Sveučilište u Zagrebu
Fakultet elektrotehnike i računarstva

Chiro

Seminarski rad iz predmeta Prevođenje programskih jezika

Zadatak broj 91

Zagreb, siječanj 2009.

Seminarski rad iz predmeta Prevođenje programskih jezika

Student: Chiro
Matični broj studenta: zzzzzzzzzz
Zadatak broj 91: Pomoću programa YACC ostvariti **sintaksnu analizu jezika kojim se opisuje ulazna datoteka programa YACC**. Pri tom nije potrebno obuhvatiti dijelove ulazne datoteke koji sadrže konstrukte C programskog koda. Nije potrebno ostvariti i leksičku analizu.

Uvod

Sintaksna analiza nekog izvornog programa je postupak procjene je li taj kod zadovoljava pravila prevođenja programskog jezika kojim je pisan. Pravila prevođenja su definirana produkcijama gramatike koje sadrže nizove završnih i nezavršnih znakova. Rezultat uspješne sintaksne analize predstavljaju parser i generativno stablo koje se može izgraditi iz produkcija gramatike.

YACC je alat za sintaksnu analizu. On prima ulaznu datoteku koja je zapravo jasno definirana specifikacija sintakse nekog jezika i prevođenjem te datoteke stvara parser u obliku datoteke izvornog C koda. Ulazna datoteka programa YACC je podijeljena u 3 cjeline: deklaracije, pravila prevođenja te pomoćne C procedure. Cjelina deklaracije sadrži deklaracije koje koristi C jezični procesor te deklaracije završnih znakova i njihovih prednosti. Pravila prevođenja su zapravo produkcije gramatike i akcije semantičkog analizatora. Pomoćne C procedure čine različiti pomoćni potprogrami sintaksnog analizatora poput potprograma postupka oporavka od pogreške. Zadatak je napraviti YACC specifikaciju za jezik kojim se opisuje ulazna datoteka programa YACC.

Ostvarenje

Za ostvarenje projektnog zadatka nije bila potrebna sva funkcionalnost specifikacije ulazne datoteke programa YACC. Tekstom zadatka je zadano da se ne moraju definirati svi dijelovi ulazne datoteke koji sadrže konstrukte C programskog koda. Tako su u cjelini deklaracije izostavljene deklaracije C jezičnog procesora. Pomoćne C procedure također nisu definirane. Zahvaljujući tome definicija sintaksnog analizatora sadrži samo deklaracije završnih znakova gramatike i pravila prevođenja.

Deklaracija završnih znakova započinje s tokenom IDENTIFIKATOR. On predstavlja sve završne znakove koji sadrže više od jednog simbola te sve nezavršne znakove jezika ulazne datoteke programa YACC. Token POC_IDENTIFIKATOR predstavlja nezavršne znakove s kojima započinju produkcije gramatike jezika ulazne datoteke. Tokeni TOKEN, START, PREC, UNION, LEFT, RIGHT, NONASSOC i TYPE predstavljaju ključne (rezervirane) riječi. Tokeni POZ, PZZ i DP predstavljaju specijalne znakove %{, %} i %%.

U ovoj sintaksnoj analizi se ne koriste završni znakovi koji zahtijevaju izričitu definiciju prednosti i asocijativnosti, no pojavila se Pomakni/Reduciraj nejednoznačnost u produkciji gramatike koja za nezavršni znak lijeve strane produkcije ima znak akcija. Nejednoznačnost je razriješena definiranjem asocijativnosti i prednosti za završne znakove IDENTIFIKATOR i '{' pri čemu znak '{' ima veću prednost od znaka IDENTIFIKATOR te se tako akcijom Pomakni razrješava nejednoznačnost. Deklaracije završavaju definicijom da početna produkcija pravila prevođenja ima nezavršni znak cjeline kao nezavršni znak lijeve strane produkcije.

U pravilima prevođenja je navedena gramatika , zajedno sa semantičkim akcijama, koja generira jezik ulazne datoteke programa YACC. YACC ima izrazito veliku funkcionalnost pa zbog toga jezik njegove ulazne datoteke pruža poprilično velike mogućnosti i slobode korisniku pri definiranju pravila prevođenja. Sintaksa tog jezika je zato poprilično fleksibilna, a nije kruta i strogo definirana kao sintaksa programskih jezika poput C-a, Jave i drugih sličnih jezika. Ta činjenica je uzrokovala probleme pri definiciji gramatike. Kako bi se ostvarila sva funkcionalnost jezika ulazne datoteke, gramatika mora biti jednostavna te ne može prijaviti sve moguće greške koje mogu nastati pri pisanju pravila prevođenja. No, te pogreške su lako uočljive pri ispisu tablica parsera odnosno

generativnog stabla.

Također se pojavio problem izbora implementacije YACC-a koja će se koristiti za ostvarenje projektnog zadatka. Većina YACC implementacija je napravljena za Linux operacijski sustav, a tek poneka za Windows. Prvotna namjera je bila koristiti BYACC (*Berkeley YACC*) , no pokazalo se da pronađena aplikacija se ne može inicijalizirati, najvjerojatnije zbog nedostatka instalacije Cygwina na korištenom Windows sustavu. Umjesto instalacije Cygwina i provjeravanja je li to jedini problem vezan uz pokretanje BYACC-a, pronađen je i korišten program Parser Generator koji posjeduje funkcionalnost BYACC-a. Spomenuti program dostupan je na sljedećem URL-u: http://www.bumblebeesoftware.com/.

Kao rješenje samostalnog studentskog projekta osim ovog dokumenta prilažem još arhivu SSP_rješenje koja sadrži YACC ulaznu datoteku (YACCparser.y) te sve datoteke koje je Parser Generator generirao prevođenjem ulazne datoteke.

Zaključak

Pomoću programa YACC je ostvarena sintaksna analiza jezika njegove ulazne datoteke. Iz rezultata projektnog zadatka se zaključuje da je YACC izrazito moćan alat za sintaksnu analizu, no za njegovo kvalitetno korištenje korisnik mora biti upoznat s osnovnom dokumentacijom i specifikacijom alata YACC te mora posjedovati barem osnovna znanja programiranja kako mu se ne bi događale elementarne pogreške pri pisanju specifikacije ulazne datoteke programa YACC koje njegova gramatika direktno ne detektira jer mora biti poprilično općenita kako bi zadovoljila širok spektar YACC-ove funkcionalnosti.