DOKUMENTACIJA

Programski prevodioci - predmetni zadatak

Osnovni podaci

Broj indeksa	Ime i prezime	Tema
SW29/2019	Aleksa Stanivuk	Prevođenje i analiziranje miniBre programa napisanih ćiriličnim pismom po sintaksi C jezika + JSON struktura + numerička kolekcija i *for each* prolazak kroz istu

Korišćeni alati

Naziv	Verzija
Flex	Verzija korištena na vežbama
Bison	Verzija korištena na vežbama
Python skripte	3
Shell skripte	/

Evidencija implementiranog dela

Osnovna ideja projekta je bila da se implementira mehanizam koji bi parsirao fajlove napisane na ćirilici i tako ih spremio za dalju obradu pomoću alata Flex i Bison. U tu svrhu je kreirana *Python* skripta koja sadrži *mapper* za mapiranje ćiriličnih karaktera na latinične. Skriptu je moguće pozvati kroz komandnu liniju i u tom slučaju joj je potrebno proslediti naziv fajla koji se parsira. Za proveru rada ovog dela projekta implementirane su osnovne funkcionalnosti poput *if-else*, *switch*, *for*, *break* i *do while*. Urađena je isključivo njihova sintaksna analiza.

Radi automatizacije procesa prevođenja sa ćirilice i procesa parsiranja i skeniranja, implementirana je i *bash* skripta koja poziva *Python* skriptu nad svim testnim fajlovima i nakon toga poziva *make test* ili *make det* funkciju radi provere sintakse i semantike navedenih fajlova.

Dodatno su implementirane i dve nove funkcionalnosti - *JSON* struktura za čuvanje podataka i kolekcija za čuvanje numeričkih vrednosti. Urađene su sintaksna i semantička analiza. Ugrađen je i *for each* prolaz kroz numeričku kolekciju tj. listu.

Detalji implementacije

Python skripta za prevođenje ćiriličnih fajlova je nazvana *cyrilics_converter*.py. Implementirana je u vidu klase koja sadrži *cyrilics_mapper* - *Python* rečnik čiji su ključevi ćirilični karakteri, a njihove vrednosti su ekvivalentni karakteri engleske latinice ili pak kombinacija dva karaktera u slučaju mapiranja jedinstvenih karaktera poput ц, ћ, џ, љ i slično. Klasa sadrži i metode: *read*, *parse_cyrilics_content*, *parse_command_line_args* i *do*. Metoda *parse_command_line_args* proverava da li je skripti prosleđen parametar pomoću komandne linije. Ukoliko jeste, prosleđeni parametar se označava kao naziv fajla za parsiranje. Zatim, metoda *read* učitava sadržaj istoimenog fajla, ukoliko on postoji. Na posletku, metoda *parse_cyrilics_content* prolazi kroz sadržaj učitanog fajla i svaki ćirilični karakter namapira na odgovarajuće karaktere iz *cyrilics_mapper* rečnika. Metoda *do* služi da bi pozvala prethodno navedene metode u odgovarajućem redosledu.

Bash skripta je implementirana tako da prolazi kroz svaki testni fajl unutar *test* foldera i nad njima poziva *cyrilics_converter* prosledjujući mu naziv tih fajlova. Ona zatim isparsiran sadržaj (sadržaj na latinici) iz konzole prosleđuje i zapisuje u novo-kreirane fajlove koji nose naziv **_parsed.mc* gde “ * ” predstavlja originalni naziv. Novi fajlovi se ne čuvaju unutar test foldera već u *root* folderu, tj. na istom nivou sa *cyrilics_mapper*-om i *bash* skriptom. Postoje dve *bash* skripte koje su gotove identične, međutim razlikuju se po poslednjoj liniji koda. U prvoj skripti se poziva *make test* metoda, dok se u drugoj poziva *make det*, radi dobijanja više informacija o izvršenju testiranja. Implementacija skripti je jasna iz njihovog naziva.

