Классическая теория компиляторов

Лекция 5

ВНУТРЕННИЕ ФОРМЫ ПРЕДСТАВЛЕНИЯ ПРОГРАММЫ

ПОЛЬСКАЯ ФОРМА

1. Если R является операндом (идентификатором или константой), то его значение заносится в стек аргументов S и осуществляется считывание следующего символа (правило "<операнд>::=идентификатор").
2. Если R – оператор, то из стека аргументов S извлекается необходимое количество аргументов; выполняется операция и результат заносится обратно в стек S.

```
R := getchar
while R != # do
case тип(R) of
        'операнд':          push(R)
        'бин.оператор':     pop(B), pop(A), Tmp := R(A,B), push(Tmp)
        'унарн.оператор':   pop(A), Tmp := R(A), push(Tmp)
endcase
R := getchar
endwhile
```

Замечание. *Существуют операторы, которые не заносят в стек результаты своей работы.*

Операторы в ПФ

Безусловные переходы

Безусловный переход на метку (аналог GOTO L)

L $BRL (Branch to label)

L – имя в таблице символов. Значение его – адрес перехода. Основная проблема при реализации этого оператора – определение адреса перехода. Возможным решением является введение фиктивного оператора, соответствующего метке, на которую будет осуществлен переход с последующим поиском его в массиве польской формы.

Безусловный переход на символ

C $BR

Условные переходы

<операнд1> <операнд2> $BRxZ|$BRM|$BRP|$BRMZ|$BRPZ|

<операнд1> – значение арифметического выражения,

<операнд2> – номер или место символа в польской цепочке (адрес).

$BRx – код оператора. При этом можно ввести самые разнообразные условия перехода. Например:

$BRZ – переход по значению 0,

$BRM – переход по значению <0,

$BRP – переход по значению >0,

$BRMZ – переход по значению <=0,

$BRPZ – переход по значению >=0,

и т.п.

При выполнении условия перехода в качестве следующего символа берется тот символ, адрес которого определяется вторым операндом. В противном случае работа продолжается как обычно.

Условные конструкции

IF <выражение> THEN <инстр1> ELSE <инстр2>

<выражение> <c1> $BRZ <инстр1> <c2> $BR <инстр2>

Предполагается, что для любого выражения 0 означает "ложь", а неравенство нулю – "истина". Здесь

<c1> – номер символа, с которого начинается <инстр2>,

<c2> – номер символа, следующего за <инстр2>,

$BR – оператор безусловного перехода на символ,

$BRZ – оператор перехода на символ c1, если значение <выражение> равно нулю).

Массивы и функции

Описание массивов

ARRAY A[L1..U1,...,Lk..Uk]

L1 U1 ... Lk Uk A $ADEC,

где $ADEC – оператор объявления массива. Оператор $ADEC имеет *переменное количество аргументов*, зависящее от числа индексов. Операнд A – адрес в таблице символов. При вычислении $ADEC из этого элемента таблицы извлекается информация о размерности массива A и, следовательно, о количестве операндов $ADEC. Отсюда понятно, почему операнд A должен располагаться непосредственно перед оператором $ADEC.

Обращение к элементу массива

A[<выр>,..,<выр>]

<выр> .. <выр> A $SUBS

Оператор $SUBS, используя элемент A таблицы символов и индексные выражения, вычисляет адрес элемента массива. Затем операнды исключаются из стека и на их место заносится новый операнд, специфицирующий тип элемента массива и его адрес.

Вызов подпрограмы

Вызов функции вида f(x1,x2):

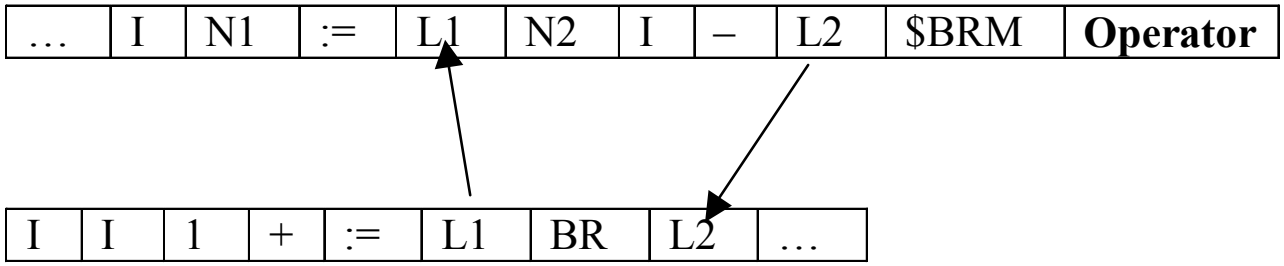
x1 x2 f $CALL,

где x1 и x1 – аргументы, f – имя функции, $CALL – команда передачи управления по адресу, определяемому именем функции. При этом предполагается, что мы заранее занесли в таблицу имен информацию о том, сколько и каких аргументов у функции f, а также ее адрес – индекс начала кода подпрограммы в массиве польской формы.

