

Алгоритмы синтаксического разбора. Польская запись

1. Общие сведения

Обычная форма выражений называется *инфиксной*. Знаки операции размещаются между операндами, с которыми они взаимодействуют, например:

$$\begin{aligned} a + b \\ a + b * c \\ (a + b) * c \end{aligned}$$

В префиксной (прямой польской) записи знаки операций записываются до операндов, например:

$$\begin{aligned} + a b \\ + a * b c \\ * + a b c \end{aligned}$$

В постфиксной (обратной польской) записи знаки операций записываются после операндов, например:

$$\begin{aligned} a b + \\ a b c * + \\ a b + c * \end{aligned}$$

Польская запись не содержит скобок.

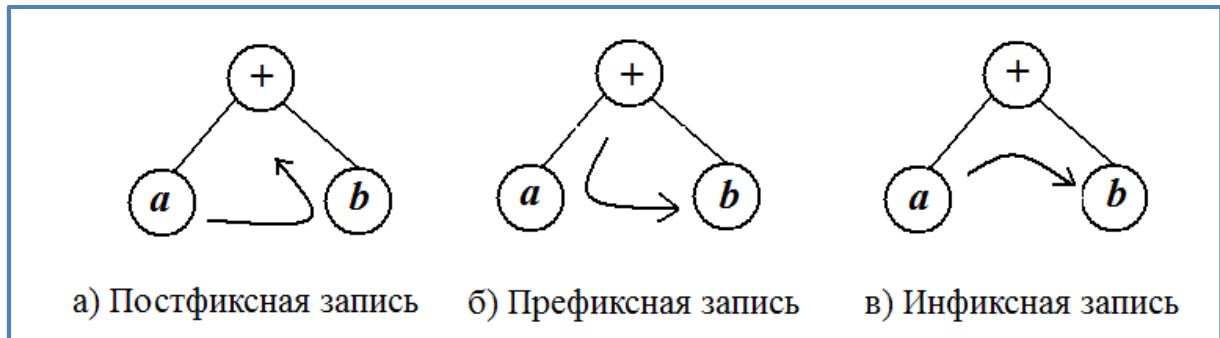
$$a + b * c - a / (a + b) \quad \Rightarrow \quad a b c * + a a b + / -$$

Выражение читается слева направо. Каждая операция выполняется над двумя операндами, непосредственно стоящими перед знаком этой операции. Последовательность операндов и знак операции в выражении заменяется результатом этой операции. Результатом вычисления всего выражения становится результат последней вычисленной операции.

Такая нотация названа польской в честь ее изобретателя — польского математика и логика **Яна Лукасевича** (Jan Łukasiewicz, 1878–1956).

Обратную польскую запись называют польской *инверсной* записью (ПОЛИЗ). ПОЛИЗ удобна:

- для вычисления выражений;
- как промежуточная форма представления выражений в трансляторе;
- как промежуточная форма представления операторов языков программирования.

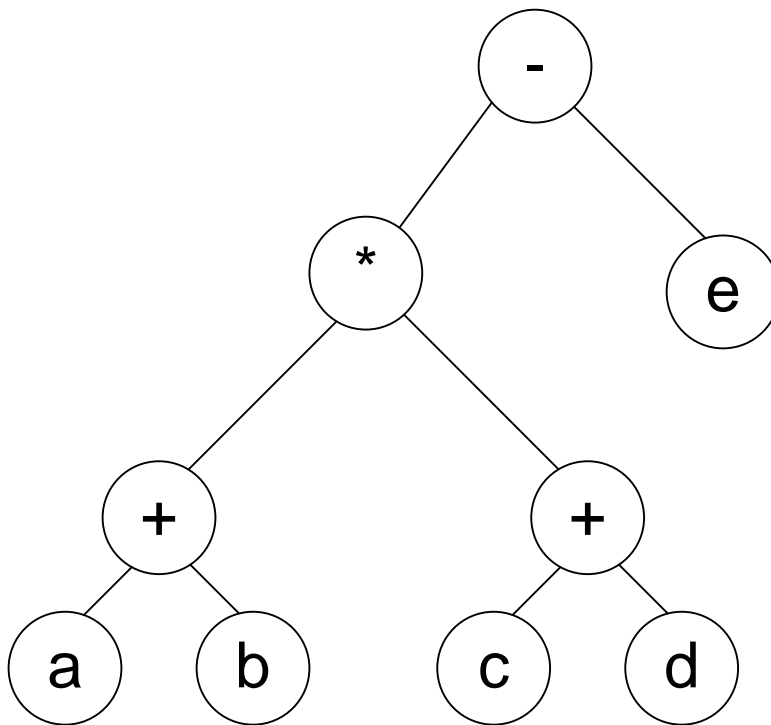


По дереву вывода легко получить обратную польскую запись выражения.

В этом дереве листья – это операнды или константы (терминальные символы), внутренние узлы – операции.

