Aleksa Todorović

Programski jezik za programiranje na prirodnom jeziku

Program translira komande unete na engleskom jeziku na konstrukcije programskog jezika PASCAL. Program vrši analizu unetog teksta i, u slučaju da ga je uspešno analizirao, vrši njegovo transliranje na PASCAL. Program omogućava proširenje gramatike i rečnika kojima raspolaže. Ovde su opisani sledeći elementi programa: algoritam za analizu unetog teksta, memorisanje gramatike, fajlovi koji se koriste pri radu, a dato je i uputstvo za korišćenje programa.

Uvod

Postoji veliki broj problema za čije je rešavanje potrebno poznavanje nekog programskog jezika ili posedovanje neke aplikacije koja će omogućiti rešavanje datog problema. Međutim, aplikacije uglavnom nisu univerzalnog karaktera tj. ne mogu da rešavaju sve probleme, a za upoznavanje programskog jezika potrebno je dosta vremena. To je i razlog zbog kojeg sam odlučio da napravim program koji bi bio što bliži programskom jeziku a za čije programiranje ne bilo potrebno veliko znanje. Ovaj program predstavlja pokušaj da se korisniku omogući rešavanje problema programiranjem na prirodnom (ovde engleskom) jeziku.

Metod Program je pravljen na programskom jeziku PASCAL. Na početku rada program može samo da izvrši tokenizaciju teksta tj. nema mogućnost transliranja unetog teksta na PASCAL kod. Međutim, tokom rada program traži objašnjenje za nepoznate reči/rečenice što omogućava proširenje njegove gramatike. Tako sam došao do nivoa na kome je moguće da se u jednoj rečenici definiše neki niz i nađe njegov najveći/najmanji elemenat. Gramatika kojom program raspolaže se nalazi u fajlu ENGLISH.LNG. Program se sastoji iz tri dela. Prvi deo vrši tokenizaciju teksta i promenu njegovog zapisa u oblik koji je lak za dalju obradu. Drugi deo programa je najvažniji deo programa koji vrši analizu teksta i proširenje gramatike. Treći deo pravi PASCAL kod datog programa.

Aleksa Todorović (1979), Sremski Karlovci, Gavrila Principa 4, učenik 2. razreda Gimnazije „Jovan Jovanović Zmaj" u Novom Sadu

Gramatika (opis sintakse i semantike)

Gramatika koju koristi ovaj program nalazi se u fajlu ENGLISH.LNG. Njena sintaksa, opisana EBNF notacijom [1], izgleda ovako:

```
$GRAMATIKA = {DEFINICIJA-TIPA} "!"
 $DEFINICIJA-TIPA = "&" OPIS-TIPA {OPIS-TIPA}
  $IME-TIPA = REC
 $OPIS-TIPA = "=" SINTAKSA-TIPA SEMANTIKA-TIPA "."
  $SINTAKSA-TIPA = SINTAKSNA-RECENICA
   $SINTAKSNA-RECENICA = SINTAKSNA-REC {SINTAKSNA
                         -REC}
    $SINTAKSNA-REC = OPISNA-REC
  $SEMANTIKA-TIPA = {SEMANTICKA-RECENICA} "@"
   $SEMANTICKA-RECENICA = PASCAL-OPIS | PREVOD
    $PASCAL-OPIS = "$" PASCAL-REC {PASCAL-REC}
     $PASCAL-REC = (""" PASCAL-DEFINICIJA """) | TIP
                   -RECI
      $PASCAL-DEFINICIJA = <definicija labele, tipa,
                                konstante ili promen-
ljive,
                           ili njen deo
    $PREVOD = REC-PREVODA {REC-PREVODA}
     $REC-PREVODA = (""" PASCAL-KOD """) | TIP-RECI
      $PASCAL-KOD = <bilo koja naredba u TURBO
                    PASCAL-u, ili njen deo>
$REC = SLOVO {SLOVO} " "
 $SLOVO = "A" | "B" | "C" | "D" | "E" | "F" | "G" |
          "H" | "I" | "J" | "K" | "L" | "M" | "N" |
          "O" | "P" | "Q" | "R" | "S" | "T" | "U" |
          "V" | "W" | "X" | "Y" | "Z"
$OPISNA-REC = REC | TIP-RECI
 $TIP-RECI = "~" IME-TIPA BROJ " "
$BROJ = CIFRA {CIFRA}
 $CIFRA = "0" | "1" | "2" | "3" | "4" | "5" | "6" |
          "7" | "8" | "9"
```