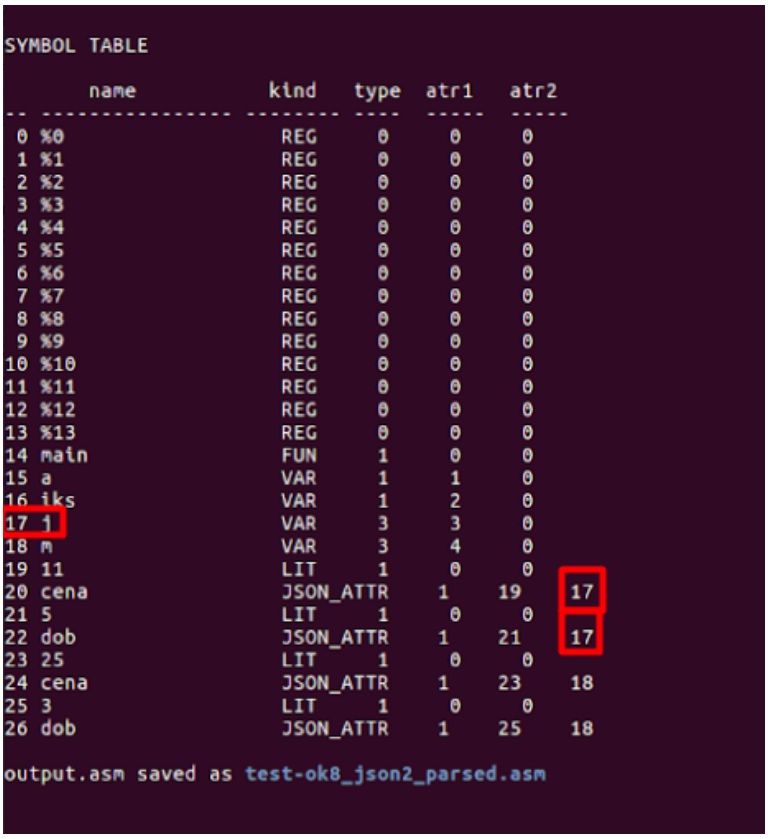
Radi verodostojnosti *miniBre* jezika, i njegove rezervisane reči su navedene na srpskom jeziku, neretko i na arhaičan nacin. Rezervisane reči na ovom nivou implementacije su: ceo (*int*), neoznačen (*uint*), ukoliko (*if*), inače (*else*), vrni (*return*), vrti (*for*), stande (*break*), dok (*while*), radi (*do*), izbori (*switch*), slučaj (*case*) i podrazumevano (*default*). Ove reči se sa ćiriličnog pisma prevode u njihove ekvivalente na latinici (npr. случај => slucsaj), a oni se zatim u *micko.l* fajlu mapiraju na standardne engleske izraze. Razlog ovog medjuproces prevođenja na latinicu je što *regex* provere ne mogu da se izvedu nad ćiriličnim konstrukcijama.

Implementacija sintaksne analize za *if-else*, *switch*, *for*, *break* i *do while* neće biti dodatno pojašnjena iz razloga što je ona odrađena na terminima vežbi. Uloga njihovog postojanja je isključivo da bi se prikazao rad ćiriličnog mapiranja, kao i pojedinih promena koje je bilo potrebno uvesti.

Za implementaciju *JSON* strukture je bilo potrebno uvesti novi tip u *defs.h* - *JSON*. Definisanje je identično kao i definisanje *int* ili *uint* promenljivih. Međutim, deklaracija je identična kao i deklarisanje stvarnih *JSON* objekata (Slika 1).

```
j = {
        цена: 11,
        доб: 5
};
```

Objektu je unutar vitičastih zagrada moguće dodati atribute u obliku *ključ : vrednost*. Objekti su zapravo *_ID* tokeni, odnosno stringovi, dok su vrednosti isključivo literali, odnosno numeričke vrednosti. Ovim atributima je kasnije moguće pristupiti po istom principu kao i *JSON* objektima - *objekat.atribut*. Implementirana je provera i sintaksnih i semantičkih grešaka. Pri dodeljivanju vrednosti, proverava se da li promenljiva sa tim nazivom uopšte i postoji u tabeli simbola, a zatim se proverava i da li je ta promenljiva tipa *JSON*. Prilikom, svakog dodavanja atributa, proverava se da li taj objekat već sadrži atribut tog naziva. To je izvršeno tako sto su atributi zapravo vrste *JSON_ATTR* i njihov *atr2* sadrži index *JSON* objekta kom pripadaju. (Slika 2)



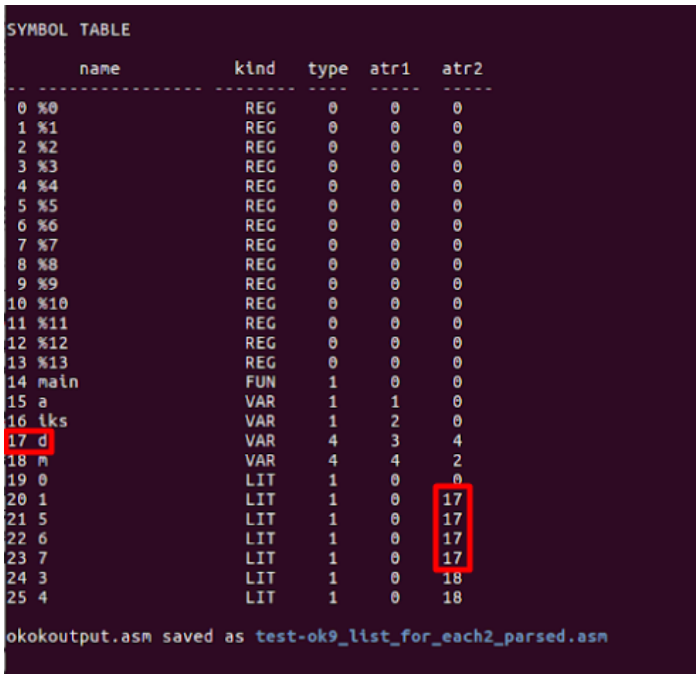
```
SYMBOL TABLE

       name           kind   type  atr1   atr2
-- ---------------- -------- ----  -----  -----
 0 %0                 REG      0     0      0
 1 %1                 REG      0     0      0
 2 %2                 REG      0     0      0
 3 %3                 REG      0     0      0
 4 %4                 REG      0     0      0
 5 %5                 REG      0     0      0
 6 %6                 REG      0     0      0
 7 %7                 REG      0     0      0
 8 %8                 REG      0     0      0
 9 %9                 REG      0     0      0
10 %10                REG      0     0      0
11 %11                REG      0     0      0
12 %12                REG      0     0      0
13 %13                REG      0     0      0
14 main               FUN      1     0      0
15 a                  VAR      1     1      0
16 iks                VAR      1     2      0
17 j                  VAR      3     3      0
18 m                  VAR      3     4      0
19 11                 LIT      1     0      0
20 cena               JSON_ATTR      1     19     17
21 5                  LIT      1     0      0
22 dob                JSON_ATTR      1     21     17
23 25                 LIT      1     0      0
24 cena               JSON_ATTR      1     23     18
25 3                  LIT      1     0      0
26 dob                JSON_ATTR      1     25     18

output.asm saved as test-ok8_json2_parsed.asm
```

