DOKUMENTACIJA

Programski prevodioci - predmetni zadatak

**Osnovni podaci**

|  |  |  |
| --- | --- | --- |
| Broj indeksa | Ime i prezime | Tema |
| SW7/2019 | Ivana Kašiković | Rječnici u mini C |

**Korišćeni alati**

|  |  |
| --- | --- |
| Naziv | Verzija |
| GCC compiler, Flex, Bison, Hipsim |  |

## Evidencija implementiranog dela

## *U okviru projekta podrzani su rjecnici u mini C-u i odradjena je sintaksa,semantika i generisanje koda. Rjecnike je potrebno globalno definisati i to je moguce uraditi sa ili bez definisanih elementima. Omogucen je pristup elementima, redefinisanje njihovih vrijednosti,kao i dodavanje novih elemenata na globalnom nivou. Uradjena su semanticke provjere za jedinstvenost naziva rjecnika, jedinstvenost kljuca u okviru rjecnika, provjera da li su ispravni tipovi podataka kljuceva i vrijednosti elemenata.*

## Detalji implementacije

Svi relevantni testovi nalaze se u korjenskom direktorijumu projekta.

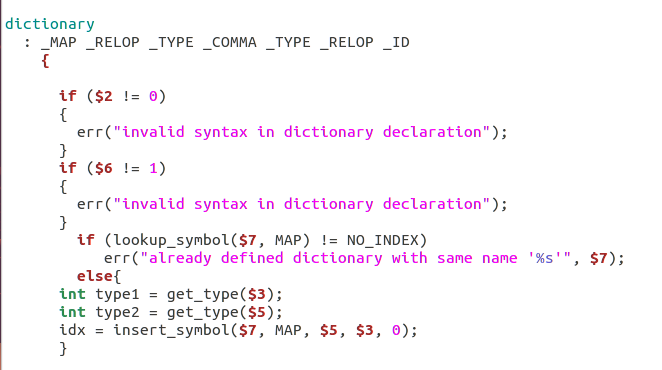
Ideja za implementaciju rjecnika zasniva se na sljedecem principu. Uveli smo novi “kind” u fajlu defs.h pod nazivom MAP i MAP\_ATR, koji su koristeni za cuvanje samih naziva rjecnika i njegovih elemenata. Prilikom nailaska na neki rjecnik prvo se vrsi semanticka provjera da li postoji vec neki rjecnik pod istim nazivom. Ako je to slucaj baca se greska.U suprotnom se vrsi cuvanje mape u tableli simbola,gdje se pod “type” cuva tip vrijednosti, a pod atributom “atr1” se cuva tip kljuca elementa,sto koristimo za provjeru tipova elemenata rjecnika.

Sintaksa za definisanje rjecnika:

map <int,unsigned> nazivMape;

map <int,int> nazivMape1 = {{1,1},{1,2}};

Primjer ubacivanja mape u tabelu simbola (Slika 1 ):

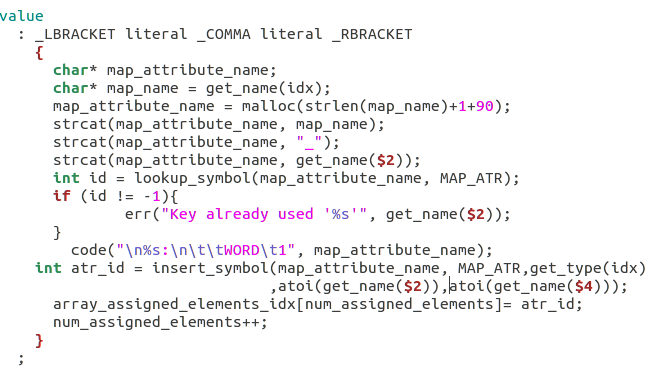


Slika Ubacivanje mape u tabelu simbola

Prilikom cuvanja elemenata rjecnika jedinstveni naziv se formira kao *nazivRjecnika\_kljucElementa* koji koristimo za id u tabeli simbola, kao i za generisanje koda pomocu kljucne rijeci WORD. Pod atributom “type” cuvamo tip vrijednosti elementa rjecnika,pod “atr1” cuvamo kljuc elementa rjecnika, a kao “atr2” cuva se vrijednost elementa rjecnika. Prije samog cuvanja vrijednosti elementa rjecnika u tabeli simbola vrsi se provjera da li postoji vec neki element u tom rjecniku sa datim kljucem i da li se tip kljuca I vrijednosti elementa rjecnika poklapa sa tipovima koji su karakteristicni za taj rjecnik.

Primjer ubacivanja elementa u tabelu simbola (Slika 2 ):

Element se pise u obliku “{1,1}”:



Slika Ubacivanje elementa mape u tabelu simbola

Redefinisanje vrijednosti rjecnika odredjeno je tako sto se generisanjem jedinstvenog naziva za element izvrsi pretraga u tabeli simbola. Uz sve neophodne provjere da li je vrijednost odgovarajuceg tipa vrsi se cuvanje nove vrijednosti elementa u atributu 2.

Na slicnom principu funkcionise i pristupanje,tj. element se uzima direktnim pristupom tabeli simbola.

Dodavanje elementa na globalnom nivou odradjeno je tako sto se odmah odvoji prostor za promjenljivu pomocu kljucne rijeci WORD I ubaci se u tabelu simbola gdje je atribut “kind” MAP\_ATR, a vrijednost se cuva u okviru globalne strukture pod nazivom “elements\_map”. Kada prevodjenje dodje do “main\_body” prolaskom kroz spomenutu strukturu vrsi se cuvanje vrijednosti elementa rjecnika pomocu naredbe *gen\_move.*

Pored toga postojalo je par ideja za dodavanje novog elementa u rjecnika u okviru funkcije:

1. Da se u sekciji *program* doda jos jedan prolaz kroz *function\_list* kako bi se u prvom prolazu izvrsilo generisanje koda za sve elemente rjecnika,a u drugom bi se pokrenuo sav ostali kod.Međutim, ovaj nacin je izazivao shift/reduce konflikt.
2. Druga ideja je bila da se u okviru main funkcije dva puta pozove yyparse, gdje bi svaki prolaz imao spomenute ciljeve,a redni broj prolaza bi se regulisao preko globalne promjenljive. Ali, pri drugom prolazu se javljao *syntax error*. Problem je bio to sto u prvom prolazu program “pojede” tokene,tako da ne ostanu tokeni za drugi prolaz. Ideja je bila da se dva puta pozove flex na neki nacin,sto nas dovodi do trece ideje.
3. Cilj je formirati novu skriptu za pokretanje programa gdje bismo imali dva y fajl, po jedan za svaki prolaz. Ali,sam zadatak je bio zahtjevan,tako da smo prilikom implementacije naisli na dosta sporednih problema,sto je dovelo do toga da implementacija ne bude u potupunosti ispunjena.

## Ideje za nastavak

## U nastavku istrazivanja mogli smo se baviti implementacijom dodavanja elemenata rjecniku u okviru funkcije. Mogli bismo implementirati jos neke funkcionalnosti poput brisanja elemenata, podrzavanje rada rjecnika sa nekim drugim tipovima podataka...

## Literatura

<https://stackoverflow.com>

<https://stackoverflow.com/questions/4384359/quick-way-to-implement-dictionary-in-c>