Elektrotehnički fakultet u Beogradu

Katedra za računarsku tehniku i informatiku

*Predmet:* Programski prevodioci 1

*Nastavnik:* dr Dragan Bojić, redovni profesor

*Asistenti:* mast. inž. Maja Vukasović,

mast. inž. Mihajlo Ogrizović

*Školska:* 2022/2023.

*Datum: 08*.02.2022.

*Autor*: Ćirković Miloš

**Projekat**

**– Kompajler za Mikrojavu –**

**OPIS PROJEKTA**

Cilj projektnog zadatka je realizacija kompajlera za programski jezik Mikrojavu. Kompajler omogućava prevodjenje sintaksno i semantički ispravnih Mikrojava programa u Mikrojava bajtkod koji se izvršava na virtuelnoj mašini za Mikrojavu. Sintaksno i semantički ispravni Mikrojava programi su definisani specifikacijom [MJ].

Programski prevodilac za Mikrojavu ima četiri osnovne funkcionalnosti: leksičku analizu, sintaksnu analizu, semantičku analizu i generisanje koda.

***Leksički analizator***treba da prepoznaje jezičke lekseme i vrati skup tokena izdvojenih iz izvornog koda, koji se dalje razmatraju u okviru sintaksne analize. Ukoliko se tokom leksičke analize detektuje leksička greška, potrebno je ispisati odgovarajuću poruku na izlaz.

***Sintaksni analizator***ima zadatak da utvrdi da li izdvojeni tokeni iz izvornog koda programa mogu formiraju gramatički ispravne sentence. Tokom parsiranja Mikrojava programa potrebno je na odgovarajući način omogućiti i praćenje samog procesa parsiranja na način koji će biti u nastavku dokumenta detaljno opisan. Nakon parsiranja sintaksno ispravnih Mikrojava programa potrebno je obavestiti korisnika o uspešnosti parsiranja. Ukoliko izvorni kod ima sintaksne greške, potrebno je izdati adekvatno objašnjenje o detektovanoj sintaksnoj grešci, izvršiti oporavak i nastaviti parsiranje.

***Semantički analizator***se formira na osnovu apstraktnog sintaksnog stabla koje je nastalo kao rezultat sintaksne analize. Semantička analiza se sprovodi implementacijom metoda za posećivanje čvorova apstraktnog sintaksnog stabla. Stablo je formirano na osnovu gramatike implementirane u prethodnoj fazi. Ukoliko izvorni kod ima semantičke greške, potrebno je prikazati adekvatnu poruku o detektovanoj semantičkoj grešci.

***Generator koda***prevodi sintaksno i semantički ispravne programe u izvršni oblik za odabrano izvršno okruženje Mikrojava VM. Generisanje koda se implementira na sličan način kao i semantička analiza, implementacijom metoda koje posećuju čvorove.

**Korisnicko uputstvo**

1. Sve biblioteke se ucitavaju preko MJCompiler>Properties>Libraries>Add JARs…

Učitati :

1. cup\_v10k.jar
2. Jflex.jar
3. Log4j-1.2.17.jar
4. mj-runtime.jar
5. Symboltable.jar
6. mj-runtime se pokreće sa podešavanjem: run Configurations...>Run>Arguments

u Program arguments uneti: test/program.obj -debug , applay, a zatim Run.

**TEST PRIMERI**

*Primer1:*

// Test301

program test301

const int nula = 0;

const int jedan = 1;

const int pet = 5;

int niz[], niz2[], a, b;

char nizch[];

int mat[][];

{

void main()

int bodovi;

bool bt;

{

bodovi = 0;

bodovi++;//1

bodovi = bodovi + jedan;//2

bodovi = bodovi \* pet;//10

bodovi--;//9

print(bodovi);//9

niz = new int[3];

niz[nula] = jedan;//niz[0] = 1;

niz[1] = 2;//niz[1] = 2;

niz[niz[jedan]] = niz[niz[0]] \* 3;//niz[niz[1]] = niz[niz[0]]\*3; -> niz[2] = niz[1]\*3 -> niz[2] = 2\*3

bodovi = niz[2]/niz[0];// bodovi =/6/1

print(bodovi);//6

print(niz[0]);//1

print(niz[1]);//2

print(niz[2]);//6

mat = new int[4][5];

read(niz[1]);

mat[0][0] = mat[0][mat[0][0]];//mat[0][0] = 0

mat[0][mat[0][0]] = 2;// mat[0][0] = 2

mat[1][jedan - niz[0]] = 5; // mat[1][1 - 1] = 5 -> mat[1][0] = 5

mat[1][1] = mat[0][0] \* mat[0][1];// mat[1][1] = 0 \* 0 -> mat[1][1] = 0

print(mat[0][0]);//2

print(mat[1][0]);//5

print(mat[1][1]);//0

nizch = new char[3];

nizch[0] = 'a';

nizch[jedan] = 'b';

nizch[pet - 3] = 'c';

print(nizch[1]); //b

print(nizch[jedan \* 2]);//c

read(bodovi); // if bodovi = 5

bodovi = -bodovi + (pet \* a / 2 - jedan) \* bodovi - (3 % 2 + 3 \* 2 - 3);

print(bodovi);// then -14

}

}

*Primer2:*

program test301

const int nula = 0;

const int jedan = 1;

const bool pet = true;

const int N = 10;

int g;

bool m;

{

void main()

int bodovi;

bool bt;

int x,y;

int niz[];

{

g = N;

y = 53;

print(g);

g--;

x = -y + g \* N;

print(x);

x = 2;

y = 4;

niz = new int[N];

niz[6] = 5;

niz[0] = 111;

niz[0] = niz[x+y];

niz[0]--;

print(g);

g++;

x = 0;

y = 0;

niz[x+y] = niz[6];

print(niz[0]);

g--;

print(g);

print(y);

print(4);

}

}

DODATNE KLASE

1. MyTab.java klasa extend-uje Tab klasu koja manjka boolen tip podatka. U njoj se isti definiše, a inicijaliuje u Semantičkoj analizi prilikom pokretanja koda.

public static final Struct ***boolType*** = new Struct(Struct.***Bool***);

public static void myInit() {

*init*();

*currentScope*.addToLocals(new Obj(Obj.***Type***, "bool", ***boolType***));

}