Radi konzistentnosti sa tematikom ćiriličnog pisma i srpskog jezika, *JSON* objekti se u polaznim fajlovima označavaju rezervisanom reči *jovan* (*JSON* => Jason (ime) => Jovan).

Kolekcija, odnosno lista numeričkih vrednosti, tj. literala se radi konzistentnosti naziva družba. Njihova definicija je identična kao i definicija *JSON* simbola i običnih promenljivih, s tim što su ovi simboli tipa *LIST*. Deklaracija se odvija tako sto se *_ID* simbola izjednači doslovno sa listom brojeva odvojenih zarezom i uokvirenih pravougaonim zagradama (Slika 3).

```
дружба д;
д = [1, 5, 6, 7];
```

Radi semantičkih provera, za svaki *LIST* simbol se u njegovom *atr2* čuva broj elemenata, a za literala koji predstavljaju elemente ovih lista, *atr2* sadrži index unutar tabele simbola liste kojoj pripadaju (Slika 4).



```
SYMBOL TABLE

       name           kind   type  atr1   atr2
-- ---------------- -------- ----  -----  -----
 0 %0                 REG     0     0      0
 1 %1                 REG     0     0      0
 2 %2                 REG     0     0      0
 3 %3                 REG     0     0      0
 4 %4                 REG     0     0      0
 5 %5                 REG     0     0      0
 6 %6                 REG     0     0      0
 7 %7                 REG     0     0      0
 8 %8                 REG     0     0      0
 9 %9                 REG     0     0      0
10 %10                REG     0     0      0
11 %11                REG     0     0      0
12 %12                REG     0     0      0
13 %13                REG     0     0      0
14 main               FUN     1     0      0
15 a                  VAR     1     1      0
16 iks                VAR     1     2      0
17 d                  VAR     4     3      4
18 n                  VAR     4     4      2
19 0                  LIT     1     0      0
20 1                  LIT     1     0     17
21 5                  LIT     1     0     17
22 6                  LIT     1     0     17
23 7                  LIT     1     0     17
24 3                  LIT     1     0     18
25 4                  LIT     1     0     18

okokoutput.asm saved as test-ok9_list_for_each2_parsed.asm
```

Implementiran je i *FOR EACH* prolaz kroz literale ovih lista. Pored sintaksne analize implementirana je i semantička provera da li su definisani simbol koji se koristi za iteriranje, kao i simbol koji označava listu - za njega se proverava i da li je tipa *LIST* (Slika 5).

```
цео а;
цео икс;
а = 0;
за сваки икс унутар д {
        а = а + икс;
}
```

Ideje za nastavak

Što se tiče nastavka razvoja parsiranja programa napisanih na ćirilici, potrebno je omogućiti da se skripte za testiranje pokreću i bez postojanja *main* rezervisane reci unutar fajla, odnosno omogućiti izmenu te reči u npr. *главни*.

Naravno, moguće je uvesti podršku za veći skup rezervisanih reči na srpskom jeziku.

Za *JSON* simbole bi bilo dobro implementirati podršku za ne-numeričke simbole, dakle *boolean* i stringove. Dodatno, omogućavanje setovanja vrednosti atributa van konstruktora bi bilo veoma korisno za korisnike.

Za *LIST* simbole važi slično. Bilo bi dobro omogućiti dodavanje, kao i oduzimanje elemenata. Prilikom implementacije je bio izvršen pokušaj implementacije *override*-ovanja ovih simbola, međutim pojavio se problem pri brisanju simbola elemenata iz tabele simbola. Naravno, kao i za *JSON*, zanimljivo bi bilo probati implementirati podršku i za ne-literalne elemente poput stringova.

Literatura

Korišteni su predmetni materijali sa vežbi i predavanja.