Практическое занятие по СПО №3

Густов Владимир Владимирович
gutstuf@gmail.com

Цитата дня: Дед Загорулько проснулся и спросил: «Сейчас октябрь или декабрь?»
(с) Егор Летов

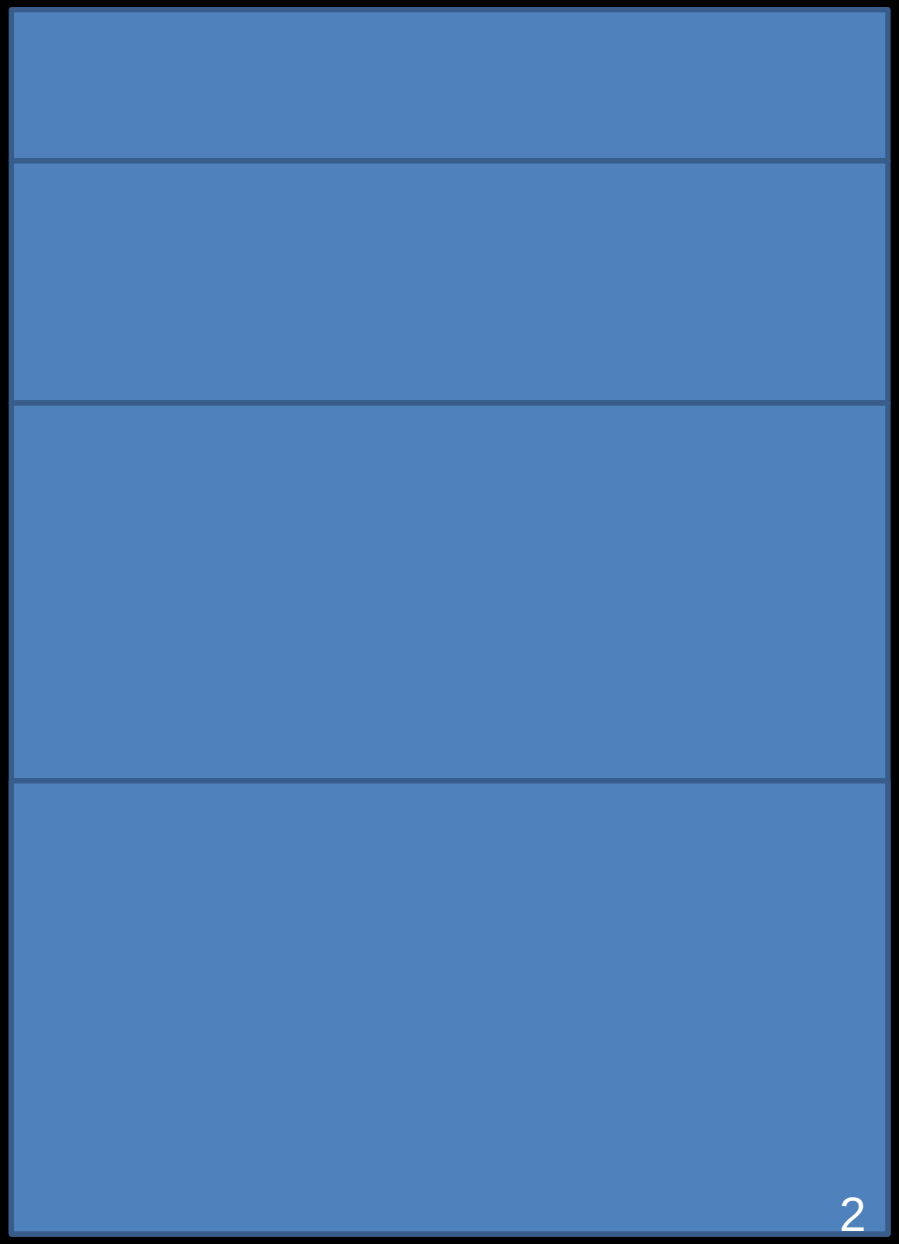
Repeat it

1.
```
program ag41n;
```

2.
```
program okey;
var 1_end : bool;
begin
end.
```

3.
```
program try_4g41n;
var oK : Boolean = true;
begin
 while oK do
   writeln(oK);
end.
```