Выполняем обход дерева снизу вверх и слева направо (для каждой вершины вначале посещается ее левое поддерево, потом правое и в последнюю очередь сама вершина) и для каждой вершины выводим ее значение.

$$(a+b)*(c+d) - e$$



$$ab+cd+*e-$$

2. Вычисление выражения в обратной польской записи за один просмотр

<i>№</i>	<i>Выражение</i>	<i>Переменные</i>	<i>Стек</i>
0	ab+cd+*e-	R ₁ =a	R ₁
1	R ₁ b+cd+*e-	R ₂ =b	R ₁ R ₂
2	R ₁ R ₂ +cd+*e-	R ₃ = R ₁ +R ₂	R ₃
3	R ₃ cd+*e-	R ₄ =c	R ₃ R ₄
4	R ₃ R ₄ d+*e-	R ₅ =d	R ₃ R ₄ R ₅
5	R ₃ R ₄ R ₅ +*e-	R ₆ = R ₄ +R ₅	R ₃ R ₆
6	R ₃ R ₆ *e-	R ₇ = R ₃ *R ₆	R ₇
7	R ₇ e-	R ₈ =e	R ₇ R ₈
8	R ₇ R ₈ -	R ₉ = R ₇ -R ₈	R ₉
9	R ₉		

3. Алгоритм построения польской записи

Определим приоритет операций:

<i>Приоритет</i>	<i>Операция</i>
1	(
1)
2	+
2	-
3	*
3	/

Алгоритм построения:

- исходная строка: **выражение**;
- результирующая строка: **польская запись**;
- стек: **пустой**;
- исходная строка просматривается слева направо;
- **операнды** переносятся в результирующую строку в порядке их следования;
- **операция** записывается в стек, если стек пуст или в вершине стека лежит отрывающая скобка;
- **операция** выталкивает все операции с **большим** или **равным** приоритетом в результирующую строку;
- **открывающая скобка** помещается в стек;
- **закрывающая скобка** выталкивает все операции до открывающей скобки, после чего обе скобки уничтожаются;
- по концу разбора исходной строки все операции, оставшиеся в стеке, выталкиваются в результирующую строку.

4. Пример.

<i>Исходная строка</i>	<i>Результирующая строка</i>	<i>Стек</i>
$(a + b) * (c + d) - e$		
$a + b) * (c + d) - e$		(
$+b) * (c + d) - e$	a	(
$b) * (c + d) - e$	a	+(
$) * (c + d) - e$	ab	+(
$* (c + d) - e$	$ab +$	
$(c + d) - e$	$ab +$	*
$c + d) - e$	$ab +$	(*
$+d) - e$	$ab + c$	(*
$d) - e$	$ab + c$	+(*
$) - e$	$ab + cd$	+(*
$-e$	$ab + cd +$	*
e	$ab + cd +*$	-
	$ab + cd +* e$	-
	$ab + cd +* e -$	

| Стек организован по принципу LIFO.

5. Расширение алгоритма построения польской записи

Легко расширить алгоритм так, чтобы он обрабатывал выражения, содержащие *вызовы функций, элементы массива, другие виды скобок* и т.п.

<i>Приоритет</i>	<i>Операция</i>
0	(
0)
1	,
2	+
2	-
3	*
3	/
4	[
4]

Алгоритм построения (пример расширения):

- исходная строка: выражение;
- результирующая строка: польская запись;
- стек: пустой;
- исходная строка просматривается слева направо;
- **операнды** переносятся в результирующую строку в порядке их следования;
- **операция** записывается в стек, если стек пуст или в вершине стека лежит отрывающая скобка;
- **операция** выталкивает все операции с *большим* или *равным* приоритетом в результирующую строку;
- **запятая** не помещается в стек, и если в стеке есть операции, то все выбираются в строку;
- **открывающая скобка** помещается в стек;
- **закрывающая скобка** выталкивает все операции до открывающей скобки, после чего обе скобки уничтожаются;
- **квадратная открывающая скобка** помещается в стек;
- **квадратная закрывающая скобка** выталкивает все до открывающей квадратной скобки и генерирует последовательность @n (индекс n указывает число операндов, разделенных запятыми);
- по концу разбора исходной строки все операции, оставшиеся в стеке, выталкиваются в результирующую строку.

Выполнение:

<i>Исходная строка</i>	<i>Результирующая строка</i>	<i>Стек</i>
$a * (b + [[c, d] + e, g]) - k/[e, f]$		
$* (b + [[c, d] + e, g]) - k/[e, f]$	a	
$(b + [[c, d] + e, g]) - k/[e, f]$	a	$*$
$b + [[c, d] + e, g]) - k/[e, f]$	a	$* ($
$+ [[c, d] + e, g]) - k/[e, f]$	ab	$* ($
$[[c, d] + e, g]) - k/[e, f]$	ab	$* (+$
$[c, d] + e, g]) - k/[e, f]$	ab	$* (+[$
$c, d] + e, g]) - k/[e, f]$	ab	$* (+[[$
$, d] + e, g]) - k/[e, f]$	abc	$* (+[[$
$d] + e, g]) - k/[e, f]$	abc	$* (+[[$
$] + e, g]) - k/[e, f]$	$abcd$	$* (+[[$
$+e, g]) - k/[e, f]$	$abcd@_2$	$* (+[$
$e, g]) - k/[e, f]$	$abcd@_2$	$* (+[+$
$, g]) - k/[e, f]$	$abcd@_2e$	$* (+[+$
$g]) - k/[e, f]$	$abcd@_2e +$	$* (+[$
$]) - k/[e, f]$	$abcd@_2e + g$	$* (+[$
$) - k/[e, f]$	$abcd@_2e + g@_2$	$* (+$
$-k/[e, f]$	$abcd@_2e + g@_2 +$	$*$
$k/[e, f]$	$abcd@_2e + g@_2 +*$	$-$
$/[e, f]$	$abcd@_2e + g@_2 +* k$	$-$
$[e, f]$	$abcd@_2e + g@_2 +* k$	$-/$
$e, f]$	$abcd@_2e + g@_2 +* k$	$-/[$
$, f]$	$abcd@_2e + g@_2 +* ke$	$-/[$
$f]$	$abcd@_2e + g@_2 +* ke$	$-/[$
$]$	$abcd@_2e + g@_2 +* ke f$	$-/[$
	$abcd@_2e + g@_2 +* ke f@_2$	$-/$
	$abcd@_2e + g@_2 +* ke f@_2/$	$-$
	$abcd@_2e + g@_2 +* ke f@_2/-$	