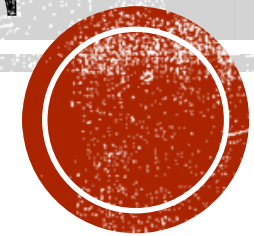
Языки программирования и методы трансляции



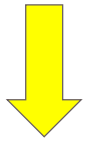
Лекция 4

2 марта 2018 г.

ЛЕКСИЧЕСКИЙ АНАЛИЗ И ТАБЛИЦА СИМВОЛОВ

поток символов

S	U	M		=		S	U	M		+		U	N	I	T		*		1	.	2	E	-	1	2	;



поток токенов

id (0)	=	id (0)	+	id (1)	*	num (1.2e-12)	;

Таблица символов

Имя	Код
sum	0
unit	1

Полная таблица символов транслятора MiniC

Имя	Код	Класс	Тип	Длина	По-умолч.	Область
a	0	var	int	None	0	-1
b	1	var	int	None	None	-1
fact	2	func	int	1	None	-1

СИНТАКСИЧЕСКИЙ АНАЛИЗ И АТОМЫ

поток лексем

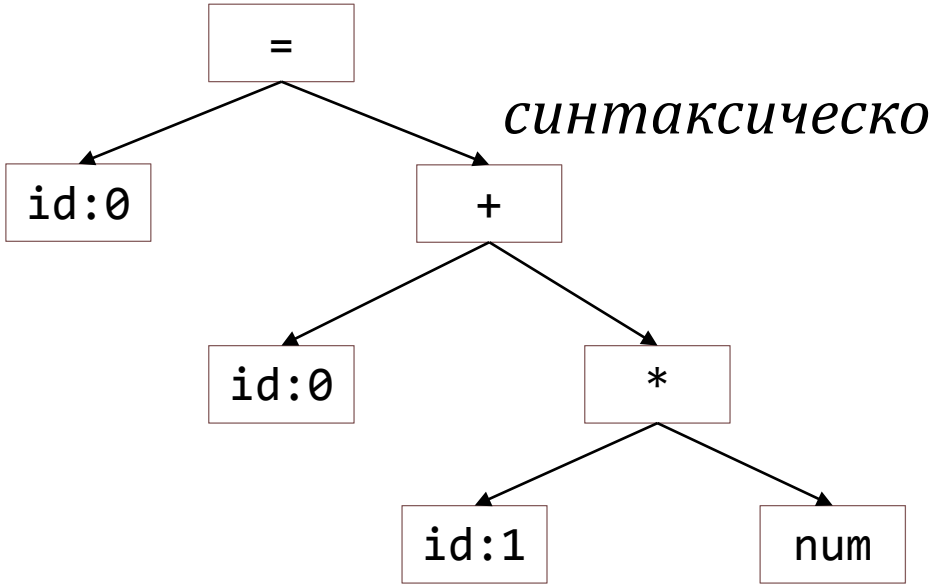
id (0)	=	id (0)	+	id (1)	*	num (1.2e-12)	;

Промежуточное представление

список атомов

```
(MULT,id:1,num,TEMP1)
(ADD,id:0,TEMP1,TEMP2)
(MOVE,TEMP2,id:0)
```

синтаксическое дерево



ТРАНСЛЯЦИЯ

Опр. 1: *Трехадресный код (атом)* – атомарная инструкция, использующая не более трех адресов. Например: $\mathbf{op}_1 = \mathbf{op}_2 * \mathbf{op}_3$

Опр. 2: *Адрес в атоме* – это либо константа, либо имя пользовательских или временных переменных, либо имя метки.

Трехадресный код не допускает вложенных инструкций.

Опр. 3: *Четверки* – способ записи трехадресного кода, при котором каждый атом представлен структурой с четырьмя полями:

```
(операция, операнд1, операнд2, результат)
```

ТРАНСЛЯЦИЯ

Опр. 4: *alloc()* – создает в таблице символов запись для временной переменной и возвращает ее номер

Опр. 5: *newlabel()* – инструкция, возвращающая номер новой метки

Опр. 6: *Метка* – псевдо-атом, вставляемый в поток атомов и используемый для обозначения нужного места для условных и безусловных переходов.

