Exercise P1. Lexical Scanner for Simplified C Programming Language

1 Aim of the Exercise

The aim of the exercise is to build a simple scanner for a much simplified version of the programming language C. The task of the analyzer is:

- to recognize tokens of C language 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 the C language from the Moodle web page of the course for the subject *Lexical Analysis*. The following files are to be found there:

- `p1c.pdf` — manual (just being read)
- `Makefile` — needed for compilation with the command `make`
- `common.h` — header file defining the greatest length of strings
- `c.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
- `c.y` — parser that is needed only for declaring tokens and for invoking the lexical analyzer
- `test1.c` — correct test program
- `test2.c` — 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 **using the ordering given below**:

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

B. detecting keywords in test programs (defined in the source code for `bison`)

C. removing blanks

D. removing one-line comments

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

F. recognition of identifiers

G. recognition of integers and floating point numbers

H. recognition of strings without start conditions

I. recognition of character constants

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

K. recognition of include directives

L. recognition of strings 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**. Make sure that recognized tokens are handed over to the parser using `return`. Tokens being removed, e.g. comments or white spaces, should not call `return`.

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          action3
  ```
- 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.c

```
/****************************************************/
/* Program ASCII - wyswietla rozszerzone kody ASCII */
/****************************************************/
#include <stdio.h>
#include ”test.h”
unsigned char uc; // zmienna sterujaca petli typu char
int fromASCII = 128, toASCII = 255;
long int x[10];
void main( void )
{
  struct data {
    int rok;
    int miesiac;
    int dzien;
  };
  data poczatek, koniec;
  int i;
  printf( ”Rozszerzone kody ASCII\n\n”);
  for ( uc = fromASCII; uc <= toASCII; uc1++ ) {
    printf( ”%3d:%2c”, uc, uc ); printf(”\n”);
```

```
  }
  int x1 = fromASCII + 2 * ( 20 +  toASCII );  /* int */
  double realTest = 12.34e-12 + .56 + 78.; /* double */
  x[0] = 1;
  for (i = 1; i < 10; i++) {
    x[i] = x[i-1] * i * i;
  }
  poczatek.rok = 2018;
  poczatek.miesiac = 10;
  poczatek.dzien = 1;
}
```

7 Test Data — File test2.c

```
/************************************************/
/* Program ASCII - wyswietla rozszerzone kody ASCII */
/************************************************/
#include <stdio.h>
#include ”test.h”
unsigned char uc; // zmienna sterujaca petli
int fromASCII = 128, toASCII = 255;
void main( void )
{
        printf(”\n\n\nRozszerzone kody ASCII\n\n”);
        for (uc = fromASCII; uc <= toASCII; uc1++)
        {
                printf(”%3d:%2c”, uc, uc);
        }
}
int x1 = fromASCII + 2 * ( 20 +  toASCII ); /* te linie /* sluza
* / wylacznie celom testowym ;-) */
double realTest = 12.34 + .56 + 78.;
*/ // nieotwarty komentarz
”Niezamknieta stala tekstowa
/* niezamkniety komentarz
```

8 Output of the Lexical Analyzer for test1.c

```
Author: First name and family name
yytext              Symbol type      Symbol value as string

Przetwarzanie dyrektywy #include <stdio.h>
Przetwarzanie dyrektywy #include ”test.h”
unsigned            KW_UNSIGNED
char                KW_CHAR
uc                  IDENT            uc
;                   ;
int                 KW_INT
fromASCII           IDENT            fromASCII
=                   =
128                 INTEGER_CONST    128
,                   ,
toASCII             IDENT            toASCII
=                   =
255                 INTEGER_CONST    255
;                   ;
long                KW_LONG
int                 KW_INT
x                   IDENT            x
[                   [
10                  INTEGER_CONST    10
```

