Zajęcia P1. Analizator leksykalny dla uproszczonego języka Turbo Pascal

1 Cel ćwiczeń

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

- rozpoznawanie elementów języka Turbo Pascal i określanie ich wartości
- usuwanie białych znaków i komentarzy
- wykrywanie wskazanych dyrektyw
- wykrywanie błędów leksykalnych

2 Czynności wstępne

Po włączeniu komputera należy wybrać system operacyjny Linux i 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 Moodle dla tematu *Analiza leksykalna* należy pobrać pliki dla języka Turbo Pascal. Znajdują się tam następujące pliki:

- `p1p.pdf` — instrukcja (właśnie czytany plik)
- `Makefile` — potrzebny do kompilacji za pomocą polecenia `make`
- `common.h` — plik nagłówkowy zawierający określenie największej długości napisów
- `p.l` — szkielet analizatora leksykalnego, który należy uzupełnić; należy zwrócić uwagę na definicję funkcji `process_token()`, którą należy wykorzystać
- `p.y` — analizator składniowy, którego jedynymi zadaniami jest deklaracja rozpoznawanych elementów końcowych oraz wywołanie analizatora leksykalnego
- `test1.pas` — program testowy poprawny w danej gramatyce
- `test2.pas` — program testowy z błędami, które powinny zostać wykryte

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

3 Zadania do wykonania

Należy uzupełnić dostarczony szkielet analizatora leksykalnego i pokazać, że działa poprawnie testując do na dostarczonych programach testowych. Analizator powinien wypisywać informacje o rozpoznanych symbolach końcowych w trzech kolumnach:

1. dopasowany tekst
2. rozpoznany symbol
3. wartość symbolu (tylko w sytuacji, gdy symbol rzeczywiście ma wartość)

Do wypisywania tych informacji służy zawarta w dostarczonym szkielecie analizatora funkcja `process_token()`. Funkcja zwraca rozpoznany symbol, dlatego w działaniu dla reguły rozpoznającej symbol powinna znaleźć się instrukcja `return process_token(. . . )` z odpowiednimi parametrami funkcji.

Dostarczony kod należy uzupełnić o następujące elementy:

A. wypisanie własnego imienia i nazwiska (w pliku dla programu `bison`)

B. wykrywanie słów kluczowych zdefiniowanych w pliku źródłowym dla programu `bison` (uwaga: Pascal nie rozróżnia wielkości liter)

C. usuwanie białych znaków

D. wykrywanie operatorów wieloznakowych (`<=`, `:=`,...) występujących w programach testowych

E. wykrywanie identyfikatorów

F. wykrywanie liczb całkowitych

G. wykrywanie liczb rzeczywistych

H. wykrywanie stałych tekstowych (napisów) bez użycia mechanizmu warunków początkowych

I. wykrywanie symboli końcowych jednoznakowych: operatorów, interpunkcji

J. wykrywanie dyrektywy dołączania plików

K. wykrywanie napisów z użyciem warunków początkowych

L. usuwanie komentarzy wielowierszowych typu { z użyciem warunków początkowych

M. usuwanie komentarzy wielowierszowych typu `(*` z użyciem warunków początkowych

N. znajdowanie znaków zamknięcia komentarza przy braku jego rozpoczęcia z użyciem warunków początkowych

O. wykrywanie niezamkniętego komentarza ze wskazaniem wiersza jego rozpoczęcia z użyciem warunków początkowych

4 Ocena

Za każdy element można dostać 1 punkt, czyli razem 15 punktów. Jeżeli ktoś nie zdąży zrobić wszystkich elementów na zajęciach, istnieje możliwość dokończenia analizatora w domu, ale **tylko elementów od K do O** i za każdy element **w domu przysługuje tylko połowa punktów**. Plik wykonany na zajęciach umieścić **na zajęciach** na platformie Moodle. UWAGA: **Analizator leksykalny potrzebny będzie na następnych zajęciach**.