Gramatika programa se sastoji iz niza definisanih tipova reči. Tip reči čini grupa opisa koji kazuju koji tekst taj tip može da zamenjuje i koje je njegovo značenje u svakom od tih slučajeva. U gramatici postoje četiri standardna tipa: COMMAND, TEXT, NUMBER i WORD. COMMAND je tip svakog teksta kojim korisnik nešto naređuje kompjuteru. Svaki uneti tekst i svaka rečenica je ovog tipa. To je ujedno i polazni tip pri analizi teksta. TEXT je tip koji ukazuje na tekst između znakova navoda. NUMBER je tip koji ukazuje na broj. WORD je tip koji ukazuje na sve ostale reči. Definicija tipa se sastoji iz navođena znaka "&" (koristi se pri pretrživanju), imena tipa, i njegovih opisa. Ime tipa se sastoji iz slova, i njegova dužina je ograničena na 64 karaktera. Program ne pravi razliku između velikih i malih slova. U opisu tipa se navodi njegova sintaksa i semantika. Sintaksa tipa se sastoji iz niza sintaksnih rečenica. Sintaksne rečenice se sastoje iz

opisnih reči. Svaka opisna reč je reč ili pokazivač na tip reči. Pokazivač na tip reči kazuje da se na tom mestu može naći čitav niz drugih reči čija je sintaksa opisana nikim od opisa datog tipa. Ovakva struktura omogućuje da se sa engleskog ne prevode samo reči ili rečenice, već i čitavi pasusi, pa i ceo tekst. Semantiku tipa čine semantičke rečenice. Ova rečenica može biti: – definicija labele, tipa, konstante ili promenljive (u PASCAL-u), ili – naredba u PASCAL-u ili njen deo U prvom i drugom slučaju će se u PASCAL fajl prepisati tekst između znakova navoda, dok će se preostali tekst analizirati kao da je unešen od korisnika. Ako se u odeljku između zagrada nađe znak "~", to znači da će se u izlaznom fajlu na tom mestu preći u novi red. U gramatici se pri navođenju tipa navodi i njegov redni broj u opisu sintakse, da bi se pri transliranju tačno znalo o kom se tipu radi. Ovo će kasnije biti objašnjeno na primeru. Dakle, ako tekst

```
PRINT "Petnica"
```

treba na ekranu da ispiše

```
Petnica
```

onda će opis gramatike za ovaj tekst biti:

```
&COMMAND
=
PRINT ~TEXT1
"WRITELN("~TEXT1 ");~"@
!
```

Već sam pomenuo da se uz svaki tip navodi i njegov redni broj u opisu sintakse. Ako, na primer, za uneti tekst

```
PRINT "Redni broj je: " "prvi"
```

treba da se ispiše samo

```
prvi
```

onda će opis gramatike koja će to da uradi biti:

```
&COMMAND
=
PRINT ~TEXT1 ~TEXT2
"WRITELN("~TEXT2 ");~"@
!
```

Ovakva struktura podataka je laka za obradu, ali joj je mana što zauzima mnogo prostora pa je zato i pristup pojedinim delovima spor. Ovi problemi će biti rešeni indeksiranjem svih reči u bazi, a za tipove će postojati i podaci o tome na kom mestu u fajlu se nalaze. Trenutno se ovi podaci kreiraju pri svakoj analizi teksta.

Algoritam za analizu teksta

Pre same analize unetog teksta, program vrši indeksiranje gramatike i unetog teksta. Naime, program u niz smešta brojeve koji označavaju početne indekse svih reči u tekstu. Program indeksira i gramatiku, ali ovde pamti samo podatke za početne redove opisa nekog tipa. Kada se izvrši indeksiranje prelazi se na analizu. Analiza programa je, u stvari, pokušaj da se na osnovu datih opisa tipova izvrši prolaz kroz uneti tekst. Nešto slično ovome imaju neki programski jezici (npr. resatisfakcija u PROLOG-u). Analiza programa se vrši iz ovih koraka:

1. a) Ako je sledeća reč tekst, broj ili neka druga reč koja nije tip, onda se ispituje da li ona odgovara opisu tekućeg tipa (tekući tip je onaj tip za koji program trenutno ispituje da li on predstavlja opis za uneti tekst).

b) Ako je sledeća reč tip, onda se tekući tip postavlja na taj tip. Ovde se proverava i da li je za reč koja se ispituje ovo njeno prvo upoređivanje sa datim tipom, pa će program raditi za dati tip samo ako jeste. Ovo se radi da ne bi došlo do stalne provere jedne reči za isti tip. Dakle, program ovde radi po principu rekurzije. Međutim, ovaj deo nije implementirana rekurzijom, već se podaci o ispitanim tipovima smeštaju u niz koji simulira rekurziju. Smatram da ovakva organizacija podataka zauzima mnogo manje memorije nego rekurzija, pa je moguće da dođe do prepunjenja steka.