4.
```
program Pog;
var null: integer;
begin
 null := 1 <> 2 + 2;
end.
```



Repeat it

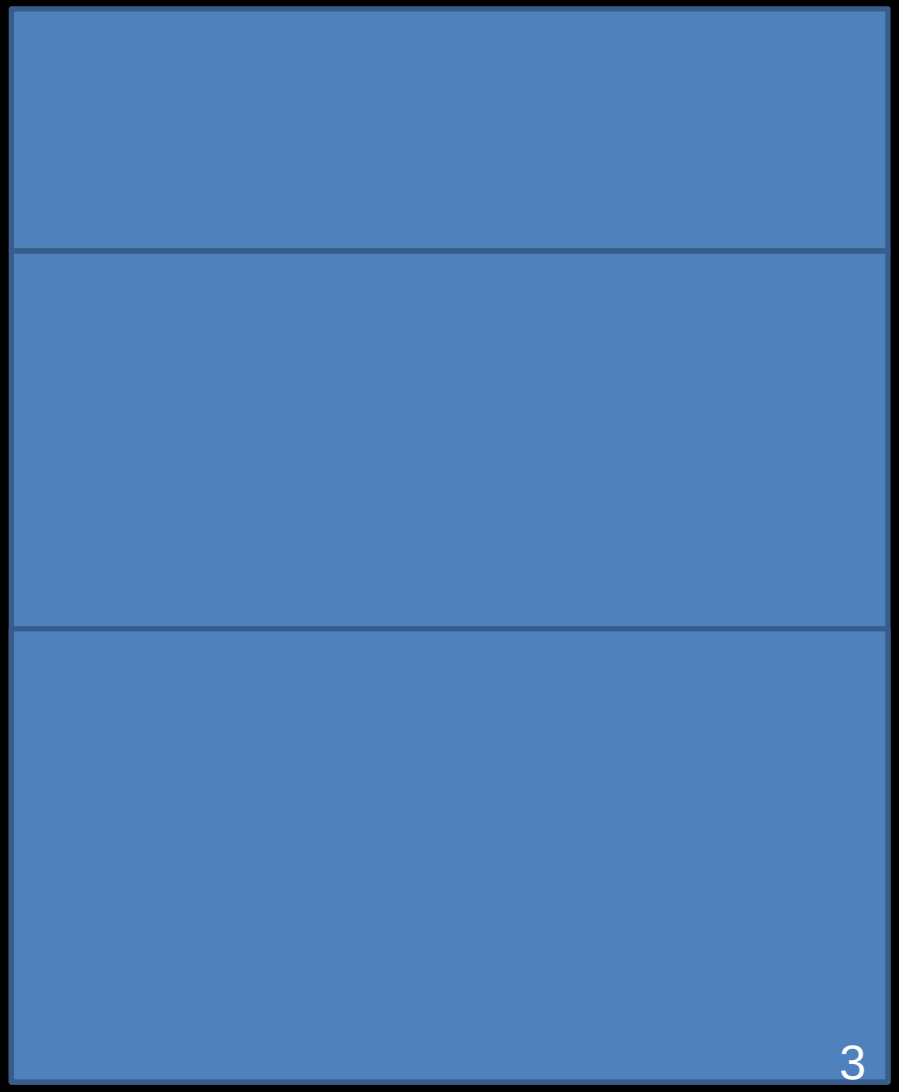
1.
```
program ag41n;
```

2.
```
program okey;
var 1_end : bool;
begin
end.
```

3.
```
program try_4g41n;
var oK : Boolean = true;
begin
 while oK do
   writeln(oK);
end.
```

4.
```
program Pog;
var null: integer;
begin
 null := 1 <> 2 + 2;
end.
```

Синтаксическая



Repeat it

1.
```
program ag41n;
```
Синтаксическая

2.
```
program okey;
var 1_end : bool;
begin
end.
```
Лексическая

3.
```
program try_4g41n;
var oK : Boolean = true;
begin
 while oK do
   writeln(oK);
end.
```

4.
```
program Pog;
var null: integer;
begin
 null := 1 <> 2 + 2;
end.
```

Repeat it

1.
```
program ag41n;
```
Синтаксическая

2.
```
program okey;
var 1_end : bool;
begin
end.
```
Лексическая

3.
```
program try_4g41n;
var oK : Boolean = true;
begin
 while oK do
   writeln(oK);
end.
```
Логическая

4.
```
program Pog;
var null: integer;
begin
 null := 1 <> 2 + 2;
end.
```

Repeat it

1.
```
program ag41n;
```
Синтаксическая

2.
```
program okey;
var 1_end : bool;
begin
end.
```
Лексическая

3.
```
program try_4g41n;
var oK : Boolean = true;
begin
 while oK do
   writeln(oK);
end.
```
Логическая

4.
```
program Pog;
var null: integer;
begin
 null := 1 <> 2 + 2;
end.
```
Семантическая

Грамматика Паскаля

```
<assignment> ::= <variable> := <expression>
<expression> ::= <simple exp>
                 | <simple exp> <rel op>
                   <simple exp>
<simple exp> ::= <term>
                 | <sign> <term>
                 | <simple exp> <add op> <term>
<term> ::= <factor> | <term> <mul op> <factor>
<factor> ::= <variable> | ( <expression> )

<rel op> ::= = | <> | < | <= | >= | >
<add op> ::= + | - | or
<mul op> ::= * | / | div | mod | and
```

