XXXI JUGOSLOVENSKA KONFERENCIJA ETAN-a, B L E D , 1—5. JUNA 1987.

Tanko Zita ,
Osnovna Privredna banka Pančevo

Gajić Mihajlo ,
Osnovna Privredna banka Pančevo

MODEL INTERACTIVNOG KALKULATORA REALIZOVAN KAO SOFTVERSKI INTRPRETER

A MODEL OF INTERACTIVE CALCULATOR PRODUCED AS A SOFTWARE INTERPRETER

SADRŽAJ : U ovom radu predstavljen je interaktivni kalkulator realizovan kao softverski interpreter. Napisan je u višem programskom jeziku i interpretira i evaluira aritmetičke izraze unutar zadate gramatike. Prezentiran metod je moguće upotrebiti za izradu interpretera bilo koje gramatike na osnovu pravila te gramatike.

ABSTRACT : In this paper is presented a interactive calculator produced as a software interpreter . It is written in high level language and interprets and evaluates arithmetic expressions derived by given grammar . Method presented is also usable in producing interpreters for any grammar based on grammar rools.

1. UVOD

Razlog za realizaciju ovakvog jednog softverskog proizvoda ležao je u potrebi da se na terminalima računara IBM 4331 korisnicima da pomoćno sredstvo u vidu kalkulatora pomoću kojeg bi mogli obavljati izračunavanja vrednosti aritmetičkih izraza. Posebno je ova potreba bila naglašena u službama koje su raspolagale terminalom i time bile u mogućnosti korišćenja računara a ipak su bile prinudjene da izračunavanja obavljaju džepnim kalkulatorima.

Iz navedenog razloga otpočelo je koncipiranje softverskog produkta koji bi na računaru IBM 4331 simulirao ponašanje hardverskih IC u džepnom digitronu. Koncipiranje takvog produkta dovelo je do potrebe proučavanja s jedne strane formalnih jezika , a s druge strane interpretera koji interpretiraju te jezike. Iz tog proučavanja i primene odgovarajućih teorijskih postavki u praksi nastao je predmet ovoga rada.

Pri definisanju kalkulatora kao interpretera aritmetičkih izraza pošlo se od teorije formalnih jezika . Poznato je da se formalni jezik može definisati skupom pravila iz kojih se deriviraju svi iskazi tog jezika. Lako je pokazati da se jezik čiji su iskazi aritmetički izrazi derivira iz sledećeg skupa pravila :

```
1) <E> ==> <T><L>          9) <P> ==> (<E>)
2) <E> ==> - <T><L>       10) <P> ==> $
3) <T> ==> <F><M>         11) <P> ==> K
4) <L> ==> $              12) <N> ==> $
5) <L> ==> <A><T><L>      13) <N> ==> S<P><N>     (1)
6) <M> ==> $              14) <A> ==> +
7) <M> ==> <x><F><M>      15) <A> ==> -
8) <F> ==> <P><N>         16) <x> ==> *
                          17) <x> ==> /
```

Na osnovu toga pri izradi kalkulatora vodjeno je računa da kalkulator evaluira sve sintaksno ispravne iskaze jezika deriviranog iz navedenih gramatičkih pravila. Da bi se obavilo izračunavanje vrednosti aritmetičkih izraza koji mogu biti ulaz u ovaj kalkulator , potrebno je najpre utvrditi da zadati izraz pripada skupu izraza deriviranom iz zadate gramatike. Ovo ispitivanje predstavlja zapravo izvršavanje dveju analiza nad ulaznim nizom - leksičke i sintaksne . Ove analize se mogu izvršiti na više načina. U slučaju kalkulatora odabran je način izvrsavanja analiza putem odgovarajućih konačnih automata.

Ovakav pristup je odabran zbog toga što je postupak : jednostavan , dozvoljava generalizaciju i pogodan je za realizaciju na računaru.

U nastavku je izložen način realizacije kalkulatora , način njegovog funkcionisanja kao i glavni moduli iz kojih se sastoji.

