

Programski prevodioci: Vežbe 3

Sadržaj

1. Uvod.....	1
2. Kompajliranje, pokretanje i testiranje.....	1
3. Napomena za .l datoteku	1
4. Problem "visećeg" else	2
5. Zadaci	4
5.1. Zadatak 1.....	4
5.2. Zadatak 2.....	4
5.3. Zadatak 3.....	4
6. Projekat.....	5

1. Uvod

Na ovim vežbama biće prikazana gramatika programskog jezika miniC.

2. Kompajliranje, pokretanje i testiranje

Kao i prethodne nedelje, rešenje se kompajlira i pokreće na sledeći način:

```
make
./syntax
```

Kako bismo izbegli da prilikom testiranja svaki put ručno kucamo test primer, moguće je test primer iskucati u nekoj datoteci, a zatim upotrebom redirekcije proslediti tu datoteku na standardni ulaz programa:

```
./syntax < ulazna_datoteka
```

3. Napomena za **.l** datoteku

Obratiti pažnju da pojedini regularni izrazi vraćaju identične tokene:

...

```
"int"           { return _TYPE; }  
"unsigned"      { return _TYPE; }
```

...

```
"+"           { return _AROP; }  
"-"           { return _AROP; }
```

```
"<"           { return _RELOP; }  
">"           { return _RELOP; }  
"<="          { return _RELOP; }  
">="          { return _RELOP; }  
"=="          { return _RELOP; }  
"!="          { return _RELOP; }
```

...

Skener je implementiran na ovakav način, kako bi se pojednostavila gramatika jezika.

Na svakom mestu u gramatici gde može da se pojavi operator **+**, sigurno je uvek validno i da se na tom mestu pojavi operator **-**. Prema tome, za potrebe sintaksne analize, nema potrebe praviti distinkciju između operatora **+** i operatora **-**, pa se zato u oba slučaja vraća isti token. Isto važi i za relacione operatore i za tipove.

4. Problem "visećeg" **else**

Posmatrajmo sledeću konstrukciju:

```
if (/* uslov */)
  if (/* uslov */)
    /* iskaz */
  else
    /* iskaz */
```

Uzmimo da je definisana sledeća gramatika za ovakvu konstrukciju:

```
if_statement
: if_part
| if_part _ELSE statement
;

if_part
: _IF _LPAREN rel_exp _RPAREN statement
;
```

Kada parser naiđe na token `_ELSE`, dolazi do `shift/reduce` konflikta:

1. Parser može izvršiti redukciju i tako završiti parsiranje unutrašnjeg `if` iskaza. Ovo bi značilo da `else` deo pripada spoljašnjoj petlji.
2. Parser može nastaviti sa preuzimanjem tokena i tek nakon kompletnog preuzimanja `else` dela izvršiti redukciju. Ovo bi značilo da `else` deo pripada unutrašnjoj petlji.

Parser ne može samostalno da odluči koji od ova 2 navedena slučaja je ispravan, pa stoga dolazi do konflikta.

Razrešavanje konflikta definisanjem prioriteta:

```
...  
  
%nonassoc ONLY_IF ①  
%nonassoc _ELSE  
  
%%  
  
if_statement  
: if_part %prec ONLY_IF ②  
| if_part _ELSE statement  
;  
  
if_part  
: _IF _LPAREN rel_exp _RPAREN statement  
;  
  
%%  
...
```

① Bison omogućava navođenje prioriteta i asocijativnosti tokena.

Moguće deklaracije su:

%left

Operatori su levo asocijativni.

%right

Operatori su desno asocijativni.

%nonassoc

Definiše samo prioritete, a ne i asocijativnost.

Posle svake deklaracije navodi se lista tokena na koje se deklaracija odnosi.

Prioriteti operatora su kontrolisani redosledom navođenja deklaracija. Token koji je prvi naveden u `.y` datoteci ima najmanji prioritet, dok poslednji token ima najveći prioritet. Prema

tome, u listingu iznad, navedeno je da token `ONLY_IF` ima manji prioritet od `_ELSE` tokena.

Bitno je naglasiti da se prioriteti i asocijativnosti odnose samo na tokene, a ne i na pojmove.

- ② Imajući u vidu da se prioriteti odnose samo na tokene, dolazimo do problema. Prvo `if_statement` pravilo ne sadrži nijedan token, već samo pojam `if_part`. Dakle, nije moguće specificirati prioritete tokena, kada token ne postoji. U ovakvim slučajevima može se navesti `%prec` modifikator praćen nazivom nekog izmišljenog tokena. Ovo znači da token `ONLY_IF` ne postoji zapravo u ulaznom tekstu, već da je to izmišljen pomoćni token koji nam omogućava da specificiramo da `_ELSE` ima veći prioritet od tog izmišljenog tokena.

Na ovaj način, razrešena je dvosmislenost gramatike. Token `_ELSE` ima veći prioritet, pa će parser nedvosmisleno odlučiti da nastavi preuzimanje tokena. Ovako podešeni prioriteti imaju za posledicu da `else` uvek pripada unutrašnjem `if` iskazu, što i jeste način na koji C i miniC programi funkcionišu.

5. Zadaci

5.1. Zadatak 1

Proširiti gramatiku `select` iskazom.

Sintaksa `select` iskaza ima oblik:

```
"select" <vars> ①  
"from" id "where" "(" <condition> ")" ";" ②
```

① Pojam `<vars>` predstavlja jednu ili više promenljivih, odvojenih zarezom.

② Pojam `<condition>` predstavlja relacioni izraz.

```
select a, b, c from details where (a > 4);
```

5.2. Zadatak 2

Proširiti gramatiku tako da se u jednoj deklaraciji može deklarirati više promenljivih, odvojenih zarezom:

```
int a, b, c;
```

5.3. Zadatak 3

Proširiti gramatiku postinkrement operatorom, koji se može primeniti samo na identifikator.

Postinkrement operator se može pojaviti:

Unutar izraza

```
a = b + c++ - 5;
```

Kao poseban iskaz

```
a++;
```

6. Projekat

U [dokumentu](https://docs.google.com/document/d/e/2PACX-1vTh-HFm3Ujw2s6Oa_bWZGeYQM87NHUpS2L90rq72DQd0QigfpDDax0-Rh01PLNKKiwx7iGQUV3Ouecu/pub) [https://docs.google.com/document/d/e/2PACX-1vTh-HFm3Ujw2s6Oa_bWZGeYQM87NHUpS2L90rq72DQd0QigfpDDax0-Rh01PLNKKiwx7iGQUV3Ouecu/pub] se nalaze osnovne informacije vezane za projekat i zajednički zadaci koje je potrebno da svi implementiraju. Svake nedelje dokument će biti dopunjavan sa obaveznim delovima koje je potrebno implementirati.