Skener

- Skener je zadužen za leksičku analizu
- Skener preuzima (skenira) znak po znak (programskog) teksta radi prepoznavanja simbola sastavljenih od zadanih znakova ili njihovih sekvenci. Pri tome:
  - ignoriše znakove koji razdvajaju simbole (delimiteri ili separatori, npr. prazno mesto, tab, *newline*) i
  - reaguje na nedozvoljene znakove ili njihove sekvence tako što prijavljuje leksičku grešku
- Skener je obično potprogram parsera i kada ga parser pozove on prepozna jedan simbol i isporučuje ga parseru (radi sintaksne analize)

Skener

- Za parser je zgodnije da mu skener umesto simbola (konkretnog niza znakova) isporuči numeričku oznaku vrste/grupe simbola koja se zove **token** (*token*)
  - za tačku (".") je dovoljno da parser dobije samo token - **`DOT`**
  - za reči (`word`) je potrebno da parser osim tokena za reč - **`WORD`** dobije i string te reči, da bi mogao da razlikuje reči
- Znači, u opštem slučaju skener isporučuje parseru **dve vrednosti** – token i vrednost simbola

Skener – simboli tekst gramatike

SIMBOL	TOKEN	VREDNOST
“.”	_DOT	-
capital_letter {small_letter}	_CAPITAL_WORD	string
small_letter {small_letter}	_WORD	string

- Definicija tokena tekst gramatike:

```
#define  _DOT           1
#define  _CAPITAL_WORD  2
#define  _WORD          3
```

Skener – dijagram prelaza

- Potreban sistematičan način prepoznavanja simbola
- Generalni – opšti skener:
  - preuzme opis simbola i na osnovu njega
  - prepoznaje i klasifikuje simbole
- Opisivanje simbola omogućuje **gramatika simbola** (*patterns*): `word`, `capital_word`, `dot` (formalna gramatika za običan tekst)
- Ona se može prikazati u grafičkom obliku pomoću **dijagrama prelaza** *(transition diagram)* koji daje osnovu za sistematičan način prepoznavanja simbola

Skener – dijagram prelaza

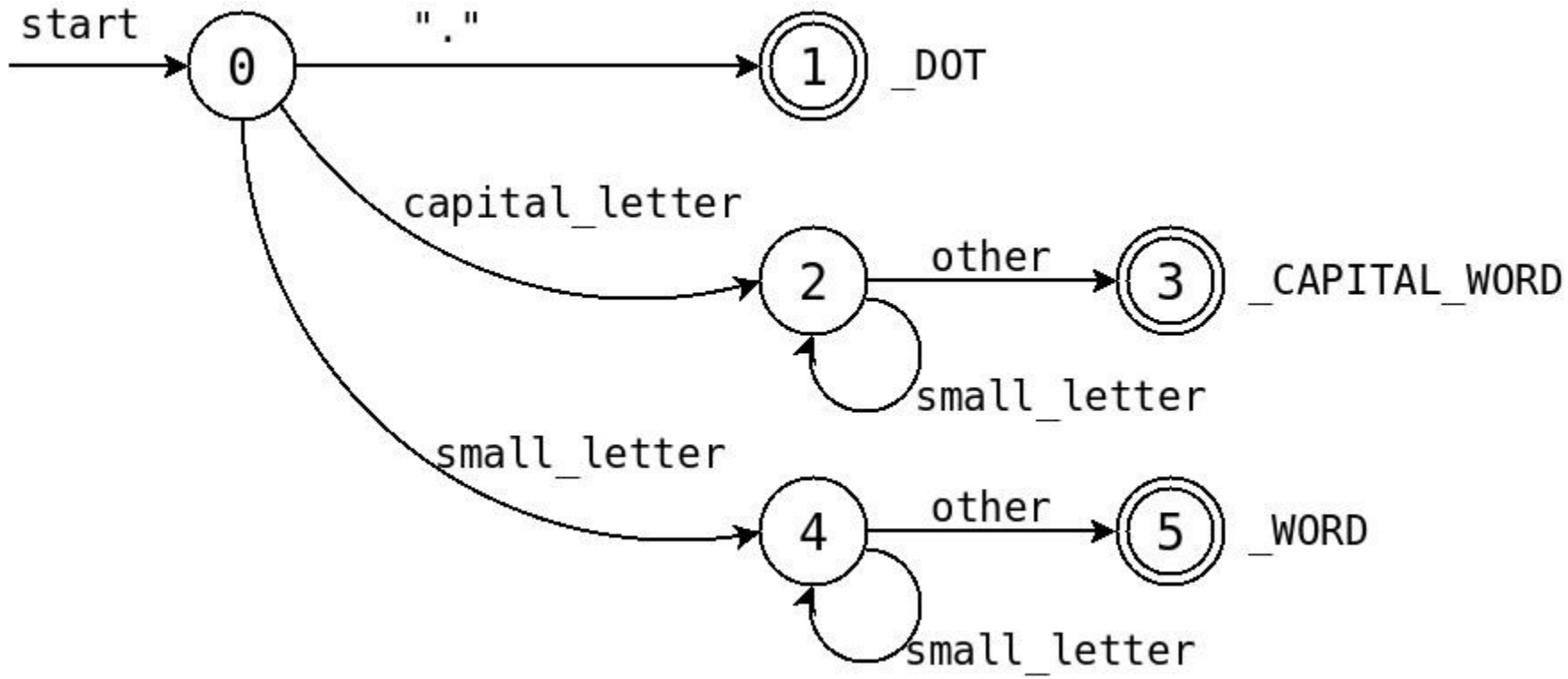
- Dijagram prelaza zahteva uvođenje stanja skenera
  - **polazno** stanje
  - po jedno **završno** stanje za svaki prepoznati simbol
  - za simbole sastavljene od jednog znaka, pojava tog znaka prevodi skener iz polaznog u završno stanje
  - za simbole sastavljene od više znakova, pojava prvog znaka prevodi skener iz polaznog u prvo **međustanje**, a pojava svakog narednog znaka prevodi skener u sledeće međustanje, kada ima više međustanja. Ako pravilo formiranja simbola dozvoljava ponavljanje pojedinih znakova, tada nakon njihove pojave skener ostaje u zatečenom međustanju.

