Exercise P1. Lexical Scanner for Simplified Turbo Pascal

1 Aim of the Exercise

The aim of the exercise is to build a simple scanner for a much simplified version of Turbo Pascal. The task of the analyzer is:

- to recognize tokens of Turbo Pascal and to determine their values
- to remove blanks and comments
- to recognize given directives
- to recognize lexical errors

2 Preliminaries

After turning on the computer, one should select Linux, and log in as *student*. One should open a console window (e.g. press `Alt-F2` and type `xterm`), create one's own directory using a command `mkdir` *family name of the user*, and a subdirectory for the current exercise. Download files for Turbo Pascal from the Moodle web page of the course for the subject *Lexical Analysis*. The following files are to be found there:

- `p1p.pdf` — manual (just being read)
- `Makefile` — needed for compilation with the command `make`
- `common.h` — header file defining the greatest length of strings
- `p.l` — skeletal lexical analyzer that needs to be completed; take a closer look at the definition of `process_token()`, which should be used in the rules
- `p.y` — parser that is needed only for declaring tokens and for invoking the lexical analyzer
- `test1.pas` — correct test program
- `test2.pas` — test program with errors that should be detected

After having completed the exercise, the directory should be removed.

3 Tasks

The supplied skeletal lexical analyzer should be extended so that it works correctly on supplied test programs. The analyzer should print information on recognized tokens in three columns:

1. matched text
2. recognized token
3. value of the token (only when it makes sense)

Function `process_token` is designed to print that information. The function returns a recognized token, so an action in a rule recognizing a token should contain **`return`** `process_token(. . . )` with appropriate parameters.

The supplied code needs to be completed with the following items:

A. printing one's own name (in the `bison` program)

B. detecting keywords defined in the bison source file `p.y` (note: Pascal is case-insensitive)

C. removing blanks

D. recognition of multi-character operators (`<=`, `:=`,. . .) that appear in test programs

E. recognition of identifiers

F. recognition of integers

G. recognition of floating point numbers

H. recognition of strings without start conditions

I. recognition of one-character tokens: operators and punctuation

J. recognition of include directives

K. recognition of strings using start conditions

L. removal of multi-line comments { using start conditions

M. removal of multi-line comments `(*` using start conditions

N. detection of comment end sequence without the beginning sequence using start conditions

O. detection of failure to close a comment with indications of the line where the comment begins

4 Grading

All items are graded as 1 point. If needed, items from K to O can be completed **at home for half a point each**. The file developed in the lab should be uploaded before the end of the class on Moodle. **The lexical analyzer will be needed for the next exercise**.

5 Start Conditions

- Start condition active at the start of the program: `INITIAL`
- Declaraction: `%x condition1, condition2,. . .`
- Matching in a start condition:
```
<con1> re1      action1;
<con1,con2,INITIAL>re2 action2;
<*>re3        działanie3
```
- changing start condition: `BEGIN condition4`
- current start condition: `YYSTATE`
- checking the current start condition after all input data has been read: in function `yywrap`, which must be defined, and which must return 1

6 Test Data — File test1.pas

```
Program ASCII; (* Wyswietla kody ASCII *)
Uses
   crt, dos;
{$I MyFile.inc}
Var
   i : Integer;
   c : Char;
   r : real;
   t : array[1..10] of integer;
   d :  record
          rok, miesiac : integer;
          dzien        :  integer;
        end;
Const   (* zakres wyswietlanych znakow *)
   minASCII = 30;
   maxASCII = 255;
Begin
   ClrScr(); (* intro na czystym ekranie *)
   Write('Kody ASCII od 30 do 255: '); WriteLn('(po 20 w wierszu):');
   For i := minASCII To maxASCII Do (* wyswietlenie zadanych kodow ASCII *)
      Write( Chr( i ) : 4 );
   ReadKey; (* czekaj na nacisniecie klawisza *)
   r := 12.34e-12 * ( 56.0 + 0.78 ); { test liczb rzeczywistych }
```

```
    i := minASCII + 2 * (20 + maxASCII );
    t[10] := 1;
    for i := 9 downto 1 do t[i] := t[i+1] * i * i;
    d.rok := 2018;
    d.dzien := 1;
    d.miesiac := d.dzien * 10;
End.
```