2. a) Ako je u prvom koraku uspešno ispitana reč, onda se prelazi na sledeću reč. U protivnom se ispitivanje vrši za sledeći slobodan opis tekućeg tipa. Sledeći slobodan opis tipa je sledeći opis tekućeg tipa ili sledeći slobodan opis prethodnog tipa (to je poslednji tip pre tekućeg za koji se vršila provera).

b) Ako je poslednja ispitivana reč bila tip, tekuća reč se ponovo ispituje za tekući tip, inače se poziva na sledeći slobodan opis tekućeg tipa.

3. Ova dva postupka se ponavljaju sve dok se ne izvrši prolaz kroz uneti tekst ili dok ne postoji sledeći slobodan opis tipa COMMAND (u koraku 2. a).

4. Ako prolaz kroz tekst nije izvršen, onda se traži nova definicija tipa COMMAND, ili nekog drugog tipa (ovu opciju bira korisnik), pa se zatim ponovo analizira uneti tekst.

U programu se koriste dve važne strukture podataka. Prva je niz slogova koji se koristi pri prelasku sa tekućeg na sledeći tip. Dužina svakog sloga je 9 bajtova: indeks tekućeg tipa (1 bajt); indeks opisa tekućeg tipa (1 bajt); mesto u sintaksi do kog se došlo sa pretragom (1 bajt); indeks sledećeg tipa (1 bajt); indeks opisa sledećeg tipa (1 bajt); indeks reči u

unetom tekstu do koje se došlo sa pretragom (2 bajta); indeks u tabeli elemenata programa (2 bajta).

Za ovaj niz se na početku analize oslobađa 63.000 bajtova sto omogućava proveru do 7.000 tipova. Druga struktura je niz u kome svaki bit označava da li je određena reč u tekstu bila početna pri ispitivanju za neki tip. Dužina ovog niza nije određena na početku rada. On se stvara na početku analize teksta. Za ovaj niz se oslobađa memorija koja je jednaka

```
Mem = Nopis * (Ntxt + 7) : 8,
```

gde je: Mem – količina memorije koja se oslobađa, Nopis – broj opisa tipova u gramatici, a Ntxt - broj reči u tekstu.

Algoritam se u dosadašnjem radu pokazao dobar. Ovde ću dati primer za koji će program ,,znati da predstavlja mrtvu petlju. Ako se u fajlu ENGLISH.LNG nađe tekst

```
&COMMAND
=
~COMMAND1
@
!
```

program će početi sa analizom. Međutim, kada bude trebalo da po drugi put počne analizu za tip COMMAND od prve reči prijaviće grešku. Ovo je i razlog postojanja drugog niza. Pre uvođenja ovog niza, program se prekidao javljajući grešku da je u radu prepunjen stek, pa se analiza nije mogla izvršiti. Program će mrtvu petlju ,,otkriti i u slučaju da više tipova pozivaju jedni druge, i da u tom lancu poslednji poziva prvog.

Ulazni, izlazni i pomocni fajlovi

Ulazni fajl za program je XXX.NPL (XXX je naziv koji daje korisnik), izlazni fajl je XXX.PAS, a pomoćni fajlovi su XXX.TOK, XXX.RES i XXX.LST.

XXX.NPL je fajl u kome se nalazi tekst koji korisnik unosi. Najbolje je da se ovaj fajl edituje u nekom tekst editoru, pošto u programu ne postoji ova mogućnost.

XXX.TOK je fajl koji se stvara posle tokenizacije teksta u fajlu XXX.NPL. Tokom tokenizacije, program ignoriše velika i mala slova uključujući i ona između znakova navoda, a zanemaruje se veći broj blanko-karaktera. Ostale karaktere program tretira kao obicne reci.

XXX.RES je fajl koji se formira posle analize programa i u njemu se nalaze podaci o tome koji je opis korišćen pri analizi nekog tipa. Uz naziv svakog tipa se nalazi i njegov indeks. Zapisano EBNF notacijom definicije izgledaju ovako:

```
$DEFINICIJA = ELEMENT "=" VREDNOST
```

```
$ELEMENT = IME-TIPA {IME-TIPA}
 $IME-TIPA = REC BROJ
  $REC = SLOVO {SLOVO}
   $SLOVO = "A" | "B" | "C" | "D" | "E" | "F" |
            "G" | "H" | "I" | "J" | "K" | "L" |
            "M" | "N" | "O" | "P" | "Q" | "R" |
            "S" | "T" | "U" | "V" | "W" | "X" |
            "Y" | "Z"
  $BROJ = CIFRA {CIFRA}
   $CIFRA = "0" | "1" | "2" | "3" | "4" | "5" |
            "6" | "7" | "8" | "9"
$VREDNOST = TIP | BROJ | TEKST
 $TIP = "~" IME-TIPA
 $TEKST = """ {KARAKTER} """
  $KARAKTER = <bilo koji znak>
```

XXX.LST je fajl u koji se upisuju PASCAL-komande i ostali podaci koji cine glavni program. Ovaj fajl se stavara pri transliranju, odnosno pri obradi fajlova XXX.RES, ENGLISH.LNG i XXX.TOK.