2. DELOVI KALKULATORA

Radi utvrdjivanja sintaksne ispravnosti u kalkulatoru se najpre izvršava leksička analiza zadatog aritmetičkog izraza , a zatim i sintaksna analiza. Na taj način pre no što se utvrdi da li je zadati izraz moguće evaluirati i pre no što se pristupi izračunavanju njegove vrednosti , aritmetički izraz se podvrgava leksičkoj a zatim sintaksnoj analizi. Ako se tokom tih analiza ne detektuje ni jedna greška - kalkulator izračunava vrednost izraza i prikazuje je na ekranu . Ako se detektuje greška , kalkulator izdaje odgovarajuću poruku i omogućuje korekciju izraza. Način korišćenja kalkulatora je interaktivan .

Da bi se izvršile pomenute aktivnosti u kalkulatoru on je struktuiran modularno i sastoji se od : modula za leksičku analizu (LA) , modula za sintaksnu analizu (SA) , modula za izračunavanje vrednosti izraza (EV) i kontrolno monitorskog modula koji obavlja I/O kao i kontrolu i koordinaciju rada ostalih modula.

Navedeni moduli izdvojeni su kao zasebne celine ali su ipak u tesnoj medjusobnoj sprezi. Najsamostalniji od njih je modul (LA) dok su modul (SA) i (EV) zbog odabranog metoda za njihovu realizaciju daleko više zavisni jedan od drugog.

2.1. AUTOMAT ZA LEKSIČKU ANALIZU

Modul (LA) realizovan je kao konačni automat sa definisanim ulazom , izlazom , stanjima i odgovarajućom funkcijom prelaza. Ulaz u leksičku analizu je ispisani niz simbola a izlaz niz detektovanih obeležja (token) ili poruka o leksičkoj neispravnosti ulaznog niza.

Radi ilustracije rada automata za leksičku analizu data je slika 1. na kojoj je prikazana funkcija prelaza. U prvom redu prikazane tabele nalaze se redom svi regularni ulazni simboli (U(j)). U prvoj koloni tabele nalaze se simbolički označena sva stanja u kojima se automat može naći (S(i)). Unutar tabele su date uredjene dvojke (A(i),B(j)) u kojima prvi član ukazuje na novo stanje u koje automat prelazi u slučaju da se nalazi u stanju S(i) i da na ulazu detektuje ulazni simbol U(j). Drugi član B(j) označava da li se prilikom ovog prelaska iz jednog stanja u drugo javlja izlaz (OUTPUT) ili ne. Stanja su označena simboličkim oznakama (skraćenice) , prisustvo izlaza sa OUT a odsustvo sa NOT.

Na primer razmotricemo situaciju kada prikazani automat analizira ulazni niz 25+3.2 . Automat se najpre nalazi u početnom stanju (PS). Ulaz prve cifre prevodi taj automat u stanje broj

(B).(U vrsti u kojoj je stanje (PS) i koloni u kojoj su cifre član tabele je (B,NOT)). Prilikom ove promene stanja nema

	zagrada	cifra	tacka	znak	zvezda
P S	ZA , NOT	B , NOT	DB , NOT	ZN , NOT	ZN1, NOT
Z A	ZA , OUT	B , OUT	DB , OUT	ZN , OUT	ZN1, OUT
B	ZA , OUT	B , NOT	DB , NOT	ZN , OUT	ZN1, OUT
D B	ZA , OUT	DB , NOT	TA , OUT	ZN , OUT	ZN1, OUT
Z N	ZA , OUT	B ,OUT	DB , OUT	ZN , OUT	ZN1, OUT
Z N 1	ZA , OUT	B , OUT	DB , OUT	ZN , OUT	ST , NOT
S T	ZA , OUT	B , OUT	DB , OUT	ZN , OUT	ZN1, OUT
T A	ZA , OUT	DB , NOT	TA , OUT	ZN , OUT	ZN1 ,OUT
S T O P	STOP, OUT	STOP,OUT	STOP,OUT	STOP,OUT	STOP,OUT

slika 1.