Skener – dijagram prelaza

- Dijagram prelaza je **usmereni graf** u čiji sastav ulaze:
  - **čvorovi** koji predstavljaju stanja skenera
    - jedan od ovih čvorova je polazni, a
    - više njih mogu biti završni čvorovi
  - usmerene **spojnice** između čvorova
    - ukazuju na moguće prelaske iz jednog u drugo stanje
  - **labele** usmerenih spojnica
    - svaka labela odgovara skupu znakova
    - podrazumeva se da spojnice koje kreću iz istog čvora imaju različite labele, odnosno da je presek njihovih labela prazan skup (ovo ograničenje obezbeđuje jednoznačnost prelazaka)
  - **znakovi** od kojih se mogu obrazovati labele

Skener – dijagram prelaza

```
dot            → "."
capital_word → capital_letter {small_letter}
word           → small_letter {small_letter}
```



Skener – dijagram prelaza

- Podrazumeva se da početno stanje skenera odgovara početnom čvoru
- Kada se ne nalazi u završnom stanju, skener preuzima znak i proverava da li on pripada skupu znakova neke od labela spojnica izlazećih iz čvora koji odgovara trenutnom stanju skenera.
  - Ako preuzeti znak pripada skupu znakova jedne od labela pomenutih spojnica, skener prelazi u stanje određeno čvorom na koga ukazuje dotična spojnica.
  - Ako ne pripada, tada je otkrivena leksička greška.

Skener – dijagram prelaza

- Kada dospe u stanje koje odgovara završnom čvoru, skener je prepoznao simbol i može da ponovo pređe u početno stanje.
- U nekim od završnih stanja skener mora da **vrati poslednje preuzeti znak**, ako on ne pripada prepoznatom simbolu, jer tada on pripada sledećem simbolu
  - na primer, ako iza znaka **<** ne sledi znak **=**, reč je o relacionom operatoru manje, pa znak iza znaka **<** ne pripada tom nego sledećem simbolu

