به نام خدا

گزارش پروژه ی کامپایلر

یاسمن سادات میرمحمد

۹۴۳۱۰۲۲

---

## ساختار کلی پروژه:

شامل lexer,parser,generator است که هر یک وظیفه ی سه فاز مختلف از کامپایلر را بر عهده دارند.

## فاز اول

### تحلیل لغوی:

در این فاز کافی است بتوانیم توکن ها را تشخیص بدهیم. برای این کار، از کتابخانه ی ply در پایتون استفاده کردم و با توجه به اطلاعات مشخص شده در فاز اول پروژه ، توکن ها را از هم جدا کردم.

لازم به ذکر است در کد نمونه یک سری توکن و عبارت وجود داشتند که در صورت پروژه به آن ها اشاره نشده بود اما من برای کارایی بیشتر کامپایلر خود، آن ها را در نظر گرفتم. همچنین هربار که توکن های فایل ورودی جاپ میشود، شماره خط و بقیه اطلاعات آن را چاپ میکنم.

برای انجام این پروژه از زبان پایتون و پکیج PLY برای استفاده از Lex و Yacc در زبان پایتون استفاده کردم. دلیل انتخاب این زبان سادگی و بیانگر بودن صورت پروژه برای مشاهده‌ی خطاها و قراردادها و سادگی یافتن باگ‌ها استفاده کردم.

در پیاده سازی این پروژه از اسلایدهای موجود در این لینک استفاده کردم.

نتیجه روی فایل ورودی sample_1 :

```
F:\Anaconda3\python.exe
D:/Compiler/Project/Final_Project/my_Impl/lexer/lexer.py
```

:Traceback (most recent call last)

LexToken(INTEGER_KW,’int’,1,0)

File “D:/Compiler/Project/Final_Project/my_Impl/lexer/lexer.py”, line 225, in <<module

LexToken(NUMBER,’1’,1,4)

:for token in lexer

Illegal character -> f

(LexToken(CLOSING_PARENTHESES,’)’,1,15

(LexToken(OPENING_BRACE,’{‘,2,18

LexToken(INTEGER_KW,’int’,3,21)

Illegal character -> f

File “F:\Anaconda3\lib\site-packages\ply\lex.py”, line 419, in next

LexToken(SEMICOLON,’;’,3,34)

()t = self.token

LexToken(INTEGER_KW,’int’,4,37)

Illegal character -> s

File “F:\Anaconda3\lib\site-packages\ply\lex.py”, line 363, in token

LexToken(SEMICOLON,’;’,4,51)

([:func.__name__, newtok.type), lexdata[lexpos

Illegal character -> f

LexToken(NUMBER,’5’,5,64)

LexToken(MUL,’*’,5,66)

ply.lex.LexError:
D:/Compiler/Project/Final_Project/my_Impl/lexer/lexer.py:175: Rule ‘t_KW’ 'returned an unknown token type ‘VOID

LexToken(NUMBER,’1’,5,68)

```
LexToken(NUMBER,'0',5,69)
LexToken(SEMICOLON,';',5,71)
Illegal character -> s
Illegal character -> f
LexToken(PLUSPLUS,'++',6,94)
LexToken(SEMICOLON,';',6,97)
Illegal character -> G
Illegal character -> s
(LexToken(CLOSING_BRACE,'}',9,144

Process finished with exit code 1
```

فاز دوم

کانفلیکت‌ها:

## Shift Reduce و Reduce Reduce

در گرامر مورد نظر کانفلیکت‌های Shift/Reduce و Reduce/Reduce وجود داشت.

یک دلیل وجود این مشکلات وجود قواعد اپسیلون در گرامر بود که با استفاده از فرمول موجود در این لینک سعی کردم اونها رو برطرف کنم.

بعضی از مشکلات دیگر در عملیات‌های محاسباتی و شروط هم باعث می‌شد تا کانفلیکت‌های دیگری رخ بدهد که پس از جست و جوف برای حل آن‌ها از تابع Precedence برای اولویت بندی استفاده شد.

فاز سوم:

تولید کد میانی

برای پیاده سازی، علاوه بر نکات مشخص شده در صورت پروژه، از این لینک استفاده کردم:

https://github.com/2020saurav/py-icg/blob/master/src/converter.py