Parser

- Zadatak parsera je da proveri da li je ulazni niz simbola (tokena), dobijen od skenera, u skladu sa gramatikom
- Ovakva provera se svodi ili na
  - (1) pokušaj da se iz polaznog pojma gramatike po njenim pravilima **izvede** niz simbola identičan ulaznom nizu simbola ili na
  - (2) pokušaj da se ulazni niz simbola **redukuje** (sažme) po pravilima gramatike u njen polazni pojam

Parser

- Prvi pristup odgovara **silaznom** (*top-down*) **parsiranju**
- Drugi pristup odgovara **uzlaznom** (*bottom-up*) **parsiranju**
- Kod silaznog parsiranja parser u toku izvođenja formira **niz izvođenja** koji se može prikazati u obliku **stabla parsiranja** (*parse tree*)
- Izvođenja mogu biti
  - (1) **s leva** (*leftmost*): u svakom koraku se zamenjuje krajnje levi pojam ili
  - (2) **s desna** (*rightmost*): u svakom koraku se zamenjuje krajnje desni pojam

Parser

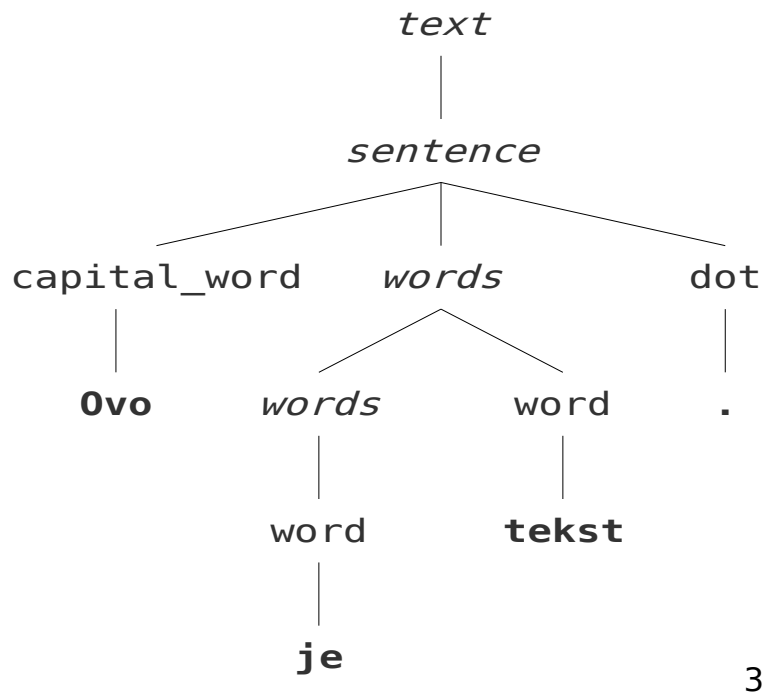
- Primeri nizova izvođenja i korespondentnih stabala parsiranja za iskaz

  Ovo je tekst.

- izvođenje s leva

text =>
text sentence =>
sentence =>
capital_word *words* dot =>
Ovo *words* dot =>
Ovo *words* word dot =>
Ovo *words* word word dot =>
Ovo word word dot =>
Ovo je word dot =>
Ovo je tekst dot
Ovo je tekst .

