DOKUMENTACIJA

Programski prevodioci - predmetni zadatak

Osnovni podaci

Broj indeksa	Ime i prezime	Tema
sw76/2019	Vladan Mikić	Lambda fun, enumeracije, levi i desni shift, logički operatori u miniC

Korišćeni alati

Naziv	Verzija
Bison	Verzija korišćena na vežbama
Flex	Verzija korišćena na vežbama
Hipsim	Verzija korišćena na vežbama

Evidencija implementiranog dela

Implementirane su sledeće funcionalnosti:

- Levi i desni shift: sintaksa, semantika, generisanje koda
- Enumeracije:
  - deklaracija: sintaksa, semantika, generisanje koda
  - dodela vrednosti enumeracije nekoj promenljivoj: sintaksa, semantika, generisanje koda
- Lambda funkcije:
  - deklaracija: sintaksa, semantika, generisanje koda
  - poziv lambda funkcije: sintaksa, semantika, generisanje koda
  - dodela povratne vrednosti lambda funkcije nekoj promenljivoj: sintaksa, semantika, generisanje koda

Detalji implementacije

Levi i desni shift

Implementirani su levi i desni logički shift operatori. Pozivaju se sa *<<{vrednost}* i *>>{vrednost}*, respektivno. Parameter u zagradama predstavlja opcionu vrednost shifta.

Sintaksa i semantika

Dodate su << i >> čije vrednosti su _LSHIFT i _RSHIFT. Datoteka *micko.y* je proširena sa dva izraza: *lshift_exp* i *rshift_exp.* Takođe, dodati su *lshift_exp* i *rshift_exp* u *exp* i u

assignment_statement. Dodavanjem *shift* izraza u *assignment_statement* omogućuje da se vrednost *shift-a* skladišti u promenljivu nad kojoj se *shift* poziva.

Generisanje koda

Generisanje koda je prošireno dodatnom funkcijom *gen_shift.* Pomenuta funkcija generiše odgovarajući *shift* na osnovu tipa promenljive nad kojom se *shift* poziva i prosleđenog tipa shifta (levi ili desni).

Testovi

U folderu *"./tests/shift"* dodati su testovi za sintaksu, semantiku i generisanje koda. Datoteke: "test-ok1.mc", "test-ok2.mc" i "test-ok3.mc" sadrže sintaksno i semantički ispravan kod i ujedno testiraju generisanje koda. Datoteke: "test-synerr1.mc" i "test-semerr1.mc" sadrže testove koje prouzrokuju sintaksne i semantičke greške.

Enumeracije

Pored levog i desnog *shift-a*, micko je proširen sa enumeracijama. Enumeracije se definišu na sledeći način: *enum ime_enuma { vrednost_1, vrednost_2 };.* Sve enumeracije je potrebno definisati posle definicije promenljivih unutar funkcije. Obavezno je da enumeracija ima bar jednu vrednost. Vrednost *enum* tipova predstavlja njihov redni broj u enumeraciji, počinjući od 0*.*

Sintaksa i semantika

U datoteku *micko.l* dodat je enum tip koji ima vrednost _ENUM i _DOT koji predstavlja tačku. U *micko.y* je dodat *enumeration* koji predstavlja definiciju enum tipa. Sintaksa navedenog tipa je sledeća: *_ENUM _ID _LBRACKET enum_values _RBRACKET _SEMICOLON*. Vrednosti enumeracije su predstavljene pomoću *enum_values.* Sintaksa dodela vrednosti enuma nekoj promenljivoj je sledeća: *_ID _ASSIGN _ID _DOT _ID _SEMICOLON* i dodata je u *assignment_statement.*
U tabeli simbola su dodate vrednosti predstavljene u Tabeli 1.

Simbol	*Name*	*Kind*	*Type*	*Atr1*	*Atr2*
ENUM	ime enuma	ENUM	NO_TYPE	broj vrednosti enuma	lokacija enuma u odnosu na stack pointer
ENUM_VAL	vrednost	ENUM_VAL	INT	indeks enuma kojoj vrednost pripada	redni broj vrednosti u enumu

Tabela 1 Enum simboli u tabeli simbola

Generisanje koda

Generisanje koda za enum je implementirano tako što je proširana funkcija *gen_sym_name* u *codegen.c* datoteci. Čita se vrednost enum vrednosti, tj. polje *atr2* koji ujedno predstavlja i njen redni broj, i generiše se kod za dodelu te vrednosti nekoj promenljivoj.

Testovi

Svi testovi su definisani u folderu “.tests/enum/”. Datoteke: “test-ok1.mc”, “test-ok2.mc” i “test-ok3.mc” sadrže sintaksno i semantički ispravan kod i ujedno testiraju generisanje koda. Datoteke: “test-synerr1.mc”, “test-synerr2.mc”, “test-semerr1.mc”, “test-semerr2.mc” i “test-semerr3.mc” sadrže testove koje prouzrokuju sintaksne i semantičke greške.