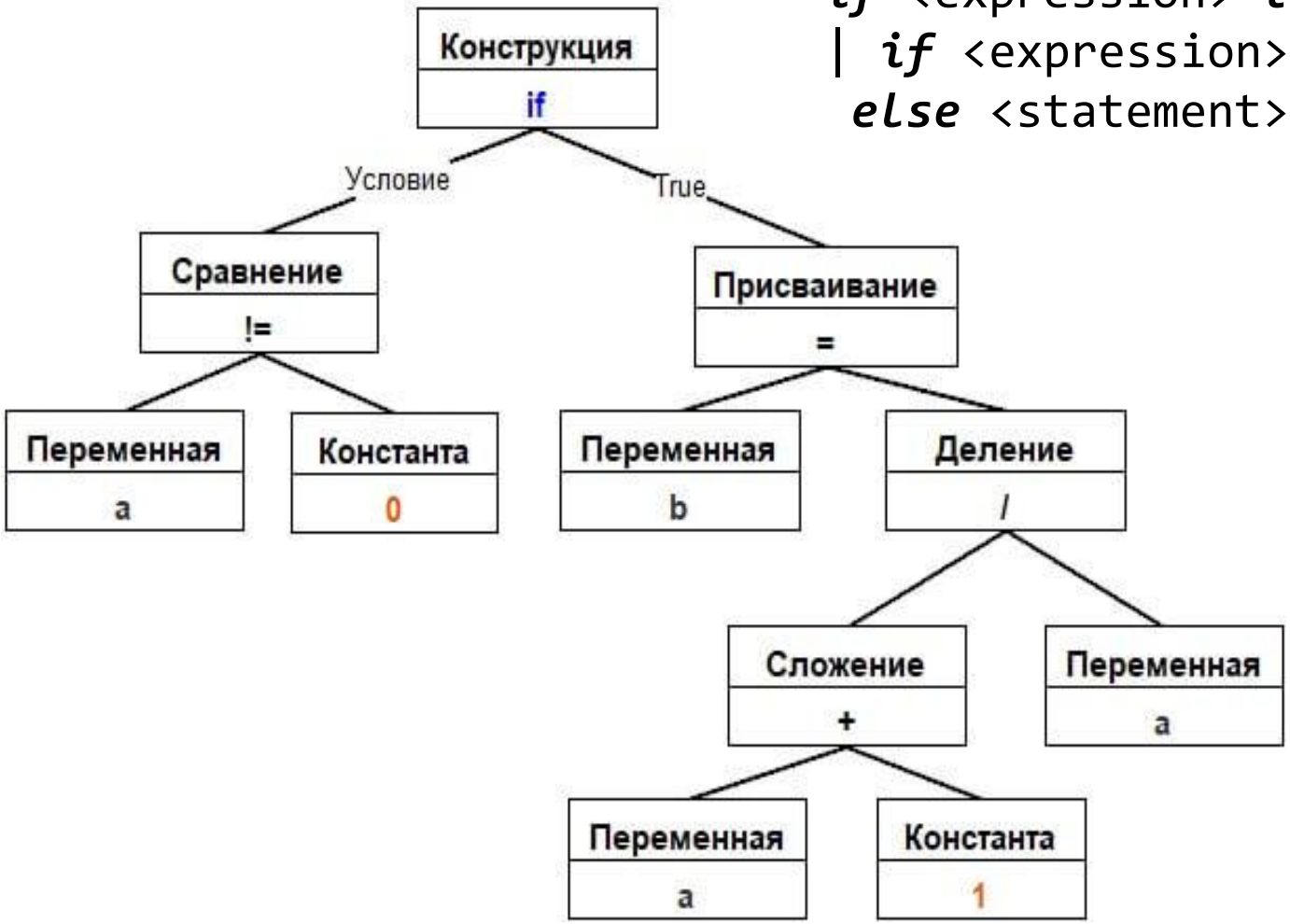
```
1) a := 1 + 2 * 3;
2) a := 2 + (3 – 1) div 1;
```

Условный оператор: if then else

Выражение:

```
if (a != 0) b = (a + 1) / a;
```

Синтаксическое дерево:

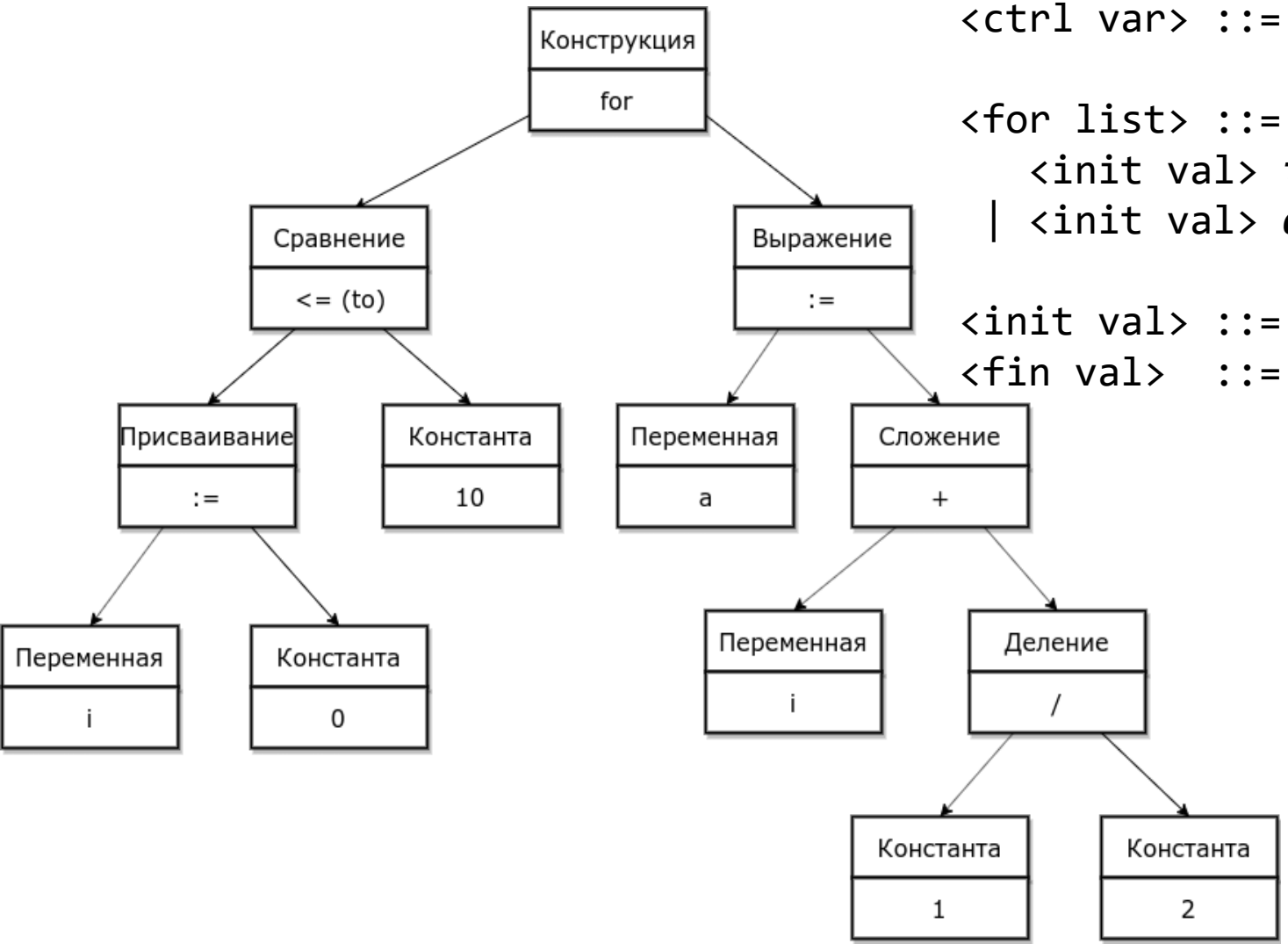



```
<if statement> ::=
if <expression> then <statement>
| if <expression> then <statement>
  else <statement>
```

Оператор цикла (со счётчиком): for to do

```
for i := 0 to 10 do
    a := i + 1 / 2;
```

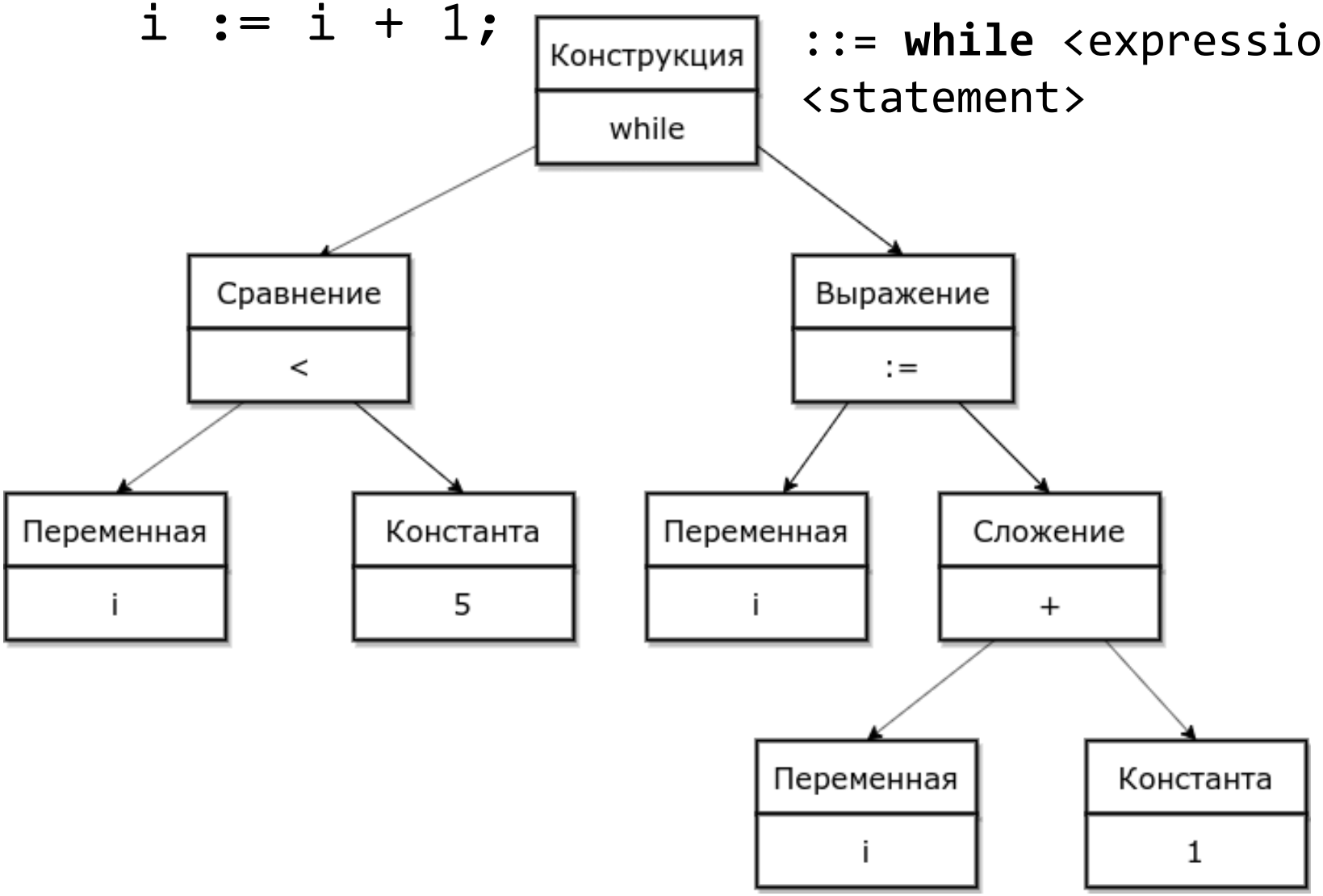
```
<for statement> ::=
for <ctrl var> :=
<for list> do <statement>

<ctrl var> ::= <id>

<for list> ::=
    <init val> to <fin val>
  | <init val> downto <fin val>

<init val> ::= <expression>
<fin val>  ::= <expression>
```