Циклы

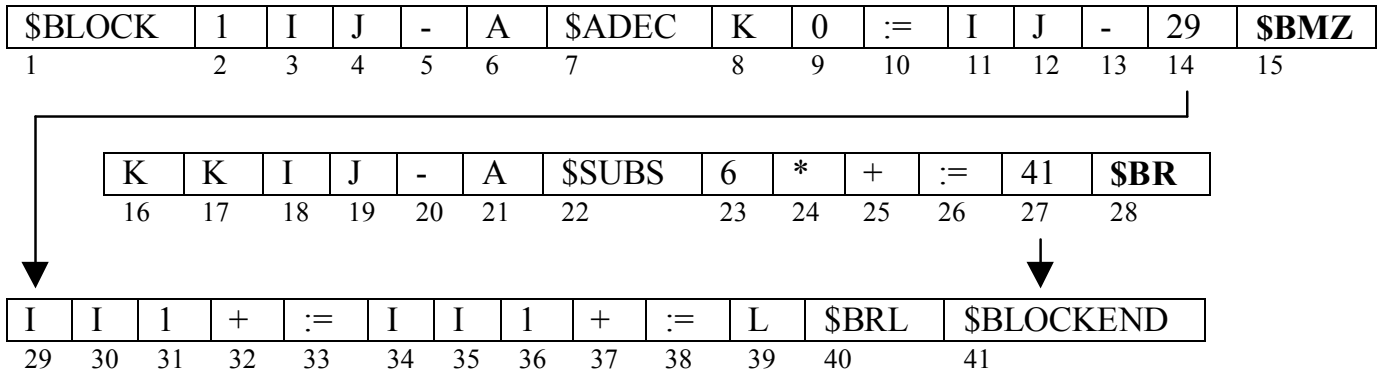
FOR I=N1 TO N2 DO Operator

```
I := N1;
L1: IF I>N2 THEN GOTO L2;
    Operator;
    I:=I+1;
    GOTO L1;
L2:
```



Пример

```
BEGIN
        INTEGER K;
        ARRAY A[1..I-J];
        K := 0;
        L:   IF I>J THEN
                K := K+A[I-J]*6
        ELSE
        BEGIN
                I := I+1;
                I := I+1;
                GOTO L;
        END
END.
```



ТЕТРАДЫ

```
BEGIN
        INTEGER K;
        ARRAY A[1..I-J];
        K := 0;
        L:   IF I>J THEN
                K := K+A[I-J]*6
        ELSE
        BEGIN
                I := I+1;
                I := I+1;
                GOTO L;
        END
END.
```

(1) $BLOCK

(2) - I, J, T1

(3) $BOUNDS 1, T1

(4) $ADEC A

(5) := 0,, K

(6) - I, J, T2

(7) $BMZ 13, T2

(8) - I, J, T3

(9) * A[T3], 6, T4 (???)

(10) + K, T4, T5

(11) := T5,, K

(12) $BR 18

(13) + I, 1, T6

(14) := T6,, I

(15) + I, 1, T7

(16) := T7,, I

(17) $BRL L

(18) $BLOCKEND

Тетрады

Оператор	Операнды	Описание
$BR	i	переход на i-ю тетраду
$BZ (BP,BM)	i, P	переход на i-ю тетраду, если P=0 (P>0,P<0)
$BG (BL,BE)	i, P1, P2	переход на i-ю тетраду, если P1>P2 (P1<P2,P1=P2)
$BRL	P	переход на тетраду, номер которой задан в P-м элементе таблицы символов
$BOUNDS	P1, P2	P1 и P2 описывают граничную пару массива
$ADEC	P	Массив описан в P. Если размерность массива равна n, то этой тетраде должны предшествовать n операторов $BOUNDS, задающих n граничных пар

```
$AINDX I
$AGET A, R
```

