Sveučilište u Zagrebu
Fakultet elektrotehnike i računarstva

Stevo-88

Seminarski rad iz predmeta Prevođenje programskih jezika

Zadatak br. 2021

Zagreb, siječanj 2010.

Seminarski rad iz predmeta Prevođenje programskih jezika

Student: Stevo-88

Matični broj studenta: -

Zadatak broj 2021: Pomoću programa *LEX* ostvariti **leksičku analizu programskog kôda za jezik *Fortran*.** Potrebno je podržati osnovne konstrukte jezika poput ulaznih i izlaznih naredbi, blokove, naredbe grananja, programske petlje, te riješiti nejednoznačnosti koje se pojavljuju u leksičkoj analizi. Izlaz iz programa su četiri tablice leksičke analize.

UVOD

Leksički analizator

Leksički analizator je prvi dio jezičnog procesora. To je jedini dio koji izravno pristupa izvornom kôdu. On grupira znakove izvornog kôda u leksičke jedinke koje zatim klasificira, usput bilježeći parametre pojedine jedinke. Također generira niz uniformnih znakova koji prikazuje klase leksičkih jedinki onim redoslijedom kojim su one zapisane u izvornom kôdu. Uz sve to, leksički analizator pronalazi greške te određuje njihovo mjesto u izvornom kôdu.

Izlaz leksičkog analizatora su tablica uniformnih znakova te tablice koje sadrže konkretne leksičke jedinke s parametrima. U ovoj implementaciji leksičkog analizatora tri su takve tablice:

- Tablica ključnih riječi, operatora i specijalnih znakova (KROS tablica),
- Tablica identifikatora,
- Tablica konstanti.

KROS tablica je fiksna – sadrži skup leksičkih jedinki specifičan za određeni programski jezik. U ovom slučaju nema parametara. U Tablicu identifikatora upisuju se varijable, imena funkcija i potprograma, labele i sl. Također nema parametara. Tablica konstanti sadrži popis brojeva, pojedinačnih znakova i stringova koji su eksplicitno napisani u izvornom kôdu. Parametar im je tip konstante (konkretno, *real*, *integer*, *complex* i *string*). Tablica uniformnih znakova u svakom retku, dakle za svaku leksičku jedinku, sadrži i pokazivač na određeni zapis u pripadajućoj tablici s informacijama o leksičkim jedinkama. Te su (četiri) tablice jedino što je potrebno daljnjem prevođenju programa.

Program LEX (*Lexical Analyser Generator*)

Lexical Analyser Generator (hr. *generator leksičkih analizatora*) jest program koji od skupa pravila zadanih kombinacijom regularnih izraza i naredaba programskog jezika *C* stvara kôd (napisan u jeziku *C*) koji, pravilno kompiliran, daje program koji analizira ulaz (datoteku ili s tipkovnice), pronalazi u njemu definirane regularne izraze te, po pronalasku svakog izraza, izvršava zadani programski kôd. Kao što mu sámo ime govori, najčešće se koristi za generiranje leksičkih analizatora.

Forma tekstualne datoteke s pravilima strogo je definirana:

```
%{

        Deklaracije funkcija i globalnih varijabli
        pisane u C-u, također i funkcija main()

%}

        Regularne definicije oblika:
        {imelexjedinke}       regularni izraz

%%

        Pravila prevođenja oblika:
        regularni izraz       kôd u C-u

%%

        Pomoćne procedure
```

Konkretna će datoteka biti pobliže opisana u dijelu Ostvarenje (str. 4).

Fortran

Fortran (*The IBM Mathematical **For**mula **Tran**slating System*) je proceduralni programski jezik pogodan za numeričke kalkulacije. Razvio ga je IBM 1950-ih godina za znanstvene aplikacije. Tijekom proteklih pola stoljeća koristi se za razne zadatke koji zahtijevaju visoke performanse, kao što su predviđanje vremena, numeričko rješavanje parcijalnih diferencijalnih jednadžbi, računalna dinamika fluida i sl. Također se koristi za *benchmarking* superračunala.

Zbog svoje dugovječnosti, Fortran ima mnoge verzije (FORTRAN, IBM 1401 FORTRAN, FORTRAN II, FORTRAN III, FORTRAN IV, FORTRAN 66, FORTRAN 77, Fortran 90, Fortran 95, Fortran 2003, Fortran 2008). Nažalost, literatura (Internet) često ne navodi verziju fortrana o kojoj piše pa je vrlo vjerojatno da je moj leksički analizator mješavina raznih verzija. No, smatram da su "moja" pravila vrlo bliska Fortranu 95:

- Nema osjetljivosti na mala/velika slova, čak ni u ključnim riječima. Popis ključnih riječi relativno je lako proširiv (inače je vrlo opširan, ovdje su uzete samo najvažnije naredbe: `"program", "implicit none", "integer", "real", "double precision", "complex", "character", "len", "logical", "parameter", "dimension", ".true.", ".false.", "if", "then", "elseif", "else", "endif", "do", "while", "exit", "print", "read", "end", "call", "contains", "subroutine", "function"`). Lako je učiniti da je npr. `"end function"` jedna kodna riječ (ako je to zgodnije sintaksnom analizatoru), ali zasad je ostavljeno ovako.
- Komentari počinju znakom uskličnika (`!`) i protežu se do kraja reda.
- Popis operatora također je lako proširiv, a trenutačno sadrži: `"+", "-", "*", "/", "**", "<", ">", "==", "=", "/=", "<=", ">=", ".lt.", ".gt.", ".eq.", ".ne.", ".le.", ".ge.", ".not.", ".and.", ".or.", ".eqv.", ".neqv."`.
- Specijalni znakovi su: `"_", ":", ";", "?", "$", "&", "%", "'", ".", ",", "(", ")"`, te znak navodnika(`"`).
- Identifikatori počinju slovom, smiju sadržavati slova, brojke i donju crtu (_) te mogu imati najviše 254 znaka.

- Brojevne konstante (*real* ili *integer*) počinju brojkom, mogu sadržavati točno jednu decimalnu točku (ne zarez), nakon koje slijede brojke. Dozvoljen je i (neovisno o postojanju decimalne točke) točno jedan znak eksponenta (E, D ili Q – označavaju preciznost), zatim opcionalno `"+"` ili `"-"` pa ponovo brojke.
- Kompleksni se brojevi pišu u obliku `(`*broj*`,`*broj*`)`. Prvi broj je realni dio, a drugi imaginarni dio kompleksnog broja. Oznake kao što su `i` ili `j` za imaginarne brojeve se ne koriste.
- Stringovi su nizovi bilo kojih znakova (osim novog reda) između dva znaka (dvostrukog ili jednostrukog) navodnika.

Nejednoznačnosti

U ovako definiranom jeziku postoje dvije nejednoznačnosti, koje je moguće riješiti provjerom lijevog konteksta:

- znak minusa kao matematički operator i kao dio (negativnog) broja, te
- kompleksni broj kao dva argumenta funkcije ili potprograma i kao stvaran kompleksni broj.

Ostvarenje

Leksički je analizator ostvaren pomoću programa Flex (slobodna verzija LEX-a) na operacijskom sustavu Linux. Krajnji se program sastoji od dvije izvršne datoteke: `analizator` (kompilirana datoteka lex.yy.cc) i `start` (program koji olakšava pravilno pokretanje analizatora). Potonji je prilično jednostavan i ne smatram potrebnim ni bitnim objašnjavati ga. Zato sada samo preostaje objasniti datoteku pravila.lex.

Pa, krenimo od kraja. *Pomoćne procedure* ne postoje – sve su procedure napisane kod deklaracija. *Pravila prevođenja* sastoje se od jednostavnog regularnog izraza definiranog u dijelu *Regularne definicije* (npr. samo `{kljucnarijec}`) i akcije koja se sastoji samo od poziva funkcije i postavljanja varijabli za provjeru lijevog konteksta (objašnjeno kasnije). Iznimke su `{prekidni}+`, koji nema definiranu akciju (dakle, samo zanemaruje prekidne znakove), te oble zagrade i zarez. Otvorena zagrada i zarez definirani su posebno jer se poslije njih, za razliku od ostalih specijalnih znakova, varijabla *ocekuj_negativni* mora postaviti na 1 (lijevi kontekst – kasnije). Zatvorena zagrada definirana je na ovaj način samo da bi bila "u paru" s otvorenom.

Regularne definicije zapravo su bit ovog cijelog rada, iako zauzimaju vrlo malo mjesta. Započinju jednostavnom klasifikacijom pojedinih znakova (ili grupa od po nekoliko određenih znakova): *novired*, *prekidni*, *slovo*, *brojka*, *specznak*, *operator* i *pmpp* (što je kratica od PlusMinusPutaPodijeljeno). Ta su četiri znaka odvojena od operatorâ zbog rješavanja nejednoznačnosti. Nastavljaju se jednostavne definicije komentara i stringa te popis ključnih riječi. Varijabla (zapravo se radi o bilo kojem identifikatoru) počinje slovom i nastavlja se slijedom slova, brojki i donje crte. Problem duljine varijable riješen je programski. Pozitivan broj definiran je kao što je opisano u podnaslovu "Fortran", a negativan je isti, sa znakom minusa na početku. Klasa *broj*

(ne baš pedagoški) označava samo realne i cijele brojeve. Zbog pohlepnosti regularnih izraza u definiciji klase *kompleksni* nakon svakog se znaka mora dopisati `{prekidni}*`, što smanjuje čitljivost.

Dolazimo do *Deklaracija*, koje zauzimaju 400-tinjak linija kôda. Sve je prilično detaljno komentirano, pa smatram suvišnim još jednom sve komentirati ovdje. No, neke je stvari bitno spomenuti. Izlaz leksičkog analizatora su četiri tablice koje se spremaju u četiri zasebne datoteke. No, one se spremaju tek na kraju programa. Tijekom izvođenja, svi su elementi svih tablica spremljeni u dinamički alocirana polja (osim KROS tablice koja je spremljena u statički alocirano polje). Ovo može predstavljati problem kod vrlo dugih programa i računala s vrlo malo memorije. Nadalje, sva su polja jednodimenzionalna, radi jednostavnosti. Pošto tablice imaju fiksan broj stupaca, napravljeno je zasebno polje za svaki stupac svake tablice. Inicijalizacija dinamičkog alociranja memorije za tablice (naredba *malloc*) nalazi se na kraju, u funkciji *main()*, jer se ne može nalaziti izvan funkcije, kao deklaracije globalnih varijabli.

