

## Расширенные Формы Бэкуса – Наура

<операции\_группы\_отношения>:: = != | = = | < | <= | > | >=  
<операции\_группы\_сложения>:: = + | - | ||  
<операции\_группы\_умножения>:: = \* | / | &&  
<унарная\_операция>:: = !  
<программа>:: = program var <описание> begin <оператор> { ; <оператор> } end.  
<описание>:: = <тип> <идентификатор> { , <идентификатор> }  
<тип>:: = int | float | bool  
<оператор>:: = <составной> | <присваивания> | <условный> |  
<фиксированного\_цикла> | <условного\_цикла> | <ввода> | <вывода>  
<составной>:: = begin <оператор> { ; <оператор> } end  
<присваивания>:: = <идентификатор> := <выражение>  
<условный>:: = if «(»<выражение> «)» <оператор> [else <оператор>]  
<фиксированного\_цикла>:: = for <присваивания> to <выражение> [step  
<выражение>] <оператор> next  
<условного\_цикла>:: = while «(»<выражение> «)» <оператор>  
<ввода>:: = readln идентификатор { , <идентификатор> }  
<вывода>:: = writeln <выражение> { , <выражение> }  
<признак\_начала\_комментария>:: = {  
<признак\_конца\_комментария>:: = }  
<выражение>:: = <операнд> {<операции\_группы\_отношения><операнд>}  
<операнд>:: = <слагаемое> {<операции\_группы\_сложения><слагаемое>}  
<слагаемое>:: = <множитель> {<операции\_группы\_умножения><множитель>}  
<множитель>:: = <идентификатор> | <число> | <логическая\_константа> |  
<унарная\_операция><множитель> | «(»<выражение>«)»  
<число>:: = <целое> | <действительное>  
<логическая\_константа>:: = true | false  
<идентификатор>:: = <буква> {<буква> | <цифра>}  
<буква>:: = A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T |  
U | V | W | X | Y | Z | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p  
q | r | s | t | u | v | w | x | y | z  
<цифра>:: = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9  
<целое>:: = <двоичное> | <восьмеричное> | <десятичное> |  
<шестнадцатеричное>  
<двоичное>:: = { / 0 | 1 / } ( B | b )  
<восьмеричное>:: = { / 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 / } ( O | o )  
<десятичное>:: = { / <цифра> / } [ D | d ]  
<шестнадцатеричное>:: = <цифра> { <цифра> | A | B | C | D | E | F | a | b |  
c | d | e | f } ( H | h )  
<действительное>:: = <числовая\_строка><порядок> |  
[<числовая\_строка>] . <числовая\_строка> [порядок]  
<числовая\_строка>:: = { / <цифра> / }  
<порядок>:: = ( E | e ) [ + | - ] <числовая\_строка>

## Формальные грамматики

Pr → program var Opis begin Oper { ; Oper } end.  
Opis → Tip Ident { , Ident }  
Oper → sostav | prisv | usl | fix | uslc | vvod | vyvod  
Tip → int | float | bool  
sostav → begin Oper { ; Oper } end  
prisv → Ident := vyraz  
usl → if (Vyraz) Oper [else Oper]  
fix → for prisv to vyraz [step vyraz] Oper next  
uslc → while (vyraz) Oper  
vvod → readln Ident { , Ident }  
vyvod → writeln vyraz { , vyraz }  
start\_kom → {  
end\_kom → }  
vyraz → Operand { otnosh Operand }  
Operand → slag { gr\_slog slag }  
slag → mnog { umnog mnog }  
mnog → Ident | chislo | const | unar mnog (vyraz)  
chislo → celoe | deystv  
const → true | false  
Ident → bookva { bookva | cifra }  
otnosh → != | == | < | <= | > | >=  
slog → + | - | ||  
umnog → \* | / | &&  
unar → !=  
bookva → A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V  
| W | X | Y | Z | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w  
| x | y | z  
cifra → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9  
celoe → dvoichnoe | vosmerich | desyatich | shestnad  
dvoichnoe → dvc B | dvc b  
dvc → 0 | 1 | dvc 0 | dvc 1  
vosmerich → vosmc O | vosmc o  
vosmc → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | vosmc0 | vosmc1 | vosmc2 | vosmc3 | vosmc4 |  
vosmc5 | vosmc6 | vosmc7  
desyatich → cifraD | cifrad | cifra  
shestnad1 → cifra | shestnsd1 cifra | shestnad1 buk16  
shestnad → shestnad1H | shestnad1h  
shestnad1Buk16 → A | B | C | D | E | F | a | b | c | d | e | f  
deystvit → chisl\_str Porydok | . Chisl\_str | Chisl\_str. Chisl\_str | Chisl\_str. Chisl\_str  
Porydok | .chisl\_str Porydok

Chisl\_str → Cifra | Chisl\_str Cifra

Porydok → E Znak Chisl\_str | e Znak Chisl\_str | E Chisl\_str | e Chisl\_str

Znak → + | -

## Вариант программы

The screenshot shows a software interface with the following components:

- Текст программы:**

```
program
var q integer
read q
q=q plus 1
write q
end
```
- Лексемы:**

```
(1,7) (4,8) (1,8) (2,8) (1,18) (4,3) (2,5) (4,1) (2,16) (4,3) (1,5) (3,8) (2,8) (1,19) (4,6) (2,7) (1,6) (2,0)
```
- Таблицы:**

