Zajęcia P2. Analizator składniowy dla uproszczonego języka Turbo Pascal

1 Cel ćwiczeń

Celem ćwiczeń jest stworzenie prostego analizatora składniowego dla bardzo uproszczonej wersji języka Turbo Pascal. Zadaniem tworzonego analizatora jest:

- rozpoznawanie konstrukcji składniowych języka Turbo Pascal
- wykrywanie błędów składniowych

2 Czynności wstępne

Po włączeniu komputera należy wybrać system operacyjny Linux i w laboratorium zalogować się jako użytkownik `student`. Należy otworzyć okno konsoli (np. nacisnąć Alt+F2 i napisać `xterm`), utworzyć własny podkatalog za pomocą polecenia `mkdir` *nazwisko użytkownika*, i podkatalog dla bieżącego ćwiczenia. Ze strony przedmiotu na platformie eNauczanie należy pobrać następujące pliki:

- `Makefile` – potrzebny do kompilacji za pomocą polecenia `make`
- `common.h` – plik nagłówkowy zawierający określenie największej długości napisów
- `p.y` – szczątkowy analizator składniowy z komentarzami i zdefiniowaną funkcją `found`
- `test.pas` – program testowy poprawny w danej gramatyce

Po zakończeniu pracy wskazane jest usunięcie utworzonego katalogu wraz z zawartością.

3 Zadania do wykonania

Do wykonania zadania potrzebny jest analizator leksykalny przygotowany w poprzednim ćwiczeniu. W przypadku braków w analizatorze leksykalnym należy je uzupełnić. Należy uzupełnić dostarczony szkielet analizatora składniowego i pokazać, że działa poprawnie testując go na dostarczonych programach testowych. Analizator powinien wypisywać informacje o rozpoznanych konstrukcjach składniowych. Do wypisywania tych informacji służy zawarta w dostarczonym szkielecie analizatora funkcja `found()`. Funkcja ma dwa parametry: nazwa rozpoznanej konstrukcji (należy tu wpisać nazwę zmiennej gramatyki) oraz argument, który ma znaczenie (jest różny od pustego napisu) dla niektórych konstrukcji, np. może być nazwą funkcji. Należy dążyć do uzyskania takiego samego wyjścia programu jak w punkcie 6.

Dostarczony kod należy uzupełnić o następujące elementy:

A. nazwa programu (PROGRAM_NAME)

B. deklaracja stałej (CONST)

C. sekcja deklaracji stałych (CONST_SECT)

D. deklaracja zmiennych (VAR)

E. sekcja deklaracji zmiennych (VAR_SECT)

F. nagłówek funkcji/procedury (FUN_HEAD)

G. definicja procedury (PROCEDURE)

H. parametr formalny (FORM_PARAM)

I. blok (BLOCK)

J. definicja funkcji (FUNCTION)

K. parametr aktualny (ACT_PARAM)

L. wywołanie funkcji (FUNCT_CALL)

M. pętla for (FOR_INSTR)

N. przypisanie (ASSIGN_INSTR)

O. instrukcja warunkowa (IF_INSTR)

Analizator składniowy można uruchamiać przyrostowo. Przypuśćmy, że gdzieś na początku gramatyki mamy regułę:

```
A: B C D
;
```

Jeżeli ją tak zapiszemy, będziemy musieli rozwinąć także wszystkie zmienne po prawej stronie reguły. Jeżeli A jest symbolem początkowym gramatyki, musimy zapisać wszystkie reguły gramatyki. Nie wszystkim udaje się ukończyć cały analizator składniowy na zajęciach, a skoro analizator nie działa, dostają 0 punktów. Możliwe jest jednak pisanie reguł analizatora przyrostowo, punkt po punkcie. W regule rozwijającej zmienną A początkowo ujmujemy punkty C i D w komentarze.

```
A: B /* C D */
;
```

Teraz musimy rozwinąć tylko zmienną B i zmienne pojawiające się w jej rozwinięciu. Analizator się skompiluje i można go przetestować. Potem można przesunąć komentarz za zmienną C. Ujmowanie w komentarze jest dużo lepszym rozwiązaniem niż pomijanie reszty reguły, gdyż widać, że reguła ma dalsze, nie używane jeszcze części.

