Programski prevodioci (vežbe)

Generatori leksičkih analizatora

Razvoj generatora leksičkih analizatora

- Lex – prvi generator leksičkih analizatora nastao u Bell laboratorijama 1975. kao deo operativnog sistema UNIX. On proizvodi skener u C programskom jeziku.
- Na Prinston univerzitetu 1996. godine razvijen je alat JLex, koji generiše izvorni kod leksičkih analizatora u programskom jeziku Java.
- JFlex je modifikacija i nadgradnja funkcionalnosti JLex-a.

Koraci u razvoju leksičkog analizatora pomoću JFlexa-a

- Kreiranje lex specifikacije koja sadrži definicje regularnih izraza reči koje se prepoznaju.
- Kreiranje Java klase Yytoken čije instance vraća leksički analizator kao rezultat svog rada.
- Kreiranje Java klase koja sadrži definicije celobrojnih identifikatora tokena.
- Generisanje java koda na osnovu kreirane specifikacije.

Java klasa koja sadrži definicije celobrojnih identifikatora tokena jezika µPascal

Vrste reči:

- Ključne reči:
  - program
  - var
  - integer
  - char
  - begin
  - end
  - read
  - write
  - if
  - then
  - else

```
public class sym
{
  public final static int EOF = 0;
  public final static int PROGRAM = 1;
  public final static int VAR = 2;
  public final static int INTEGER = 3;
  public final static int CHAR = 4;
  public final static int BEGIN = 5;
  public final static int END = 6;
  public final static int READ = 7;
  public final static int WRITE = 8;
  public final static int IF = 9;
  public final static int THEN = 10;
  public final static int ELSE = 11;
```

Java klasa koja sadrži definicije celobrojnih identifikatora tokena jezika μPascal

- Identifikatori
- Konstante
- Operatori:
  - +
  - *
  - :=
- Separatori:
  - (
  -)
  - ;
  - ,
  - .
  - :

```
    public final static int ID = 12;
    public final static int CONST = 13;
    public final static int PLUS = 14;
    public final static int MUL = 15;
    public final static int ASSIGN = 16;
    public final static int LEFTPAR = 17;
    public final static int RIGHTPAR = 18;
    public final static int SEMICOLON = 19;
    public final static int COMMA = 20;
    public final static int DOT = 21;
    public final static int COLON = 22;
}
```

Sadržaj klase Yytoken

- Klasa “Yytoken” obično sadrži:
  - Izdvojenu reč,
  - Njeno značenje (celobrojni identifikator značenja),
  - Liniju koda iz koje je reč izdvojena,
  - Početna pozicija u liniji od koje je reč izdvojena,
  - Pozicija u liniji na kojoj se izdvojena reč završava,
  - Dodatni atrubut simbola…

Implementacija klase Yytoken

```
class Yytoken
{
  public int m_index;
  public String m_text;
  public int m_line;
  public int m_charBegin;
  public Object value;

  Yytoken (int index, String text, int line,
    int charBegin)
  {
    m_index = index;
    m_text = text;
    m_line = line;
    m_charBegin = charBegin;
  }
```

Implementacija klase Yytoken

```
  public String toString() {
    return "Text    : " + m_text +
           "\nindex : " + m_index +
           "\nline  : " + m_line +
           "\ncBeg. : " + m_charBegin;
  }
}
```

Format .jflex specifikacije

Korisnički kod

%%

Opcije i deklaracije

%%

Leksička pravila

Korisnički kod

- Skup import i package iskaza koji se direktno prepisuju na početak generisanog fajla bez provere njihove sintaksne ispravnosti.

Opcije i deklaracije

Opcije i deklaracije sadrže:

- Opcije za podešavanje generisanog leksera:
  - JFlex direktive i
  - Java kod koji se uključuje u različite delove leksera.
- Deklaracije posebnih početnih stanja
- Definicije makroa

Najčešće korišćene JFlex direktive

Opcije kojima se podešavaju paramemetri generisane klase:

- **`%class <ImeKlase>`**

  Definiše ime generisane klase – po *default*-u ime generisane klase je **`yylex`**.

- **`%implements <intrefejs1>[,<interfejs2>][,…]`**

  Ukoliko je ova direktiva navedena generisana klasa implementira navedene interfejse.

- **`%extends <ImeKlase>`**

  Ukoliko je ova direktiva navedena generisana klasa nasledjuje navedenu klasu.

Najčešće korišćene JFlex direktive

Opcije kojima se podešavaju paramemetri generisane funkcije za leksičku analizu:

- **`%function <ImeFunkcije>`**

  Definiše ime generisane funkcije koja vrši izdvajanje jedne reči – po default-u ime generisane funkcije je **`yylex()`**.

