Лекция 20

БГТУ, ФИТ, ПОИТ, 3 семестр, Языки программирования

Алгоритмы синтаксического разбора. Польская запись

Обычная форма выражений называется ***инфиксной.*** Знаки операции размещаются между операндами, например:

a + b
a + b*c
(a + b)*c

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

+ a b
+ a * b c
**** + a b c***

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

a b +
a b c * +
a b + c *

Польская запись не содержит скобок.

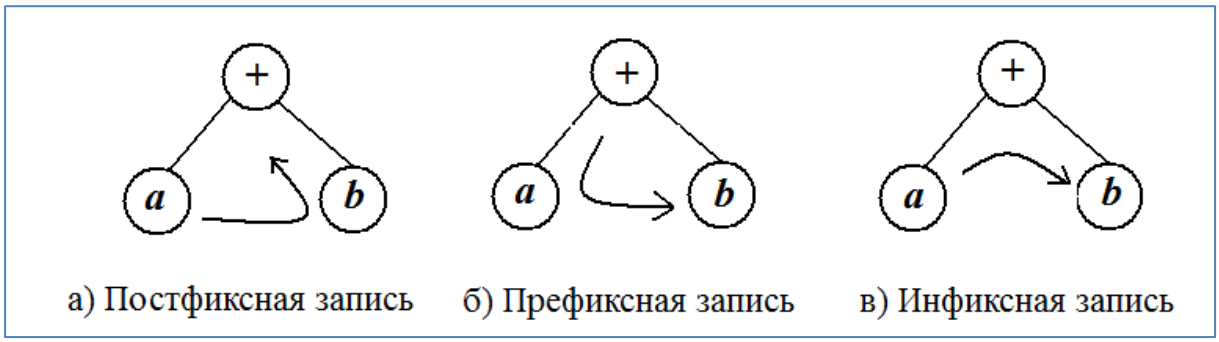
a + b*c – a/(a + b) => ***a b c * + a a b + / –***

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:

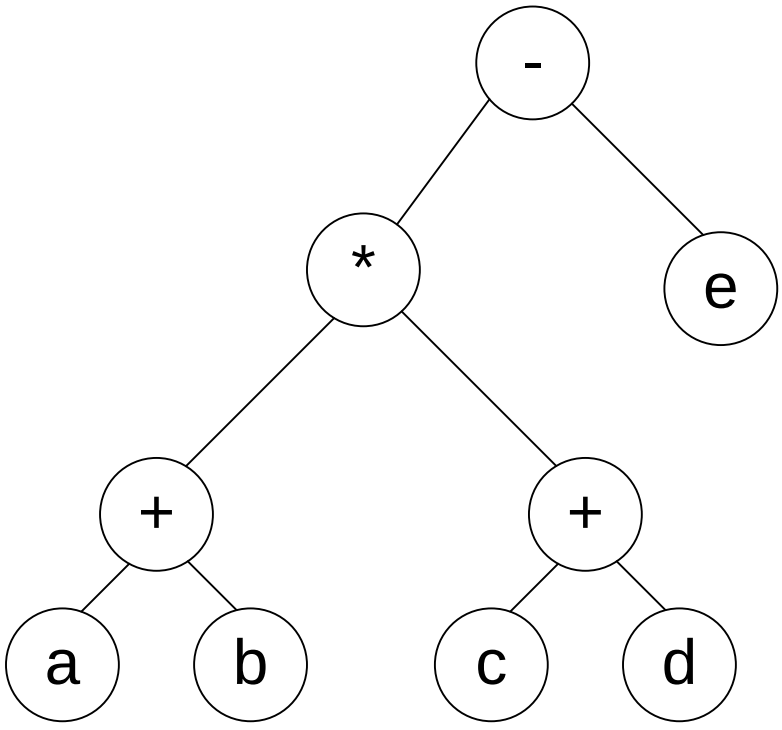
- для вычисления выражений;
- как промежуточная форма представления выражений в трансляторе;
- как промежуточная форма представления операторов языков программирования.



