Sveučilište u Zagrebu

Fakultet elektrotehnike i računarstva

kate

**Samostalni studentski projekt iz predmeta**

**„Prevođenje programskih jezika“**

Zadatak broj 2050.

Zagreb, lipanj 2010.

# Samostalni studentski projekt iz predmeta “Prevođenje programskih jezika”

**Student:** kate

**Matični broj studenta:**

**Zadatak broj 2050:** Pomoću programa za generiranje leksičkih analizatora (LEX ili slični alat) ostvariti leksičku analizu jezika CSV (engl. *Comma-Separated Values*). Potrebno je podržati osnovne konstrukte jezika, te riješiti nejednoznačnosti koje se pojavljuju u leksičkoj analizi. Dodatno, potrebno je pomoću akcija ograničiti korištenje cjelobrojnih brojeva na brojeve do 10 znamenki te ispisati najveći cijeli broj u dokumentu. Izlaz iz programa su četiri tablice leksičke analize. Samostalno odrediti leksičke jedinke potrebne u sintaksnoj analizi. Dokumentaciji priložiti izvorni kod i izvodivu verziju analizatora kao i nekoliko pripremljenih primjera kojima se analizator može isprobati.

# UVOD

### Leksička analiza

Leksička analiza je prvi korak u radu jezičnog procesora. To je linearna analiza znakova izvornog programa koja provjerava jesu li leksičke jedinke u izvornom programu pravilno napisane. Leksički analizator čita znakove izvornog programa, grupira ih u leksičke jedinke, zapisuje parametre u tablice znakova i generira niz uniformnih znakova. Leksičke jedinke razvrstane su prema klasama: ključne riječi, operatori, specijalni znakovi, konstante i identifikatori. Leksičke jedinke u istoj klasi imaju zajednička pravila koja određuju koji su dozvoljeni oblici za tu klasu.

Rezultat leksičke analize su četiri tablice: tablica uniformnih znakova, tablica identifikatora, tablica konstanti i tablica ključnih riječi, operatora i specijalnih znakova. Ove tablice su dostupne ostalim koracima rada jezičnog procesora koji nemaju izravan pristup tekstu izvornog programa.

Tablica uniformnih znakova je osnovna tablica leksičke analize. U toj tablici su upisani svi uniformni znakovi onim redoslijedom kojim su leksičke jedinke zadane u izvornom programu. Jedan zapis u tablici sadrži uniformni znak leksičke jedinke i kazaljku koja pokazuje na mjesto zapisa leksičke jedinke u odgovarajućoj tablici.

U ostalim tablicama za svaku leksičku jedinku postoji točno jedan zapis koji se sastoji od rednog broja i same leksičke jedinke. Ostale informacije o lesksičkim jedinkama se dodaju u daljnjim koracima rada jezičnog procesora.

### LEX (Lexical Analyzer Generator)

LEX je program koji omogućava automatiziranu izgradnju leksičkog analizatora. LEX učitava datoteku sa definicijama leksičkih pravila (*lex.l*), obrađuje ih i stvara datoteku sa kôdom u programskom jeziku C (*lex.yy.c*). Ta datoteka se kopira i prevodi u jednom od C prevodioca, i tako se dobiva izvršna datoteka spremna za korištenje. Format datoteke *lex.l* je sljedeći:

*{definicije}*

*%%*

*{pravila prevođenja}*

*%%*

*{pomoćne procedure}*

Područje za **definicije** može sadržavati tri dijela:

1. C kôd – tekst napisan između graničnika %{ i %} program LEX izravno prepiše na početak datoteke lex.yy.c
2. Regularni izrazi – ovdje navedeni izrazi mogu se koristiti u pravilima prevođenja tako da se samo navede njihovo ime.
3. Stanja – ako je potrebno razrješavanje nejednoznačnosti pretraživanjem lijevog konteksta leksički analizator mora imati stanja. Ovdje se ta stanja inicijaliziraju, tako da se na početku reda napiše *%s* i nakon toga se navedu imena stanja.

