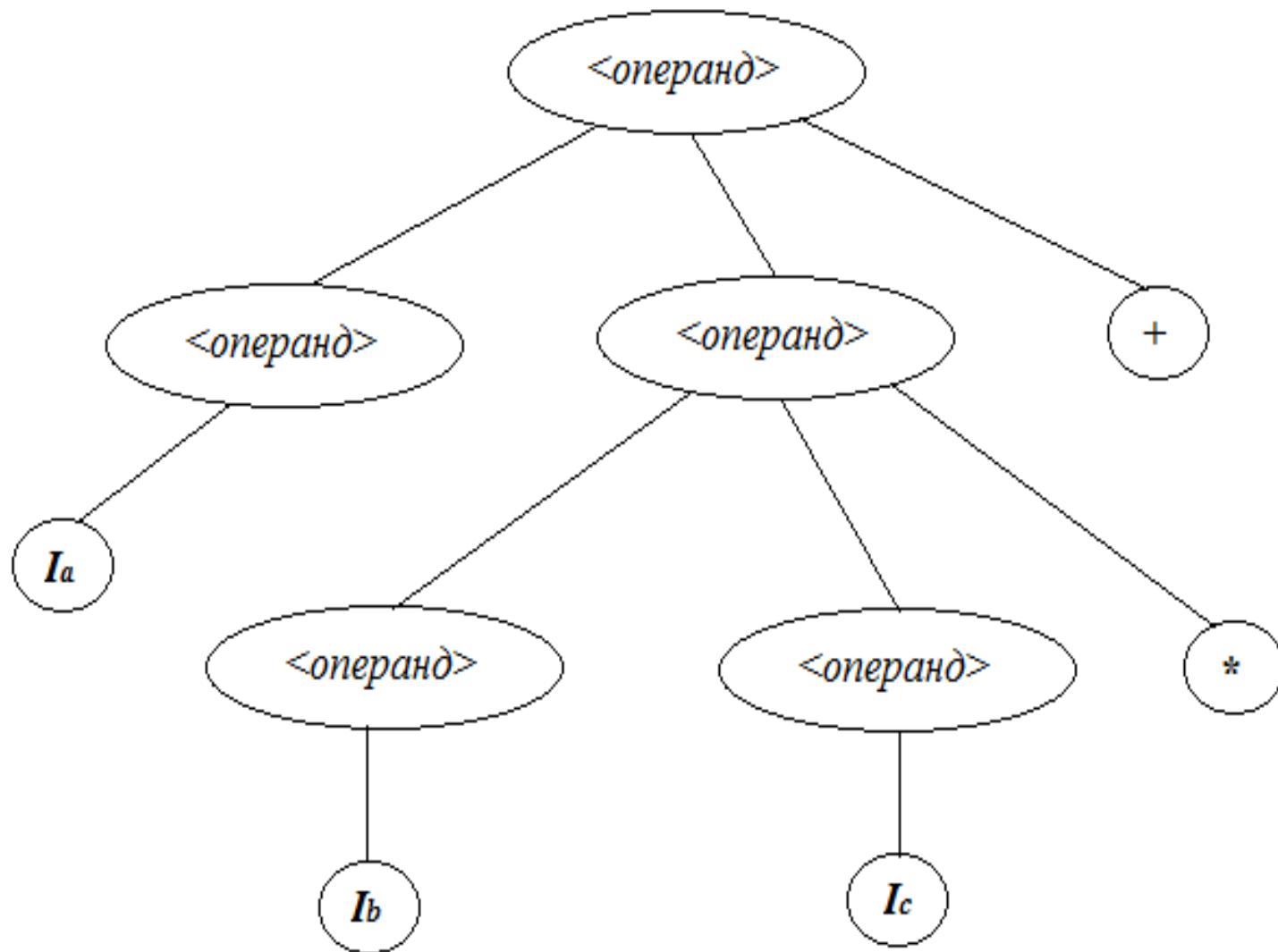
## **6.1. Цепочечный перевод**

- Рассмотрим трансляцию на примере перевода арифметических выражений из одной формы записи в другую.
  - Обычный способ записи арифметических выражений называется *инфиксная запись*. В этом случае знаки операций записываются между операндами. Существуют и другие способы записи арифметических выражений, в частности *постфиксная запись* или *обратная польская запись*.
  - При использовании постфиксной записи знаки операций записываются после операндов:
  - |                         |                           |
|-------------------------|---------------------------|
| <i>Инфиксная запись</i> | <i>Постфиксная запись</i> |
| $a + b * c$             | $a b c * +$               |
  - Выражение в постфиксной форме вычисляется следующим образом. Мы движемся по цепочке, состоящей из операндов и знаков операций, слева направо. Как только встречается знак операции, берутся два предшествующих ему операнда, выполняется операция, и результат помещается на место этих операндов и знака операции. Далее продолжается движение слева направо, и выполняются аналогичные действия. Ещё одной особенностью постфиксной записи является то, что здесь не требуются скобки для задания приоритета операций.