АТОМЫ

(ADD, x, y, z)	z = x+y
(SUB, x, y, z)	z = x-y
(MUL, x, y, z)	z = x*y
(DIV, x, y, z)	z = x/y
(NEG, x,, z)	z = -x
(AND, x, y, z)	z = x && y
(OR, x, y, z)	z = x || y
(NOT, x,, z)	z = !x
(MOV, x,, z)	z = x

(LBL,,, L1)	установить метку L1 на следующую по порядку инструкцию (кроме LBL)
(EQ, x, y, L1)	если x == y перейти на L1
(NE, x, y, L1)	если x != y перейти на L1
(GT, x, y, L1)	если x > y перейти на L1
(LT, x, y, L1)	если x < y перейти на L1
(GE, x, y, L1)	если x >= y перейти на L1
(LE, x, y, L1)	если x <= y перейти на L1
(JMP,,, L1)	перейти на инструкцию L1
(IN,,, x)	ввод с клавиатуры значения в x
(OUT,,, x)	вывод на экран значения x

АТОМЫ

Пример 1

Выражение `Res = A + B * C + D` будет транслировано в

```
(MUL,B,C,TEMP1)
(ADD,A,TEMP1,TEMP2)
(ADD,TEMP2,D,TEMP3)
(MOV,TEMP3,,Res)
```

или более точно:

```
(MUL,2,3,5)
(ADD,1,5,6)
(ADD,6,4,7)
(MOV,7,,0)
```

Таблица символов

Имя	Код
Res	0
A	1
B	2
C	3
D	4
TEMP1	5
TEMP2	6
TEMP3	7

РЕАЛИЗАЦИЯ

$Expr_p \rightarrow Term_q \ Elist_{q,p}$

$Elist_{p,q} \rightarrow + \ Term_r \ \{ADD\}_{p,r,s} \ Elist_{s,q}$ $\quad s \leftarrow Alloc()$

$Elist_{p,q} \rightarrow \varepsilon$ $\quad q \leftarrow p$

$Term_p \rightarrow Factor_q \ Tlist_{q,p}$

$Tlist_{p,q} \rightarrow * \ Factor_r \ \{MULT\}_{p,r,s} \ Tlist_{s,q}$ $\quad s \leftarrow Alloc()$

$Tlist_{p,q} \rightarrow \varepsilon$ $\quad q \leftarrow p$

$Factor_p \rightarrow (\ Expr_p \ )$

$Factor_p \rightarrow ident_p$

РЕАЛИЗАЦИЯ

$Expr_p \rightarrow Term_q\ Elist_{q,p}$

```
void Expr (int & p)
      {  int q;
         if (inp=='(' || inp==Ident)
            {  Term (q);                          // apply rule 1
               Elist (q,p);
            }                                     // end rule 1
         else Reject();
      }
```