"A[1,2]+B[J]"

```
    $AINDX 1
    $AINDX 2
    $AGET A, T1
    $AINDX J
    $AGET B, T2
    + T1, T2, T3
```

Префиксная форма записи

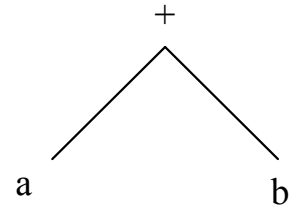
Определение префиксной формы записи (ПрФ):

- Если инфиксное выражение E представляет собой один операнд **a**, то ПрФ выражения E - это **a**.
- Если инфиксное выражение E1@E2, где @ - знак операции, а E1 и E2 - инфиксные выражения для операндов, то ПрФ этого выражения - это @E1'E2', где E1', E2' - ПрФ выражений E1 и E2.
- Если (E) есть инфиксное выражение, то ПрФ этого выражения есть E.

Примеры

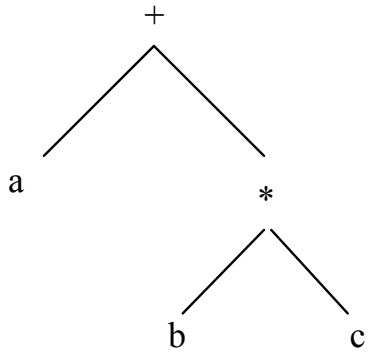
Пример 1. $E_{инф}$ = a + b

- По определению ПрФ выражения E1@E2 - это @E1'E2', где E1',E2' - ПрФ выражений E1 и E2. ПрФ для E1 и E2 – это
  E1' = a, E2' = b
- Окончательно:
  +ab



Пример 2. $E_{инф}$ = a + b*c

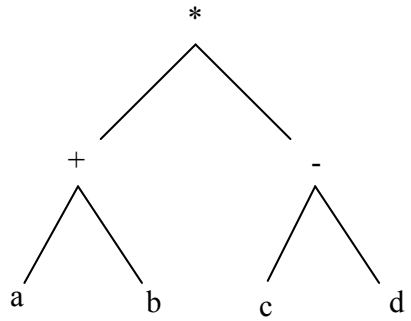
- По определению ПрФ выражения E1@E2 - это @E1'E2', где E1',E2' - ПрФ выражений E1 и E2. ПрФ для E1 и E2 – это
  E1' = a, E2' = *bc
- Окончательно:
  +a*bc



Примеры

Пример 3. $E_{инф}$ = (a + b) * (c - d)

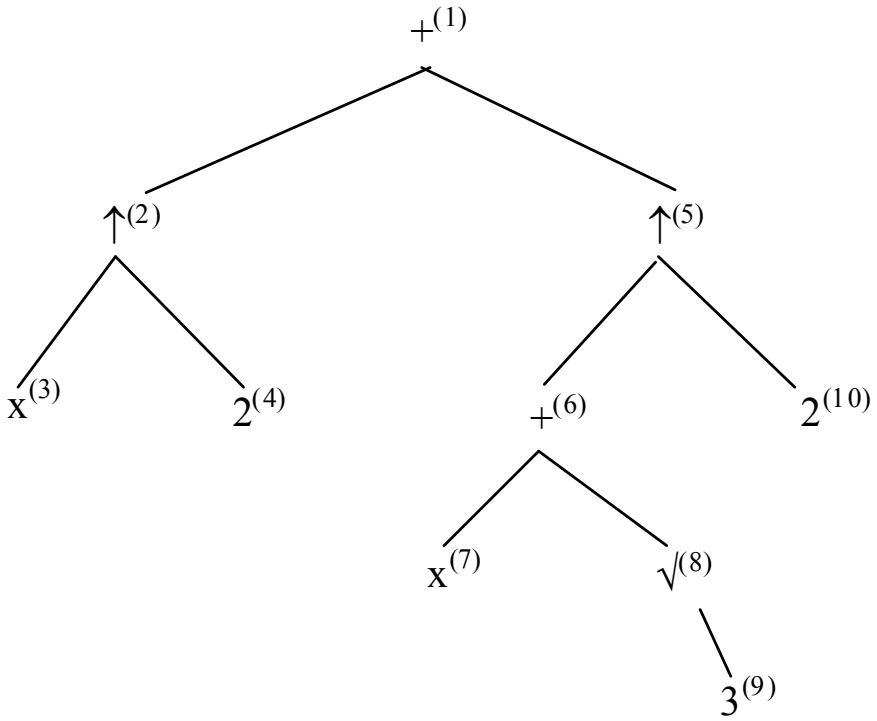
- Представляем E в виде E1*E2, где E1 = (a + b), E2 = (c - d).
- По определению ПрФ выражения E1@E2 - это @E1'E2', где E1',E2' - ПрФ выражений E1 и E2. ПрФ для E1 и E2 – это E1' = +ab, E2' = -cd
- Окончательно: *+ab-cd



