Programski Prevodioci - Projekat

Cilj projektnog zadatka je individualno implementirati interpreter i generator koda za Pascal programski jezik. Potrebno je generisati kod u C programskom jeziku koji ne sme sadržati sintaksne greške. Savet je modifikovati implementaciju urađenu na vežbama. Projekat se sastoji iz dve faze. Prva faza podrazumeva generisanje apstraktnog sintaksnog stabla (vežbe 4 i 5), dok je druga faza posvećena implementaciji interpretera (vežbe 9 i 10) i generatora koda (vežbe 6).

Prva faza [18 poena] - Potrebno je podržati sledeće celine:

- Promenljive i tipovi podataka (integer, real, boolean, char, string)
- Binarne i unarne aritmetičke operacije (+, -, *, /, div, mod)
- Binarne i unarne logičke operacije (not, and, or, xor)
- Relacione operacije (=, <>, <, <=, >, >=)
- If-Else naredba
- Repeat-Until naredba sa instrukcijama za kontrolu toka (break, continue)
- While-Do naredba sa instrukcijama za kontrolu toka (break, continue)
- For-Do naredba sa instrukcijama za kontrolu toka (to, downto, break, continue)
- Funkcije za rad sa brojevima (inc, dec, ord, chr)
- Funkcije za rad sa stringovima (length, insert)
- Funkcije za rad sa konzolom (read, write, readln, writeln)
- Korišćenje nizova primitivih tipova podataka
- Korisnički definisane funkcije i procedure (function, procedure, exit)

Druga faza [32 poena] - Potrebno je podržati sledeće celine:

- Interpreter Pascal koda - potrebno je podržati sve celine iz prve faze
- Generator C koda - potrebno je podržati sve celine iz prve faze
- Prijavljivanje grešaka:
    - Sintaksne greške pronađene leksičkom analizom izvornog koda
    - Korišćenje nedeklarisane promenljive
    - Korišćenje nekompatibilnih tipova
    - Detektovanje beskonačne rekurzije

Testiranje implementacije

Potrebno je testirati implementaciju koristeći predefinisane datoteke sa izvornim kodovima što se može postići pomoću priloženog koda u nastavku. Generisani kod i interpreter treba da završe izračunavanje svakog test primera u 5 sekundi. Po isteku roka, svi radovi će automatski biti testirani nad tim datotekama i ocenjeni po broju tačnih ispisa. Uz testiranje, vršiće se detekcija plagijarizma. Plagirani radovi će biti ocenjeni sa nula poena. Obavezno je korišćenje argumenata komandne linije za unos putanja do datoteka sa izvršnim i generisanim kodovima. Za ove potrebe je pogodno iskoristiti priloženi kod u nastavku. Neposredno pre slanja projekta je potrebno obrisati sve pozive print funkcije kako bi se prilikom testiranja na standardnom izlazu prikazao samo traženi ispis.

Rokovi za predaju projekta

Projektni zadatak je potrebno poslati u *ipynb* formatu pod nazivom *ime-prezime-indeks* pomoću formulara pre početka prve (druge) kolokvijumske nedelje. Odbrana projektnog zadatka je obavezna i održaće se u prvoj (drugoj) kolokvijumskoj nedelji u terminu predviđenom za ovaj predmet.

Prilog

Prva ćelija - Korišćenje argumenata komandne linije:

```
# ACINONYX - BEGIN

DEBUG = True # OBAVEZNO: Postaviti na False pre slanja projekta

if DEBUG:
   test_id = '01' # Redni broj test primera [01-16]
   path_root = '/content/Datoteke/Druga faza/'
   args = {}
   args['src'] = f'{path_root}{test_id}/src.pas' # Izvorna PAS datoteka
   args['gen'] = f'{path_root}{test_id}/gen.c' # Generisana C datoteka
else:
   import argparse
   arg_parser = argparse.ArgumentParser()
   arg_parser.add_argument('src') # Izvorna PAS datoteka
   arg_parser.add_argument('gen') # Generisana C datoteka
   args = vars(arg_parser.parse_args())

with open(args['src'], 'r') as source:
   text = source.read()
   lexer = Lexer(text)
   tokens = lexer.lex()
   parser = Parser(tokens)
   ast = parser.parse()
   symbolizer = Symbolizer(ast)
   symbolizer.symbolize()
   generator = Generator(ast)
   generator.generate(args['gen'])
   runner = Runner(ast)
   runner.run()

# ACINONYX - END
```

Druga ćelija - Testiranje interpretera i generatora koda:

```
# GRADER - BEGIN
# 1. Preuzeti direktorijum Datoteke sa materijala
# 2. Postaviti arhivu u content direktorijum u okviru sesije
# 3. Napraviti i pokrenuti ćeliju sa komandom !unzip 'arhiva.zip'
# 4. Postaviti DEBUG promenljivu na False
# 5. Zakomentarisati linije koje počinju znakom !
# 6. Preuzeti notebook kao .py datoteku i imenovati je main.py
# 7. Postaviti main.py na putanju na koju pokazuje path_root
# 8. Postaviti DEBUG promenljivu na True
# 9. Otkomentarisati linije koje počinju znakom !
# 10. Pokrenuti grader.sh pokretanjem ove ćelije
if DEBUG:
   path_grader = f'{path_root}grader.sh'
   !chmod +x '{path_grader}' # Dozvola za izvršavanje
   !bash '{path_grader}' '{path_root}' # Pokretanje gradera
# GRADER - END
```