Оператор цикла (с предусловием): while do

```
while i < 5 do
    i := i + 1;
```

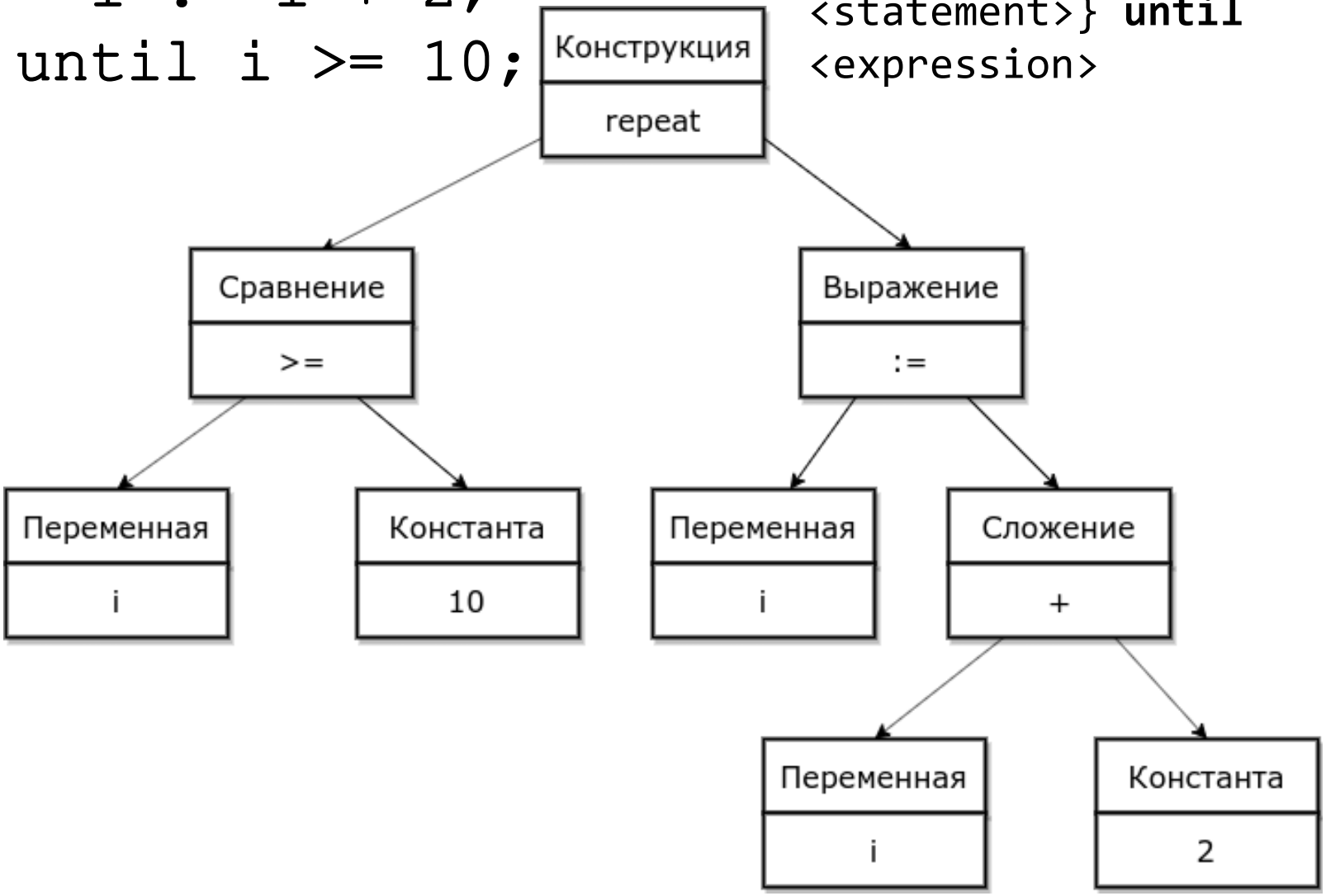
<while statement>
::= **while** <expression> **do**
<statement>