nikakvog izlaza (ulazna cifra se čuva u internom baferu). Ulazak sledeće cifre dok je automat u stanju (B) ne menja stanje i ne izaziva izlaz. Nailazak znaka + izaziva promenu stanja uz izdavanje izlaza (u tabeli odgovarajući član je (Z,OUT)), te se automat nakon toga nalazi u stanju znak (Z) , a kao izlaz daje obeležje detektovano kao broj. Nailazak sledeće cifre izaziva prelazak u stanje b i izdavanje sadržaja internog bafera (znak +). Nailazak tačke prevodi automat u stanje decimalnog broja (DB)

Matrica MAT (i , j)

	zagrada	cifra	ta\ka	znak	zvezda
1	(2 , 0)	(3 , 0)	(4 , 0)	(5 , 0)	(6 , 0)
2	(2 , 1)	(3 , 1)	(4 , 1)	(5 , 1)	(6 , 1)
3	(2 , 1)	(3 , 0)	(4 , 0)	(5 , 1)	(6 , 1)
4	(2 , 1)	(4 , 0)	(8 , 1)	(5 , 1)	(6 , 1)
5	(2 , 1)	(3 , 1)	(4 , 1)	(5 , 1)	(6 , 1)
6	(2 , 1)	(3 , 1)	(4 , 1)	(5 , 1)	(7 , 0)
7	(2 , 1)	(3 , 1)	(4 , 1)	(5 , 1)	(6 , 1)
8	(2 , 1)	(3 , 1)	(8 , 1)	(5 , 1)	(6 , 1)
9	(9 , 1)	(9 , 1)	(9 , 1)	(9 , 1)	(9 , 1)

slika 2.

takodje bez izdavanja izlaza. Najzad simbol kraja ulaznog niza izaziva prelazak u stanje STOP uz izdavanje sadržaja internog bafera.

Na tom principu obavlja se leksička analiza ma kog ulaznog niza . Uz detekciju i razdvajanje obeležja tokom leksičke analize se utvrdjuju i klase obeležja (operand , operator , zagrada ili decimalna tačka) te se takav niz obeležja dalje predaje na sintaksnu analizu.

Programska realizacija ovog automata je dovoljno jednostavna i može se prikazati sledećim algoritmom (na osnovu slike 2.)

```
stanje = 1 ;
DO i = 1 TO broj karaktera ulaznog niza ;
A :     DO j = 1 TO 18 ;
            IF ulaz(i) = tab(j) THEN LEAVE A ;
        END ;
        IF j > 18 THEN DO ;
             CALL ERROR ("nedozvaoljen ulaz") ;
             RETURN ;
        END ;
        IF M ( stanje , klasa , 2 ) = "1" THEN CALL OUT ;
        stanje = M ( stanje , klasa , 1 ) ;
END ;
```

gde je:

M(i,j,1) prvi član uredjenog para matrice MAT(i,j) a M(i,j,2) drugi član uredjenog para sa napred objašnjenim značenjem.

ulaz( ) - niz ulaznih karaktera (znakova)

tab( ) - niz dozvoljenih ulaznih simbola (zagrade, cifre , znaci operacija +,-, / i * i tačka)

klasa - klasa ulaznog simbola (jedna od navedenih)

ERROR - procedura koja izvestava o greški

OUT - procedura koja izdaje sadržaj internog bafera

Na taj način ulazni niz se prolaskom kroz ovako definisan automat razlaže na obeležja koja se prosledjuju na dalju analizu.

2.2. AUTOMAT ZA SINTAKSNU ANALIZU

U sintaksnoj analizi korišćen je top - down parser, koji polazi od startnog simbola gramatike i višestrukom primenom, predikcija, prevodi ga u ulaznu rečenicu.

Top - down parser za zadatu rečenicu jezika (konkretno aritmetički izraz) izvede njenu levu derivaciju iz startnog simbola i pri tome odredi redosled predikcije (pravila prepisivanja) tj. definiše parsno stablo.