Područje za **pravila prevođenja** se može gledati kao tablica u kojoj prvi stupac predstavlja regularni izraz, a drugi akciju koja se izvrši ako je izraz prepoznat. Akcija može biti jedna naredba C-a ili više naredbi zatvorenih vitičastim zagradama. Omogućeno je razrješavanje nejednoznačnosti pretraživanjem lijevog ili desnog konteksta. Područje za pravila prevođenja započinje i završava znakovima *%%*.

U područje za **pomoćne procedure** pišu se funkcije pomoću kojih se grade akcije koje su dio pravila prevođenja. Tekst napisan u ovom dijelu izravno se prepisuje u datoteku lex.yy.c.

Glavni program može biti napisan u području definicija ili pomoćnih procedura, a u glavnom programu se poziva leksički analizator pozivom funkcije *yylex()*.

### CSV (Comma Separated Values)

CSV je jednostavan format koji se koristi za zapisivanje tablica baze podataka u datoteku. Svaki zapis u tablici je jedna linija teksta u datoteci, a svaka vrijednost polja je od sljedeće vrijednosti odvojena zarezom. Format CSV je široko podržan pa se često koristi za prijenos tabličnih podataka među različitim računalnim programima koji ga podržavaju. CSV datoteka se često koristi prilikom prebacivanja informacija iz baze podataka u tablični proračun. Format CSV podržavaju gotovo svi programi za tablični proračun i sustavi za upravljanje bazom podataka, a podržavaju ga i mnogi programski jezici.

Ne postoji jedinstveni standard za CSV format, pa se može naići više inačica tog formata. Sljedeća pravila odnose se na najčešće korištenu formu CSV-a:

* Stupci tablice se odvajaju zarezom, a redovi znakom za novi red (\n). Zapisi mogu i ne moraju biti zatvoreni dvostrukim navodnicima

*0036434202,Ana Sovic,2.R1*

*“0036543043“,“Ivan Anic“,“2.E1“*

* Da bi polje moglo sadržavati specijalne znakove (zarez, znak za novi red i dvostruke navodnike) mora biti zatvoreno dvostrukim navodnicima.

*0036434202,Ana Sovic,“2.R1, 2.R4*

*2.E2“*

* Ako je unos za neko polje prazan niz on mora biti zatvoren dvostrukim navodnicima.

*0036434202,““,2.R1*

* Ako vrijednost polja sadrži dvostruke navodnike oni moraju biti dupli (““).

*“0036434202“,“Ana ““Velika““ Sovic“,“2.R1“*

* Praznine su sastavni dio zapisa i nesmiju biti ignorirane.

*0036434202, Ana Sovic , 2.R1* nije isto ovome *0036434202,Ana Sovic,2.R1*

# OSTVARENJE

Ostvareni leksički analizator čita izvorni kod iz ulazne datoteke i vrši leksičku analizu nad danim kodom. Analizator razlikuje dvije vrste leksičkih jedinki: KROS (ključne riječi, operatori i specijalni znakovi) i KON (konstante). Ne razlikuje identifikatore jer jezik koji se prevodi ne sadrži identifikatore nego samo specijalne znakove (zarez, znak za novi red, dvostruki navodnici) i konstante.

Ulaz u program je datoteka ulazna, a izlaz su datoteke „kros.txt“ koja sadrži tablicu ključnih riječi, operatora i specijalnih znakova, „kon.txt“ koja sadrži tablicu konstanti, i „uniformni\_znakovi.exe“ koja sadrži tablicu uniformnih znakova. Zapis u tablici uniformnih znakova se sastoji od tipa uniformnog znaka i rednog broja njegovog zapisa u tablici znakova, a zapisi u tablicama znakova se sastoje od rednog broja i samog znaka.

Na početku datoteke nalaze se regularne definicije koje opisuju leksičke jedinke koje su dopuštene u izvornom programu.

*broj [0-9]+*

*predznak [-+]*

*integer {predznak}?{broj}*

*float {predznak}?{broj}\.{broj}*

*znak [^,\n"]+*

*znakUNavodnicima \"[^"]\*\"*

*praznina [ \t]+*

Regularni izrazi integer i float opisuju brojčane konstante, a izrazi znak i znakUNavodnicima opisuju znakovne konstante. Specijalni znakovi nisu definirani u ovom dijelu nego su napisani direktno u pravilima prevođenja.

