

Sveučilište u Zagrebu
Fakultet elektrotehnike i računarstva

Mihej Komar

**Seminarski rad iz predmeta
Prevođenje programskih jezika**

Zadatak broj 68

Zagreb, siječanj 2009.

Seminarski rad iz predmeta Prevođenje programskih jezika

Student: Mihej Komar

Matični broj studenta: 0036426836

Zadatak broj 68: Izgraditi program simulator leksičkog analizatora zasnovan na tablici prijelaza ϵ -NKA prema algoritmu iz udžbenika ("Jezični procesori 2", str. 60). Ulaz u program simulator je skup regularnih izraza kojima se opisuju klase dozvoljenih leksičkih jedinki. Radi jednostavnosti pretpostaviti da u jeziku zadanom regularnim izrazima nema nejednoznačnosti. Program simulator treba omogućiti izvođenje simulacije po koracima, te praćenje stanja kazaljki *početak*, *završetak*, *posljednji* i *izraz*.

Uvod

Cilj ovog samostalnog studentskog rada bio je izraditi dio jezičnog stroja – leksičku analizu ulaznog programa. Leksička analiza je prvi korak jezičnog stroja – učitava datoteku koju je korisnik upisao te kao rezultat vraća niz leksičkih jedinki. Najčešće se leksički analizatori kreiraju u obliku DKA automata, a u ovom radu je bio cilj napraviti nešto drugačije ostvarenje, tj. bilo je potrebno koristiti ϵ -NKA automat. Takvi automati se ne koriste jer za svaki prijelaz određuju skup stanja. Prednost nad DKA automatom je manja potrošnja memorije jer je potrebno spremati puno manji broj stanja.

Ostvarenje

Aplikacija je ostvarena pomoću programskog jezika Java. Program se sastoji od više razreda koji se nalaze u više paketa:

- **hr.komar.ppj.ssp.lexer** – paket sadrži razrede koji ostvaruju leksički analizator,
- **hr.komar.ppj.ssp.gui** – paket koji sadrži sve razrede potrebne za implementaciju grafičkog sučelja prema korisniku i
- **hr.komar.ppj.ssp.model** – paket koji sadrži pomoćne objekte koji se koriste u oba paketa.

Leksički analizator









U paketu `hr.komar.ppj.ssp.lexer` se nalaze razredi:

