Sveučilište u Zagrebu
Fakultet elektrotehnike i računarstva

Gunslinger

Seminarski rad iz predmeta
Prevođenje programskih jezika

Zadatak br. 2025

Zagreb, siječanj 2011.

Sadržaj

1. Zadatak …….……3
2. Uvod …….……4
   - Leksička analiza ……....…..…..……..4
   - LEX ……....…….5
   - PLY (Python LEX-YACC) ……...……...……...……..5
   - Programski jezik C ……..……6
3. Programsko ostvaranje ……...….....……6
4. Upute za korištenje ……....……8
5. Zaključak ……...……8
6. Literatura ……..……..9

1. Zadatak

Student: Gunslinger

Matični broj studenta:

Zadatak 2025:

Pomoću programa LEX izgraditi jednostavan program za statističku analizu programa pisanih u jeziku C. Statistički program treba učitati izvorni program i u izlaznu datoteku ispisati broj redaka, broj (različitih) identifikatora, broj (različitih) konstanti te učestalost uporabe pojedinih ključnih riječi, operatora i specijalnih znakova.

2. Uvod

Leksička analiza

Leksička analiza je prvi korak jezičnog procesora. To je jedini dio koji izravno pristupa izvornom kôdu. On grupira znakove izvornog kôda u leksičke jedinke koje zatim klasificira, usput bilježeći parametre pojedine jedinke. Također generira niz uniformnih znakova koji prikazuje klase leksičkih jedinki onim redoslijedom kojim su one zapisane u izvornom kôdu. Uz sve to, leksički analizator pronalazi greške te određuje njihovo mjesto u izvornom kôdu. Izlaz leksičkog analizatora su tablica uniformnih znakova te tablice koje sadrže konkretne leksičke jedinke s parametrima. Leksički analizator tipično gradi četiri tablice:

• Tablica ključnih riječi, operatora i specijalnih znakova (KROS tablica)

• Tablica identifikatora

• Tablica konstanti

• Tablica uniformnih znakova

KROS tablica je fiksna – sadrži skup leksičkih jedinki specifičan za određeni programski jezik. U ovom slučaju nema parametara. U tablicu identifikatora upisuju se varijable, imena funkcija i potprograma, labele i sl. Također nema parametara. Tablica konstanti sadrži popis brojeva, pojedinačnih znakova i stringova koji su eksplicitno napisani u izvornom kôdu. Parametar im je tip konstante (float, integer, char i string). Tablica uniformnih znakova u svakom retku, dakle za svaku leksičku jedinku, sadrži i pokazivač na određeni zapis u pripadajućoj tablici s informacijama o leksičkim jedinkama. Te su (četiri) tablice jedino što je potrebno daljnjem prevođenju programa.

LEX

Lexical Analyser Generator (hr. generator leksičkih analizatora) jest program koji od skupa pravila zadanih kombinacijom regularnih izraza i naredaba programskog jezika C stvara kôd (napisan u jeziku C) koji, pravilno kompiliran, daje program koji analizira ulaz (datoteku ili s tipkovnice), pronalazi u njemu definirane regularne izraze te, po pronalasku svakog izraza, izvršava zadani programski kôd. Kao što mu sámo ime govori, najčešće se koristi za generiranje leksičkih analizatora.

Forma tekstualne datoteke s pravilima strogo je definirana:

```
%{
     Deklaracije funkcija i globalnih varijabli
     pisane u C-u, također i funkcija main()
%}
     Regularne definicije oblika:
     {ime lex jedinke}   regularni izraz
%%
     Pravila prevođenja oblika:
     regularni izraz     kôd u C-u
%%
     Pomoćne procedure
```

No, u ovom radu koristi se Python implementacija LEX-a, koji je nešto drugačije strukturiran, što je opisano u poglavlju Programsko ostvarenje.

PLY (Python LEX-YACC)

PLY je implementacija alata LEX i YACC (Yet Another Compiler Compiler) napisana za Python, u Python-u. Sastoji se od dva glavna modula: lex.py i yacc.py. Lex.py modul se koristi za razbijanje ulaznog teksta u zbirku takozvanih tokena koji se određuju prema pravilima napisanih regularnih izraza. Yacc.py se koristi za prepoznavanje sintakse jezika specificirane u obliku kontekstno neovisne gramatike.

U ovom radu koristi se samo lex.py, pošto je potrebna samo statička analiza ulaznog programa.

Programski jezik C

Programski jezik **C** spada u proceduralne programske jezike koji je razvijen u ranim 70-im godinama 20. stoljeća, ali se dosta mijenjao tokom godina te je u više navrata neformalno i formalno standardiziran. Kao jedan od najvažnijih jezika u povijesti komercijalne računalne industrije, **C** je do danas ostao jedini programski jezik prilagođen za sve računalne platforme, od malih sustava pa do mrežnih superračunala. Programi napisani u njemu vrlo su bliski načinu rada hardvera te u načelu zahtijevaju od programera dobro razumijevanje rada procesora, memorije, ulazno-izlaznih sklopova itd. No, rad s registrima procesora i adresiranje memorije apstrahirani su pomoću koncepta varijabli i pokazivača što uz eksplicitne kontrolne strukture i funkcije znatno olakšava programiranje u odnosu na izravno programiranje u strojnim jezicima.