## Грамматика польских выражений

- Множество польских выражений с операциями + и \* можно породить с помощью следующей грамматики:
- 
- $\langle \text{операнд} \rangle \rightarrow \langle \text{операнд} \rangle \langle \text{операнд} \rangle +$
- $\langle \text{операнд} \rangle \rightarrow \langle \text{операнд} \rangle \langle \text{операнд} \rangle *$
- $\langle \text{операнд} \rangle \rightarrow I$
- 
- Для приведённого примера дерево вывода имеет следующий вид:

## Синтаксическое дерево



# Механизм перевода инфиксной записи в постфиксную

- Рассмотрим теперь, как мог бы работать некоторый механизм, осуществляя перевод инфиксной записи в постфиксную. Для нашего выражения последовательность действий будет следующая:
  - 
  - Читать(*a*) Печатать(*a*) Читать(+) Читать(*b*) Печатать(*b*) Читать(\*)
  - Читать(*c*) Печатать(*c*) Печатать(\*) Печатать(+)
  - 
  - Слово Читать не надо воспринимать слишком буквально, обычно это эквивалентно сдвигу по входной цепочке. Вместо Печатать мы будем указывать выдачу символа путём заключения его в фигурные скобки.
  - Таким образом, последовательность действий ввода и вывода для нашего выражения будет выглядеть следующим образом:
    - 
    - $a \{a\} + b \{b\} * c \{c\} \{*\} \{+\}$
    - 
    - Если теперь выбрать только действия вывода, то получим
    - $\{a\} \{b\} \{c\} \{*\} \{+\}$

## Исходная грамматика

- А после выполнения самих действий вывода получаем постфиксную запись
- 
- $a \ b \ c \ * \ +$
- 
- Мы рассмотрели обработку одного конкретного выражения. Нам же нужно переводить произвольные выражения.
- Вернёмся к грамматике для арифметических (инфиксных) выражений:
- 
- $\langle E \rangle \rightarrow \langle E \rangle + \langle T \rangle$
- $\langle E \rangle \rightarrow \langle T \rangle$
- $\langle T \rangle \rightarrow \langle T \rangle * \langle P \rangle$
- $\langle T \rangle \rightarrow \langle P \rangle$
- $\langle P \rangle \rightarrow (\langle E \rangle)$
- $\langle P \rangle \rightarrow I$
- 
-

# Модифицированная грамматика

- Например, чтобы напечатать идентификатор после того, как он прочитан, правило 6 изменяется следующим образом:
  - 
  - $\langle P \rangle \rightarrow I \{ I \}$
  -
- Чтобы напечатать знак сложения после того, как напечатаны оба его операнда, правило 1 изменяется следующим образом:
  - 
  - $\langle E \rangle \rightarrow \langle E \rangle + \langle E \rangle \{ + \}$
  -
- В итоге получаем следующую грамматику:
  - 
  - $\langle E \rangle \rightarrow \langle E \rangle + \langle T \rangle \{ + \}$
  - $\langle E \rangle \rightarrow \langle T \rangle$
  - $\langle T \rangle \rightarrow \langle T \rangle * \langle P \rangle \{ * \}$
  - $\langle T \rangle \rightarrow \langle P \rangle$
  - $\langle P \rangle \rightarrow (\langle E \rangle)$
  - $\langle P \rangle \rightarrow I \{ I \}$

## Вывод цепочки

- Полученная грамматика представляет собой то, что обычно называют *транслирующей грамматикой* или *грамматикой перевода*. Поскольку действия, заключённые в скобки, в данном случае являются выдачей символа, такие грамматики часто называют *грамматиками цепочечного перевода*.
- *Транслирующей грамматикой* или *грамматикой перевода* называется контекстно-свободная грамматика, множество терминальных символов которой разбито на множество входных символов и множество символов действия.
- Цепочки языка, порождаемого транслирующей грамматикой, представляют собой последовательности символов действия.
- Рассмотрим, как осуществляется перевод выражения  $a + b * c$ .
- Вывод этого выражения по исходной грамматике выглядит следующим образом:
  - 
  - $\langle E \rangle \Rightarrow \langle E \rangle + \langle T \rangle \Rightarrow \langle T \rangle + \langle T \rangle \Rightarrow^* \langle P \rangle + \langle T \rangle * \langle P \rangle \Rightarrow$
  - $= \Rightarrow \langle P \rangle + \langle P \rangle * \langle P \rangle \Rightarrow^* I_a + I_b * I_c$
  - 
  - т.е.  $a + b * c$ .

## Вывод на основе транслирующей грамматики

- В случае транслирующей грамматики получаем:
  - 
  - $\langle E \rangle \Rightarrow \langle E \rangle + \langle T \rangle \{+\} \Rightarrow \langle T \rangle + \langle T \rangle \{+\} \Rightarrow \langle T \rangle + \langle T \rangle * \langle P \rangle \{*\} \{+\} \Rightarrow$ 
    - $= \Rightarrow \langle P \rangle + \langle T \rangle * \langle P \rangle \{*\} \{+\} \Rightarrow \langle P \rangle + \langle P \rangle * \langle P \rangle \{*\} \{+\} \Rightarrow$
    - $= \Rightarrow I_a \{I_a\} + \langle P \rangle * \langle P \rangle \{*\} \{+\} \Rightarrow I_a \{I_a\} + I_b \{I_b\} * \langle P \rangle \{*\} \{+\} \Rightarrow$
    - $= \Rightarrow I_a \{I_a\} + I_b \{I_b\} * I_c \{I_c\} \{*\} \{+\}$
  - 
  - Оставляя только символы действия, получаем:
    - $\{I_a\} \{I_b\} \{I_c\} \{*\} \{+\}$
    -
  - Теперь, выполнив эти действия, получим постфиксную форму  $abc^*+$ .