РЕАЛИЗАЦИЯ

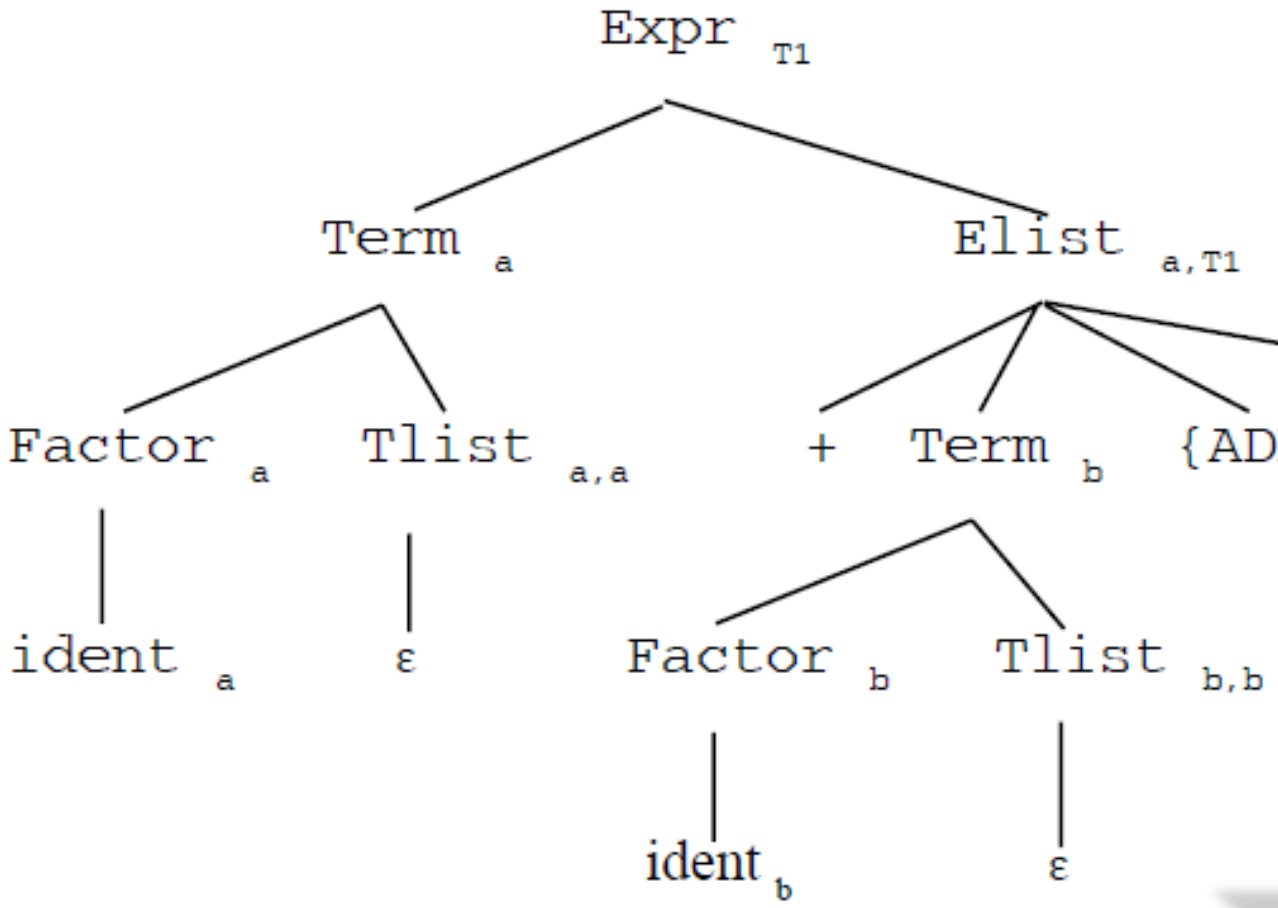
$Elist_{p,q} \rightarrow + \; Term_{r} \; \{ADD\}_{p,r,s} \; Elist_{s,q}$ $\quad s \leftarrow Alloc()$

$Elist_{p,q} \rightarrow \varepsilon$ $\quad q \leftarrow p$

```
void Elist (int p, int & q)
      {  int r,s;
         if (inp=='+')
            {  token.getToken();
               inp = token.getClass();    // apply rule 2
               Term (r);
               s = Alloc();
               Out (ADD, p, r, s);
               Elist (s,q);
            }                                // end rule 2
         else if (inp=='↩' || inp==')') q = p;  // rule 3
         else Reject();
      }
```

АТОМЫ

Пример 2. Аннотированное дерево вывода для **a+b**




```
Expr_p → Term_q Elist_q,p
Elist_p,q → + Term_r {ADD}_p,r,s Elist_s,q      s ← Alloc()
Elist_p,q → ε                                   q ← p
Term_p → Factor_q Tlist_q,p
Tlist_p,q → * Factor_r {MULT}_p,r,s Tlist_s,q   s ← Alloc()
Tlist_p,q → ε                                   q ← p
Factor_p → ( Expr_p )
Factor_p → ident_p
```

АТОМЫ

Пример 3. Построить аннотированное дерево вывода для (a+b)*c

```
Expr_p → Term_q Elist_q,p
Elist_p,q → + Term_r {ADD}_p,r,s Elist_s,q      s ← Alloc()
Elist_p,q → ε                                   q ← p
Term_p → Factor_q Tlist_q,p
Tlist_p,q → * Factor_r {MULT}_p,r,s Tlist_s,q   s ← Alloc()
Tlist_p,q → ε                                   q ← p
Factor_p → ( Expr_p )
Factor_p → ident_p
```

ВЫРАЖЕНИЯ

Как транслировать в список атомов сравнение? Например **x > y**

```
(MOV,1,,T1) // предположим, что результат – истина
(GT,x,y,L1) // x > y? Если да, то пропустить MOV
(MOV,0,,T1) // иначе результат ложный
(LBL,,,L1)
```

T1 – временная переменная для хранения результата сравнения
L1- метка

ВЫРАЖЕНИЯ

Как написать транслирующую грамматику для сравнения?