7 Test Data — File test2.pas

```
Program ASCII; (* Wyswietla kody ASCII *)
Uses
        crt, dos;
{$I MyFile.inc}
Var
        i : Integer;
        c : Char;
        r : real;
Const   (* zakres wyswietlanych znakow *)
        minASCII = 30;
        maxASCII = 255;
Begin
        ClrScr(); (* intro na czystym ekranie *)
        Writeln( 'Kody ASCII od 30 do 255: (po 20 w wierszu):' );
        For i := minASCII To maxASCII Do (* wyswietlenie zadanych kodów ASCII *)
                Write( Chr( i ) : 4 );
        ReadKey; (* czekaj na nacisniecie klawisza *)
        r := 12.34 * ( 56.0 + 0.78 ); { test liczb rzeczywistych }
        i := minASCII + 2 * (20 + maxASCII );
        *) { nieotwarty komentarz }
        }  { nieotwarty komentarz }
        { komentarz
        wielowierszowy 1 }
        (* komentarz
        wielowierszowy 2 *)
        { niezamkniety komentarz ...
End.
```

8 Output of Lexical Analyzer for test1.pas

```
Autor: Imie i Nazwisko
yytext              Typ elementu    Wartość elementu znakowo

Program             KW_PROGRAM
ASCII               IDENT           ASCII
;                   ;
Uses                KW_USES
crt                 IDENT           crt
,                   ,
dos                 IDENT           dos
;                   ;
Processing directive INCLUDE
Var                 KW_VAR
i                   IDENT           i
:                   :
Integer             KW_INTEGER
;                   ;
c                   IDENT           c
:                   :
Char                KW_CHAR
;                   ;
```

```
r                        IDENT              r
:                        :
real                     IDENT              real
;                        ;
t                        IDENT              t
:                        :
array                    KW_ARRAY
[                        [
1                        INTEGER_CONST      1
..                       RANGE
10                       INTEGER_CONST      10
]                        ]
of                       KW_OF
integer                  KW_INTEGER
;                        ;
d                        IDENT              d
:                        :
record                   KW_RECORD
rok                      IDENT              rok
,                        ,
miesiac                  IDENT              miesiac
:                        :
integer                  KW_INTEGER
;                        ;
dzien                    IDENT              dzien
:                        :
integer                  KW_INTEGER
;                        ;
end                      KW_END
;                        ;
Const                    KW_CONST
minASCII                 IDENT              minASCII
=                        =
30                       INTEGER_CONST      30
;                        ;
maxASCII                 IDENT              maxASCII
=                        =
255                      INTEGER_CONST      255
;                        ;
Begin                    KW_BEGIN
ClrScr                   IDENT              ClrScr
(                        (
)                        )
;                        ;
Write                    IDENT              Write
(                        (
'Kody ASCII od 30 doSTRING_CONST            'Kody ASCII od 30 do 255: '
)                        )
;                        ;
WriteLn                  IDENT              WriteLn
(                        (
'(po 20 w wierszu):'STRING_CONST            '(po 20 w wierszu):'
)                        )
;                        ;
For                      KW_FOR
i                        IDENT              i
:=                       ASSIGN
minASCII                 IDENT              minASCII
To                       KW_TO
maxASCII                 IDENT              maxASCII
Do                       KW_DO
Write                    IDENT              Write
(                        (
Chr                      IDENT              Chr
(                        (
i                        IDENT              i
)                   )
:                   :
4                   INTEGER_CONST    4
)                   )
;                   ;
ReadKey             IDENT            ReadKey
;                   ;
r                   IDENT            r
:=                  ASSIGN
12.34e-12           FLOAT_CONST      12.34e-12
*                   *
(                   (
56.0                FLOAT_CONST      56.0
+                   +
0.78                FLOAT_CONST      0.78
)                   )
;                   ;
i                   IDENT            i
:=                  ASSIGN
minASCII            IDENT            minASCII
+                   +
2                   INTEGER_CONST    2
*                   *
(                   (
20                  INTEGER_CONST    20
+                   +
maxASCII            IDENT            maxASCII
)                   )
;                   ;
t                   IDENT            t
[                   [
10                  INTEGER_CONST    10
]                   ]
:=                  ASSIGN
1                   INTEGER_CONST    1
;                   ;
for                 KW_FOR
i                   IDENT            i
:=                  ASSIGN
9                   INTEGER_CONST    9
downto              KW_DOWNTO
1                   INTEGER_CONST    1
do                  KW_DO
t                   IDENT            t
[                   [
i                   IDENT            i
]                   ]
:=                  ASSIGN
t                   IDENT            t
[                   [
i                   IDENT            i
+                   +
1                   INTEGER_CONST    1
]                   ]
*                   *
i                   IDENT            i
*                   *
i                   IDENT            i
;                   ;
d                   IDENT            d
.                   .
rok                 IDENT            rok
:=                  ASSIGN
2018                INTEGER_CONST    2018
;                   ;
d                   IDENT            d
```

```
.                      .
dzien                  IDENT            dzien
:=                     ASSIGN
1                      INTEGER_CONST    1
;                      ;
d                      IDENT            d
.                      .
miesiac                IDENT            miesiac
:=                     ASSIGN
d                      IDENT            d
.                      .
dzien                  IDENT            dzien
*                      *
10                     INTEGER_CONST    10
;                      ;
End                    KW_END
.                      .
```