- stablo parsiranja za izvođenje s leva

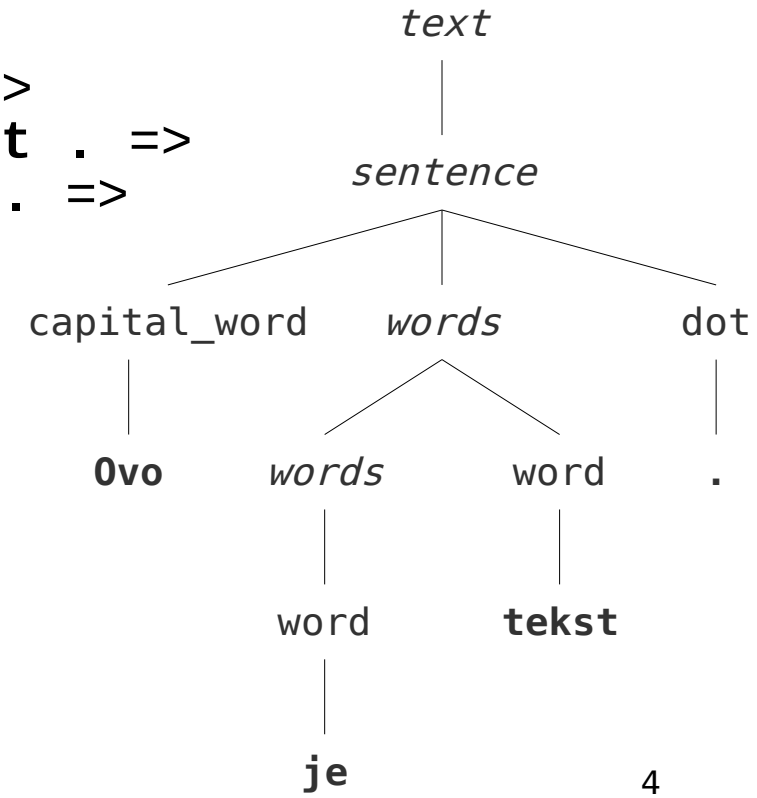


Parser

- izvođenje s desna

```
text =>
text sentence =>
text capital_word words dot =>
text capital_word words . =>
text capital_word words word . =>
text capital_word words tekst . =>
text capital_word words word tekst . =>
text capital_word words je tekst . =>
text capital_word je tekst . =>
text Ovo je tekst . =>
Ovo je tekst .
```

- stablo parsiranja za izvođenje s desna



Parser – silazno parsiranje

- Kod silaznog parsiranja parser na svakom koraku
  - preuzima s leva jedan simbol iz ulaznog niza simbola i
  - proverava da li gramatika predviđa pojavu preuzetog simbola u dotičnom koraku
- Na primer, parser bi za ulaz **`Ovo je tekst.`** mogao da:
  - preuzme prvi simbol `capital_word` (**`Ovo`**) i
    - ustanovi da pravila *`text`* i *`sentence`* omogućuju njegovu pojavu na samom početku
  - kao drugi simbol preuzme `word` (**`je`**)
    - koji se uklapa u pravilo *`words`*
  - preuzme treći simbol `word` (**`tekst`**) i uklopi ga u pravilo *`words`*
  - preuzme simbol `dot` (**`.`**)
    - koji se uklapa u pravilo *`sentence`*, odnosno *`text`*
  - prijavi uspešno završeno parsiranje

Parser – silazno parsiranje

- U toku silaznog parsiranja parser konstruiše stablo parsiranja prikazano na prethodnim slajdovima
- Za silazno parsiranje je zgodno primeniti izvođenje s leva
  - jer se tada simboli sa ulaza preuzimaju u prirodnom redosledu

Parser – uzlazno parsiranje

- Postupak redukcije kod uzlaznog parsiranja je inverzan (suprotan) postupku izvođenja kod silaznog parsiranja
  - u praksi je prihvaćen postupak redukcije koji je inverzan postupku izvođenja s desna (da bi se simboli iz ulaznog niza simbola mogli preuzimati u prirodnom redosledu)
- Kod uzlaznog parsiranja parser na svakom koraku
  - s leva preuzima jedan simbol iz ulaznog niza simbola,
  - dodaje ga na kraj prethodno formirane sekvence pojmova i/ili simbola i
  - proverava da li je novoformirana sekvenca identična desnoj strani nekog pravila gramatike
  - ako jeste, parser redukuje (zamenjuje) novoformiranu sekvencu pojmom sa leve strane pomenutog pravila

Parser

- Primer uzlaznog parsiranja:

Ovo je tekst . =>
text **Ovo je tekst .** =>
text capital_word **je tekst .** =>
text capital_word *words* **je tekst .** =>
text capital_word *words* word **tekst .** =>
text capital_word *words* **tekst .** =>
text capital_word *words* word **.** =>
text capital_word *words* **.** =>
text capital_word *words* dot =>
text sentence =>
text

Parser

- I u toku uzlaznog parsiranja parser konstruiše (izvrnuto) stablo parsiranja

Parser

- Sekvenca pojmova i/ili simbola, koja je kandidat za redukciju, može da se formira na **steku** tako što se na stek smeštaju jedan za drugim njeni elementi
- Tako se na vrhu steka uvek nalazi završni element dela sekvence koja može biti redukovana
- U okviru redukcije redukovani deo sekvence se skida sa steka, a umesto njega se na stek smešta pojam u koga se pomenuti deo sekvence može redukovati

