Programski prevodioci: Projekat Šifra:PP-JN9SS

Sadržaj

1. Napomena

Za svaki zadatak potrebno je uraditi:

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

2. Zadatak 1

Omogućiti upotrebu logičkih operatora (and, or) i definisanje uslovnih izraza sa upotrebom ovih operatora.

Primer:

```
if (a > 3 and b < 4){
}
while ( c < 5 or a > 4 and v != 3){
}

NOTE: Nije neophodno da pravite while iskaz, osim ukoliko to nije traženo u nekom od
narednih zadataka.
```

3. Zadatak 2

Proširiti jezik FOR iskazom koji ima sledeći oblik:

```
"for" "(" <type> <id> "=" <lit1> ":" <lit2> ")"
    <statement>
```

Gde:

- <type> predstavlja tip podatka
- <id> predstavlja naziv promenljive, iterator petlje
- <lit1> predstavlja literal, inicijalna vrednost iteratora
- <lit2> predstavlja literal, granica iteratora
- <statement> predstavlja iskaz

Realizovati semantičke provere:

1. <id> treba da bude lokalna promenljiva za for iskaz (sledeći for iskaz može da definiše iterator sa istim imenom).
2. <id>, <lit1>, <lit2> moraju biti istog tipa.

Izvršavanje:

- Inicijalizacija iteratora se vrši samo jednom, pre prvog izvršavanja petlje (<lit1>).
  - Ako je lit2 >= lit1:
  - Na početku svake iteracije treba proveriti da li je iterator veći od drugog literala (<lit2>), ako nije izvršiti telo petlje.
  - Inkrement iteratora se vrši nakon izvršenja tela petlje.
- Ako je lit2 < lit1:
  - Na početku svake iteracije potrebno je proveriti da li je iterator manji od drugog literala (<lit2>), ako nije izvršiti telo petlje.
  - Dekrement iteratora se vrši nakon izvršenja tela petlje.

Omogućiti i ugnježdene for iskaze.

Primer:

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

4. Zadatak 3

Proširiti jezik CHECK iskazom koji ima sledeći oblik:

```
"check" "(" <check_expression> ")" "{"
    "case" <constant_expression> "=>" <case_body> ["break" ";"]
    ...
    ["otherwise" "=>" <otherwise_statement>]
"}"
```

Gde:

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

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

Realizovati sledeće semantičke provere:

1. Promenljiva u <check_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 <check_expression>

Izvršavanje:

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

Primer:

```
check (a){
    case 1 =>
        a = a + 5;
        break;
    case 5 =>
    {
        b = 3;
    }
    otherwise =>
        a = a + b;
}
```