**DOKUMENTACIJA**

Programski prevodioci - predmetni zadatak

**Osnovni podaci**

|  |  |  |
| --- | --- | --- |
| Broj indeksa | Ime i prezime | Tema |
| SW 24/2019 | Matija Zarić | Callback funkcije u miniC jeziku |

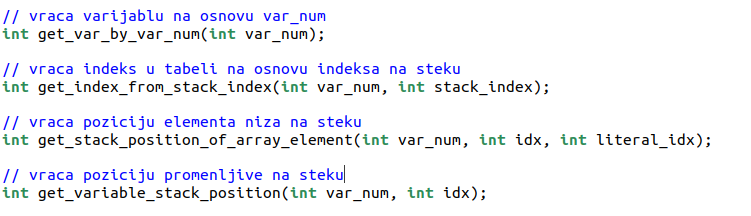
**Korišćeni alati**

|  |  |
| --- | --- |
| Naziv | Verzija |
| Bison, Flex | 3.5.1, 2.6.4 |

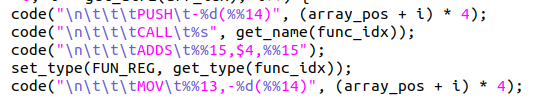
## Evidencija implementiranog dela

* **Nizovi –** sintaksa, semantika, generisanje koda
* **For each petlja za nizove** – sintaksa, semantika, generisanje koda
* **Callback funckije** – sintaksa, semantika, generisanje koda

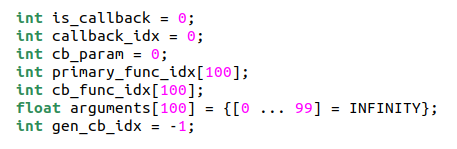
## Detalji implementacije i nedostaci

* **Nizovi**
  + Sintaksa, semantika i generisanje koda
    - U micko.l fajlu dodato je pronalaženje tokena “[” i “]” kao \_SLBRACKET i \_SRBRACKET.
    - U defs.h dodata je nova vrednost KIND enumeracije, koja je ARRAY i ima pridruženu vrednost 0x40.
    - U micko.y omogućena su dva načina definisanja niza, prvi je u vidu: \_TYPE \_ID \_SLBRACKET literal \_SRBRACKET \_SEMICOLON, a drugi je u vidu \_TYPE \_ID \_SLBRACKET \_SRBRACKET \_ASSIGN \_LBRACKET literal\_list \_RBRACKET \_SEMICOLON za definisanje niza sa unapred zadatim vrednostima.
    - Da bi se podržala sintaksa nizova korišćene su sledeće promenljive: niz arrays, brojač array\_idx i brojač array\_elem\_num. U nizu arrays se smeštaju indeksi nizova u tabeli simbola. Brojač array\_idx označava broj definisanih nizova, a brojač array\_elem\_num označava broj elemenata za nizove koji se definišu sa predefinisanim vrednostima.
    - Nizovi se smeštaju u tabelu simbola na isti način kao i promenljive. Razlika je u tome što se za atr2 u tabeli za niz dodeljuje vrednost broja elemenata u nizu.
    - Za generisanje koda potrebno je bilo izmeniti sve segmente miniC kompajlera sa vežbi. Što se tiče slobodnog prostora na steku, ako postoje nizovi, on se računa na izmenjen način. Sabira se broj elemenata svih nizova i broj promenljivih i ta količina lokacija se ostavlja na steku za lokalne promenljive.
    - Za generisanje koda kod predefinisanih nizova, potrebne su bile sledeće globalne promenljive: nizovi arrays\_decl, array\_decl\_vars i brojači: array\_decl\_cnt i array\_decl\_idx. U niz arrays\_decl se smeštaju indeksi iz tabele simbola nizova koji su predefinisani, u niz array\_decl\_vars se smeštaju indeksi iz tabele simbola svih literala za sve predefinisane nizove, brojač array\_decl\_idx predstavlja broj predefinisanih nizova, dok brojač array\_decl\_cnt predstavlja broj indeksa literala u nizu array\_decl\_vars.
    - U symtab.h dodate su deklaracije sledećih funkcija:
    - Svi sintaksni šabloni su izmenjeni da podrže nizove. Šablon num\_exp je imao najviše promena pa ću njega detaljnije objasniti.
    - Pre svega da bi se omogućila provera tipova izraza u num\_exp korišćene su sledeće globalne promenljive: niz type\_check\_numexp i brojači type\_check\_arrays i type\_check. U niz type\_check\_numexp su smeštani indeksi iz tabele simbola svih izraza unutar num\_exp, brojač type\_check predstavlja broj izraza, a brojač type\_check\_arrays predstavlja broj korišćenih elemenata nizova unutar izraza kao i broj promenljivih ukoliko u funkciji postoje nizovi. Ova izmena je napravljena zato što semantička vrednost izraza, ako ima veze sa promenljivom ili elementom niza, možda neće predstavljati indeks elementa u tabeli simbola već će predstavljati poziciju te promenljive na steku.
    - Takođe da bi se podržalo generisanje koda za nizove dopunjena je funkcija gen\_sym\_name tako da podrži poslate pozicije promenljivih na steku.
  + Testiranje
    - Nizovi su zastupljeni u svim testovima tačnosti, testovi koji treba da prođu. Oni se nalaze unutar foldera oktests i pokazuju upotrebljivost svih starih elemenata zajedno sa proširenjem u vidu nizova. U folderu sanity-test nalazi se test sa elementima iz osnovnog miniC jezika, koji kada se pokrene pokazuje da je on i dalje funkcionalan.
    - U folderu errortests se nalaze testovi sa sintaksnim i semantičkim greškama, u njima su prikazane sve situacije sintaksnih i semantičkih grešaka koje mogu da se naprave upotrebom nizova.
* **For each petlja za nizove**
  + Sintaksa, semantika i generisanje koda
    - U fajlu micko.l dodato je prepoznavanje tokena „.“ i „forEach“ i oni vraćaju vrednosti \_DOT odnosno \_FOREACH.
    - Za prepoznavanje sintakse for each iskaza za nizove dodat je šablon for\_each koji izgleda ovako: \_ID \_DOT \_FOREACH \_LPAREN \_ID \_RPAREN \_SEMICOLON. Prvi \_ID predstavlja naziv niza, a drugi predstavlja naziv funkcije koja se poziva nad svakim elementom niza.
    - Funkcija koja se poziva nad elementima niza mora da prima isti tip parametra kao što je tip niza i takođe da vrati isti tip.
    - Semantičke greške se mogu javiti ako nisu ispoštovani tipovi ili ako ne postoji zadati niz odnosno zadata funkcija.
    - Za generisanje koda se za svaki element niza redom dodaje