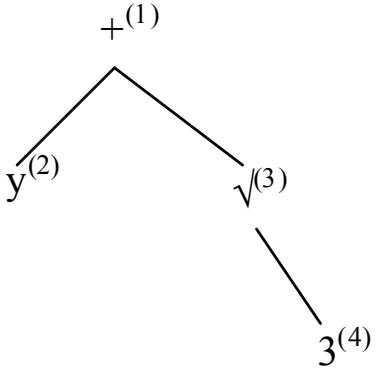
Пример 4. $E_{инф} = \log\left(y + \sqrt{y^2 - b / \sin x}\right)$

$E_{преф}$=log + y √ - ↑ y 2 / b sin x

Примеры



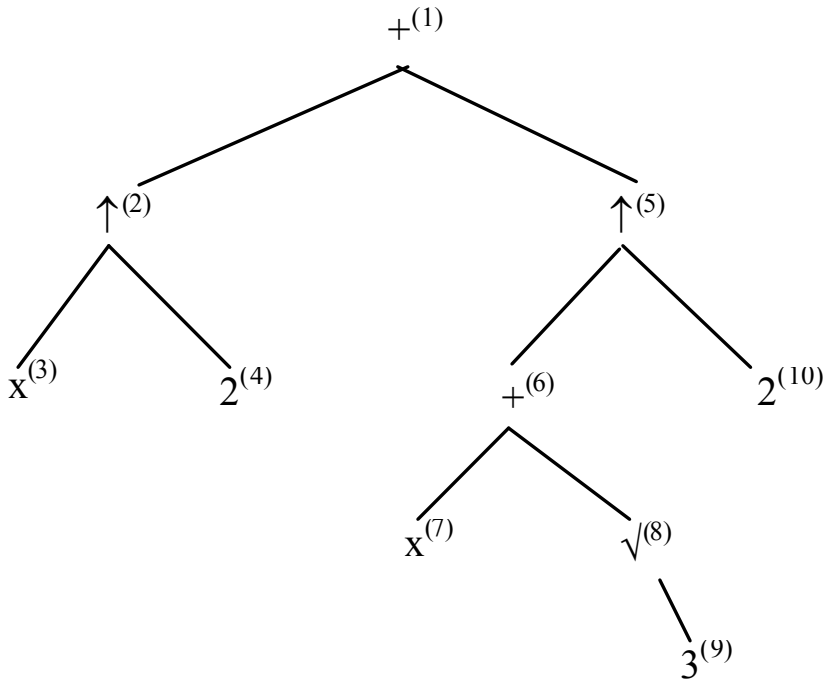
$E_{инф} = x^2+(x+\sqrt{3})^2$



$E_{инф} = y+\sqrt{3}$

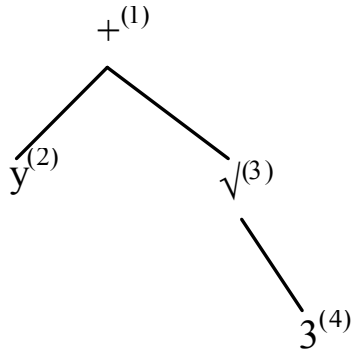
Примеры

$E_{инф} = x^2+(x+\sqrt{3})^2$



1	2	3	4	5	6	7	8	9	10
+	↑	x	2	↑	+	x	√	3	2

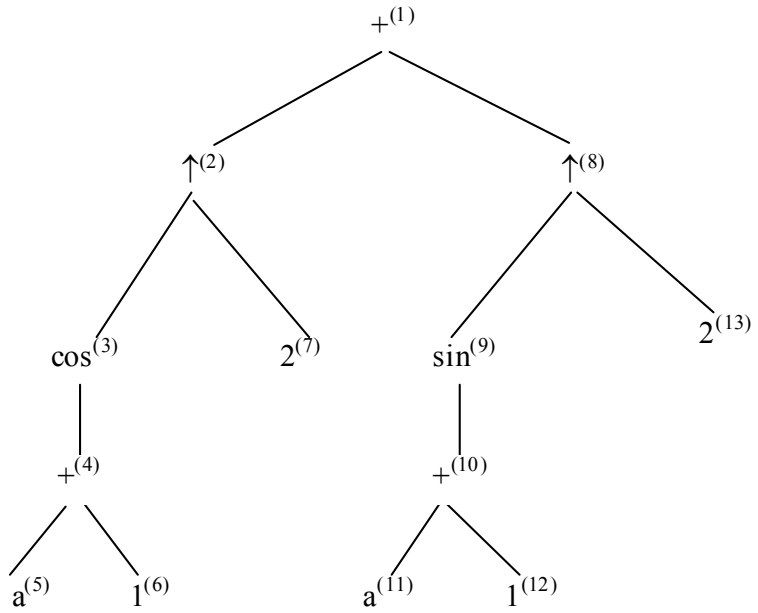
$E_{инф} = y+\sqrt{3}$



1	2	3	4
+	y	√	3

Примеры

Пример 6. $E_{инф} = \cos^2(a+1)+\sin^2(a+1)$



1	2	3	4	5	6	7	8	9	10	11	12	13
+	↑	cos	+	a	1	2	↑	sin	+	a	1	2

Основная теорема префиксной формы записи

Последовательность символов S является правильно построенной префиксной формой, если и только если:

1. ранг(S)=-1
2. ранг(последовательности слева от S)≥0

Причем считается, что:

ранг(оператора) = вес(оператора)-1,

ранг(пустой последовательности) = 0,

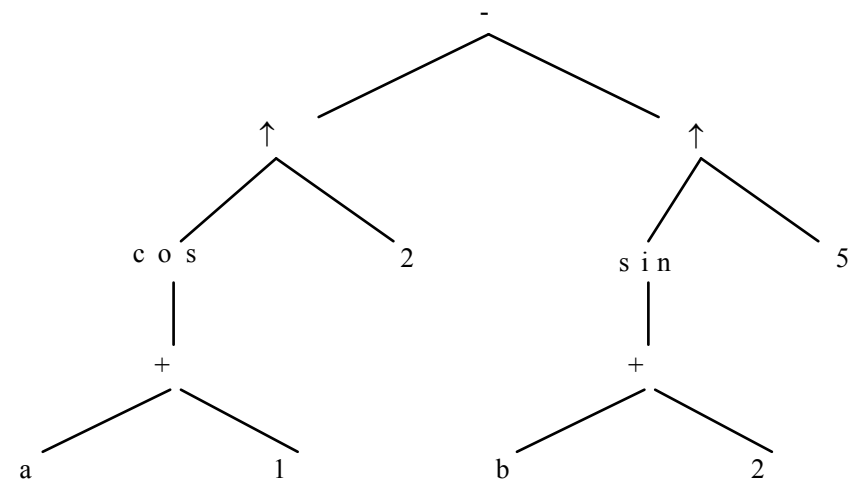
ранг(n-го символа) = n-1,

ранг(переменной) = ранг(константы) = -1,

ранг(S1 соединенной с S2) = ранг(S1)+ранг(S2).

Пример

Пример 7. $E_{инф} = \cos^2(a+1)-\sin^5(b+2)$



№	1	2	3	4	5	6	7	8	9	10	11	12	13
