МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Московский авиационный институт
(национальный исследовательский университет)

Институт №8
Компьютерные науки и прикладная математика

Кафедра 806 «Вычислительная математика и программирование»

КУРСОВОЙ ПРОЕКТ

по дисциплине «Системы программирования»

Студент: А. Ю. Кварацхелия
Преподаватель: В. В. Киндинова
Группа: М8О-201Б-21
Дата:
Оценка:

(Подпись студента)

(Подпись преподавателя)

Москва, 2023

Разработка СУ-схемы, переводящей выражение из инфиксной записи в постфиксную

Дадим определение некоторым терминам, чтобы внести ясность в постановку задачи.

СУ-схемой - схемой синтаксически управляемого перевода - называют пятёрку объектов

$$Tb = (V, \Sigma, \Delta, P, S),$$

где V - конечное множество нетерминальных символов, Σ - конечный входной алфвит, Δ - конечный выходной алфавит, P - конечное множество правил вида

$$A \vdash \alpha,\ \beta, \quad \alpha \in (V \cup \Sigma)*, \quad \beta \in (V \cup \Delta)*$$

и вхождение нетерминалов в цепочку β образуют перестановку вхождений нетерминалов в цепочку α.

Инфиксной записью выражения называется привичный нам вид выражения

$$A + B,$$

где A и B - некоторые операнды, $+$ - некоторый оператор, в выражении конечное количество операторов и операндов.

Постфиксной же записью называется такая запись, в которой оператор следует за операндами, для которых он определён. Например

$$A + B \implies AB + \quad \text{или} \quad A + B * C \implies ABC * +.$$

Существует два метода перевода выражения из инфиксной записи в постфиксную: «скобочный» и использующий стэк (магазин). Если бы требовалось реализовать МП-преобразователь, переводящей выражение из инфиксной записи в постфиксную, мы бы обратились ко второму методу. В нашем случае, СУ-схемы не располагают магазинной памятью, поэтому ничего не остаётся кроме как использовать первый метод.

Опишем «скобочный метод» перевода:

1. Расставить скобки так, чтобы при равенстве приоритетов операторов порядок их исполнения не изменился.
2. Для каждой тройки (левая скобка, оператор, правая скобка) заменить правую скобку оператором, оператор, ранее стоящий между двух скобок, удалить, как и левую скобку.
3. Продолжать П.2, пока в выражении присутствуют скобки. Напомним, что постфиксная запись не требует скобок, ввиду однозначности определения последовательности исполнения операторов.

Пример работы алгоритма:

$$A + B * C \implies A + (B * C) \implies (A + (B * C)) \implies (A + BC*) \implies ABC * +.$$

Преобразуем алгоритм для большей простоты реализации СУ-схемы - опустим шаг с расстановкой скобок, заменяя часть выражения сразу постфиксной записью, опираясь на известные приоритеты операторов.

$$A + B * C \implies A + BC* \implies ABC * +$$

По условию существует девять групп операторов, распределённых по приоритетам:

1. $\uparrow$
2. $*$, $\div$
3. $+$, $-$
4. $\leq$, $<$, $=$, $>$, $\geq$
5. $\neg$
6. $\wedge$
7. $\vee$
8. $\rightarrow$
9. $\equiv$.

Преобразуем нетривиальный пример:

$$A \vee B + C \leq D \neg E \implies A \vee BC+ \leq D \neg E \implies A \vee BC + D \leq \neg E \implies$$
$$\implies A \vee BC + D \leq E \neg \implies ABC + D \leq E \neg \vee$$

Таким образом, работу СУ-схемы можно описать следующим образом: Пусть некоторая часть выражения уже обработана, тогда по определению постфиксной записи существует оператор, имеющий меньший приоритет, чем оператор, обрабатываемый в данный момент, тогда алгоритм сводится к тому, чтобы завести оператор, обрабатываемый в данный момент, за оператор, имеющий приоритет наибольший, но меньший, чем приоритет обрабатываемого оператора, иначе оператор станет крайним символом цепочки.

Рассмотрим тривиальную ситуацию, в которой нужно привести к обратной польской записи выражение $a + a * a$. В таком случае правила СУ-схемы будут выглядеть следующим образом:

№	Символ	Входная цепочка	Выходная цепочка
1	E	$E + T$	$ET+$
2	E	T	T
3	T	$T * F$	$TF*$
4	T	F	F
5	F	a	a
6	F	E	E

Приведём вывод данной СУ-схемы:

$$(E,\, E) \Longrightarrow^{1} (E + T,\, ET+) \Longrightarrow^{2} (T + T,\, TT+) \Longrightarrow^{4}$$
$$\Longrightarrow^{4} (F + T,\, FT+) \Longrightarrow^{5} (a + T,\, aT+) \Longrightarrow^{4} (a + F,\, aF+) \Longrightarrow^{6}$$
$$\Longrightarrow^{6} (a + E,\, aE+) \Longrightarrow^{2} (a + T,\, aT+) \Longrightarrow^{3} (a + T * F,\, aTF * +) \Longrightarrow^{5}$$
$$\Longrightarrow^{5} (a + T * a,\, aTa * +) \Longrightarrow^{4} (a + F * a,\, aFa * +) \Longrightarrow^{6} (a + a * a,\, aaa * +).$$