Parser

- Primer korišćenja steka (stek se puni s desna u levo)

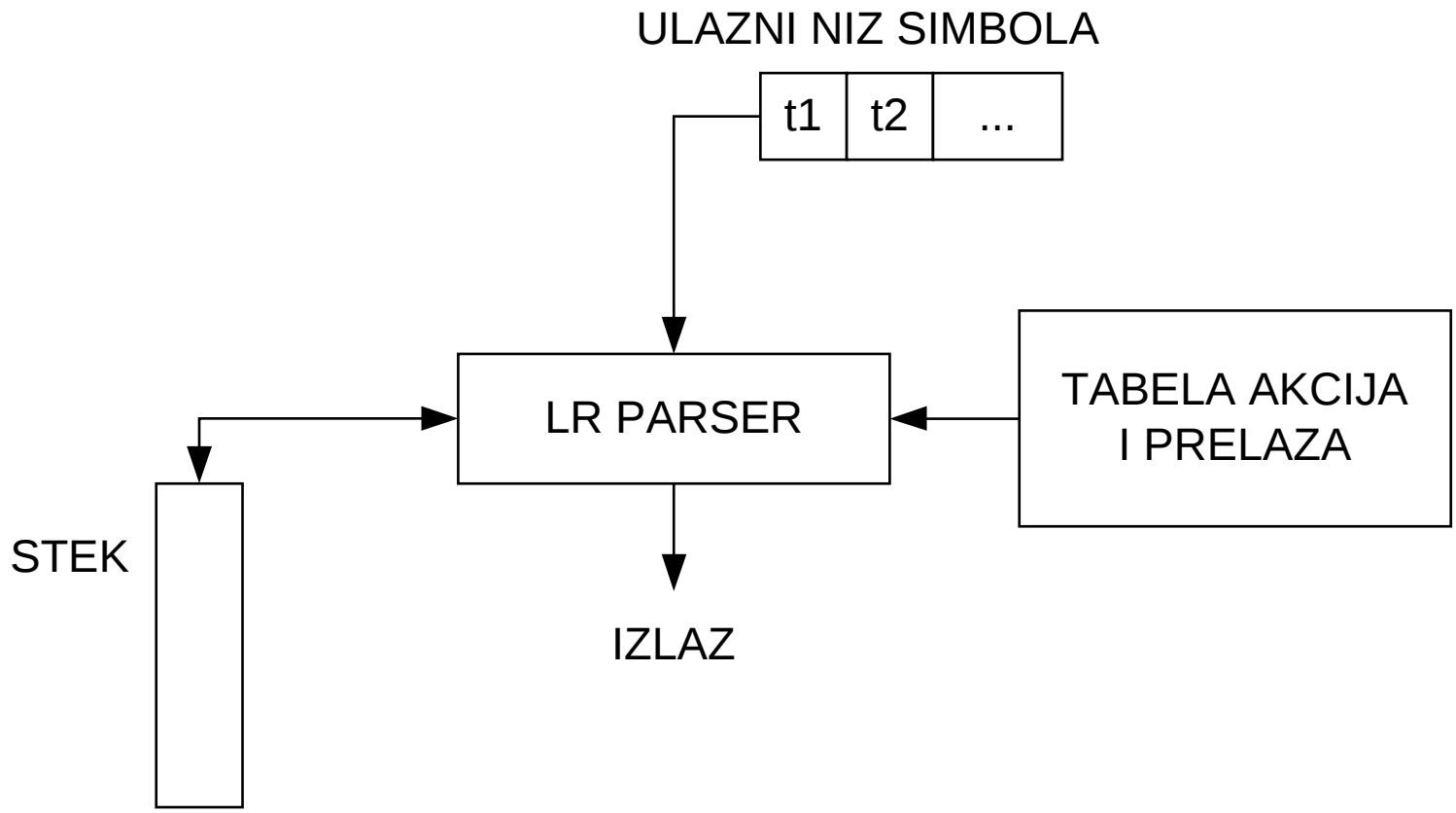
Sadržaj steka	Akcija
ε	redukcija
text	redukcija
text capital_word	smeštanje tokena capital_word (**Ovo**)
text capital_word ε	redukcija
text capital_word *words*	redukcija
text capital_word *words* word	smeštanje tokena word (**je**)
text capital_word *words*	redukcija
text capital_word *words* word	smeštanje tokena word (**tekst**)
text capital_word *words*	redukcija
text capital_word *words* dot	smeštanje tokena dot (**.**)
text sentence	redukcija
text	redukcija

