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

Prevođenje programskih jezika

1. Laboratorijska vježba

Tema prve laboratorijske vježbe je leksička analiza programskog koda. Vaš zadatak je programski ostvariti leksički analizator za jednostavan programski jezik *PJ* koji je opisan u nastavku upute.

Jezik *PJ*

Jezik *PJ* omogućuje cjelobrojno računanje uz proizvoljan broj cjelobrojnih varijabli za pohranu međurezultata. Na primjer, program za računanje zbroja kubova prvih deset prirodnih brojeva u jeziku *PJ* bi se mogao napisati ovako:

```
// stavi sumu kubova prvih deset prirodnih brojeva u varijablu rez
n = 10 // varijable ne treba deklarirati prije inicijalizaije
rez = 0
za i od 1 do n
  rez = rez + i*i*i
az
```

U prikazanom primjeru ključne riječi označene su masnim slovima. Ključne riječi u jeziku su

za
od
do
az (obrnuto od **za**; zatvara blok petlje)

Jezik podržava samo linijske komentare koji počinju leksičkom jedinkom `//` i protežu se do kraja retka. Programi se pišu koristeći ASCII znakove. Imena varijabli (identifikatori) počinju znakom engleske abecede nakon čega dolazi proizvoljan broj znakova engleske abecede i dekadskih znamenaka. Konstante u jeziku su isključivo cjelobrojne, dekadske i nenegativne (veće ili jednake nuli; negativne brojeve dobiva se prefiksiranjem unarnim operatorom `-`). Za pridruživanje se koristi operator `=`. S lijeve strane operatora pridruživanja nalazi se jedan identifikator, a s desne strane se nalazi izraz u kojem se mogu pojavljivati varijable i konstante uz binarne aritmetičke operatore `+`, `-`, `*` i `/` i oble zagrade za određivanje redoslijeda izračunavanja. Jezik osim binarnih podržava i unarne operatore `+` i `-`, ali oni se u leksičkoj analizi ne razlikuju od binarnih operatora, tj. pripadaju istoj klasi leksičkih jedinki. Između leksičkih jedinki mogu se nalaziti bjeline i to znak praznine (space), tab (\t) ili znak za novi redak (\n). Neke od navedenih značajki nisu bitne za leksičku analizu, ali su navedene kako biste dobili jasniju sliku o jeziku.

Primjer programa sa složenijim izrazima prikazan je u nastavku:

```
x = 0
za y od -10 do 10
  x = (x+y)*10 + x/10
```

```
  x = x*x - 3
az
y = x * (x - 123)
```

Programsko rješenje laboratorijske vježbe sa standardnog ulaza treba pročitati program u jeziku *PJ*. Na standardni izlaz treba ispisati niz uniformnih znakova pri čemu je svaki uniformni znak u svojem retku. Uz svaki uniformni znak treba ispisati pripadni broj retka u kojemu se leksička jedinka nalazi (retci se broje od 1) i samu leksičku jedinku tj. grupirani niz znakova ulaznog programa. Ove tri komponente svakog retka trebaju biti odvojene jednim razmakom.

Uniformni znakovi koje leksički analizator treba ispisivati su:

IDN za identifikatore
BROJ za konstante
OP_PRIDRUZI za operator =
OP_PLUS za operator +
OP_MINUS za operator -
OP_PUTA za operator *
OP_DIJELI za operator /
L_ZAGRADA za lijevu zagradu (
D_ZAGRADA za desnu zagradu)
KR_ZA za ključnu riječ za
KR_OD za ključnu riječ od
KR_DO za ključnu riječ do
KR_AZ za ključnu riječ az

Na primjer, za ulaz

```
// stavi sumu kubova prvih deset prirodnih brojeva u varijablu rez
n = 10 // varijable ne treba deklarirati prije inicijalizaije
rez = 0
za i od 1 do n
  rez = rez + i*i*i
az
```

leksički analizator treba ispisati:

```
IDN 2 n
OP_PRIDRUZI 2 =
BROJ 2 10
IDN 3 rez
OP_PRIDRUZI 3 =
BROJ 3 0
KR_ZA 4 za
IDN 4 i
KR_OD 4 od
BROJ 4 1
KR_DO 4 do
IDN 4 n
IDN 5 rez
OP_PRIDRUZI 5 =
```

```
IDN 5 rez
OP_PLUS 5 +
IDN 5 i
OP_PUTA 5 *
IDN 5 i
OP_PUTA 5 *
IDN 5 i
KR_AZ 6 az
```

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. Ulazna točka za rješenja u Javi je razred LeksickiAnalizator, a za Python datoteka LeksickiAnalizator.py. Za ostale jezike je organizacija koda proizvoljna (naravno, mora postojati točno jedna funkcija/metoda main itd.).

Ograničenja na način ostvarenja leksičkog analizatora

Način ostvarenja leksičkog analizatora je proizvoljan, **ali nije dozvoljeno korištenje biblioteka i alata koji implementiraju ključne dijelove gradiva predmeta. Specifično, nije dozvoljeno koristiti regex biblioteke niti postojeće generatore leksičkih analizatora.** Jedan od mogućih pristupa za rješavanje vježbe je za svaku klasu leksičkih jedinki definirati pripadni konačni automat. Definirane automate može se iskoristiti za ostvarenje leksičkog analizatora koristeći jedan od dva algoritma leksičkog analizatora iz udžbenika. Kako jezik ima mali broj klasa leksičkih jedinki, ukupni broj stanja i prijelaza automata također će biti malen. U rješenju slobodno možete koristiti bilo kakav **vlastiti** kod (npr. za automate iz laboratorijske vježbe iz UTR-a).