Таким образом, «по индукции» опишем СУ-схему, переводящую выражение из инфиксной записи в ПОЛИЗ:

№	Символ	Входная цепочка	Выходная цепочка
1	A	$A \equiv B$	$AB \equiv$
2	A	B	B
3	B	$B \to C$	$BC \to$
4	B	C	C
5	C	$C \vee D$	$CD\vee$
6	C	D	D
7	D	$D \wedge E$	$DE\wedge$
8	D	E	E
9	E	$E\neg F$	$EF\neg$
10	E	F	F
11	F	$F = G$	$FG =$
12	F	G	G
13	G	$G + O$	$GO+$
14	G	O	O
15	O	$O * P$	$OP*$
16	O	P	P
17	P	$P \uparrow Q$	$PQ \uparrow$
18	P	Q	Q
19	Q	a	a
20	Q	A	A

Итак, решение почти готово, осталось добавить оставшиеся символы из учтёных групп приоритетности и исправить правило №9, учитывая одноместность оператора $\neg$.

№	Символ	Входная цепочка	Выходная цепочка
1	A	$A \equiv B$	$AB \equiv$
2	A	B	B
3	B	$B \to C$	$BC \to$
4	B	C	C
5	C	$C \vee D$	$CD\vee$
6	C	D	D
7	D	$D \wedge E$	$DE\wedge$
8	D	E	E
9	E	$\neg EF$	$E\neg F$
10	E	F	F
11	F	$F \leq F_1$	$FF_1 \leq$
12	F	F_1	F_1
13	F_1	$F_1 < F_2$	$F_1F_2 <$
14	F_1	F_2	F_2
15	F_2	$F_2 = F_3$	$F_2F_3 =$
16	F_2	F_3	F_3
17	F_3	$F_3 > F_4$	$F_3F_4 >$
18	F_3	F_4	F_4
19	F_4	$F_4 \geq G$	$F_4G \geq$
20	F_4	G	G
21	G	$G + G_1$	GG_1+
22	G	G_1	G_1
23	G_1	$G_1 - O$	G_1O-
24	G_1	O	O
25	O	$O * O_1$	OO_1*
26	O	O_1	O_1
27	O_1	$O_1 \div P$	$O_1P\div$
28	O_1	P	P
29	P	$P \uparrow Q$	$PQ\uparrow$
30	P	Q	Q
31	Q	a	a
32	Q	A	A

Чтобы проверить корректность описанной СУ-схемы, обработаем следующую цепочку:

$$a \to a * a \uparrow \neg a < a = a \equiv a \vee a.$$

$$(A,\, A) \implies (B,\, B) \implies (B \to C,\, BC \to) \implies \cdots \implies (a \to C,\, aC \to) \implies$$
$$\cdots \implies (a \to Q,\, aQ \to) \implies (a \to A,\, aA \to) \implies \cdots \implies (a \to O * O_1,\, aO * O_1 \to) \implies$$
$$\cdots \implies (a \to a * O_1,\, aa * O_1 \to) \cdots \implies (a \to a * P \uparrow Q,\, aa * PQ \uparrow \to) \implies \cdots \implies$$
$$\implies (a \to a * P \uparrow a,\, aa * Pa \uparrow \to) \implies (a \to a * A \uparrow a,\, aa * Aa \uparrow \to)$$
$$\implies (a \to a * \neg EF \uparrow a,\, aa * F \neg Ea \uparrow \to) \implies \cdots$$

```
var sdt = new mySDTSchemata(new List<Symbol>() { "E","T", "F" }
        new List<Symbol>() { "a","+", "*" },
        new List<Symbol>() { "a","+", "*" },
        "E");

sdt.addRule(new Symbol("E"),
        new List<Symbol>() { "E","+","T" },
        new List<Symbol>() { "E","T","+" });
sdt.addRule(new Symbol("E"),
        new List<Symbol>() { "T" },
        new List<Symbol>() { "T" });
sdt.addRule(new Symbol("T"),
        new List<Symbol>() { "T", "*", "F" },
        new List<Symbol>() { "T", "F", "*" });
sdt.addRule(new Symbol("T"),
        new List<Symbol>() { "F" },
        new List<Symbol>() { "F" });
sdt.addRule(new Symbol("F"),
        new List<Symbol>() { "a" },
        new List<Symbol>() { "a" });
sdt.addRule(new Symbol("F"),
        new List<Symbol>() { "E" },
        new List<Symbol>() { "E" });
```