Kompilując częściowo zrealizowany analizator składniowy możemy napotkać na problemy związane z dyrektywą `%type` wskazującą zmienne, które nie zostały jeszcze rozwinięte w żadnej regule. Należy wówczas ująć dyrektywę w komentarze do czasu dopisania odp. reguł.

4 Ocena

Za każdy można dostać 1 punkt, czyli 15 punktów na zajęciach. Punkty będą przyznawane dopiero po rozmowie z prowadzącym.

5 Dane testowe — plik test.pas

```
Program Testing;
(*Uses crt, dos;*)
Const (* range of displayed characters *)
        minASCII = 30;
        maxASCII = 255;
        tekst = 'test string';
Var
   c : Char;
   r : real;
   i, i1, _i, _00 : Integer;
   t : array[1..10] of integer;
   d :  record
           year, month : integer;
           day         :  integer
        end;

Procedure Empty_Without_Parameters;
Begin
End;

Function Empty_With_Parameters( a : Integer, c : Char, r : Real ) : Integer;
Begin
End;

Procedure With_Declarations;
Const
   r1 = 12.34;
   r2 = 0.56;
   r3 = 78.0;
Var
```

```
31    s : String;
32    t : array[1..10] of integer;
33    d :  record
34            year, month : integer;
35            day         :  integer
36         end;
37 Begin
38 End;
39
40
41 Begin (* main block *)
42    Empty_Without_Parameters;
43    Empty_With_Parameters( 123, 'c', 12.34 );
44    ClrScr; (* intro opn clear screen *)
45    Writeln( 'Kody ASCII (30-255):' );
46    For i := minASCII To maxASCII Do (* display of given ASCII codes *)
47       Write( Chr( i ), '   ' );
48    ReadKey; (* wait for a key press *)
49    i := ( i1 + 3 ) * _00;
50    (* conditional instruction *)
51    if  a > 10
52       then
53       b := a;
54    if ( a > 1 )
55       then
56       b := a
57    else
58       b := 1;
59    if ( a > b )
60       then
61       if ( a > c )
62          then
63          m := a
64       else
65          m := c
66       else
67          if ( b > c )
68             then
69             m := b
70          else
71             m := c;
72    t[10] := 1;
73    for i := 9 downto 1 do t[i] := t[i+1] * i * i;
74    d.year := 2018;
75    d.day := 1;
76    d.month := d.day * 10;
77 End.
```