Lambda funkcije

Lambda funkcije su ujedno predtavljale i najobimniji deo projekta. Sintaksa implementirane *lambda* funkcije podseća na *python lambda* funkcije. Definiše se na sledeći način: *lamda ime_promenljive = lambda(lista_parametara) : telo_funkcije;*. Svi parametri moraju biti istog tipa i obavezno je da *lambda* funkcija ima bar jedan parametar.

Sintaksa i semantika

Datoteka *micko.l* je proširena sa *lambda* tokenom koji ima vrednost _LAMBDA. Takođe su u *defs.h* dodate vrste simbola za *lambda* funkciju i *lamda* parametar, a to su: LFUN i LPAR. U datoteku *micko.y* dodat je *lambda_statement* iskaz. Prilikom kreiranja nove *lambda* funkcije proverava se da li već postoji *lambda* funkcija sa istim imenom, a ukoliko ne, kreira se novi niz za tip parametara *lambda* funkcije, koji se koristi kako bi se proverilo da li je svaki tip parametara isti. Pored toga, dodaje se u tabelu simbola lambda funkcija što je prikazano u Tabeli 2. *Lambda* parametri se u tabelu simbola dodaju odma nakon *lambda* funkcije kojoj pripadaju, to je prikazano u Tabeli 3.

Name	*Kind*	*Type*	*Atr1*	*Atr2*
ime lambda promenljive	LFUN	tip lambde, isti je kao tip parametara	broj parametara	redni broj lambde

Tabela 2 LFUN simbol u tabeli simbola

Name	*Kind*	*Type*	*Atr1*	*Atr2*
ime parametra	LPAR	tip parametra	indeks lambda funkcije kojoj pripada u tabeli simbola	redni broj paramtera

Tabela 3 LPAR simbol u tabeli simbola

Generisanje koda

Kako bi se generisao kod za lambda funkciju, bilo je potrebno dodati labele za lambda funkcije i za nastavak tela funkcije u kojoj se lambda deklarišu. To je urađeno tako što je nakon završetka deklaracije lambda funkcije povećan brojač *main_counter* koji koji prati koji je trenuti redni broj tela funkcije. Kod za *lambda* funkciju se generiše tako što se generiše prvo labela *@lambda_promenljiva_redni-br-lambde*, gde se inicijalizuju parametri. Nakon toga se generiše labela *@lambda_promenljiva_redni-br-lambde_body* gde dalje generiše telo lambda funkcije. Na kraju se generiše labela *@lambda_promenljiva_redni-br-lambde_exit* koja predstavlja izlaz iz *lambda* funkcije i nakon toga se generiše i labela

@main_body_main_counter koja dalje predstavlja nastavak koda tela funkcije. Ukoliko ne postoji kod između npr. deklaracija dve *lambda* funkije, kod ispod *@main_body_main_counter* će sadržati samo skok naredbu na labelu tela funkcije koja sadrži neki kod. Kako bi se generisala imena simbola *lambda* parametara, proširena je *gen_sym_name* funkcija u *codegen.c* sa slučajem za LPAR, koji pronađe odgovarajuću lokaciju *lambda* parametra na *stack*-u i generiše odgovarajući kod.

Testovi

Svi testovi su definisani u folderu ".tests/lambda/". Datoteke: "test-ok1.mc", "test-ok2.mc" i "test-ok3.mc" sadrže sintaksno i semantički ispravan kod i ujedno testiraju generisanje koda. Datoteke: "test-synerr1.mc", "test-synerr2.mc",, "test-synerr3.mc", "test-semerr1.mc", "test-semerr2.mc", "test-semerr3.mc" i "test-semerr4.mc" sadrže testove koje prouzrokuju sintaksne i semantičke greške.

Ideje za nastavak

Moglo bi se dodati podrška za aritmetički levi i desni *shift.*
Što se tiče enumeracije bilo bi poželjno da se proširi mogućnosti dodele vrednosti tj. promene indeksa vrednosti *enum*-a. Sintaksa bi izgledala ovako: *enum state { started=1, finished=-1 }*. Takođe bilo bi dobro da se može dobaviti i string reprezentacija tj. ime te vrednosti, tj. ako uradim sledeću dodelu: *int x = state.started;*, da mogu uraditi *name(x)*, ili nešto slično.
Lambda funkcije bi se poboljšati jednostavnijom sintaksom. Moglo bi se promenuti: *lambda x = lambda(int a, int b) : a - b + 1;* u *lambda x = (int a, int b) : a - b + 1;*.

Literatura

"Python Lambda". *W3schools.Com*, 2022, https://www.w3schools.com/python/python_lambda.asp.
"Enumeration (Or Enum) In C - Geeksforgeeks". *Geeksforgeeks*, 2022, https://www.geeksforgeeks.org/enumeration-enum-c/.
Stoyanov, Yasen. "Logical Vs. Arithmetic Shift - Open4tech". *Open4tech*, 2022, https://open4tech.com/logical-vs-arithmetic-shift/.