Parser

- Sličnost rada skenera i parsera
  - za parser je simbol ono što je za skener znak
  - za parser se mogu uvesti stanja i podrazumevati da pojava nekog simbola u nekom stanju može prevesti parser u sledeće stanje ili može biti protumačena kao greška
- I parser se može realizovati kao simulator konačnog automata čiju aktivnost usmerava tabela akcija i prelaza
- Praktičan značaj uzlaznog parsiranja proizlazi iz činjenice da taj postupak koriste **LR parseri** (***L****eft-to-right scanning of the input, constructing a* ***R****ightmost derivation in reverse*)
- LR parseri imaju svojstvo opštosti i implementiraju se kao konačni automati

Parser – LR parser

- Model LR parsera



Parser – LR parser

- Na svakom koraku svoje aktivnosti LR parser se nalazi u jednom od stanja iz skupa mogućih stanja
  - podrazumeva se da se LR parser nalazi u početnom stanju na početku svoje aktivnosti
- Stanja su odabrana tako da svako stanje registruje napredak, ka nekoj od mogućih redukcija, koji je LR parser napravio na datom koraku
  - znači postepeno napredovanje koje parser napravi ka nekoj redukciji se registruje u obliku niza stanja kroz koja on prolazi u toku pomenutog napredovanja
- Svoja stanja LR parser čuva na steku umesto odgovarajućih simbola i/ili pojmova
  - na vrhu steka se nalazi važeće stanje,
  - ispod njega prethodno stanje,
  - itd. za preostala stanja iz niza stanja kroz koja je LR parser prošao

Parser – LR parser

- Tabela akcija i prelaza upravlja aktivnošću LR parsera
- Broj njenih redova je određen brojem različitih stanja
- Broj njenih kolona je određen brojem različitih simbola/tokena
- U preseku reda stanja S_i i kolone simbola t_j nalazi se element tabele koji određuje akciju LR parsera kada on u stanju S_i na početku ulaznog niza simbola pronađe simbol t_j
- Svaki od elemenata tabele akcija i prelaza sadrži oznaku jedne od moguće četiri akcije LR parsera
- Pod pretpostavkom da je LR parser u stanju S_i i da se na početku ulaznog niza simbola nalazi simbol t_j, moguće akcije LR parsera su:

stanja	simboli / tokeni		
	t1	...	tj
stanje 0			
...			
stanje i			

Parser – LR parser

1. LR parser prelazi u stanje navedeno u elementu [S_i,t_j], smešta oznaku tog stanja na stek i pomera (***shift***) simbol t_j sa početka ulaznog niza simbola
2. LR parser redukuje (***reduce***) niz od n stanja sa vrha steka i proizvede odgovarajući izlaz. Dužina n niza redukovanih stanja je određena sadržajem elementa [S_i, t_j], a novo stanje LR parsera zavisi od stanja koje na steku prethodi redukovanom nizu stanja (od vrste redukcije). Simbol t_j ostaje na početku ulaznog niza simbola.
3. LR parser objavljuje uspešno prepoznavanje ulaznog niza simbola (***accept***)
4. LR parser objavljuje da je ulazni niz simbola pogrešan (***error***)

- Zbog prve dve akcije, parsiranja koja obavljaju LR parseri se nazivaju *shift-reduce* parsiranja

