Programski prevodioci: Projekat Šifra:PP-Q0Ga

Sadržaj

1. Napomena

Za svaki zadatak potrebno je uraditi:

1. Sintaksnu analizu
2. Semantičku analizu
3. Generisanje koda

2. Zadatak 1

Proširiti gramatiku jezika tako da omogući definisanje lokalnih promenljivih unutar bloka.

Realizovati semantičke provere:

1. Lokalna promenljiva koja je definisana unutar bloka važi od momenta deklaracije do kraja tog bloka.

```
int foo()  {
    int x;
    int y;

    {
        int x;
        ...
    }
}
```

3. Zadatak 2

Proširiti jezik FOR iskazom koji ima sledeći oblik:

```
"for" "(" <type> <id> "=" <lit1> "to" <lit2> ["step" <lit3> ]")"
    <statement>
```

Gde:

- <type> predstavlja tip podatka
- <id> predstavlja naziv promenljive, iterator petlje
- <lit1> predstavlja literal, početnu vrednost iterator
- <lit2> predstavlja literal, gornju granicu do koje ide iterator
- <lit3> predstavlja literali, korak za koji se menja iterator
- <statement> predstavlja iskaz

"step" <lit3> je opcioni deo.

Realizovati semantičke provere:

1. <id> treba da bude lokalna promenljiva za for iskaz (sledeći for iskaz može da definiše iterator sa istim imenom).
2. <lit1>, <lit2> i <lit3> moraju biti istog tipa kao i <id>.
3. Vrednost <lit1> mora da bude manja od vrednosti <lit2>.

Izvršavanje:

- Inicijalizacija iteratora se vrši samo jednom, pre prvog izvršavanja petlje. Iterator se inicijalizuje na vrednost prvog literala(<lit1>).
- Na početku svake iteracije potrebno je proveriti da li je iterator manji od drugog literala (<lit2>), ako jeste izvršiti telo petlje.
- Nakon izvršavanja tela petlje, iterator uvećati za 1 ili za korak <lit3> ukoliko je korak definisan.

Omogućiti i ugnježdene for iskaze.

Primer:

```
int zbir = 0;
int razlika = 0;
for (int i = 1 to 5 step 2){
    zbir = zbir + i;
    razlika = razlika - i;
}
for (int i = 0 to 3 )
    razlika = zbir - i;
```

4. Zadatak 3

Proširiti jezik SWITCH iskazom koji ima sledeći oblik:

```
"switch" "[" <switch_expression> "]" "{"
    "case" <constant_expression> "->" <case_body> ["finish" ";"]
    ...
    ["otherwise" "->" <otherwise_statement>]
"}"
```

Gde:

- <switch_expression> predstavlja ime promenljive
- <constant_expression> predstavlja konstantu
- <case_body> predstavlja iskaz (statement)
- <otherwise_statement> predstavlja iskaz (statement)

Mora postojati bar jedna case naredba. Finish naredba se opciono može pojaviti samo na kraju case naredbe. Otherwise naredba je opciona i može se pojaviti samo posle svih case naredbi

Realizovati sledeće semantičke provere:

1. Promenljiva u <switch_expression> mora biti prethodno deklarisana
2. Konstante u svim case iskazima moraju biti jedinstvene
3. Tip konstante u case naredbi mora biti isti kao tip promenljive u <switch_expression>

Izvršavanje:

- Na početku switch iskaza se izvrši provera vrednosti promenljive switch_expression.
- U zavisnosti od te vrednosti preusmerava se tok izvršavanja na telo odgovarajuće case naredbe.
- Ukoliko se na kraju case naredbe nalazi finish naredba, tok izvršavanja se preusmerava na kraj switch iskaza; a ako je finish naredba izostavljena, "propada" se na izvršavanje sledeće case naredbe.
- Otherwise naredba se izvršava ukoliko se vrednost switch promenljive razlikuje od svih konstanti navedenih u svim case naredbama.

Primer:

```
switch [a] {
    case 1 ->
        a = a + 5;
        finish;
    case 5 ->
    {
        b = 3;
    }
    otherwise ->
        a = a - 1;
}
```