Prilikom same realizacije korišćen je takozvani PREDIKTIVNI PARSER s obzirom da je kalkulator realizovan na PL/I jeziku koji ne podržava rekurziju, pa je bilo neophodno napraviti parser sa eksplicitnim stekom, odnosno prediktivni parser.

Prediktivni parser za svoj rad koristi sledeće komponente:

Bafer ulazne rečenice (B) - koji je koji je formiran prilikom leksičke analize i u koji se smešta ulazna rečenica sa oznakom $ na kraju rečenice.

Stek gramatičkih simbola (SY) - u kome se drže rečenični oblici (derivacije) koji se dobijaju u toku sintaksne analize, s tim da je najleviji simbol uvek na vrhu steka. Stek je inicijalizovan $ koji označava dno steka.

Parsing matrica P (I,L) sadrži informacije o gramatici. Svi

neterminali I (koji se nalaze na vrhu steka gramatičkih simbola SY) odgovaraju redovima parsing matrice, dok terminali L (na koje ukazuje pokazivač ulaznog bafera) odgovaraju kolonama date matrice. Ukoliko se na preseku date vrste i kolone date matrice nalazi neki broj , tada taj broj označava redni broj predikcije koju treba primeniti u suprotnom prekida se sintaksna analiza, jer ulazna rečenica ne pripada datom jeziku.

2.3. AUTOMAT ZA EVALUACIJU IZRAZA

Automat za evaluaciju izraza realizovan je korišćenjem inverzne Poljske notacije postavljajući operator u nizu što je kasnije moguće tj. takozvanom notacijom ("KASNI OPERATOR"). Prilikom prevodjenja standardne notacije u inverznu Poljsku notaciju korišćen je stek (lifo lista) u koji se smeštaju operatori, dok se u drugom steku čuvaju operandai nad kojima se vrši evaluacija.

Evaluacija se vrši parcijalno u momentu kada iz steka izlazi poslednji upisani operator, u zavisnosti da li je operator unarni ili binarni , primenjuje se nad poslednjim odnosno poslednja dva operanda. Izračunata vrednost postaje novi operand koji zamenjuje prethodni odnosno prethodna dva operanda.

Na ovaj način se praktično standardna notacija ne prevodi u inverznu Poljsku notaciju već se za evaluaciju izraza koristi tehnika inverzne Poljske notacije.

Na osnovu gramatike definisane pravilima (1) i prediktivnog parsera datog slikom 3. analiziraćemo rečenicu K + K uz pomoć tabele date slikom 4. vršeći sintaksnu analizu i odamah zatim evaluaciju izraza. Prva kolona tabele na slici 4. označava redni broj koraka u sintaksnoj analizi,druga kolona prikazuje stek gramatike simbola (SY) ,treća kolona prikazuje bafer ulazne rečenice (B) s tim što podvučeno obeležje označava token koji se analizira, četvrta kolona prikazuje redni broj predikcije koju treba primeniti a peta samu predikciju.

Prvi korak - u na vrhu (SY) nalazi se startni simbol gramatike E analizira se obeležje K , na osnovu prediktivne matrice slika 3. vidi se da u preseku vrste koja počinje sa E i kolone sa obeležjem K stoji redni broj predikcije koju treba primeniti.

Drugi korak - startni simbol E vadi se sa vrha steka i zamenjuje primenjenom predikcijom u prethodnom koraku s tim što se najlevlji simbol stavlja na vrh steka.Primenjuje se predikcija 5.

Treći korak - neterminal T vadi se sa vrha steka i zamenjuje primenjenom predikcijom u prethodnom koraku.Primenjuje se predikcija 8.

Četvrti korak - neterminal F vadi se sa vrha steka i zamenjuje primenjenom predikcijom u prethodnom koraku.Primenjuje se predikcija 12 prilikom koje neterminal postaje terminal. Sada kad smo identifikovali obeležje K pozivamo modul koji vrši evaluaciju izraza (EV). Ovaj modul prepoznaje da je K operand i smešta ga u stek nakon čega ovaj modul vraća kontrolu sintaksnom analizatoru

Peti korak -neterminal P vadi se sa vrha steka i analizira se sledeće obeležje +, primenjuje se predikcija 10.

