§ 7. Атрибутная грамматика проверки типов данных в программе

Цель: для фрагмента программы на некотором языке программирования высокого уровня, синтаксис которого похож на Pascal, выполнить статическую проверку типов. Значение атрибута .type указывает на результат проверки. Если значение атрибута равно type_error, то фрагмент программы содержит ошибку типов. Если значение отлично от type_error, то программа успешно прошла проверку.

Назовем APG атрибутную грамматику проверки типов.

В начале программы будут стоять команды объявления типа идентификаторов.

1	$P \rightarrow D; S$	
2	$D \rightarrow D; D$	
3	$D \rightarrow x: T$	
4	$T \rightarrow real$	
5	$T \rightarrow int$	
6	$T \rightarrow \uparrow T$	
7	$T \rightarrow array[n]\ of\ T_1$	
8	$T \rightarrow T_1 \mapsto T_2$	

1	$P \rightarrow D; S$	P.type:=S.type
2	$D \rightarrow D; D$	
3	$D \rightarrow x: T$	ADDTYPE(x.entry, T.type)
4	$T \rightarrow real$	T.type:= real
5	$T \rightarrow int$	T.type:= int
6	$T \rightarrow \uparrow T_1$	T.type:=ptr(T_1)
7	$T \rightarrow array[n]$ of T_1	T.type:= array(n.val, T_1)
8	$T \rightarrow T_1 \mapsto T_2$	T.type:= T_1.type $\mapsto$ T_2.type

Команды, выполняемые в программе:

9	$S \rightarrow S_1; S_2$	
10	$S \rightarrow x:=E$	
11	$S \rightarrow$ if E then S_1	
12	$S \rightarrow$ while E do S_1	
13	$E \rightarrow n$	
14	$E \rightarrow x$	
15	$E \rightarrow E_1 < E_2$	
16	$E \rightarrow E_1\uparrow$	
17	$E \rightarrow E_1[E_2]$	
18	$E \rightarrow E_1(E_2)$	

9	$S \rightarrow S_1; S_2$	$S.type := \begin{cases} S_2.type, & \text{если } S_1.type = void \\ type\_error \end{cases}$
10	$S \rightarrow x := E$	$S.type := \begin{cases} void, & \text{если } TYPE(x.entry) = E.type \\ type\_error \end{cases}$
11	$S \rightarrow$ if E then S_1	$S.type := \begin{cases} S_1.type, & \text{если } E.type = bool \\ type\_error \end{cases}$
12	$S \rightarrow$ while E do S_1	$S.type := \begin{cases} S_1.type, & \text{если } E.type = bool \\ type\_error \end{cases}$
13	$E \rightarrow n$	E.type:= int
14	$E \rightarrow x$	E.type:=TYPE(x.entry)
15	$E \rightarrow E_1 < E_2$	$E.type := \begin{cases} bool, & \text{если } E_1.type = E_1.type = int \\ bool, & \text{если } E_1.type = E_1.type = real \\ type\_error \end{cases}$
16	$E \rightarrow E_1 \uparrow$	$E.type := \begin{cases} t, & \text{если } E_1.type = ptr(t) \\ type\_error \end{cases}$

17	$E \rightarrow E_1[E_2]$	$E.type := \begin{cases} t, & \text{если } E_2.type = int,\ E_1.type = array(N, t) \\ type\_error \end{cases}$
18	$E \rightarrow E_1(E_2)$	$E.type := \begin{cases} t, & \text{если } E_2.type = s,\ E_1.type = s \mapsto t \\ type\_error \end{cases}$

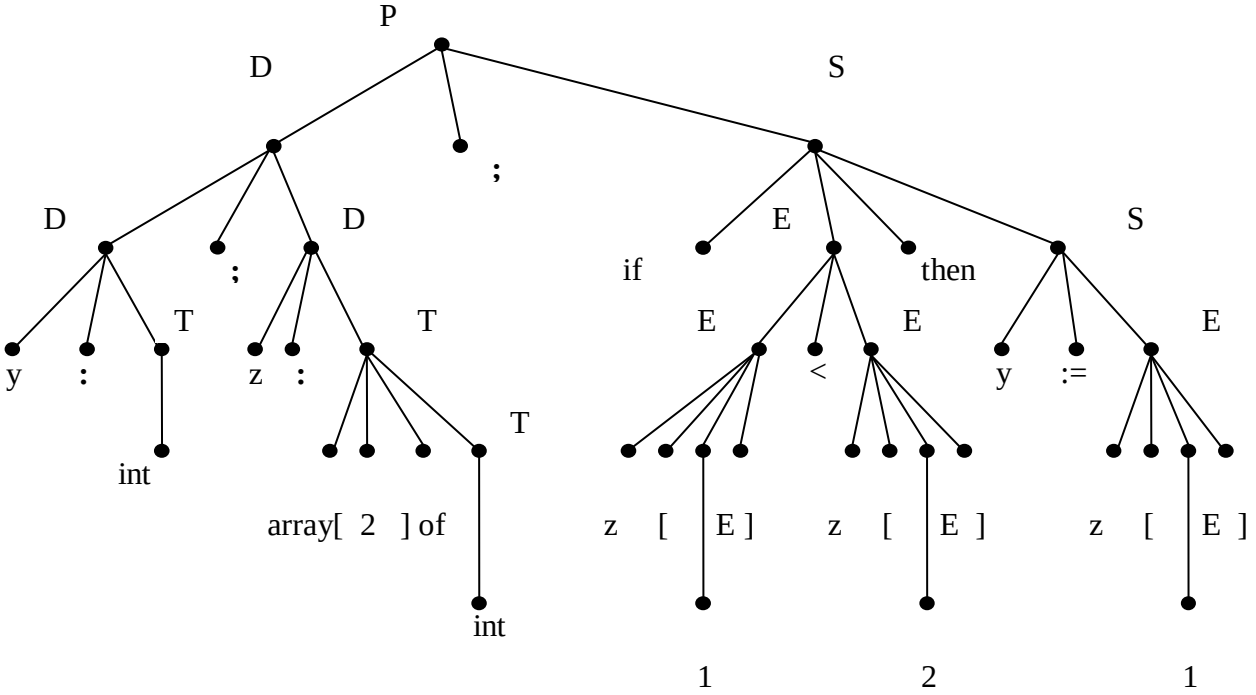
Для решения задач в этой грамматике не хватает присваивания значению массива:

19	$S \rightarrow R:=E$	$S.type:=\begin{cases} void, & \text{если } R.type = E.type \\ type\_error \end{cases}$
20	$R \rightarrow E_1[E_2]$	$R.type:=\begin{cases} t, & \text{если } E_2.type = int,\ E_1.type = array(N, t) \\ type\_error \end{cases}$

Пример (стр. 229, Шур, Замятин).
Дана программа:

```
y: int;
z: array[2] of int:
if z[1] < z[2] then y:=z[1]
```

Построим дерево вывода этой программы в грамматике APG.

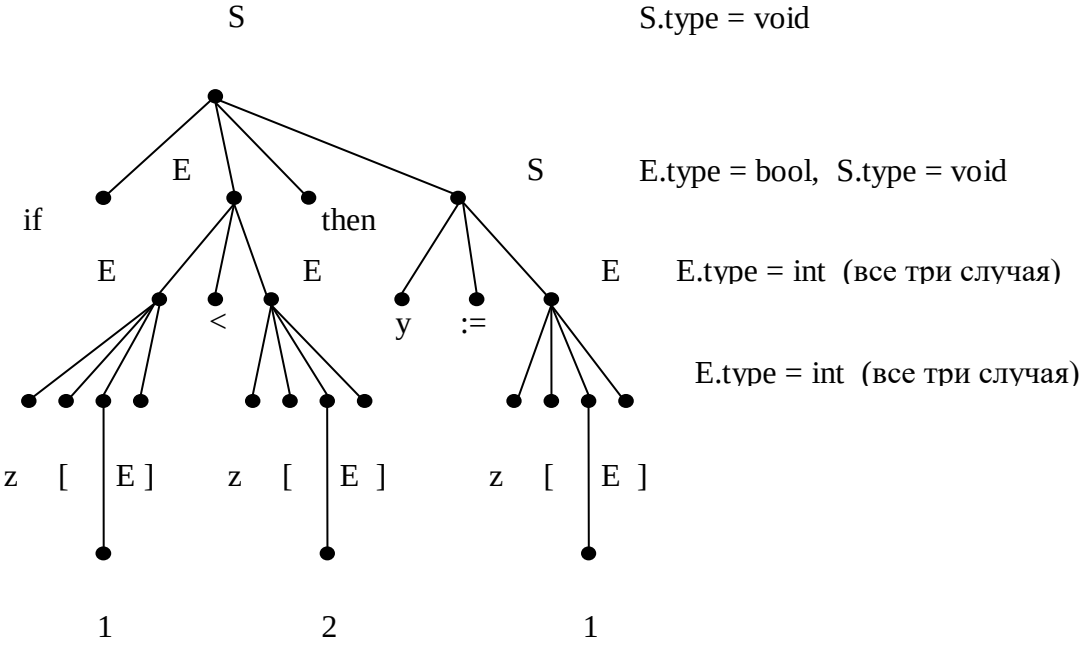

P
D
S
;
D
;
D
E
S
if
then
T
T
E
E
E
y
:
z
:
<
y
:=
T
int
array[2] of
z [E]
z [E]
z [E]
int
1
2
1

При обходе левого поддерева с корнем D будет два раза выполняться процедура ADDTYPE, первый раз с аргументами ADDTYPE(y, int), второй раз – ADDTYPE(z, array(2, int)).

В результате в таблице символов появятся строки:

Id	**Тип**	
y	int	…
z	array(2,int)	…

При обходе правового поддерева с корнем S будет вычислено значение атрибута S.type:



§ 8. Преобразование типов

В реальных языках программирования требования на типы аргументов команд менее жесткие.

В обращении с типами используется инструмент преобразования типов, иногда выполняющийся компилятором автоматически (неявно).

1. Присваивание.

10	$S \rightarrow x:=E$	$S.type:=\begin{cases} void, & \text{если } TYPE(x.entry) = E.type \\ type\_error \end{cases}$
19	$S \rightarrow R:=E$	$S.type:=\begin{cases} void, & \text{если } R.type = E.type \\ type\_error \end{cases}$

Вместо сравнения типов идентификтора x или нетерминала R с типом E можно обратиться к процедуре, выполняющей преобразование выражения, выводимого из E к нужному типу. Если преобразование невозможно, процедура выдаст результат type_error, который будет присвоен значению S.type. Если преобразование возможно, процедура выдаст результат void, который будет присвоен значению S.type.

Аналогично можно сделать в обращении к элементу массива $E \rightarrow E_1[E_2]$, или в сравнении «меньше» $E \rightarrow E_1 < E_2$.

2. Присваивание результата сложения.
В команде вида z:= x + y, применяемых во всех языках программирования, идентификаторы x и y могут быть как целыми, так и действительными числами. Действительные числа могут двух видов: real (короткое) и double (длинное). Также может встретиться «+» для операции конкатенации строк.

В этом случае производится преобразование типов x и y, так, чтобы результат операции можно было бы получить.