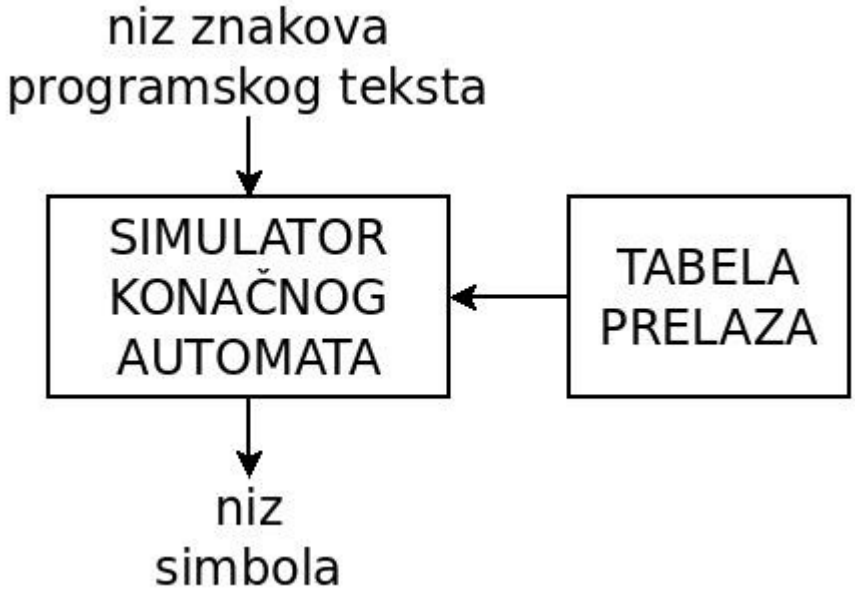
Skener – tabela prelaza

- Dijagram prelaza može biti prikazan i u obliku **tabele prelaza**
- Tabela prelaza određuje akciju skenera na pojavu nekog znaka u zadanom stanju
- Redove tabele prelaza označavaju redni brojevi polaznog stanja i međustanja, a njene kolone označavaju znakovi
- Svaki element tabele prelaza sadrži ili redni broj stanja u koje skener prelazi nakon pojave odgovarajućeg znaka ili crticu kao oznaku da u datom stanju nije predviđena pojava odgovarajućeg znaka

STANJE	ZNAK		
	"."	capital_letter	small_letter
0	1	2	4
2	3	3	2
4	5	5	4

Skener - deterministički konačni automat

- Dijagram/tabela prelaza definiše **deterministički konačni automat** (*deterministic finite automata, DFA*) koji ima unapred zadan konačan broj stanja i konačan broj jednoznačnih prelaza između ovih stanja
- Skener može imati oblik funkcije koja simulira takav automat.
- Da bi ovakav skener prepoznao simbole konkretne gramatike simbola on mora imati na raspolaganju odgovarajuću tabelu prelaza:



Skener – regularni izrazi

- Tabelu prelaza u potpunosti definiše odgovarajući opis gramatike simbola.
- Za ovu svrhu se koriste **regularne gramatike**. Njihova pravila imaju oblik **regularnih definicija** koje sadrže ime za određenu vrstu simbola i njen opis u formi **regularnog izraza** (*regular expression*):

  `ime → regularni izraz`

- Oblikovanje regularnih izraza
  - pojedinačni znakovi predstavljaju regularne izraze
    - regularnom izrazu **`a`** odgovara string "a"
  - spajanjem pojedinačnih znakova nastaju novi regularni izrazi
    - regularnom izrazu **`abc`** odgovara string "abc"

Skener – regularni izrazi

- znak **`*`** označava da se njemu prethodeći regularni izraz navodi nijednom, jednom ili više puta
  - regularnom izrazu **`abc*`** odgovaraju stringovi "ab", "abc", "abcc", "abccc", ...
- znak **`+`** označava da se njemu prethodeći regularni izraz navodi jednom ili više puta
  - regularnom izrazu **`abc+`** odgovaraju stringovi "abc", "abcc", "abccc", ...
- znak **`?`** označava da se njemu prethodeći regularni izraz navodi nijednom ili jednom
  - regularnom izrazu **`abc?`** odgovaraju stringovi "ab" i "abc"

Skener – regularni izrazi

- znak `|` označava alternative
  - regularnom izrazu `ab|cd` odgovaraju stringovi "ab" i "cd"
- male zagrade omogućuju grupisanje
  - regularnom izrazu `a(b|c)d` odgovaraju stringovi "abd" i "acd"
- uglaste zagrade omogućuju navođenje klase znakova
  - regularni izraz `[0123456789]` odgovara regularnom izrazu `(0|1|2|3|4|5|6|7|8|9)`, znači svakoj od pojedinačnih cifara
- znak `-` omogućuje skraćeno navođenje klase znakova
  - regularni izraz `[0-9]` odgovara regularnom izrazu `[0123456789]`