Оператор цикла (с постусловием): repeat until

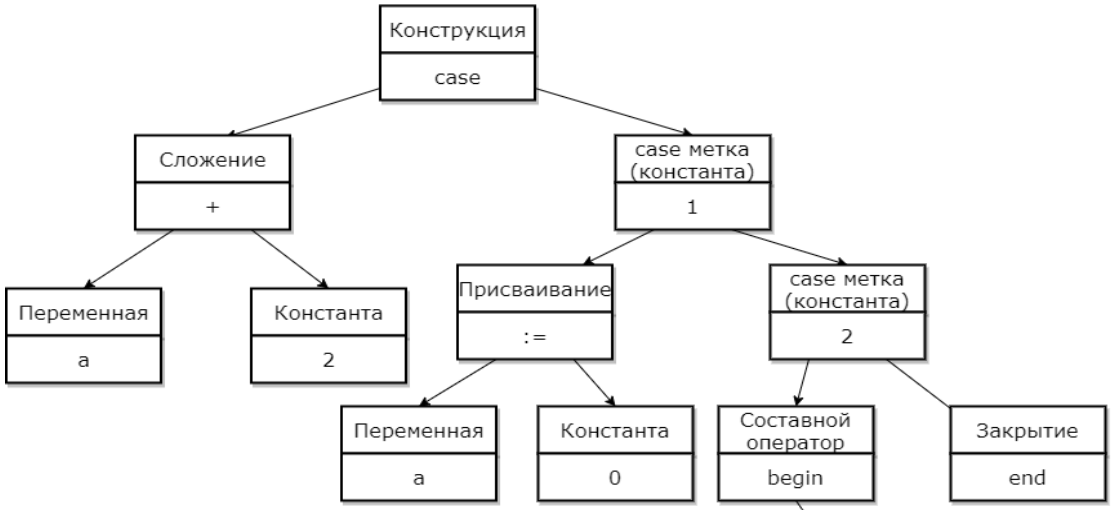
```
repeat
   i := i + 2;
until i >= 10;
```

```
<repeat statement>
::= repeat <statement> {;
<statement>} until
<expression>
```

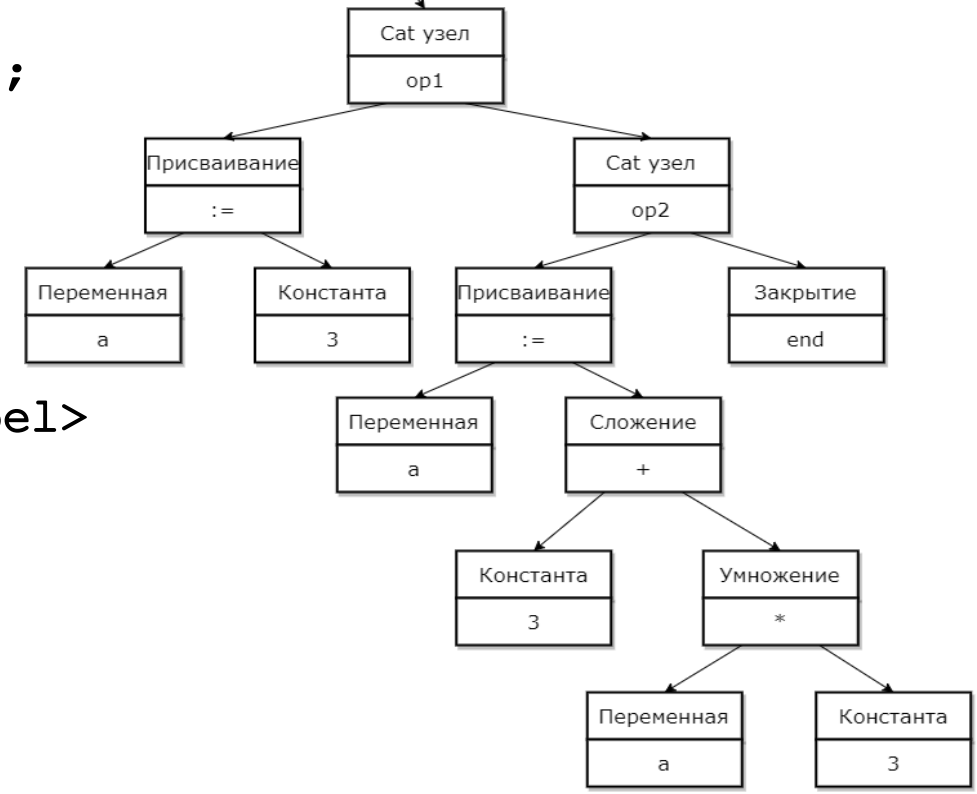


Оператор выбора: case of end

```
case a + 2 of
1: a := 0;
2: begin
   a := 3;
   a := 3 + a * 3;
   end;
end;
```