Parser – LR parser

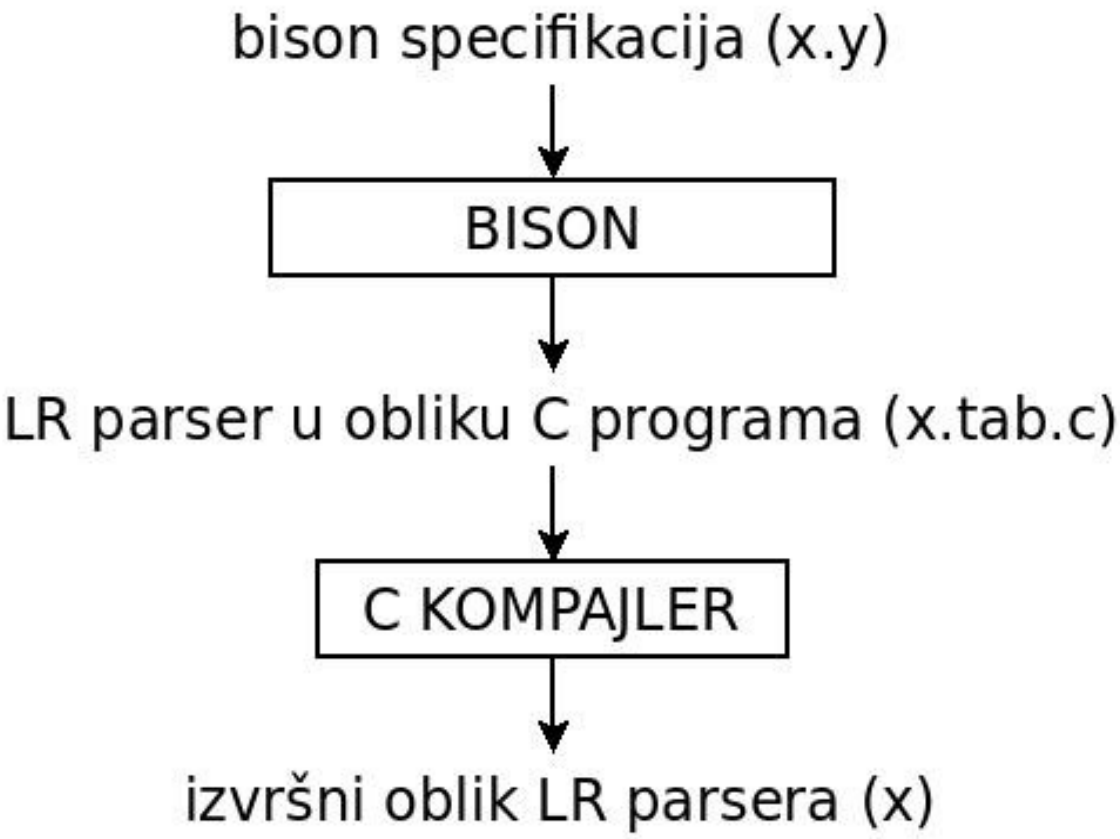
- LR parseri se međusobno razlikuju po tabeli akcija i prelaza
  - ova tabela je zavisna od gramatike
  - ona se oblikuje tako da se za polazno stanje LR parsera ustanovi koji simboli su prihvatljivi u tom stanju
  - za njih se odrede nova stanja u koja LR parser prelazi iz početnog stanja pri pojavi pomenutih simbola
  - za ostale simbole se konstatuje da njihova pojava u polaznom stanju predstavlja grešku
  - za svako od novih stanja se ponavlja prethodni postupak, uz napomenu da pojava određenih simbola u pomenutim stanjima može da dovede i do redukcije ili do uspešnog prepoznavanja ulaznog niza simbola
- LR parseri se mogu automatski izgenerisati ako se na osnovu zadate gramatike automatski proizvede tabela akcija i prelaza i tako definiše ponašanje generisanog LR parsera. To je zadatak **generatora LR parsera**.

Parser – generator LR parsera

- Domet generisanog LR parsera je prepoznavanje ispravnih ili pogrešnih nizova simbola
- Akciju parsera nakon prepoznavanja nizova simbola, odnosno njegov izlaz u toj situaciji treba da definiše korisnik
  - on to može da uradi tako što će generatoru LR parsera uz pravila navoditi i segmente programa koji određuju akcije LR parsera nakon prepoznavanja pomenutih pravila:

    `pravilo opis akcije`
- Pošto su opisi akcija segmenti programa u kojima se koriste konstante, promenljive i funkcije, generatoru LR parsera moraju biti saopštene i njihove definicije
- U okviru pravila se navode tokeni, pa se i njihove definicije moraju saopštiti generatoru LR parsera
- Primeri generatora LR parsera su
  - **Yacc** (***Y****et* ***A****nother* ***C****ompiler* ***C****ompiler*) kompajler
  - **Bison** kompajler

Parser - Bison



Parser - Bison

- Izgled Bison specifikacije (izvornog programa)

Parser - Bison

- Bison kompajler generiše LR parser u obliku C funkcije:

  ```
  int yyparse(void);
  ```

- Povratna vrednost funkcije **`yyparse`** različita od nule ukazuje na grešku u parsiranju
- Tokeni se zadaju u obliku

  ```
  %token   ime tokena
  ```