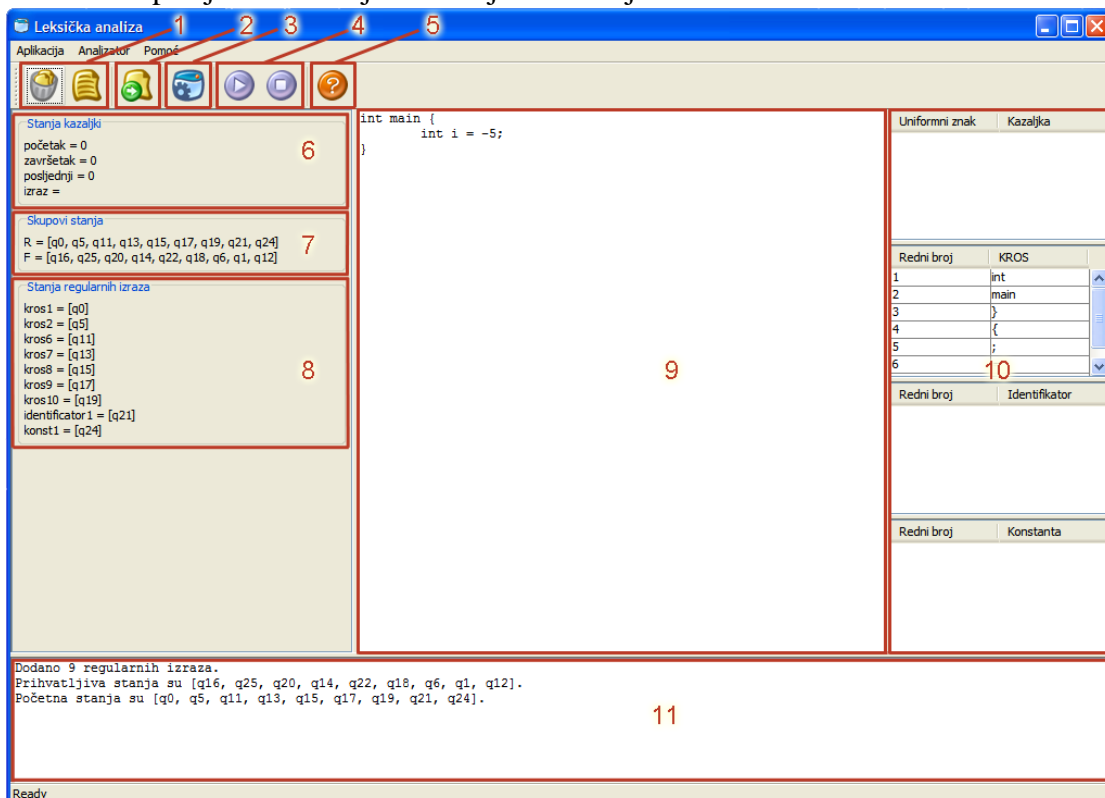
- **LexicalAnalyzer.java** – razred predstavlja leksički analizator te sadrži metode kojima se izvršava analiza upisanog programa. Povezan je s grafičkim sučeljem te time predstavlja „most“ između ostvarenja leksičkog analizatora i grafičkog sučelja. Sadrži sve regularne izraze (`RegularExpression`) u obliku liste.
- **NfaEpsilon.java** – razred predstavlja ϵ -NKA automat i sadrži metode kojima se izvršavaju prijelazi unutar automata. Sadrži sva stanja i ima informaciju koja stanja su početna ili prihvatljiva.
- **NfaEpsilonGenerator.java** – razred se koristi prilikom izgradnje ϵ -NKA automata. Pri inicijalizaciji objekta prima regularni izraz pomoću kojeg se izgrađuje ϵ -NKA automat. Generiranje se radi rekurzivno, tako da se u svakom koraku trenutni regularni izraz dijeli na više jednostavnijih. Nakon generiranja stvara se objekt `NfaEpsilon` koji se koristi u nastavku programa.
- **RegularExpression.java** – razred predstavlja regularni izraz upisan kao ulaz u program. Sprema regularni izraz, ime, tip i generirani ϵ -NKA automat u obliku `NfaEpsilon` objekta.
- **State.java** – razred predstavlja jedno stanje te ima informacije o svim prijelazima. Sadrži i metode kojima se može izračunati ϵ -OKRUŽENJE stanja.
- **Token.java** – razred predstavlja jedan redak KROS tablice, tablice konstanti ili tablice identifikatora.

Regularni izraz se stvara pozivanjem konstruktora razreda `RegularExpression` koji prima ime i tip izraza te sam regularni izraz. Pritom se stvara objekt `NfaEpsilon` koji pak poziva razred `NfaEpsilonGenerator`. Skup svih regularnih izraza se sprema u listu i proslijeđuje se objektu `LexicalAnalyzer`. Time je leksički analizator spreman za rad.

Grafičko sučelje

Grafičko sučelje je izvedeno pomoću alata Swing. Glavni prozor aplikacije je prikazan na slici (Slika 1: Glavni prozor aplikacije). Sastoji se od više dijelova:

1. Gumb  ima funkciju brisanja svih podataka (briše sve regularne izraze te sve vrijednosti postavlja na početne. Drugi gumb () ima funkciju postavljanja svih vrijednosti na pokazne – pokazuju primjer kako program radi.
2. Gumb  otvara novi prozor u koji se upisuju regularni izrazi. Više o njemu slijedi u nastavku.
3. Gumb  pokreće leksički analizator te upisani program analizira do kraja.
4. Gumb  pokreće i nastavlja leksičku analizu u načinu rada *korak po korak*, a gumb  prekida. Nakon što je leksička analiza završena, potrebno je pritisnuti gumb  kako bi program izašao iz načina rada *korak po korak*.
5. Pritiskom na gumb  otvara se pomoć.
6. Na ovom mjestu se ispisuju stanja kazaljki *Početak*, *Završetak*, *Posljednji* i *Izraz* kao što je bilo traženo u tekstu zadatka.
7. Ovdje se ispisuju stanja u kojima je trenutačno leksički analizator (R) i prihvatljiva stanja (F).
8. Prikazuje koja od stanja, u kojima je trenutačno leksički analizator, pripadaju kojem regularnom izrazu.
9. Upisuje se ulazni program koji se leksički analizira. Za vrijeme rada *korak po korak* nije moguće mijenjati upisane vrijednosti.
10. Ovdje se ispisuju rezultati leksičke analize u obliku tablica (tablica uniformnih znakova, KROS znakova, identifikatora i konstanti).
11. Tu se ispisuju informacije o obavljenim radnjama.



Slika 1: Glavni prozor aplikacije

Primjer prozora za unos regularnih izraza je prikazan je na slici (Slika 2: Prozor za unos regularnih izraza). Sastoji se od više dijelova:

1. Popis upisanih regularnih izraza. U popisu se prikazuje ime, tip i sam regularni izraz.
2. Dodavanje novog regularnog izraza se sastoji od upisivanja imena (manja kućica), regularnog izraza (veća kućica), odabira tipa regularnog izraza i pritiska na gumb Dodaj.
3. Brisanje postojećih regularnih izraza je vrlo jednostavno – sastoji se od odabira regularnog izraza i pritiska na gumb Makni.
4. Nakon što je korisnik gotov s upisom regularnih izraza treba pritisnuti gumb Spremi i završi, čime su regularni izrazi spremi za uporabu.

Ime	Regularni izraz	Tip
kros1	int	KROS
kros2	main	KROS
kros6	}	KROS
kros7	{	KROS
kros8	;	KROS
kros9	=	KROS
kros10	-	KROS
identifikator1	(a b c d e f g h i j k l m n o p...	Identifikator
konst1	(0 1 2 3 4 5 6 7 8 9)*	Konstanta

Dodavanje novog regularnog izraza

Ime: Regularni izraz: Tip: ☒ KROS ☐ Identifikator ☐ Konstanta

Brisanje regularnog izraza

Slika 2: Prozor za unos regularnih izraza

Detaljnije informacije o svakom razredu i njihovim metodama može se vidjeti u *javadoc* dokumentaciji koja se može naći u datoteci `javadoc/index.html`.

Problemi

Prilikom stvaranja aplikacije bilo je više problema. Jedan od najvećih problema bio je kako izraditi ϵ -NKA automat, a riješen je izmišljanjem vlastitog algoritma koji se pokazao dobrim za jednostavniju primjenu. Ne troši previše memorije, dovoljno je brz pa se ne primjećuje njegovo izvršavanje (stvara se kad korisnik pritisne `Dodaj` u prozoru za dodavanje regularnih izraza). Resursi korišteni pri generiranju se u kasnijem radu s programom ne koriste pa ih *garbage collector* kasnije izbriše. Drugi veći problem je bio provođenje algoritma prema algoritmu iz udžbenika (“Jezični procesori 2”, str. 60). S obzirom na to što u mojem algoritmu praznine, nove redove i tabulatore mijenjam u jedinstveni znak, a ne samo da ih brišem, bilo je potrebno izvršiti manje modifikacije algoritma. U osnovi ostao je jednak, a rezultat je bolji. Ostali problemi s kojima sam se susreo nisu predstavljali veći vremenski gubitak.

Zaključak

Budući da sam ovaj samostalni studentski projekt odlučio napraviti što temeljitije i praktičnije te sam u njega uložio znatan trud, rezultat je opsežniji i kvalitetniji program koji radi kompletnu leksičku analizu temeljenu na ϵ -NKA automatu. U grafičkom sučelju se prikazuju svi relevantni podaci – vrijednosti najvažnijih kazaljki, stanja u kojima je automat i generirane tablice. Ima preko 2000 linija koda (preko 70000 znakova), kod je pisan samodokumentirajuće, a postoji i *javadoc* dokumentacija.