5 Przypomnienie warunków początkowych

- Warunek początkowy aktywny na początku programu: `INITIAL`
- Deklaracja (w pierwszej części): `%x warunek1, warunek2,...`
- Dopasowanie w określonym stanie:
  ```
  <war1> re1      działanie1;
  <war1,war2,INITIAL>re2 działanie2;
  <*>re3        działanie3
  ```
- zmiana warunku początkowego: `BEGIN warunek4`
- bieżący warunek początkowy: `YY_START`
- sprawdzenie bieżącego warunku po odczytaniu wszystkich danych wejściowych: w funkcji `yywrap`, którą należy zdefiniować i która musi zwrócić wartość 1

6 Dane testowe — plik test1.pas

```
1  Program ASCII; (* Wyswietla kody ASCII *)
2  Uses
3     crt, dos;
4  {$I MyFile.inc}
5  Var
6     i : Integer;
7     c : Char;
8     r : real;
9     t : array[1..10] of integer;
10    d :  record
11            rok, miesiac : integer;
```

```
12            dzien        :  integer;
13          end;
14 Const   (* zakres wyswietlanych znakow *)
15    minASCII = 30;
16    maxASCII = 255;
17 Begin
18    ClrScr(); (* intro na czystym ekranie *)
19    Write('Kody ASCII od 30 do 255: '); WriteLn('(po 20 w wierszu):');
20    For i := minASCII To maxASCII Do (* wyswietlenie zadanych kodow ASCII *)
21       Write( Chr( i ) : 4 );
22    ReadKey; (* czekaj na nacisniecie klawisza *)
23    r := 12.34e-12 * ( 56.0 + 0.78 ); { test liczb rzeczywistych }
24    i := minASCII + 2 * (20 + maxASCII );
25    t[10] := 1;
26    for i := 9 downto 1 do t[i] := t[i+1] * i * i;
27    d.rok := 2018;
28    d.dzien := 1;
29    d.miesiac := d.dzien * 10;
30 End.
```

7 Dane testowe — plik test2.pas

```
1  Program ASCII; (* Wyswietla kody ASCII *)
2  Uses
3          crt, dos;
4  {$I MyFile.inc}
5  Var
6          i : Integer;
7          c : Char;
8          r : real;
9  Const   (* zakres wyswietlanych znakow *)
10         minASCII = 30;
11         maxASCII = 255;
12 Begin
13         ClrScr(); (* intro na czystym ekranie *)
14         Writeln( 'Kody ASCII od 30 do 255: (po 20 w wierszu):' );
15         For i := minASCII To maxASCII Do (* wyswietlenie zadanych kodów ASCII *)
16                 Write( Chr( i ) : 4 );
17         ReadKey; (* czekaj na nacisniecie klawisza *)
18         r := 12.34 * ( 56.0 + 0.78 ); { test liczb rzeczywistych }
19         i := minASCII + 2 * (20 + maxASCII );
20         *) { nieotwarty komentarz }
21         } { nieotwarty komentarz }
22         { komentarz
23         wielowierszowy 1 }
24         (* komentarz
25         wielowierszowy 2 *)
26         { niezamkniety komentarz ...
27 End.
```

8 Wyjście analizatora leksykalnego dla test1.pas

```
1  Autor: Imie i Nazwisko
2  yytext              Typ elementu     Wartość elementu znakowo
3
4  Program             KW_PROGRAM
5  ASCII               IDENT            ASCII
6  ;                   ;
7  Uses                KW_USES
8  crt                 IDENT            crt
9  ,                   ,
```

```
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
;                       ;
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 Wyjście analizatora leksykalnego dla 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
*                    *
(                    (
56.0                 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
)                    )
;                    ;
Nieoczekiwane zamknięcie komentarza w wierszu 20
Nieoczekiwane zamknięcie komentarza w wierszu 21
Niezamknięty komentarz otwarty w wierszu 26
```