*Integer* je cijeli broj sa proizvoljnim brojem znamenaka, koji može ali ne mora imati predznak. *Float* je realni broj koji može imati predznak, i mora imati cijeli i decimalni dio odvojen znakom točke. U nekim verzijama formata CSV umjesto točke se stavlja zarez, ali se u tom slučaju riječi odvajaju znakom dvotočke. Međutim ovaj leksički analizator podržava format u kojem se točka koristi za decimalne brojeve, a zarez za odvajanje polja tablice.

Regularni izraz *znak* pokriva sve konstante koje se mogu koristiti ako nisu ograničene dvostrukim navodnicima. U tom slučaju konstanta ne smije sadržavati niti jedan specijalni znak (zarez, znak za novi red i dvostruki navodnik). *ZnakUNavnodnicima* predstavlja konstantu koja je zatvorena dvostrukim navodnicima. Takva konstanta smije sadržavati zarez i znak za novi red, ali ne smije sadržavati dvostruke navodnike. Moguće je da nema konstante nego se leksička jedinka sastoji samo od para dvostrukih navodnika, što označava prazno polje. Ako konstanta zatvorena navodnicima sadržava dvostruke navodnike kao znakove konstante, oni moraju biti dupli prema pravilima formata CSV. U tom slučaju će leksički analizator prepoznati tri niza, odnosno tri leksičke jedinke, umjesto jedne.

Regularni izraz *praznina* koristi se za izbacivanje praznih znakova oko navodnika. Ako se na početku ili na kraju konstante nalaze praznine i ako konstanta nije označena navodnicima, onda se i te praznine uzimaju kao dio konstante. Ako je konstanta označena navodnicima onda se praznine oko nje moraju izbaciti.

Osim regularnih definicija u području definicija se nalaze inicijalizacije potrebnih varijabli, prototipi kasnije korištenih funkcija i implementacija funkcije main().

Nakon definicija dolaze pravila prevođenja.

*, |*

*\n {pretraziKROS(yytext);}*

Ako je pronađena leksička jedinka specijalni znak onda je tu jedinku potrebno dodati u tablicu ključnih riječi, operatora i specijalnih znakova (ako već nije dodana) i u tablicu uniformnih znakova. Dodavanje znaka u tablicu uniformnih znakova izvršava se u funkciji *dodajUnifromni(char \*tip, int rBr)*. Kako bi se znalo nalazi li se ta leksička jedinka već u tablici KROS pretražuje se datoteka *kros.txt*, i pronađena leksička jedinka se uspoređuje sa svim jedinkama zapisanim u tablici. Taj postupak se izvršava u funkciji *pretraziKROS(char \*s)*. Ako traženi zapis još ne postoji, on se dodaje u funkciji *dodajKROS(char \*s).*

*{praznina}/[\n,"] {;}*

Ranije je već objašnjeno za što se koristi regularni izraz praznina. Da bi leksički analizator znao kada treba izbaciti praznine, a kada su one dio konstanti koristi se pretraživanje desnog konteksta. Ako se nakon praznina naiđe na zarez ili znak za kraj reda znači da se ta praznina nalazi iza konstante. Ako se iza praznine nalazi dvostruki navodnik onda se ona nalazi ispred konstante. Kada se pronađe praznina ne događa se ništa, što znači da se ta praznina zanemaruje.

*{integer} {cijeliBroj(yytext, yyleng);}*

*{float} |*

*{znak} |*

*{znakUNavodnicima} {dodajUniformni("KON ", brKON++); dodajKON(yytext);}*

Ako je pronađena leksička jedinka konstanta postupak je isti kao i za specijalne znakove, samo se dodavanje zapisa obavlja u funkciji dodajKON(char \*s). Ako je pronađena konstanta cijeli broj osim uređivanja tablica potrebno je i provjeriti broj znamenaka i izbaciti brojeve sa više od deset znamenki. Uz to program mora ispisati najveći cijeli broj pronađen u izvornom programu. Akcije specifične za cijele brojeve implementirane su u funkciji *cijeliBroj(char \*s, int brZnamnaka)*.