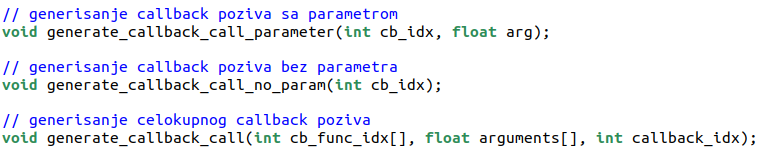
sledeći kod:



* + - U prethodnom isečku array\_pos predstavlja mesto početnog elementa niza na steku. Nakon izvršene funkcije iz registra za povratnu vrednost se vrednost smešta u isti taj element.
  + Testiranje
    - U testovima su pokriveni slučajevi korišćenja forEach iskaza nad elementima niza. U folderu oktests, testovi 5, 6, 7 i 11 prikazuju primere korišćenja ovog iskaza.
    - U folderu errortests, test za sintaksnu grešku 4 pokazuje prijavu sintaksne greške prilikom korišćenja forEach iskaza, dok testovi 6 i 7 za semantičke greške pokrivaju slučajeve u kojima se prijavljuje semantička greška.
  + Ograničenja
    - Ograničenje upotrebe ovog iskaza u gramatici proširenog miniC jezika jeste to što funkcija koja se prosleđuje mora imati iste tipove parametra i povratne vrednosti kao i tip niza. Takođe ograničenjem se može smatrati to što vrednosti neće biti reprezentativne ukoliko se ranije ne definišu početne vrednosti svakog elementa niza.
* **Callback funkcije**
  + Sintaksa, semantika, generisanje koda
    - Da bi se leksiči podržale callback funkcije u micko.l fajl dodate je prepoznavanje ključnih reči „void“ i „callback“. „Void“ je dodat u defs.h kao dodatan tip, za povratnu vrednost callback funkcije, a o ovome će biti više reči dalje. Za ključnu reč „callback“ prepoznaje se token \_CALLBACK.
    - Za sintaksno prepoznavanje callback iskaza dodat je callback\_statement u obliku: \_CALLBACK \_LPAREN argument \_RPAREN \_SEMICOLON.
    - Za potrebe sintaksne, sematiče analize kao i generisanja koda, korišćene su sledeće promenljive:



* + - Funkcija koja prima callback kao parametar se na isti način dodaje u tabelu simbola kao i ostale funkcije, samo se za vrednost atr2 postavlja tip povratne vrednosti callback funkcije, a to je VOID.
    - Kada funkcija treba da primi callback funkciju kao parametar, za parametar funkcije se navodi ključna reč callback. Prilikom prepoznavanja tog parametra u niz primary\_func\_idx dodaje se indeks iz tabele simbola te funkcije.
    - Kada se u kodu naiđe na callback\_statement vrednost is\_callback se postavlja na 1. Ovo omogućuje da se argument posmatra drugačije nego kod ostalih poziva funkcije. On se, ako već nije registar, smešta u registar i indeks registra iz tabele simbola se dodaje u niz arguments na poziciju callback\_idx.
    - Pošto nisu podržane sinhrone callback funkcije, o čemu će biti više reči kasnije, tu se završava deo vezan za callback funkciju sve dok se ne dođe do mesta poziva funkcije i prosleđivanja druge funkcije kao parametra.
    - Kada se dođe do poziva funkcije i prosleđivanja druge kao parametra, vrednost cb\_param se stavlja na 1 da bi se omogućile semantičke provere za parametar funkcije koji je naziv druge funkcije. Zatim se u izrazu \_ID koji skener pronalazi kada nađe naziv funkcije kao parametar, vrši dodavanje indeksa iz tabele simbola funkcije koja se kao parametar prosleđuje u niz cb\_func\_idx na poziciju koja se odredi nakon što se pronađe pozicija funkcije, kojoj se prosleđuje kao parametar, u nizu primary\_func\_idx.
    - Generisanje koda za poziv callback funkcije se vrši nakon povratka iz primarne funkcije kojoj je callback prosleđen.
    - U fajl codegen.h dodate su sledeće funkcije za generisanje koda kod poziva callback funkcija:



* + - Funkcija generate\_callback\_call se poziva kada treba generisati kod za callback funkciju, a funkcije generate\_callback\_call\_no\_param i generate\_callback\_call\_parameter vrše samo ispis u .asm fajl i poziva ih funkcija generate\_callback\_call.
  + Testiranje
    - U folderu oktests, testovi 8, 9, 10, 11 i 12 sadrže primere uspešnih testova korišćenja callback\_poziva. Način funkcionisanja i generisanja koda kod ove implementacije može se videti korišćenjem hipsim alata za simulaciju asemblera kao i gledanjem samog .asm fajla.
    - U folderu errortests, sintaksni test 3 pokazuje primer prijave sintaksne greške kod callback funkcija. Semantički testovi 8, 9, 10 sadrže primere prijave semantičkih grešaka kod callback poziva.
  + Ograničenja
    - Postoje tri ograničenja callback funkcija u mojoj implementaciji. Prvo ograničenje je nepostojanje sinhronih callback funkcija, tj. kod za generisanje callback poziva se ne dodaje kada se taj poziv desi, nego se dodaje onda kada se završi funkcija koja je poziva.
    - Drugo ograničenje je što povratna vrednost callback funkcije mora biti void, jer ne postoji sinhron poziv callback funkcije, pa samim tim nema potrebe za vođenjem računa o povratnoj vrednosti.
    - Treće ograničenje jeste to da se ne može pozvati callback funkcija više puta unutar funkcije koja je prima kao parametar. Ovo je posledica smeštanja registra sa argumentom u niz arguments i onda kada bi se drugi put pozvala callback funkcija argument koji se prosleđuje bi bio upisan u niz preko argumenta koji se prosleđuje prvom pozivu i onda bi oba poziva primala argument koji se šalje samo drugom pozivu.

Ideje za nastavak

Ideja za sinhroni način pozivanja callback funkcija jeste implementacija dva prolaska kroz polazni kod. U prvom prolazu bi se beležili podaci o pozivima funkcija sa callback funkcijom kao parametrom i beležilo bi se koja se funkcija prosleđuje kao callback. Ovim bi nestala potreba za pojedinim globalnim promenljivama vezanim za praćenje koja callback funkcija se prosleđuje, a takođe bi se omogućio poziv callback funkcije na onom mestu na kom se to vrši. Takođe bi bilo uklonjeno i ograničenje da povratna vrednost callback funkcije mora biti void. Bilo bi otklonjeno i ograničenje na 1 mogući poziv callback funkcije, jer bi se u prvom prolazu zapamtili parovi glavna funkcija-callback funkcija, pa bi se kod svakog callback poziva mogli slati različiti parametri i ne bi bilo potrebe za nizom arguments.

## Literatura

* Materijali sa vežbi i predavanja.
* <https://www.w3schools.com/jsref/jsref_foreach.asp> – za forEach petlje po uzoru na javascript.
* <https://www.w3schools.com/js/js_callback.asp> – za callback funkcije po uzoru na javascript