C je jezik opće namjene, što znači da se u njemu može napraviti apsolutno sve: od rješavanja zadataka, do pisanja drivera, operacijskih susatava, tekst procesora ili igara. C, kao jezik, ni u čemu ne ograničava. Omogućuje i uključivanje naredbi pisanih asemblerski, zbog čega je zajedno s mogućnošću direktnog pristupa pojedinim bitovima, bajtovima ili cijelim blokovima memorije, pogodan za pisanje sistemskog softvera. Zbog tih karakteristika C je među popularnijim programskim jezicima i rabe ga mnogi programeri. Rezultat toga je postojanje velikog broja prevoditelja za C i alata te stalno dostupne pomoći na internetu. Programi pisani u C-u su prenosivi (mogu se prevoditi i izvršavati na različitim porodicama računala uz minimalne ili nikakve ispravke) i obično su vrlo brzi. Postoje mnogi prevoditelji za jezik C, a jedan od najšire korištenih je GNU C Compiler.

U ovom radu vrši se jednostavna statička analiza jezika C.

3. Programsko ostvarenje

U ovom poglavlju bit će dokumentirana struktura glavnog programa (clex.py) koja je zapravo standardna struktura LEX programa pisanih u PLY-u.

Clex.py sastoji se od nekoliko bitnih dijelova, koji će ovdje biti navedeni u najjednostavnijem obliku:

Popis tokena:

```
tokens = ('IDENTIFIKATOR', 'BROJ', 'PLUS', 'MINUS')
```

Popis tokena sadrži potpunu listu jedinki koje leksički analizator prepoznaje. Također, ime

tokena mora biti važeći identifikator; ‘BROJ’ je pravilno ime tokena, dok znak ‘-‘ nije.

Popis rezerviranih riječi:

```
rezervirane_rijeci = {
'if' : 'IF',
'int' : 'INT',
'float' : 'FLOAT',
'for' : 'FOR'
}
```

Gotovo svi jezici, pa tako i C, imaju definirane ključne riječi koje se ne mogu koristiti kao imena varijabli ili funkcija. Stoga, potrebno je takve riječi odmah popisati kako se ne bi mogle upotrijebiti za imena identifikatora.

Popis specijalnih znakova:

```
t_COMMA = r','
t_COLON = r':'
t_SEMICOLON = r';'
```

Koristeći jednostavne regularne izraze, pridružujemo određene specijalne znakove njihovim prethodno popisanim tokenima.

Definicije složenih tokena:

```
def t_INUMBER(t):
      r'0(?!\d)|([1-9]\d*)'
      return t


def t_CHARACTER(t):
       r"'\w'"
      return t
```

Specijalni znakovi se lako prepoznaju, no za razne brojeve, identifikatore i nizove znakova moramo definirati posebne funkcije kako bi naznačili kako želimo da ih se prepoznaje. To naravno činimo regularnim izrazima.

Popis tokena koji se ignoriraju:

```
def t_WHITESPACE(t):
      r'[ \t]+'
     pass


def t_NEWLINE(t):
      r'\n+'
     t.lineno += len(t.value)
```

U ovom dijelu definiramo funkcije za izraze koje želimo ignorirati. Takve funkcije ne vraćaju prepoznatu jedinku, no možemo (kao što je ovdje stavljeno isključivo za primjer) definirati razne podrutine ukoliko je to potrebno.

Glavni dio programa:

Glavni dio ovog programa zapravo nije potreban za normalan rad PLY-a, ali je potreban za ovaj zadatak. Pozivom metode lex.py-a

```
lex.input(strings)
```

leksičkom analizatoru dajemo jedan niz znakova koji se sastoji od svih učitanih linije ulazne datoteke, bez znakova novog reda, a pomoću

```
token = lex.token()
```

dobavljamo iz analizatore tokene koje je prepoznao. Ostatak programa služi samo da prepoznavanje i ispis traženih podataka – statičke analize programa pisanog u C-u, osim poziva metode

```
lex.lex()
```

kojom se pokreće leksički analizator.

5. Upute za korištenje

Program se može pokrenuti na dva načina:

1. Iz komandne linije: `python clex.py test.c`
2. Dvoklikom. Ako želimo na taj način pokrenuti program, potrebno je u njemu (clex.py) odkomentirati liniju `#file = open("IME_DATOTEKE")` te staviti komentar na liniju `file = open(sys.argv[1])`

Program je predan tako da se pokreće iz komandne linije. Također, u istom direktoriju moraju se nalaziti datoteke clex.py i lex.py. Put do C programa (u primjeru "test.c") može se zadati u argumentu ili kodu.

6. Zaključak

Python LEX-YACC je alat koji je iznimno lako koristiti, uz relativno slabo poznanstvo samog jezika. Struktura kojom se definiraju leksičke jedinke jezika je iznimno razumljiva, kao i sam Python. Također, alat uvelike olakšava leksičku analizu jezika, i bilo mi je zadovoljstvo koristiti ga.

7. Literatura

- PLY (Python LEX-YACC) službena dokumentacija (David M. Beazley): http://www.dabeaz.com/ply/ply.html#ply_nn1
- Parsing with PLY (Dalke Scientific Software, 2010.): http://www.dalkescientific.com/writings/NBN/parsing_with_ply.html