Общее правило

Пусть $a_1,a_2,\ldots,a_n$ – операции первого приоритета, $b_1,b_2,\ldots,b_m$ – операции второго приоритета и т.д. Тогда правило построения грамматики следующее:

$X \rightarrow X\ a_1\ Y \mid X\ a_2\ Y \mid \ldots \mid X\ a_n\ Y$

$Y \rightarrow Y\ b_1\ Z \mid Y\ b_2\ Z \mid \ldots \mid Y\ b_m\ Z$

…

X – нетерминал, отвечающий за «вывод» операций первого уровня приоритета, Y – второго и т.д. Далее избавляемся от левой рекурсии.

ВЫРАЖЕНИЯ

Как написать транслирующую грамматику для сравнения?
Пример для **x > y**

$D \rightarrow E > E$ 

$D \rightarrow E\ D'$

$D' \rightarrow > E \mid \varepsilon$

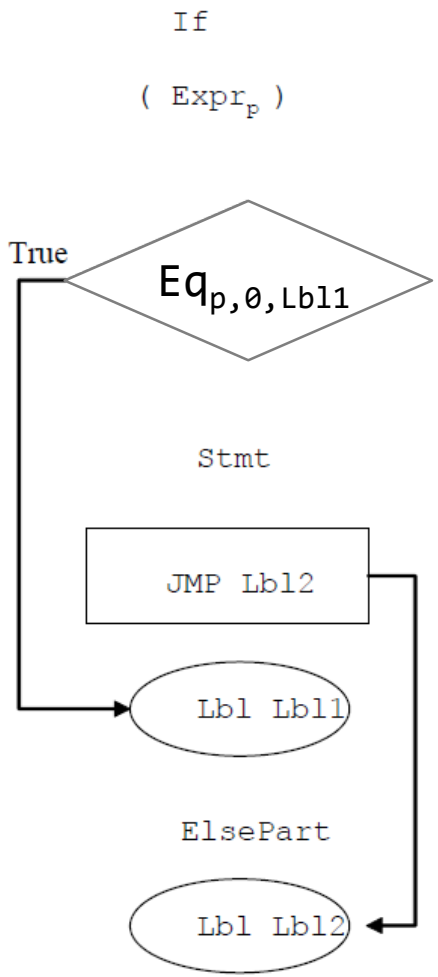
$D_p \rightarrow E_q\ D'_{rs}$ r = q; p = s

$D'_{pq} \rightarrow < E_r\ \{MOV\}_{1,,T1}\ \{GT\}_{p,r,Lbl1}\ \{MOV\}_{0,,T1}\ \{LBL\}_{Lbl1}$