- **`%int`** ili **`%integer`**

  Postavlja povratni tip funkcije koja vrši leksičku analizu na tip int.

- **`%type <ImeTipa>`**

  Postavlja povratni tip funkcije koja vrši leksičku analizu na navedeni tip.

Ukoliko ni jedna od prethodne dve direktive nije navedena, podrazumenvani povratni tip leksičkog analizatora je Yytoken.

Najčešće korišćene JFlex direktive

Opcije kojima se uključuju brojači linija i karakatera:

- **`%line`**

  Uključuje brojač linija u izvornom fajlu. Generiše se promenljiva **`yyline`** (tipa **`int`**) koja pamti redni broj tekuće linije (brojanje počinje od 0).

- **`%char`**

  Uključuje brojač karaktera u izvornom fajlu. Generiše se promenljiva **`yychar`** (tipa **`int`**) koja pamti redni broj prvog karaktera izdvojene reči (brojanje počinje od 0).

- **`%column`**

  Uključuje brojač karaktera u tekućoj liniji. Generiše se promenljiva **`yycolumn`** (tipa **`int`**) koja pamti redni broj prvog karaktera izdvojene reči u tekućoj liniji (brojanje počinje od 0).

Najčešće korišćene JFlex direktive

Opcije kojima se nalaže generisanje samostalne aplikacije (tj. generisanje main metoda):

- **`%debug`**

  Generiše se main metod koji poziva leksički analizator sve dok se ne dodje do kraja ulaznog fajla. Na standardni izlaz (tj. u Java konzoli) ispisuju se svi izdvojeni tokeni kao i kod akcije koja je tom prilikom izvršena.

- **`%standalone`**

  Generiše se main metod koji poziva leksički analizator sve dok se ne dodje do kraja ulaznog fajla. Ispravno prepoznati tokeni se ignorišu, a na standardni izlaz se ispisuju neprepoznati delovi ulaznog fajla.

Definicje delova koda koji se umeću u generisanu klasu

- **`%{`**

  `<code>`

  `%}`

  Navedeni deo koda se ugradjuje u generisanu klasu. Tu se definišu dodatni atributi i metodi klase.
- **`%init{`**

  `<code>`

  `%init}`

  Navedeni deo koda se ugradjuje u konstruktor generisane klase.
- **`%eofval{`**

  `<code>`

  `%eofval}`

  Navedeni deo koda se ugradjuje u generisanu funkciju koja vrši leksičku analizu na mestu koje odgovara akciji kada se prepozna EOF. Ovaj deo koda treba da vrati vrednost koja signalizira da se došlo do kraja koda koji se analizira.

Definicije posebnih početnih stanja

- Prilikom prepoznavanja svake nove reči, polazi se od unapred definisanog početnog stanja konačnog automata.
- Po default-u, početno stanje generisanog konačnog automata je YYINITIAL.
- Nekada se javlja situacija da iste reči imaju različita značenja zavisno od konteksta u kojem su upotrebljene.
- Da bi se ta specijalna značenja pojedinih reči prepoznala uvode se nova početna stanja konačnog automata koji prepoznaje reči jezika.

Vrste posebnih početnih stanja

- Inclusive states:

  `%s[tate] <StateName> [, <StateName>,…]`

  Ova početna stanja ne moraju biti potpuno definisana –
  tj. počev od ovih stanja samo pojedine reči se tumače na
  poseban način, dok se tumačenja ostalih reči
  preuzimaju iz default-nog početnog stanja.

- Exclusive states:

  `%x[state] <StateName> [, <StateName>,…]`

  Ova početna stanja moraju biti potpuno definisana –
  tj. kada analiza počinje od nekog od ovih stanja, sve reči jezika
  imaju drugačija značenja.

Definisanje makroa

- Makroi definisani u ovom delu koriste se u definiciji leksičkih pravila. Format za definisanje makroa:

```
<ImeMakroa> = <RegularniIzraz>
```

Leksička pravila

- Za svako završno stanje u konačnom automatu kreira se po jedno pravilo oblika:

```
[<NizStanja>]<Uzorak> { <Akcija> }
```

Gde:

- **<NizStanja>** definiše početna stanja iz kojih se navedeni uzorak prepoznaje. Ukoliko je niz stanja izostavljen, uzorak se odnosi na početno stanje YYINITIAL. Niz stanja se navodi u sledećem formatu:

```
< <ImeStanja1>[,<ImeStanja2>,…]>
```

- **<Uzorak>** predstavlja regularni izraz reči koja se ovim uzorkom izdvaja.
- **<Akcija>** deo Java koda koji se izvršava kada se navedeni uzorak prepozna.