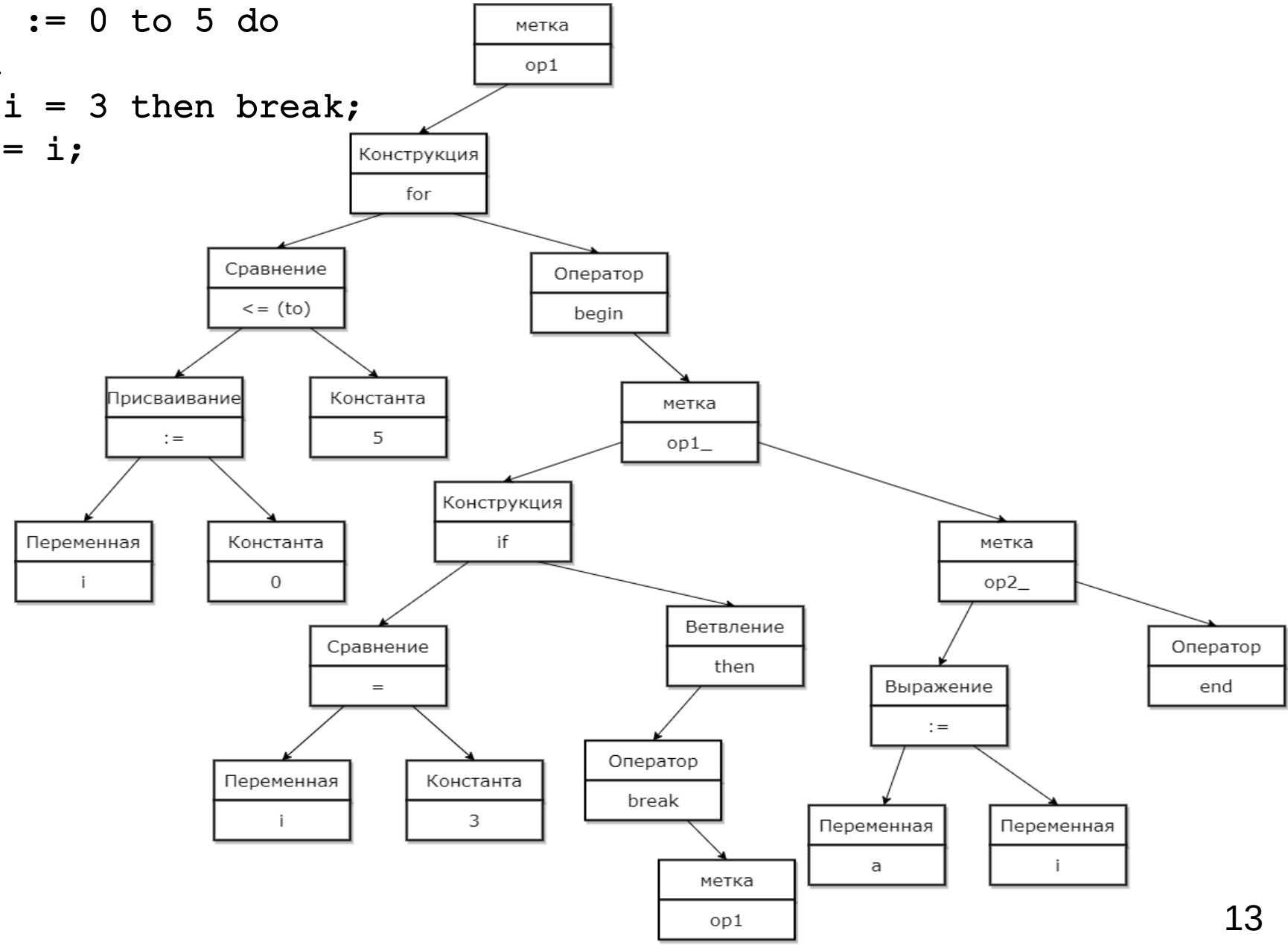



```
<case state> ::= case
  <expression> of <case list> {;
<case list> } end

<case list> ::= <case label
 list> : <statement> | <empty>

<case label list> ::= <case label>
{, <case label> }

<case label> ::= <constant>
```



