Fakultet elektrotehnike i računarstva
Zavod za elektroniku, mikroelektroniku, računalne i inteligentne sustave

Prevođenje programskih jezika

Ak. God. 2014/2015

3. Laboratorijska vježba

Tema treće laboratorijske vježbe je semantička analiza. Vaš zadatak je programski ostvariti semantički analizator za programski jezik *PJ,* koji je opisan u uputi za prvu laboratorijsku vježbu. S obzirom na jednostavnost jezika *PJ*, zadaća semantičkog analizatora bit će provjera djelokruga i životnog vijeka varijabli, odnosno jesu li sve varijable definirane u trenutku korištenja.

Djelokrug deklaracija i životni vijek varijabli

Jezik *PJ* po klasifikaciji iz udžbenika koristi statičko pravilo djelokruga bez ugniježđenih procedura, ali podržava ugniježđene blokove, isto kao C.

Životni vijek globalnih varijabli počinje u trenutku njihove definicije, a završava na kraju izvođenja programa. Za varijable lokalne nekom bloku, životni vijek počinje njihovom definicijom i završava na kraju tog bloka. Trenutak definicije varijable u jeziku *PJ* je prva pojava varijable s lijeve strane u naredbi pridruživanja. U navedenom primjeru u prvom retku naredbom pridruživanja definiramo varijablu `x`. U drugom retku programa varijabla x se koristi te se definira varijabla `y`.

```
x = 3      //definicija varijable x
y = x + 3  //korištenje varijable x
```

`Za` - petlja definira zasebni blok naredbi unutar kojeg vrijedi djelokrug varijabli tog bloka naredbi. U navedenom primjeru varijabla `x` je definirana unutar cijelog programa, dok je varijabla `i` definirana samo unutar tijela **`za`** - petlje.

```
//primjer varijable definirane unutar bloka naredbi
x = 2
za i od 1 do 5
 x = x * i;
az
```

Lokalno ime može sakriti ime iz ugnježđujućeg djelokruga. U navedenom primjeru varijabla `x` ima djelokrug samo unutar tijela **`za`** - petlje, unutar kojeg je definirana. Isto tako, varijabla `i` unutar bloka **`za`** – **`az`** skriva varijablu `i` definiranu izvan bloka. Tako varijabla `i` unutar tijela **`za`** - petlje poprima vrijednosti od 1 do `n` (10), dok u ostatku programa ima vrijednost 15.

```
//primjer skrivanja varijabli
n = 10
rez = 0
i = 15
za i od 1 do n
 rez = rez + i*i*i    //i ∈ {1..n}
 x = 5
az
//i = 15
```

Semantičke pogreške

Ukoliko semantička pravila nisu zadovoljena, dolazi do semantičke pogreške. U jeziku *PJ* moguće su semantičke pogreške isključivo u slučaju korištenja nedefiniranih varijabli. U navedenom primjeru koristi se nedefinirana varijabla `Z`, što dovodi do semantičke pogreške.

```
//primjer korištenja nedefinirane varijable
y = z * 3
```

Semantički analizator mora voditi računa i o životnom vijeku varijabli. U navedenom primjeru dolazi do semantičke pogreške jer se koristi varijabla `i` definirana unutar tijela `za` - petlje nakon što je prestao njen životni vijek.

```
//korištenje varijable izvan njenog životnog vijeka
x = 1
n = 10
za i od 1 do n
 x = i * 3
az
y = i + 1
```

Ulaz i izlaz programa sintaksnog analizatora

Ulaz semantičkog analizatora je generativno stablo dobiveno sintaksnom analizom izvornog programa iz jezika *PJ* u istom formatu kao izlaz iz sintaksnog analizatora u drugoj laboratorijskoj vježbi. Tijekom ove upute će se radi ilustracije pojavljivati odsječci programskog koda u jeziku *PJ*, ali sintaksni analizator neće dobivati na ulaz takav programski kod, već rezultat sintaksne analize nad njim.

Semantički analizator treba obići generativno stablo i na standardni izlaz za svako korištenje varijabli ispisati broj retka u kojem se ta varijabla koristi te broj retka u kojem je varijabla definirana. Definicija varijable se **ne smatra** korištenjem te za nju ne treba ispisati navedene podatke. Odnosno, semantički analizator ne mora dati ispis za pojavljivanje varijable ako se ona nalazi s lijeve strane znaka jednakosti u pridruživanju, već samo sa desne strane.

Ispis treba biti u sljedećem obliku:
`<br_retka_koristenja> <br_retka_definicije> <leksicka_jedinka>`

Broj retka korištenja te leksička jedinka definirane su u čvoru generativnog stabla. Semantički analizator treba interno pamtiti podatke o mjestu definicije varijabli.

U slučaju pogreške treba ispisati broj retka i leksičku jedinku u kojoj je otkrivena prva semantička pogreška. Semantički analizator ne treba provoditi nikakav postupak oporavka od pogreške. Drugim riječima, čim otkrije prvu semantičku pogrešku i obavi odgovarajući ispis, semantički analizator prestaje s radom. Ispis greške treba biti u sljedećem obliku:
`err <br_retka> <leksicka_jedinka>`

Primjer

U ovom poglavlju prikazana je sintaksna analiza sljedećeg jednostavnog *PJ* programa:

```
n = 5
rez = 0
za i od n do n+5
 rez = rez - i*i + i/3
az
```

Leksičkom analizom dobili bismo

```
IDN 1 n
OP_PRIDRUZI 1 =
BROJ 1 5
IDN 2 rez
OP_PRIDRUZI 2 =
BROJ 2 0
KR_ZA 3 za
IDN 3 i
KR_OD 3 od
IDN 3 n
KR_DO 3 do
```

```
IDN 3 n
OP_PLUS 3 +
BROJ 3 5
IDN 4 rez
OP_PRIDRUZI 4 =
IDN 4 rez
OP_MINUS 4 -
IDN 4 i
OP_PUTA 4 *
IDN 4 i
OP_PLUS 4 +
IDN 4 i
OP_DIJELI 4 /
BROJ 4 3
KR_AZ 5 az
```

U skladu s navedenim sintaksnim pravilima, program je sintaksno ispravan i grafički prikaz njegovog generativnog stabla možete vidjeti ovdje. Rezultat sintaksne analize, *preorder* obilazak stabla, može se vidjeti ovdje. Ovaj je program semantički ispravan te izlaz semantičkog analizatora za dani program možete vidjeti u nastavku:

```
3 1 n
3 1 n
4 2 rez
4 3 i
4 3 i
4 3 i
```

U trećem retku programa dva puta se koristi varijabla `n` definirana u prvom retku programa. U četvrtom retku programa koristi se varijabla `rez` definirana u drugom retku, te tri puta varijabla `i` definirana u trećem retku.

Savjeti za implementaciju

U programskoj implementaciji možete iskoristiti strukturu podataka generativnog stabla koju ste ostvarili u 2. laboratorijskoj vježbi, kao i metode stvaranja i obilaska istog.

Predaja rješenja na SPRUT

Rješenje se predaje kao zip arhiva u kojoj se nalazi sav potreban kod za prevođenje/izvođenje. Vrijede ista pravila kao na UTR-u (specifično, u **Javi nemojte koristiti pakete**). Ulazna točka za rješenja u Javi je razred SemantickiAnalizator, a za Python datoteka SemantickiAnalizator.py. Za ostale jezike je organizacija koda proizvoljna (naravno, mora

postojati točno jedna funkcija/metoda main itd.). Vremensko ograničenje za izvođenje semantičkog analizatora je dvije minute (što je daleko više od očekivanog vremena izvođenja).