6 Wyjście analizatora składniowego dla test.pas

```
1  Author: First and last name
2  yytext              Token type      Token value as string
3
4  Program             KW_PROGRAM
5  Testing             IDENT           Testing
6  ;                   ;
7  ===== FOUND: PROGRAM_NAME 'Testing'=====
8  Const               KW_CONST
9  minASCII            IDENT           minASCII
10 =                   =
11 30                  INTEGER_CONST   30
12 ===== FOUND: CONST 'minASCII'=====
13 ;                   ;
14 maxASCII            IDENT           maxASCII
```

```
=                       =
255                     INTEGER_CONST   255
===== FOUND: CONST 'maxASCII'=====
;                       ;
tekst                   IDENT           tekst
=                       =
'test string'           STRING_CONST    'test string'
===== FOUND: CONST 'tekst'=====
;                       ;
Var                     KW_VAR
===== FOUND: CONST_SECT =====
c                       IDENT           c
:                       :
Char                    KW_CHAR
===== FOUND: VAR =====
;                       ;
r                       IDENT           r
:                       :
real                    KW_REAL
===== FOUND: VAR =====
;                       ;
i                       IDENT           i
,                       ,
i1                      IDENT           i1
,                       ,
_i                      IDENT           _i
,                       ,
_00                     IDENT           _00
:                       :
Integer                 KW_INTEGER
===== FOUND: VAR =====
;                       ;
t                       IDENT           t
:                       :
array                   KW_ARRAY
[                       [
1                       INTEGER_CONST   1
..                      RANGE
10                      INTEGER_CONST   10
]                       ]
of                      KW_OF
integer                 KW_INTEGER
===== FOUND: VAR =====
;                       ;
d                       IDENT           d
:                       :
record                  KW_RECORD
year                    IDENT           year
,                       ,
month                   IDENT           month
:                       :
integer                 KW_INTEGER
;                       ;
day                     IDENT           day
:                       :
integer                 KW_INTEGER
end                     KW_END
===== FOUND: VAR =====
;                       ;
Procedure               KW_PROCEDURE
===== FOUND: VAR_SECT =====
Empty_Without_ParameIDENT               Empty_Without_Parameters
;                       ;
===== FOUND: FUN_HEAD 'Empty_Without_Parameters'=====
Begin                   KW_BEGIN
End                     KW_END
```

```
===== FOUND: BLOCK =====
===== FOUND: PROCEDURE 'Empty_Without_Parameters'=====
;                    ;
Function             KW_FUNCTION
Empty_With_ParameterIDENT            Empty_With_Parameters
(                    (
a                    IDENT           a
:                    :
Integer              KW_INTEGER
===== FOUND: FORM_PARAM =====
,                    ,
c                    IDENT           c
:                    :
Char                 KW_CHAR
===== FOUND: FORM_PARAM =====
,                    ,
r                    IDENT           r
:                    :
Real                 KW_REAL
===== FOUND: FORM_PARAM =====
)                    )
===== FOUND: FUN_HEAD 'Empty_With_Parameters'=====
:                    :
Integer              KW_INTEGER
;                    ;
Begin                KW_BEGIN
End                  KW_END
===== FOUND: BLOCK =====
===== FOUND: FUNCTION 'Empty_With_Parameters'=====
;                    ;
Procedure            KW_PROCEDURE
With_Declarations    IDENT           With_Declarations
;                    ;
===== FOUND: FUN_HEAD 'With_Declarations'=====
Const                KW_CONST
r1                   IDENT           r1
=                    =
12.34                FLOAT_CONST     12.34
===== FOUND: CONST 'r1'=====
;                    ;
r2                   IDENT           r2
=                    =
0.56                 FLOAT_CONST     0.56
===== FOUND: CONST 'r2'=====
;                    ;
r3                   IDENT           r3
=                    =
78.0                 FLOAT_CONST     78.0
===== FOUND: CONST 'r3'=====
;                    ;
Var                  KW_VAR
===== FOUND: CONST_SECT =====
s                    IDENT           s
:                    :
String               KW_STRING
===== FOUND: VAR =====
;                    ;
t                    IDENT           t
:                    :
array                KW_ARRAY
[                    [
1                    INTEGER_CONST   1
..                   RANGE
10                   INTEGER_CONST   10
]                    ]
of                   KW_OF
```

```
integer                 KW_INTEGER
===== FOUND: VAR =====
;                       ;
d                       IDENT           d
:                       :
record                  KW_RECORD
year                    IDENT           year
,                       ,
month                   IDENT           month
:                       :
integer                 KW_INTEGER
;                       ;
day                     IDENT           day
:                       :
integer                 KW_INTEGER
end                     KW_END
===== FOUND: VAR =====
;                       ;
Begin                   KW_BEGIN
===== FOUND: VAR_SECT =====
End                     KW_END
===== FOUND: BLOCK =====
===== FOUND: PROCEDURE 'With_Declarations'=====
;                       ;
Begin                   KW_BEGIN
Empty_Without_ParameIDENT           Empty_Without_Parameters
;                       ;