```
]                        ]
;                        ;
void                     KW_VOID
main                     IDENT              main
(                        (
void                     KW_VOID
)                        )
{                        {
struct                   IDENT              struct
data                     IDENT              data
{                        {
int                      KW_INT
rok                      IDENT              rok
;                        ;
int                      KW_INT
miesiac                  IDENT              miesiac
;                        ;
int                      KW_INT
dzien                    IDENT              dzien
;                        ;
}                        }
;                        ;
data                     IDENT              data
poczatek                 IDENT              poczatek
,                        ,
koniec                   IDENT              koniec
;                        ;
int                      KW_INT
i                        IDENT              i
;                        ;
printf                   IDENT              printf
(                        (
"Rozszerzone kody ASSTRING_CONST            "Rozszerzone kody ASCII\n\n"
)                        )
;                        ;
for                      KW_FOR
(                        (
uc                       IDENT              uc
=                        =
fromASCII                IDENT              fromASCII
;                        ;
uc                       IDENT              uc
<=                       LE
toASCII                  IDENT              toASCII
;                        ;
uc1                      IDENT              uc1
++                       INC
)                        )
{                        {
printf                   IDENT              printf
(                        (
"%3d:%2c"                STRING_CONST       "%3d:%2c"
,                        ,
uc                       IDENT              uc
,                        ,
uc                       IDENT              uc
)                        )
;                        ;
printf                   IDENT              printf
(                        (
"\n"                     STRING_CONST       "\n"
)                        )
;                        ;
}                        }
int                      KW_INT
x1                       IDENT              x1
```

```
=             =
fromASCII     IDENT           fromASCII
+             +
2             INTEGER_CONST   2
*             *
(             (
20            INTEGER_CONST   20
+             +
toASCII       IDENT           toASCII
)             )
;             ;
double        KW_DOUBLE
realTest      IDENT           realTest
=             =
12.34e-12     FLOAT_CONST     12.34e-12
+             +
.56           FLOAT_CONST     .56
+             +
78.           FLOAT_CONST     78.
;             ;
x             IDENT           x
[             [
0             INTEGER_CONST   0
]             ]
=             =
1             INTEGER_CONST   1
;             ;
for           KW_FOR
(             (
i             IDENT           i
=             =
1             INTEGER_CONST   1
;             ;
i             IDENT           i
<             <
10            INTEGER_CONST   10
;             ;
i             IDENT           i
++            INC
)             )
{             {
x             IDENT           x
[             [
i             IDENT           i
]             ]
=             =
x             IDENT           x
[             [
i             IDENT           i
-             -
1             INTEGER_CONST   1
]             ]
*             *
i             IDENT           i
*             *
i             IDENT           i
;             ;
}             }
poczatek      IDENT           poczatek
.             .
rok           IDENT           rok
=             =
2018          INTEGER_CONST   2018
;             ;
poczatek      IDENT           poczatek
.             .
```

```
miesiac             IDENT            miesiac
=                   =
10                  INTEGER_CONST    10
;                   ;
poczatek            IDENT            poczatek
.                   .
dzien               IDENT            dzien
=                   =
1                   INTEGER_CONST    1
;                   ;
}                   }
```

9 Output of the Lexical Analyzer for test2.c

```
Author: First name and family name
yytext              Symbol type      Symbol value as string

Processing directive #include <stdio.h>
Processing directive #include "test.h"
unsigned            KW_UNSIGNED
char                KW_CHAR
uc                  IDENT            uc
;                   ;
int                 KW_INT
fromASCII           IDENT            fromASCII
=                   =
128                 INTEGER_CONST    128
,                   ,
toASCII             IDENT            toASCII
=                   =
255                 INTEGER_CONST    255
;                   ;
void                KW_VOID
main                IDENT            main
(                   (
void                KW_VOID
)                   )
{                   {
printf              IDENT            printf
(                   (
"\n\n\nRozszerzone kSTRING_CONST     "\n\n\nRozszerzone kody ASCII\n\n"
)                   )
;                   ;
for                 KW_FOR
(                   (
uc                  IDENT            uc
=                   =
fromASCII           IDENT            fromASCII
;                   ;
uc                  IDENT            uc
<=                  LE
toASCII             IDENT            toASCII
;                   ;
uc1                 IDENT            uc1
++                  INC
)                   )
{                   {
printf              IDENT            printf
(                   (
"%3d:%2c"           STRING_CONST     "%3d:%2c"
,                   ,
uc                  IDENT            uc
,                   ,
uc                  IDENT            uc
)                   )
;                   ;
}                   }
}                   }
int                 KW_INT
x1                  IDENT            x1
=                   =
fromASCII           IDENT            fromASCII
+                   +
2                   INTEGER_CONST    2
*                   *
(                   (
20                  INTEGER_CONST    20
+                   +
toASCII             IDENT            toASCII
)                   )
;                   ;
double              KW_DOUBLE
realTest            IDENT            realTest
=                   =
12.34               FLOAT_CONST      12.34
+                   +
.56                 FLOAT_CONST      .56
+                   +
78.                 FLOAT_CONST      78.
;                   ;
Unexpected closure of a comment in line 19
String opened in line 20 not closed in the same line
Missing comment closure for comment opened in line 21
```