Specijalni znaci (metakarakteri) koji se koriste u definisanju regularnih izraza

znak	značenje
*	Prethodni znak ili grupa se ponavlja nijednom ili više puta.
|	Izbor alternative (bira se između simbola (ili grupe) koji prethodi i simbola (ili grupe) koji sledi.
+	Prethodni znak ili grupa se ponavlja jednom ili više puta. a+ <=> aa*
?	Prethodni znak se može, ali ne mora pojaviti. a? <=> a | ε
{n}	Prethodni znak ili grupa se ponavlja tačno n puta. a{3} <=> aaa
{n,m}	Prethodni znak ili grupa se ponavlja minimalno n, a maksimalno m puta. a{2,4} <=> aa | aaa | aaaa

Specijalni znaci (metakarakteri) koji se koriste u definisanju regularnih izraza

znak	značenje
{name}	Poziv makroa (makro je kraći simbolički naziv nekog regularnog izraza koji je ranije definisan). Makro je npr. cifra = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 a {cifra}+ je poziv tog makroa koji definiše dekadni ceo broj.
[]	Alternativa – izbor jednog od navedenih simbola. [0123456789] <=> 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 Unutar [] **ne sme** se koristiti **|** ili **,** za nabrajanje.
-	Sa prethodnim i narednim znakom definiše opseg simbola i može se koristiti samo unutar alternative. [0-9] <=> [0123456789] [0-9A-F] <=> [0123456789ABCDEF]

Specijalni znaci (metakarakteri) koji se koriste u definisanju regularnih izraza

znak	značenje
^	Unutar zagrada [] - koristi se na početku alternative i označava negaciju skupa znakova koji sledi – vrši se izbor jednog od simbola koji nisu navedeni u alternativi. [^0-9] – bilo koji znak osim dekadnih cifara
^	Van zagrada [] - ukoliko se nalazi na početku regularnog izraza, označava da se taj regularni izraz primenjuje samo za izdvajanje podstringova (particija) sa početka linije.
$	Koristi se na kraju regularnog izraza i označava da se taj regularni izraz primenjuje samo za prepoznavanje podnizova koji se nalaze na kraju linije.
()	Grupisanje simbola
\	Poništava specijalno značenje metakaraktera koji sledi.
" "	Poništavaju specijalna dejstva svih metakaraktera između.

Specijalni znaci (metakarakteri) koji se koriste u definisanju regularnih izraza

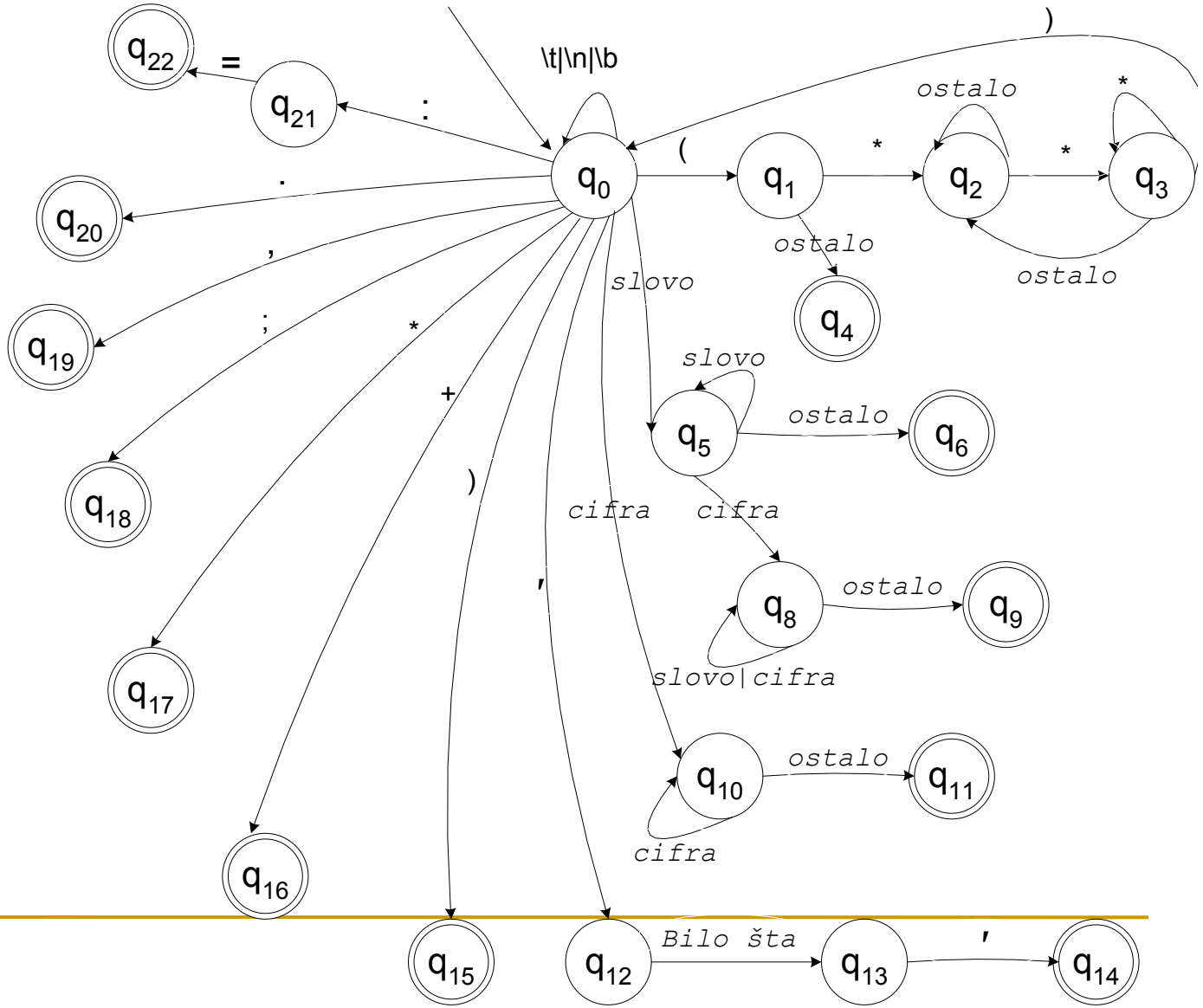
znak	značenje
!	Negira znak koji sledi - na toj poziciji se može pojaviti svaki znak osim navedenog. !a <=> [^a]
~	Prihvata sve simbole do pojave znaka ili grupe koja sledi, uključujući i taj znak/grupu.
.	Zamenjuje bilo koji znak. . <=> [^]