===== FOUND: FUNCT_CALL 'Empty_Without_Parameters'=====
Empty_With_ParameterIDENT           Empty_With_Parameters
(                       (
123                     INTEGER_CONST   123
===== FOUND: ACT_PARAM =====
,                       ,
'c'                     STRING_CONST    'c'
===== FOUND: ACT_PARAM =====
,                       ,
12.34                   FLOAT_CONST     12.34
===== FOUND: ACT_PARAM =====
)                       )
===== FOUND: FUNCT_CALL 'Empty_With_Parameters'=====
;                       ;
ClrScr                  IDENT           ClrScr
;                       ;
===== FOUND: FUNCT_CALL 'ClrScr'=====
Writeln                 IDENT           Writeln
(                       (
'Kody ASCII (30-255)STRING_CONST    'Kody ASCII (30-255):'
===== FOUND: ACT_PARAM =====
)                       )
===== FOUND: FUNCT_CALL 'Writeln'=====
;                       ;
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
)                       )
===== FOUND: FUNCT_CALL 'i'=====
===== FOUND: ACT_PARAM =====
```

```
===== FOUND: FUNCT_CALL 'Chr'=====
===== FOUND: ACT_PARAM =====
,                       ,
'   '                   STRING_CONST    '   '
===== FOUND: ACT_PARAM =====
)                       )
===== FOUND: FUNCT_CALL 'Write'=====
===== FOUND: FOR_INSTR =====
;                       ;
ReadKey                 IDENT           ReadKey
;                       ;
===== FOUND: FUNCT_CALL 'ReadKey'=====
i                       IDENT           i
:=                      ASSIGN
(                       (
i1                      IDENT           i1
+                       +
3                       INTEGER_CONST   3
)                       )
*                       *
_00                     IDENT           _00
;                       ;
===== FOUND: ASSIGN_INSTR 'i'=====
if                      KW_IF
a                       IDENT           a
>                       >
10                      INTEGER_CONST   10
then                    KW_THEN
b                       IDENT           b
:=                      ASSIGN
a                       IDENT           a
;                       ;
===== FOUND: ASSIGN_INSTR 'b'=====
===== FOUND: IF_INSTR =====
if                      KW_IF
(                       (
a                       IDENT           a
>                       >
1                       INTEGER_CONST   1
)                       )
then                    KW_THEN
b                       IDENT           b
:=                      ASSIGN
a                       IDENT           a
else                    KW_ELSE
===== FOUND: ASSIGN_INSTR 'b'=====
b                       IDENT           b
:=                      ASSIGN
1                       INTEGER_CONST   1
;                       ;
===== FOUND: ASSIGN_INSTR 'b'=====
===== FOUND: IF_INSTR =====
if                      KW_IF
(                       (
a                       IDENT           a
>                       >
b                       IDENT           b
)                       )
then                    KW_THEN
if                      KW_IF
(                       (
a                       IDENT           a
>                       >
c                       IDENT           c
)                       )
then                    KW_THEN
```

```
m                       IDENT           m
:=                      ASSIGN
a                       IDENT           a
else                    KW_ELSE
===== FOUND: ASSIGN_INSTR 'm'=====
m                       IDENT           m
:=                      ASSIGN
c                       IDENT           c
else                    KW_ELSE
===== FOUND: ASSIGN_INSTR 'm'=====
===== FOUND: IF_INSTR =====
if                      KW_IF
(                       (
b                       IDENT           b
>                       >
c                       IDENT           c
)                       )
then                    KW_THEN
m                       IDENT           m
:=                      ASSIGN
b                       IDENT           b
else                    KW_ELSE
===== FOUND: ASSIGN_INSTR 'm'=====
m                       IDENT           m
:=                      ASSIGN
c                       IDENT           c
;                       ;
===== FOUND: ASSIGN_INSTR 'm'=====
===== FOUND: IF_INSTR =====
===== FOUND: IF_INSTR =====
t                       IDENT           t
[                       [
10                      INTEGER_CONST   10
]                       ]
:=                      ASSIGN
1                       INTEGER_CONST   1
;                       ;
===== FOUND: ASSIGN_INSTR 't'=====
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
;                       ;
===== FOUND: ASSIGN_INSTR 't'=====
===== FOUND: FOR_INSTR =====
d                       IDENT           d
.                       .
year                    IDENT           year
```

```
:=                      ASSIGN
2018                    INTEGER_CONST    2018
;                       ;
===== FOUND: ASSIGN_INSTR 'd'=====
d                       IDENT            d
.                       .
day                     IDENT            day
:=                      ASSIGN
1                       INTEGER_CONST    1
;                       ;
===== FOUND: ASSIGN_INSTR 'd'=====
d                       IDENT            d
.                       .
month                   IDENT            month
:=                      ASSIGN
d                       IDENT            d
.                       .
day                     IDENT            day
*                       *
10                      INTEGER_CONST    10
;                       ;
===== FOUND: ASSIGN_INSTR 'd'=====
End                     KW_END
===== FOUND: BLOCK =====
.                       .
===== FOUND: Complete program =====
```