XXX.PAS je fajl koji se stvara spajanjem podataka o definicijama labela, tipova, konstanti i promenljivih sa fajlom XXX.LST.

Uputstvo za korišćenje programa

Program se može startovati sa i bez parametra. U slučaju da se navede, parametar označava ime programa (program za ime računa samo slova, cifre i znake „- i „_). Ako se parametar ne navede, program će tražiti unos imena programa. Program će zatim ispitati da li fajl XXX.ENG (XXX je ime programa) postoji, i tražiće unos programa ako taj fajl ne postoji. Posle ovoga se vrši analiza unetog teksta. Ako program ne uspe da razume sta je korisnik uneo, tražiće objašnjenje za datu rečenicu. Međutim, ne mora se u ovom slučaju opisivati cela rečenica. Korisnik može da odabere koji tip reči želi da definiše unošenjem njegovog imena na pitanje

```
What type will you define (default is COMMAND)?
```

Zatim se unosi nova sintaksa odabranog tipa. Unos sintakse završava pritiskom na taster ENTER. Program će sam izvršiti indeksiranje tipova i ispitati da li ovakva definicija već postoji. Dalje se traži unos semantike. Posebnu funkciju u opisu semantike ima znak ~, koji označava da na tom mestu u fajlu XXX.PAS treba da se pređe u novi red. Opis semantike se završava navođenjem znaka @. Pri navođenju tipova se koristi sledeća struktura (EBNF notacija):

```
$TIP-RECI = "~" REC BROJ " "
 $REC = SLOVO {SLOVO}
  $SLOVO = "A" | "B" | "C" | "D" | "E" | "F" | "G" |
           "H" | "I" | "J" | "K" | "L" | "M" | "N" |
           "O" | "P" | "Q" | "R" | "S" | "T" | "U" |
```

```
          "V" | "W" | "X" | "Y" | "Z"
$BROJ = CIFRA {CIFRA}
 $CIFRA = "0" | "1" | "2" | "3" | "4" | "5" | "6" |
          "7" | "8" | "9"
```

Gramatika se može proširivati i direktnim upisom u fajl ENGLISH.LNG, ali treba paziti da se ne pogreši, jer to može dovesti do rezultata koji nisu očekivani.

Zaključak

U ovoj verziji program omogućava rešavanje jednostavnijih problema. Program ne garantuje sintaksnu ispravnost PASCAL koda koji je napravljen. U daljem radu program bi trebalo da bude proširen novim standardnim tipovima LIST (lista objekata), VARIABLE (promenljive) i SENTENCE (opis teksta koji treba da se izvrši). Veliki nedostatak programa je u gramatici, jer je fajl u kome se ona nalazi velik što oduzima mnogo vremena pri radu programa. Ovaj problem bi se mogao rešiti indeksiranjem pojmova, tako da bi se izbeglo ponavljanje reči, a i pretraga bi bila brža. Još jedan od problema koji nije rešen je mogućnost definisanja funkcija i procedura u PASCAL-u, što će verovatno biti urađeno. Tokom rada sam konstatovao i to da je često potrebno da se jedan tip definiše na više načina da bi uneti tekst mogao da se analizira. Zbog toga će u program biti ubačena mogućnost da se jedan tip definiše više puta zaredom, a ne da se čeka da program svaki put ispočetka analizira uneti tekst. Posle ovoga će tipove reči biti zamenjeni klasama reči, što će korisniku omogućiti da se uvedu sledeći elementi: selekcija tipova i njehovih opisa za određeni pokazivač na tip reči; stvaranje više izlaznih vrednosti jednog tipa; dodatno ispitivanje da li dati opis stvarno predstavlja opis za deo unetog teksta.

Ovaj program nije dao velike rezultate što opravdavam time da mi je ovo prvi put da se bavim ovakvim problemom, ali ću zato nasataviti da radim na ovom projektu kako bih napravio program koji bi mogao da se primeni u praksi.

Literatura

[1] Čabarkapa, Milan. 1994. *Osnovi programiranja u PASCAL-u sa ekstenzijama TURBO PASCAL-a*, 3. dopunjeno izdanje. Beograd: Građevinska Knjiga

[2] Kunčak, Viktor. 1993. PLS - programski jezik za pisanje simulacija. *Petničke sveske*, 33/I: 111-48

[3] Klerer, Melvin 1987. *User Oriented Languages – Analisys & Design*. Macmillan Publishing Company

[4]

Aleksa Todorović

Natural Program Language

This program translates commands entered in english to PASCAL programming language. Program performs text analysing and, if he done it succesfully, he will produce PASCAL source program. Program can expand his gramatic and vocabulary.