По дереву вывода легко получить обратную польскую запись выражения. В этом дереве листья – это операнды или константы, 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-

1. **Вычисление выражения в обратной польской записи за один просмотр**

0	ab+cd+*e-	R_1=a
1	R_1b+cd+*e-	R_2=b
2	R_1R_2+cd+*e-	R_3= R_1+R_2
3	R_3cd+*e-	R_4=c
4	R_3R_4d+*e-	R_5=d
5	$R_3R_4R_5$+*e-	R_6= R_4+R_5
6	R_3R_6*e-	R_6= R_4+R_5
7	R_3R_6*e-	R_7= R_3*R_6
8	R_7e-	R_8=e
9	R_7R_8-	R_9= R_7-R_8
10	R_9	

2. **Алгоритм построения польской записи**

Определим приоритет операций:

Приоритет	Операция
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;
- **отрывающая 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.

3. Пример.

Исходная строка	Результирующая строка	Стек
$(a+b)*(c+d)-e$		
$a+b)*(c+d)-e$		(
$+b)*(c+d)-e$	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.

4. Расширение алгоритма построения польской записи

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

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

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;
- **запятая** не помещается в стек, и если в стеке операции, то все выбираются в строку;
- **отрывающая скобка** помещается в стек;
- **закрывающая скобка** выталкивает все операции до открывающей скобки, после чего обе скобки уничтожаются;
- **квадратная закрывающая** скобка выталкивает все до открывающей и генерирует @n (индекс n указывает число операндов, разделенных запятыми);
- по концу разбора исходной строки все операции, оставшиеся в стеке, выталкиваются в результирующую строку.

Исходная строка	Результирующая строка	Стек
$a * \left(b + \left[[c,d] + e,g\right]\right) - k/[e,f]$		
$* \left(b + \left[[c,d] + e,g\right]\right) - k/[e,f]$	a	
$\left(b + \left[[c,d] + e,g\right]\right) - k/[e,f]$	a	$*$
$b + \left[[c,d] + e,g\right]) - k/[e,f]$	a	$* ($
$+\left[[c,d] + e,g\right]) - k/[e,f]$	ab	$* ($
$\left[[c,d] + e,g\right]) - k/[e,f]$	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	$* (+[[$
$] + e,g]) - k/[e,f]$	$abcd$	$* (+[[$
$+e,g]) - k/[e,f]$	$abcd@_2$	$* (+[$
$e,g]) - k/[e,f]$	$abcd@_2$	$* (+[+$
$,g]) - k/[e,f]$	$abcd@_2e$	$* (+[+$
$g]) - k/[e,f]$	$abcd@_2e +$	$* (+[$
$]) - k/[e,f]$	$abcd@_2e + g$	$* (+[$
$) - k/[e,f]$	$abcd@_2e + g@_2$	$* (+$
$-k/[e,f]$	$abcd@_2e + g@_2 +$	$*$
$k/[e,f]$	$abcd@_2e + g@_2 +*$	$-$
$/[e,f]$	$abcd@_2e + g@_2 +* k$	$-$
$[e,f]$	$abcd@_2e + g@_2 +* k$	$-/$
$e,f]$	$abcd@_2e + g@_2 +* k$	$-/[$
$,f]$	$abcd@_2e + g@_2 +* k\mathrm{e}$	$-/[$
$f]$	$abcd@_2e + g@_2 +* k\mathrm{e}$	$-/[$
$]$	$abcd@_2e + g@_2 +* k\mathrm{e}\, f$	$-/[$
	$abcd@_2e + g@_2 +* k\mathrm{e}\, f@_2$	$-/$
	$abcd@_2e + g@_2 +* k\mathrm{e}\, f@_2/$	$-$
	$abcd@_2e + g@_2 +* k\mathrm{e}\, f@_2/-$	