*void cijeliBroj(char \*s, int brZnamenki)*

*{*

*if (brZnamenki < 10)*

*{*

*dodajUniformni("KON ",brKON++);*

*dodajKON(yytext);*

*int broj = atoi(s);*

*if (broj > maxCijeliBroj)*

*maxCijeliBroj = broj;*

*}*

*else*

*printf("Greska! Cijeli broj %s ima vise od 10 znamenki.\n", s);*

*}*

Funkciji se preko parametara predaje pronađeni cijeli broj, međutim ne kao broj nego kao niz znakova, a uz to se predaje i duljina tog niza. Duljina niza izravno govori o broju znamenki pronađenog broja, međutim da bi taj broj mogli koristit treba ga prebaciti u integer. To radi funkcija atoi(char \*s). Ako pronađeni cijeli broj ima više od deset znamenki on se zanemaruje (ne sprema se u tablice znakova) i ispisuje se poruka o pogrešci. Ako je broj ispravan, odnosno ima manje od deset znamenki, zapisuje se u tablicu znakova ako već nije zapisan, i dodaje se u tablicu uniformnih znakova. Uz to se provjerava i da li je veći od do tada najvećeg pročitanog cijelog broja, ako je onda on postaje najveći pročitani cijeli broj. Ovdje postoji jedno ograničenje, ako je najveći broj u izvornom programu negativan ispis najvećeg broja neće biti ispravan, nego će se ispisati nula. To je iz razloga što se varijabla *maxCijeliBroj* ne može inicijalizirati na negativnu vrijednost.

Nakon što leksički analizator završi sa radom u glavnom programu se ispisuje poruka da je analiza upsješno izvedena i koji je najveći pročitani cijeli broj.

Leksički analizator rađen je pod Windows okruženjem. Da bi se program mogao izvesti na računalu trebaju biti instalirani program Flex i neki prevodilac jezika C. Ako je prevodilac koji se koristi gcc program se izvodi upisivanje sljedećih naredbi u Command prompt:

*flex mojssp.l*

*gcc –o skener lex.yy.c –lfl*

*skener*

Naredbom flex mojssp.l dobivamo datoteku lex.yy.c. Sljedećom naredbom dobivamo izvršnu datoteku skener koju pokrećemo trećom naredbom. Da bi se mogla uspješno izvesti druga naredba potrebno je kopirati biblioteku *libfl.a* iz lib foldera programa flex u lib folder programa gcc.

# ZAKLJUČAK

Izradom laboratorijskih vježbi i ovog projekta dobila sam potpuniju sliku o leksičkoj analizi. Jezik za koji sam u ovom projektu trebala izgraditi leksički analizator je bio na relativno jednostavan. Što se tiče samog jezika najviše problema mi je zadavalo definiranje regularnih izraza za konstante koje su ograničene dvostrukim navodnicima. Bilo je mnogo pokušaja prije nego sam otkrila dosta jednostavne regularne izraze koji će prepoznati sve dozvoljene konstante. Još uvijek je ostao problem navodnika unutar konstante ograničene navodnicima, što izgrađeni jezični procesor prepoznaje kao tri odvojene konstante. Nije ni pokriven oporavak od pogreške, ali bi leksički analizator za ovaj jezik i tako mogao otkriti iznimno mali broj pogrešaka. Večina pogrešaka bi bila prepoznata pod ispravne jedinke.

Za izgradnju leksičkog analizatora sam koristila program Flex. Sa izvođenjem programa je bilo puno više problema nego sa njegovim pisanjem. Na internetu se može naći mnoštvo literature kako se piše program, ali je jako malo podataka o njegovom izvođenju na operacijskom sustavu Windows. Nakon čitanja velikog broja članaka uspjela sam prevesti program u C datoteku, međutim nisam uspjela prevesti tu datoteku pomoću programa MS Visual studio, nego sam instalirala prevodilac gcc.