Općenito, izrazi iz *Pravila prevođenja* šalju se u tri funkcije: *fidn()*, *fkros()* i *fkonst()*. Funkcija *fkonst()* jedina prima parametar na temelju kojeg odlučuje o akciji. To je napravljeno samo da bi se pojednostavnila forma pravila prevođenja. Funkcija *main()* sadrži definicije varijabli, poziv funkcije *yylex()*, te ispis tablicâ u datoteku i oslobađanje dinamički alocirane memorije. Pošto se ovaj rad neće primijenjivati u praksi, datoteke su formatirane tako da bi bile lako čitljive ljudima, a ne daljnjoj obradi programa.

Isprogramirana je i "infrastruktura" za upravljanje greškama – funkcija koja, ovisno o tipu greške (greška ili upozorenje) ispisuje prikladnu poruku na *stderr* (obično zaslon). Poruka sadrži broj linije izvornog kôda, opis greške (koji se funkciji predaje) te sámu leksičku jedinku koja je izazvala grešku. Jedine greške čije je pronalaženje implementirano su upozorenje o nemogućnosti pronalaženja ulaza u KROS tablici (što je nemoguće, osim uz loše modificiranje programa), te obavijest o predugom identifikatoru. U prvom slučaju, riječ koja nije pronađena smatra se identifikatorom, a u drugom se predugi identifikator preskače. U oba se slučaja nastavlja s leksičkom analizom.

Rješavanje nejednoznačnosti

Nejednoznačnosti su riješene korištenjem varijabli *ocekuj_negativni* i *bio_iden*, obje inicijalno postavljene na nulu. Prva se postavlja na jedinicu samo nakon operatora koji nije plus, minus, puta ni podijeljeno, te otvorene zagrade i zareza. U svim se ostalim slučajevima (osim u slučaju prekidnog znaka) postavlja na nulu. Drugim riječima, samo nakon otvorene zagrade, zareza i većine operatorâ očekuje se da će znak minusa nakon kojeg slijedi broj značiti negativni broj, a ne operaciju oduzimanja pa pozitivan broj. Nakon bilo čega drugoga, ne očekuje se negativan broj.

Druga se nejednoznačnost javlja kod dva broja odvojena zarezom, sve unutar oblih zagrada, što se može protumačiti kao kompleksni broj ili kao argument funkcije ili potprograma. Rješavanje ove nejednoznačnosti je jednostavnije – potrebno je samo "dignuti zastavicu" *bio_iden* poslije svakog identifikatora (*bio_iden* je skraćeno od "prije ove je leksičke jedinke *bio_iden*tifikator"). Pošto su samo dvije varijable za rješavanje nejednoznačnosti, činilo mi se preglednim sve njihove promjene staviti u *Pravila prevođenja*.

Zaključak

Tijekom izrade ovog zadatka naišao sam na mnoge probleme. Kao što sam već spomenuo, bilo je teško naći dobar izvor s jednostavno i opširno definiranim osnovnim svojstvima jezika Fortran. Rezultat toga bio je rad s više različitih izvora koji većinom ne navode o kojoj verziji Fortrana govore. Vrlo je zbunjujuće raditi s kontradiktornim izvorima. Mojoj sveopćoj zbunjenosti potpomoglo je i temeljito nepoznavanje Fortrana (prvi se puta susrećem s tim jezikom).

Također, ne postoji jedan jedini standard o pisanju regularnih izraza. Tako će u raznim aplikacijama jednake stvari biti različito ostvarene, a neke funkcionalnosti nisu dostupne u svim aplikacijama. Na kraju je jedini način bio naći izvor koji govori o regularnim izrazima posebno za program Flex. Mislim da jednom tako rasprostranjenom pojmu treba ozbiljna standardizacija.

Treća je "tehnička" stvar vrlo čudna situacija oko programa *LEX* (a ista je stvar i s *Yacc*-om). Naime, pretraživanjem Interneta nemoguće je naći službenu web-stranicu, firmu koja drži prava na program, čak niti mogućnost preuzimanja programa. Nakon što sam saznao da zapravo želim koristiti Flex, bilo je teško saznati kakve veze Adobe ima s tim, zašto se dva različita programa jednako zovu, te zašto je potpora za Windows tako loša.

Nakon što sam prevazišao ova tri problema, muku su mi još samo zadavali malo "zahrđalo" znanje *C*-a i nedovoljno definirana granica između leksičkog i sintaksnog analizatora. Bilo kako bilo, moj leksički analizator obavlja većinu zadaća koje treba – možda ne pretjerano učinkovito ili dovoljno detaljno ili za točno određenu verziju Fortrana – ali obavlja ih.