- Podrazumeva se da je prvo pravilo polazno
  - redukcijom po ovom pravilu parser završava rad

Parser - Bison

- Podrazumeva se da su pravila navedena u BNF notaciji
- Pravilo:

pojam → alternativa$_1$ | alternativa$_2$ | ... | alternativa$_n$

Bison prihvata u obliku:

```
pojam :  alternativa1     { /* C opis akcije */ }
      |  alternativa2     { /* C opis akcije */ }
         ...
      |  alternativan     { /* C opis akcije */ }
      ;
```

- Uputno je da se ponavljanja označavaju levom rekurzijom

Parser – primer text count 1

- Primer parsera – odrediti broj reči i rečenica u ulaznom tekstu, po tekst gramatici (**`count1`**)

Parser - Bison

- Bison podrazumeva da svaki pojam i simbol iz pravila poseduju vrednost
  - vrednost simbola određuje skener, a
  - vrednost pojmova određuje parser
- Radi čuvanja vrednosti pojmova i simbola, čija stanja se nalaze na steku, Bison predviđa poseban **stek za vrednosti**.
- Relativne pozicije stanja koja odgovaraju pojmovima/ simbolima na **steku stanja** i relativne pozicije odgovarajućih vrednosti na steku vrednosti su identične
- Radi rukovanja ovim vrednostima Bison predviđa posebnu notaciju
  - vrednost pojma s leve strane pravila označava sa **$$**
  - vrednost pojmova i simbola sa desne strane pravila označava sa **$**$_{\mathbf{i}}$ (**i** je redni broj pojma ili simbola na desnoj strani pravila, posmatrano s leva u desno)
- **$**$_{\mathbf{i}}$ označava lokacije ispod vrha steka, a nakon njihove redukcije **$$** označava vrh steka

Parser - Bison

- Za pravilo

  sentence → _CAPITAL_WORD words _DOT

  - **$$** označava vrednost pojma **sentence**,
  - **$1** vrednost simbola **_CAPITAL_WORD**,
  - **$2** vrednost pojma **words**, a
  - **$3** vrednost simbola **_DOT**
- Podrazumeva se da su vrednosti pojmova/simbola s desne strane pravila definisane i da se pomoću njih definiše vrednost pojma s leve strane pravila. Na primer:

  ```
  $$ = $1 + $3;
  ```
- Ako se u pravilu ne naznači drugačije podrazumeva se da važi

  ```
  $$ = $1
  ```

Parser - Bison

- Svaka akcija na desnoj strani pravila ima svoju vrednost **$i**
  - **A → B { a = 5; } C { $$ = 1; }**
    $$ $1 $2 $3 $4
- Vrednost akcije se definiše tako što se unutar same akcije definiše vrednost **$$**
  - **A → B { $$ = 5; } C { brojac = $2; }**
- **$$** u poslednjoj akciji se odnosi na vrednost simbola sa leve strane
  - **A → B { $$ = 5; } C { $$ = $2; }**
  - **A → B { $$ = 5; } C { $$ = 1; }**

Parser - broj reči i rečenica

- Primer parsera – odrediti broj reči i rečenica u ulaznom tekstu, po tekst gramatici (**`count2`**)

- Funkcija **`yyparse`** automatski poziva funkciju **`yylex`**
- Poziv funkcije **`free`** je potreban za dealokaciju memorije koja je zauzeta u skeneru posredstvom poziva funkcije **`strdup`**, a čija adresa je prosleđena parseru posredstvom globalne promenljive **`yylval`** (i kojoj se pristupa preko metapromenljive **$$_i$$**)

Parser - broj reči i rečenica

- Za ulaz:

  "Ovo je tekst."

  izlaz je:

Parser – Bison

- Opcije za pozivanje Bison-a:
  - ako se Bison specifikacija zove **`name.y`**
  - bez opcija generiše C fajl **`name.tab.h`** sa parserom u funkciji **`yyparse()`**
  - **`-v`** (*verbose*), generiše se datoteka **`name.output`** koja sadrži “čitljiv” opis tabele akcija i prelaza parsera tj. opis konačnog automata, u kojoj su opisani konflikti
  - **`-d`** (*definitions*), generiše C zaglavlje **`name.tab.h`** sa definicijom promenljive **`yylval`** i definicijama tokena
    - kada se ovaj fajl uključi u skener, mogu se koristiti tokeni u njemu