Programski prevodioci: Vežbe 8

Sadržaj

1. Uvod

U ovoj nedelji biće rađeni zadaci vezani za generisanja koda.

2. Zadaci

2.1. Zadatak 1: `switch` iskaz

Proširiti miniC gramatiku `switch` iskazom.

Sintaksa `switch` iskaza ima oblik:

```
"switch" "(" switch_expression ")" "{"
  "case" constant_expression ":" case_body [ "break" ";" ]
  "case" constant_expression ":" case_body [ "break" ";" ]
     ...
  [ "default" ":" default_body ]
"}"
```

- `switch_expression` predstavlja ime promenljive koja prethodno mora biti deklarisana.
- `constant_expression` predstavlja konstantu.
- `case_body` i `default_body` predstavljaju iskaz. Postoji bar jedna `case` naredba.
- `default` naredba se može pojaviti samo nakon `case` naredbi (kao poslednja).
- `break` naredba se može pojaviti samo na kraju `case` naredbe.

Realizovati sledeće semantičke provere:

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

Izvršavanje:

- Na početku `switch` iskaza se izvrši provera vrednosti promenljive u zagradama.
- 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 `switch` iskaza; a ako je `break` naredba izostavljena, "propada" se na izvršavanje sledeće `case` naredbe.
- `default` naredba se izvršava ukoliko se vrednost `switch` promenljive razlikuje od svih konstanti navedenih u svim `case` naredbama

Primer 1:

```
switch (state) {
  case 10: { s = 1; } break;
  case 20: s = 2;
  default: s = 0;
}
```

Primer 2:

```
switch (state) {
  case 10: s = 1; break;
  case 20: { s = 2; }
}
```

2.2. Zadatak 2: `iterate`

Proširiti miniC iskaze iterator iskazom koji ima sledeći oblik:

```
    "iterate" <name> <lit1> "to" <lit2> <statement>
```

Gde:

- <name> iterator, predstavlja ime lokalne promenljive ili parametra
- <lit1> literal koji predstavlja korak
- <lit2> literal koji predstavlja kraj iteracije
- <statement> predstavlja iskaze

Izvršavanje:

- Pre početka petlje treba postaviti iterator (name) na vrednost 1.
- Tačnost relacije se proverava na početku svake iteracije i izvrsava se dokle god je <name> manje ili jednako <lit2>
- Nakon svake iteracije, vrednost iteratora se uvećava za korak lit1.

Omogućiti i ugnježdene iterate iskaze.

Primer:

```
int x;
int y;
y=0;
iterate x 3 to 20 {
  y = x + y;
}
```