9 Output of Lexical Analyzer for test2.pas

```
Autor: Imie i Nazwisko
yytext                 Typ elementu     Wartość elementu znakowo

Program                KW_PROGRAM
ASCII                  IDENT            ASCII
;                      ;
Uses                   KW_USES
crt                    IDENT            crt
,                      ,
dos                    IDENT            dos
;                      ;
Przetwarzanie dyrektywy INCLUDE
Var                    KW_VAR
i                      IDENT            i
:                      :
Integer                KW_INTEGER
;                      ;
c                      IDENT            c
:                      :
Char                   KW_CHAR
;                      ;
r                      IDENT            r
:                      :
real                   IDENT            real
;                      ;
Const                  KW_CONST
minASCII               IDENT            minASCII
=                      =
30                     INTEGER_CONST    30
;                      ;
maxASCII               IDENT            maxASCII
=                      =
255                    INTEGER_CONST    255
;                      ;
Begin                  KW_BEGIN
ClrScr                 IDENT            ClrScr
(                      (
)                      )
;                      ;
Writeln                IDENT            Writeln
(                      (
'Kody ASCII od 30 doSTRING_CONST        'Kody ASCII od 30 do 255: (po 20 w wierszu):'
)                      )
;                      ;
```

```
For                    KW_FOR
i                      IDENT            i
:=                     ASSIGN
minASCII               IDENT            minASCII
To                     KW_TO
maxASCII               IDENT            maxASCII
Do                     KW_DO
Write                  IDENT            Write
(                      (
Chr                    IDENT            Chr
(                      (
i                      IDENT            i
)                      )
:                      :
4                      INTEGER_CONST    4
)                      )
;                      ;
ReadKey                IDENT            ReadKey
;                      ;
r                      IDENT            r
:=                     ASSIGN
12.34                  FLOAT_CONST      12.34
*                      *
(                      (
56.                    FLOAT_CONST      56.
+                      +
.78                    FLOAT_CONST      .78
)                      )
;                      ;
i                      IDENT            i
:=                     ASSIGN
minASCII               IDENT            minASCII
+                      +
2                      INTEGER_CONST    2
*                      *
(                      (
20                     INTEGER_CONST    20
+                      +
maxASCII               IDENT            maxASCII
)                      )
;                      ;
Unexpected closure of comment in line 17
Unexpected closure of comment in line 19
Comment opened in line 23 not closed
```