Funkcije generisanog lexer-a

funkcija	dejstvo
String yytext()	Vraća poslednju prepoznatu reč.
void yybegin(int state)	Postavlja novo početno stanje.
int yybegin()	Vraća trenutno početno stanje.
int yylength()	Vraća dužinu izdvojene reči.

Zadatak: Kreirati JFlex specifikaciju za generisanje leksičkog analizatora jezika μPascal

Graf prelaza odgovarajućeg konačnog automata:



JFlex specifikacija

```
// import sekcija
package compiler;
%%
// sekcija opcija i deklaracija
%class MPLexer
%function next_token
%line
%column

%debug

%eofval{
  return new Yytoken( sym.EOF, null, yyline, yycolumn);
%eofval}
```

JFlex specifikacija

```
%{
   // dodatni članovi generisane klase
   KWTable kwTable = new KWTable();
   Yytoken getKW()
   {
      return new Yytoken( kwTable.find( yytext() ),
         yytext(), yyline, yycolumn );
   }
%}
```

JFlex specifikacija

```
// dodatna početna stanja
%xstate COMMENT

// makroi
slovo = [a-zA-Z]
cifra = [0-9]

%%
// sekcija leksičkih pravila
\(\*          { yybegin( COMMENT ); }
<COMMENT>\*\)  { yybegin( YYINITIAL ); }
<COMMENT>.     { ; }
[\t\n ]        { ; }
\(             { return new Yytoken( sym.LEFTPAR,
                  yytext(), yyline, yycolumn ); }
\)             { return new Yytoken( sym.RIGHTPAR,
                  yytext(), yyline, yycolumn ); }
```

Jflex specifikacija

```
//operatori
\+          { return new Yytoken( sym.PLUS,
               yytext(), yyline, yycolumn ); }
\*          { return new Yytoken( sym.MUL,
               yytext(), yyline, yycolumn ); }
//separatori
;           { return new Yytoken( sym.SEMICOLON,
               yytext(), yyline, yycolumn ); }
,           { return new Yytoken( sym.COMMA,
               yytext(), yyline, yycolumn ); }
\.          { return new Yytoken( sym.DOT,
               yytext(), yyline, yycolumn ); }
:=          { return new Yytoken( sym.ASSIGN,
               yytext(), yyline, yycolumn ); }
```

Jflex specifikacija

```
// ključne reči
{slovo}+    { return getKW(); }
// identifikatori
{slovo}({slovo}|{cifra})* { return new Yytoken(
                    sym.ID, yytext(),
                    yyline, yycolumn );
// konstante
{cifra}+    { return new Yytoken( sym.CONST,
                yytext(), yyline, yycolumn ); }
'[^]'       { return new Yytoken( sym.CONST,
                yytext(), yyline, yycolumn ); }
// obrada leksičkih grešaka
.     { System.out.writeln( "ERROR: " + yytext() ); }
```