Programski prevodioci

Vežbe 6

Sadržaj

1. Uvod

Na ovim vežbama vežbaćemo sintaksu i semantiku.

2. Zadaci

2.1. Zadatak 1 postinkrement kao iskaz

Proširiti miniC gramatiku postinkrement operatorom, koji se može pojaviti kao poseban iskaz. Postinkrement se može primeniti samo na promenljive i parametre, a ne i na funkcije.

```
a++;
```

Realizovati semantičku proveru:

1. Identifikator nad kojim se primenjuje postinkrement mora biti prethodno deklarisana promenljiva ili parametar.

2.2. Zadatak 2 lokalne promenljive unutar bloka

Proširiti miniC gramatiku tako da omogući definisanje lokalnih promenljivih unutar bloka i realizovati adekvatne semantičke provere:

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

Na kraju bloka, promenljive iz tog bloka je potrebno izbrisati iz tabele simbola.

2. U miniC jeziku je dozvoljeno da postoje 2 ili više promenljivih sa istim nazivom, *ako* se one nalaze u različitim blokovima. Jedina *nedozvoljena* situacija su 2 promenljive sa istim nazivom, koje su deklarisane u istom bloku.

Primer:

```
int main() {
  int x; ①
  int y;

  x = 2;
  y = 3;

  {
    int x; ②
    x = 5;
```

```
    y = x + y; ③
  }
  return x + y; ④
}

int main() {
  int x;
  x = 2;
  {
    int z;
    x = 5;
  }
  return x + z; ⑤
}
```

① Lokalna promenljiva `x`

② Nova promenljiva `x`, lokalna za ovaj blok.

③ Koristi se unutrašnja promenljiva `x` i spoljašnja promenljiva `y`. Prema tome, rezultat izraza je `y = 8`.

④ Koriste se spoljašnji `x` i `y`. Prema tome, rezultat izraza je `10`.

⑤ Greška—promenljiva `z` je lokalna za blok, pa u ovom trenutku više nije vidljiva.

POMOĆ:

- Funkcija `print_symtab()` radi prikaz tabele simbola.
- Kolona `atr2` u tabeli simbola za lokalne promenljive (`kind=VAR`) je neiskorišćena.
- Funkcija `clear_symbols(begin_index)` radi brisanje tabele simbola od elementa `begin_index` do kraja tabele.
- Funkcija `get_last_element()` vraća indeks poslednjeg popunjenog elementa iz tabele simbola.

2.3. Priprema za kolokvijum > Zadatak 3: `branch` iskaz

Proširiti jezik BRANCH iskazom koji ima sledeći oblik:

```
"branch" "(" <var> ";" <const1> [ "," <const2> "," ... <constN> ] ")"
    "do_start" <statement> "do_end"
    ...
    "do_start" <statement> "do_end"
```

Gde:

- <var> predstavlja ime promenljive

- <const1>, <const2>, ..., <constN> predstavljaju konstante
- <statement> predstavlja iskaz

Može postojati više konstanti odvojenih zarezom, ali mora postojati barem jedna. Može postojati više "do_start" <statement> "do_end", a mora postojati barem jednom.

Realizovati sledeće semantičke provere:

1. <var> mora biti prethodno deklarisana promenljiva ili parametar
2. Konstante const1, ... , constN moraju biti istog tipa kao i var
3. Konstante const1, ... , constN moraju biti jedinstvene .

Primer:

```
    branch ( a ; 1 , 3 , 5 )
        do_start a = a + 1; do_end
        do_start a = a + 3; do_end
        do_start a = a + 5; do_end
```

2.4. Zadatak 4 funkcija sa više parametara

Proširiti miniC gramatiku funkcijama sa više parametara

```
int foo(int p, unsigned b, unsigned c){
    return p;
}
```

Poziv funkcije

```
a = foo(2, 3u, a+b);
```

Realizovati sledeće semantičke provere:

1. Može se pozvati samo postojeća funkcija
2. Prilikom poziva funkcije obratiti pažnju da tipovi i broj parametara budu odgovarajući