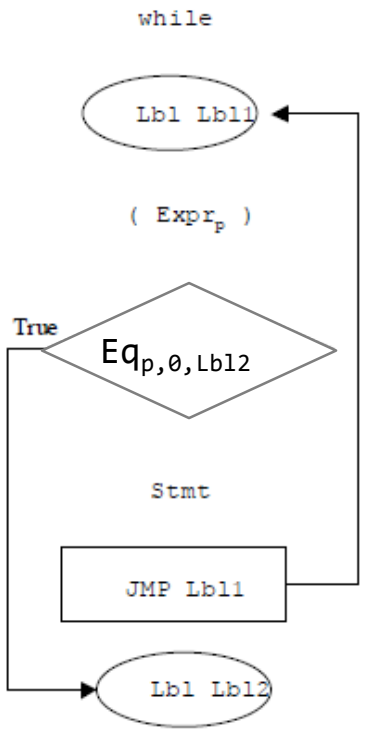
T1 = alloc() Lbl1 = newlabel() q = T1

ЦИКЛЫ И ВЕТВЛЕНИЯ

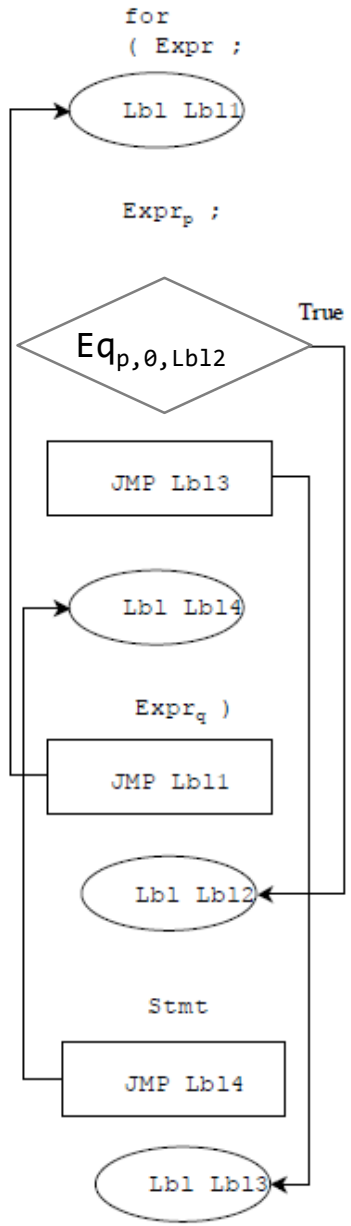
if stmt



while stmt



for stmt



УПРАЖНЕНИЯ

УПРАЖНЕНИЯ

Упражнение 1
Напишите список атомов, который будет сгенерирован для следующих инструкций `res = (b+c)*(2*c-a)`

```
(ADD, x, y, z)
(SUB, x, y, z)
(MUL, x, y, z)
(DIV, x, y, z)
(NEG, x,, z)
(AND, x, y, z)
(OR, x, y, z)
(NOT, x,, z)
(MOV, x,, z)
(LBL,,, L1)
(EQ, x, y, L1)
(NE, x, y, L1)
(GT, x, y, L1)
(LT, x, y, L1)
(GE, x, y, L1)
(LE, x, y, L1)
(JMP,,, L1)
(IN,,, x)
(OUT,,, x)
```

УПРАЖНЕНИЯ

Упражнение 2
Напишите список атомов, который будет сгенерирован для следующих инструкций `if (a==b) while (x<y) Stmt`

```
(ADD, x, y, z)
(SUB, x, y, z)
(MUL, x, y, z)
(DIV, x, y, z)
(NEG, x,, z)
(AND, x, y, z)
(OR, x, y, z)
(NOT, x,, z)
(MOV, x,, z)
(LBL,,, L1)
(EQ, x, y, L1)
(NE, x, y, L1)
(GT, x, y, L1)
(LT, x, y, L1)
(GE, x, y, L1)
(LE, x, y, L1)
(JMP,,, L1)
(IN,,, x)
(OUT,,, x)
```

УПРАЖНЕНИЯ

Упражнение 3
Напишите список атомов, который будет сгенерирован для следующих инструкций `for (j = 1; j<=i; j = j+1) Stmt`

```
(ADD, x, y, z)
(SUB, x, y, z)
(MUL, x, y, z)
(DIV, x, y, z)
(NEG, x,, z)
(AND, x, y, z)
(OR, x, y, z)
(NOT, x,, z)
(MOV, x,, z)
(LBL,,, L1)
(EQ, x, y, L1)
(NE, x, y, L1)
(GT, x, y, L1)
(LT, x, y, L1)
(GE, x, y, L1)
(LE, x, y, L1)
(JMP,,, L1)
(IN,,, x)
(OUT,,, x)
```

УПРАЖНЕНИЯ

Упражнение **4**

Напишите список атомов, который будет сгенерирован для следующих инструкций

```
if (a==b) for (i=1; i<=20; i=i+1) Stmt1
else while (i>0) Stmt2
```

```
(ADD, x, y, z)
(SUB, x, y, z)
(MUL, x, y, z)
(DIV, x, y, z)
(NEG, x,, z)
(AND, x, y, z)
(OR, x, y, z)
(NOT, x,, z)
(MOV, x,, z)
(LBL,,, L1)
(EQ, x, y, L1)
(NE, x, y, L1)
(GT, x, y, L1)
(LT, x, y, L1)
(GE, x, y, L1)
(LE, x, y, L1)
(JMP,,, L1)
(IN,,, x)
(OUT,,, x)
```

УПРАЖНЕНИЯ

Упражнение 5
Напишите список атомов, который будет сгенерирован для следующих инструкций `if (a==b) if (b>0) Stmt1 else while (i>0) Stmt2`

```
(ADD, x, y, z)
(SUB, x, y, z)
(MUL, x, y, z)
(DIV, x, y, z)
(NEG, x,, z)
(AND, x, y, z)
(OR, x, y, z)
(NOT, x,, z)
(MOV, x,, z)
(LBL,,, L1)
(EQ, x, y, L1)
(NE, x, y, L1)
(GT, x, y, L1)
(LT, x, y, L1)
(GE, x, y, L1)
(LE, x, y, L1)
(JMP,,, L1)
(IN,,, x)
(OUT,,, x)
```