Šesti korak - neterminal N vadi se sa vrha steka ,primenjuje se predikcija 7.

Sedmi korak - neterminal M vadi se sa vrha steka ,primenjuje se predikcija 3.

Osmi korak - neterminal L vadi se sa vrha steka i zamenjuje primenjenom predikcijom u prethodnom koraku.Primenjuje se predikcija 14 prilikom koje neterminal postaje terminal. Sada kad

	(	)	+	-	*	/	S	K	$
E	1	-	-	2	-	-	-	1	-
T	5	-	-	-	-	-	-	5	-
L	-	4	3	3	-	-	-	-	4
M	-	7	7	7	6	6	-	-	7
F	8	-	-	-	-	-	-	8	-
P	11	13	13	13	13	13	13	12	13
N	-	10	10	2	10	10	9	-	10
A	-	-	14	15	-	-	-	-	-
X	-	-	-	-	16	17	-	-	-

slika 3.

KORAK	SY	B	M	PREDIKCIJA
1	$E	K+K$	1	E => TL
2	$LT	K+K$	5	T => FM
3	$LMF	K+K$	8	F => PN
4	$LMNP	K+K$	12 (EV)	P => K
5	$LMN	K+K$	10	N => $
6	$LM	K+K$	7	M => $
7	$L	K+K$	3	L => ATL
8	$LTA	K+K$	14 (EV)	A => +
9	$LT	K+K$	5	T => FM
10	$LMF	K+K$	8	F => PN
11	$LMNP	K+K$	12 (EV)	P => K
12	$LMN	K+K$	10	N => $
13	$LM	K+K$	7	M => $
14	$L	K+K$	4	L => $
15	$			

slika 4.

smo identifikovali obeležje + pozivamo modul koji vrši evaluaciju izraza (EV). Ovaj modul prepoznaje da je + operator i smešta ga u stek nakon čega ovaj modul vraća kontrolu sintaksnom analizatoru
Koraci 9, 10, 11, 12, 13 identični su sa koracima 2, 3, 4, 5, 6

U petnaestom koraku primenom predikcije 4 u steku SY ostaje simbol koji označava dno steka, čime je uspešno završena sintaksna analiza nakon čega se poziva modul (EV) koji vrši evaluaciju izraza nad preostalim operandima primenom operatora koje uzima redom sa vrha steka čime se izvrši evaluacija izraza.

3. ZAKLJUCAK

Kao što se iz izloženog može videti primenjen metod za razvoj kalkulatora je u prilićnoj meri formalizovan te bi mogao dati odgovarajuće rezultate i kad bi se primenio na ma koji drugi model gramatike definisan na način kako je definisana gramatika u ovom slučaju. Ako bi se za neku drugu gramatiku definisala drugačija leksička i sintaksna pravila (ali takodje matricom prelaza , predikcijama itd.) to bi ujedno bilo i sve što bi trebalo izmeniti a da prikazani automat funkcioniše i daje korektne rezultate.
Iz svega iznetog može se zaključiti da je u ovom radu prezentiran jedan metod za razvoj interpretera iskaza zadate gramatike koji je u znatnoj meri formalizovan i automatizovan. Posebno su ove osobine prisutne u modulu za leksičku analizu. Uvedena formalizacija i automatizacija omogućuju odredjenu generalizaciju tj. primenu istog metoda i na druge gramatike pod uslovom da su zadate na isti način. Ograničenja koja postoje su pre svega prisutna u delu evaluacije vrednosti. U ovom primeru obradjivana gramatika bila je gramatika aritmetičkih izraza sa uobičajenim računskim operacijama. U slucaju primene metoda na neku drugu gramatiku deo za evaluaciju bi se morao prilagoditi pravilima evaluacije u toj gramatici.

4. LITERATURA

/1/ Gries D. : "Compiler Design for Digital Computers", John Wiley and sons, New York , 1976

/2/ Brown P.J. :"Writing Interactive Compilers and Interpreters", John Wiley and sons , New York , 1981