№	Символьное выражение	№	Результат
0	plus	0	
1	min	1	-
2	or	2	.
3	multi	3	,
4	div	4	E
5	mod	5	!
6	end	6	(
7	for	7	)
8	integer	8	{
9	real	9	}
10	boolean	10	^

№	Имя	№	Идентификаторы	Описание	Тип
0	+	0	q	0	

## Вариант программы 2

The screenshot shows a software interface with the following components:

- Текст программы:**

```
{ max,x,y:integer; read(x, y); if x > y then max as x else max as y; write(max) }
```
- Лексемы:**

```
(1;0) (1;1) (1;19) (4;0) (2;12) (4;1) (2;12) (4;2) (2;10) (1;4) (1;11) (2;14) (4;1) (2;12) (4;2) (2;15) (2;10) (1;6) (4;1) (2;4) (4;2) (1;7) (4;0) (1;16) (4;1) (1;8) (4;0) (1;16) (4;2) (2;10) (1;12) (2;14) (4;0) (2;15) (1;5) (2;11)
```
- Ошибки:**

```
->
Лексический анализ: Успешно.
->
Синтаксический: Успешно.
```

### Вариант программы 3

Лексический анализатор

Таблица служебных слов (ТШ):

#	##
1	program
2	var
3	begin
4	end.
5	int

Таблица ограничителей (ТЛ):

#	##
1	!=
2	==
3	<
4	<=
5	>

Исходный текст:

```

program
var int max, x, y
begin
  readln x, y
  if (x > y) max:
= x else max:= y
  writeln max

```

Результат анализа:

```

(1, 1) (1, 2) (1, 5) (3, 1) (2, 16) (3, 2)
(2, 16) (3, 3) (1, 3) (1, 15) (3, 2) (2,
16) (3, 3) (1, 8) (2, 17) (3, 2) (2, 5)
(3, 3) (2, 18) (3, 1) Отсутствует end.

```

Таблица идентификаторов (ТИ):

#	##
1	max
2	x
3	y

Таблица чисел (ТН):

#	##

Открыть файл

Проверить код

### Вариант программы 4

TAF

0 - Keywords:

```

0: dim
1: integer
2: real
3: boolean
4: read
5: write
6: ass
7: if
8: NE
9: then
10: else
11: end

```

Code:

```

dim a, b, c integer :
dim d, e, f real :
dim g, h, i boolean:
read(a, b, c) :
write(a) :
write(b) :
write(c) :
a ass 28 :
d ass 2.34 :
f ass .0232:
if a NE b then
a ass b :
c ass b|
else
b ass a :
c ass a;
end

```

Lexems:

```

[[1: 0), (4: 0), (2: 0), (4: 1), (2: 0), (4: 2), (1: 1), (2:
1), (1: 0), (4: 3), (2: 0), (4: 4), (2: 0), (4: 5), (1: 2),
(2: 1), (1: 0), (4: 6), (2: 0), (4: 7), (2: 0), (4: 8), (1:
3), (2: 1), (1: 4), (2: 2), (4: 0), (2: 0), (4: 1), (2: 0),
(4: 2), (2: 3), (2: 1), (1: 5), (2: 2), (4: 0), (2: 3), (2:
1), (1: 5), (2: 2), (4: 1), (2: 3), (2: 1), (1: 5), (2: 2),
(4: 2), (2: 3), (2: 1), (4: 0), (1: 6), (3: 1), (2: 1), (4:
3), (1: 6), (3: 2), (2: 1), (4: 5), (1: 6), (3: 0), (2: 1),
(1: 7), (4: 0), (1: 8), (4: 1), (1: 9), (4: 0), (1: 6), (4:
1), (2: 1), (4: 2), (1: 6), (4: 1), (1: 10), (4: 1), (1: 6),
(4: 0), (2: 1), (4: 2), (1: 6), (4: 0), (2: 4), (1: 11), ]

```

2 - Digits:

```

0: .0232 (0x1.7c1bda5119c
1: 28 (11100)
2: 2.34 (0x1.2b851eb851eb

```

1 - Delimiters:

```

0: ,
1: :
2: (
3: )
4: ;

```

Log:

Tokens:

```

[[@0,0:2='dim',<18>,1:0]
[[@1,4:4='a',<44>,1:4]
[[@2,5:5='',<36>,1:5]
[[@3,7:7='b',<44>,1:7]
[[@4,8:8='',<36>,1:8]
[[@5,10:10='c',<44>,1:10]
[[@6,12:12='integer',<19>,1:12]
[[@7,20:20=':',<37>,1:20]
[[@8,22:24='dim',<18>,2:0]
[[@9,26:26='d',<44>,2:4]
[[@10,27:27='',<36>,2:5]
[[@11,29:29='e',<44>,2:7]
[[@12,30:30='',<36>,2:8]

```

3 - Identifiers:

#	Name	Type
0	a	integer (I...
1	b	integer (I...
2	c	integer (I...
3	d	real (FLO...
4	e	real (FLO...
5	f	real (FLO...
6	g	boolean (...
7	h	boolean (...
8	i	boolean (...