Симв	-	↑	cos	+	a	1	2	↑	sin	+	b	2	5
Ранг	1	1	0	1	-1	-1	-1	1	0	1	-1	-1	-1

Суммарный ранг равен -1. Для нахождения **границы терма**, образованного, например, оператором cos, начнем суммировать ранги, начиная с этого оператора. Мы остановимся в позиции № 6. Таким образом, граница терма - с 3 по 6 позиции (cos + a 1).

Вычисление префиксной формы записи

1. Прочитать очередной символ входной цепочки R.
2. Если входной символ - оператор, то занести его в стек
3. Если входной символ - операнд, то:

3.1. Отыскать в стеке последний оператор ОР.

3.2. Если этому оператору хватает операндов (тех, что находятся в стеке после ОР плюс текущий операнд R), то извлечь из стека оператор ОР, соответствующие операнды вместе с R и выполнить операцию. Результат поместить в стек. Вернуться к п.3.1.

иначе поместить символ R в стек.

4. Вернуться к П1.

Необходим стек, умеющий хранить как вычисляемые значения, так и операторы.

№	Вход	Текущий символ R	Стек
1	*+ab-cd		#
2	+ab-cd	*	#*
3	ab-cd	+	#*,+
4	b-cd	a	#*,+,a
5	-cd	b	#*, (a+b)
6	cd	-	#*, (a+b), -
7	d	c	#*, (a+b),-, c
8		d	#*(a+b), (c-d)
9			#(a+b)*(c-d)