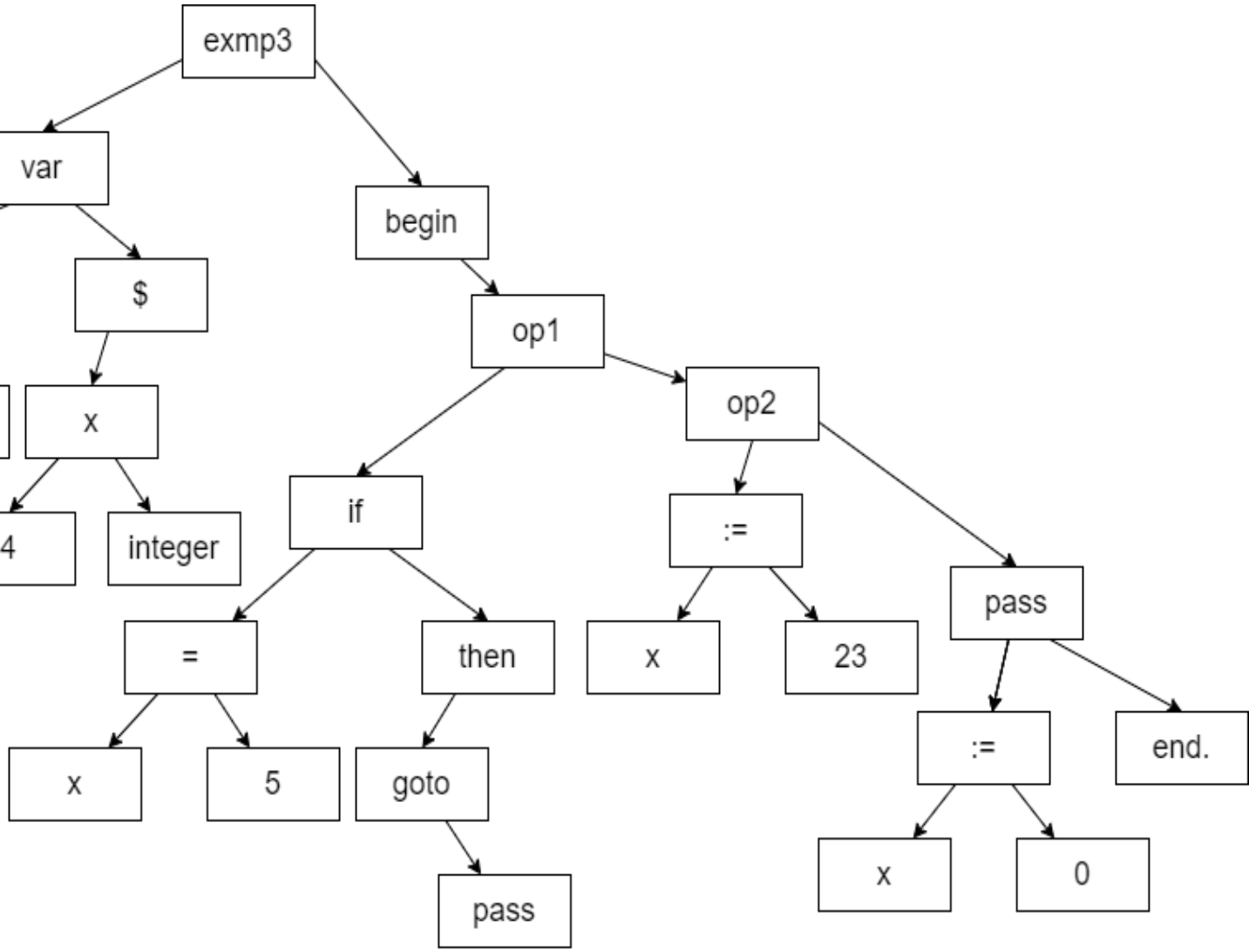
Оператор break

```
for i := 0 to 5 do
begin
  if i = 3 then break;
  a := i;
end;
```



Оператор безусловного перехода: goto

```
program exmp3;
label pass;
var x : integer = 4;
begin
  if x = 5 then
    goto pass;
  x := 23;
  pass:
  x := 0;
end.
```



GOTO

```
<block> ::= <label declaration part>
            <constant definition part>
            <var decl part> <statement part>

<label declaration part> ::= <empty>
                           | label <label> {, <label>} ;

<label> ::= <unsigned integer> | <id>

<simple statement> ::= <assignment statement>
                     | <procedure statement>
                     | <go to statement>
                     | <empty statement>
<go to statement> ::= goto <label>
```

```
program ag41n;
    var
        b : integer;
        a, d : integer;
    begin
        begin
            a := 2;
        end;
        begin
            begin
                b := 3
            end;
        end
    end.
```

Compile time

```
program surprise;
   var _end : integer = 4;
   var a    : array[-3..2] of integer;
   begin
      a[-3] := 2;
      writeln(a[-3]);
      writeln(a[_end]);
      writeln(a[4]);
   end.
```

<program> ::= **program** <identifier> ; <block> .

<block> ::= <variable declaration part> <statement part>

<statement part> ::= <compound statement>
<compound statement> ::= **begin** <statement> { ; <statement> } **end**
<statement> ::= <simple statement> | <structured statement>
<structured statement> ::= <compound statement> | <if state> | <repet state>
<if state> ::= if <exp> then <statement> | .. else <statement>
<repet state> ::= <while state> | <repeat state> | <for state>
<while state> ::= while <exp> do <statement>
<repeat state> ::= repeat <state> {; <state>} until <exp>
<for state> ::= for <id> := <for list> do <state>