Skener – regularni izrazi

- znak `.` označava bilo koji znak osim znaka za novi red
  - regularnom izrazu `.*` odgovaraju svi mogući stringovi koji mogu da se pojave u jednom redu, uključujući i prazan red
- znak `\` omogućuje da se posebni znakovi tretiraju kao obični
  - regularnom izrazu `a\.b` odgovara string "a.b"
- vitičaste zagrade omogućuju definisanje broja ponavljanja
  - regularni izraz `a{2,4}` opisuje stringove "aa", "aaa" i "aaaa"

Skener – regularni izrazi

- Primeri regularnih izraza
  - regularni izraz `[0-9]+` opisuje prirodne brojeve i nulu
  - regularni izraz `-?[0-9]+` opisuje cele brojeve
  - regularni izraz `[01]+` opisuje binarne brojeve

Skener – generator skenera

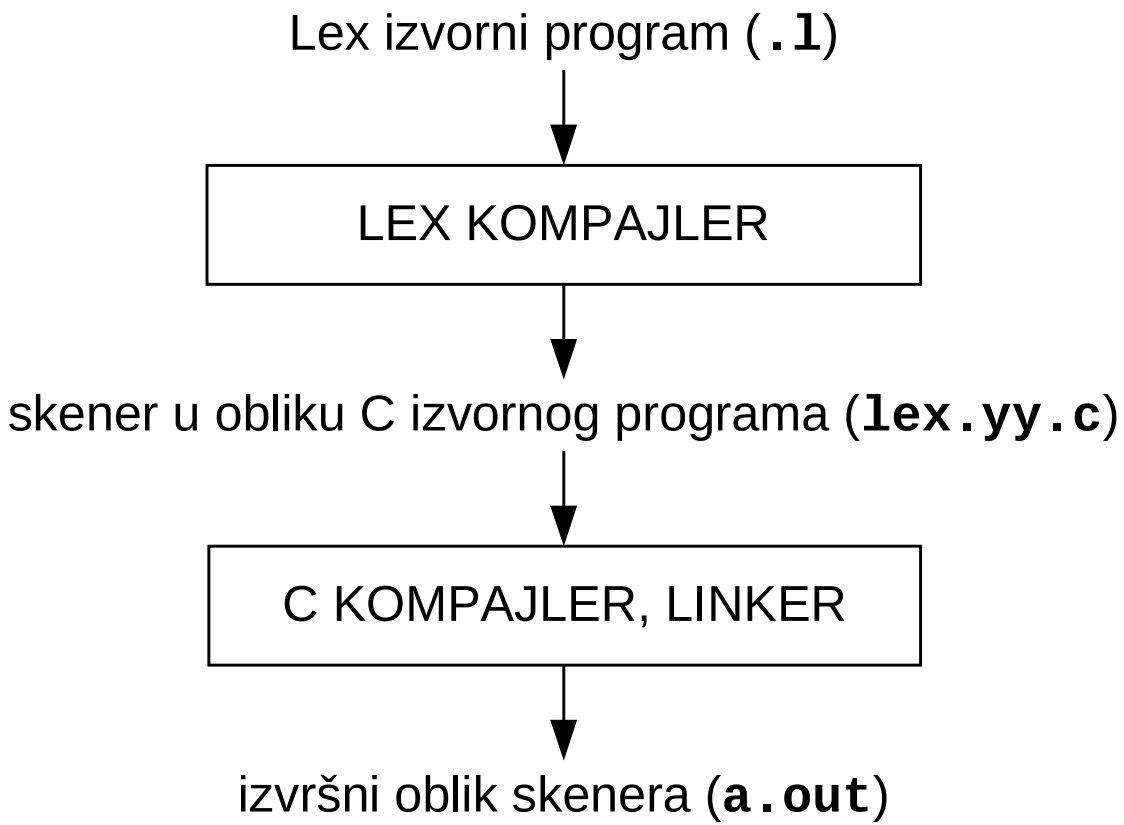
- Opis simbola u formi regularnog izraza je dovoljan za automatsko generisanje tabele prelaza namenjene za skener koji ima oblik simulatora konačnog automata, pa može da prepoznaje opisane simbole. Generator tabele prelaza određuje ponašanje ovakvog skenera, pa se naziva i **generator skenera**.
- Domet generisanog skenera je prepoznavanje simbola. Međutim, nakon dolaska skenera u završno stanje, treba da usledi njegova **akcija** nakon prepoznavanja simbola. Nju zna i opisuje korisnik u obliku segmenta programa. Takvi korisnički opisi akcija skenera moraju biti saopšteni generatoru skenera, da bi izgenerisani skener mogao na željeni način da reaguje na prepoznavanje simbola. Opisi takvih akcija se uparuju sa opisom odgovarajućih simbola u obliku:

`regularni izraz` **`opis akcije`**

Skener – generator skenera

- Pošto su opisi akcija segmenti programa koji koriste konstante, promenljive i funkcije, pored opisa akcija, generatoru skenera moraju biti saopštene odgovarajuće definicije konstanti, promenljivih i funkcija, korišćenih u opisu akcija
- Radi preglednosti regularnih izraza u njima se koriste imena navedena u regularnim definicijama, koje prethode regularnim izrazima
- Primer generatora skenera je **Lex kompajler** (*Lex compiler*)

Skener - Lex


Lex izvorni program (.l)
LEX KOMPAJLER
skener u obliku C izvornog programa (lex.yy.c)
C KOMPAJLER, LINKER
izvršni oblik skenera (a.out)

Skener - Lex

- Izgled Lex izvornog programa (Lex specifikacije)

Skener - Lex

- Lex kompajler generiše skener u obliku C funkcije **yylex**:

```
int yylex(void);
```

- Lex kompajler oblikuje funkciju **yylex** na osnovu regularnih izraza i njima pridruženih segmenata C programa koji opisuju željene akcije
- Funkciju **yylex** Lex kompajler smešta u izlaznu datoteku **lex.yy.c** zajedno sa neizmenjenim prvim i trećim delom specifikacije (definicijama C konstanti, C promenljivih i pomoćnih C funkcija)
- Za Lex kompajler znakovi **<** i **>** imaju posebno značenje (označavaju stanja skenera), pa se moraju koristiti između navodnika ("**<**" i "**>**") kada se koriste kao obični znakovi
- Lex kompajler zahteva definiciju funkcije **yywrap()** koja opisuje ponašanje skenera kada naiđe na EOF znak

Skener - Lex

- Nakon poziva funkcije **`yylex`**, u toku njenog izvršavanja, ponavlja se prepoznavanje simbola i izvršavanje zadanih akcija sve dok se u okviru zadanih akcija ne izvrši **`return`** iskaz. Tek tada sledi povratak iz ove funkcije.
- Automatski povratak iz ove funkcije se dešava kada se u toku preuzimanja znakova naiđe na kraj datoteke, a u tom slučaju povratna vrednost funkcije je **`0`**.

Skener - Lex

- Uz **yylex** funkciju Lex kompajler definiše i globalne promenljive:

char *yytext	omogućuje pristup stringu poslednje prepoznatog simbola
int yyleng	sadrži dužinu stringa poslednje prepoznatog simbola
int yylineno	sadrži broj trenutne linije
int yylval	služi za prenos vrednosti prepoznatog simbola

- String simbola treba preuzeti odmah po prepoznavanju, jer se daljim skeniranjem promenljiva **yytext** popuni stringom sledeće prepoznatog simbola!

Skener - Lex

- Podrazumeva se da se token prepoznatog simbola vraća kao povratna vrednost funkcije **`yylex`**, a da se vrednost prepoznatog simbola vraća posredstvom globalne promenljive **`yylval`**
- Funkcija **`yylex`** pronalazi
  - 1) najduži string koji odgovara nekom regularnom izrazu
    - ako pronađeni string odgovara nekolicini regularnih izraza, ovo pravilo omogućuje, na primer, da se **`<=`** prepozna kao manje ili jednako, a ne kao manje
  - 2) pripisuje ga prvonađenom regularnom izrazu
    - rezervisana reč **`if`** će se prepoznati kao rezervisana reč, a ne kao identifikator, jasno pod uslovom da regularni izraz **`if`** prethodi regularnom izrazu identifikatora **`[a-zA-Z][a-zA-Z0-9]*`**

Skener – primer izlaza skenera

```
Ovo je tekst.
Za skeniranje.
```

```
_____________________________________________________________________
START

             Ovo  TOKEN:  _CAPITAL_WORD   value:  Ovo
              je  TOKEN:  _WORD           value:  je
           tekst  TOKEN:  _WORD           value:  tekst
               .  TOKEN:  _DOT
              Za  TOKEN:  _CAPITAL_WORD   value:  Za
      skeniranje  TOKEN:  _WORD           value:  skeniranje
               .  TOKEN:  _DOT

STOP
```

- Lex specifikacija (`primeri/text/scanner/scanner.l`)