**COMPILER**

**TERM PROJECT**

박준석

20200267

소프트웨어학부

1. 밑에는 사용한 CFG입니다.

01 : S -> CODE

02 : CODE -> VDECL CODE' | FDECL CODE' | CDECL CODE' | ^

03 : CODE' -> VDECL CODE' | FDECL CODE' | CDECL CODE' | ^

04 : VDECL -> vtype id semi | vtype ASSIGN semi

05 : ASSIGN -> id assign RHS

06 : RHS -> EXPR | literal | character | boolstr

07 : EXPR -> TERM EXPR'

08 : EXPR' -> addsub TERM EXPR' | multdiv FACTOR EXPR' | ^

09 : TERM -> FACTOR TERM'

10 : TERM' -> multdiv FACTOR TERM' | ^

11 : FACTOR -> lparen EXPR rparen | id | num | boolstr

12 : FDECL -> vtype id lparen ARG rparen lbrace BLOCK RETURN rbrace | vtype id lparen ARG rparen

13 : ARG -> vtype id MOREARGS | ^

14 : MOREARGS -> comma vtype id MOREARGS | ^

15 : BLOCK -> STMT BLOCK | ^

16 : STMT -> VDECL | ASSIGN semi | if lparen COND rparen lbrace BLOCK rbrace ELSE | while lparen COND rparen lbrace BLOCK rbrace

17 : COND -> EXPR comp EXPR

18 : ELSE -> else lbrace BLOCK rbrace | ^

19 : RETURN -> return RHS semi | ^

20 : CDECL -> class id lbrace ODECL rbrace

21 : ODECL -> VDECL ODECL' | FDECL ODECL' | ^

22 : ODECL' -> VDECL ODECL' | FDECL ODECL' | ^

2. 프로그램 실행 방법.

우선 프로그램을 실행하기 위한 환경을 구축해야 합니다. 먼저, ‘python -m venv venv’ 를 실행하여 ‘python virtualenv’ 환경을 갖춘 후에 ‘pip install -r requirements.txt’ 로 필요한 라이브러리 목록을 설치해줍니다. 이러면 프로그램을 실행 할 준비는 끝납니다.

먼저 input\_code라는 txt 파일에 parsing 하고자 하는 코드를 넣습니다. 그 다음에 console 창에 “python lexer.py”를 실행하면 tokens.py라는 코드를 토큰으로 만든 파일이 생성됩니다. 그 다음에는 console 창에 “python slr\_parser.py cfg.txt tokens.txt” 명령어를 실행하면 slr-parsing table이 생성되고 token의 parsing 과정과 parse tree를 출력하게 됩니다. 그리고 명령어를 “python slr\_parser.py -g cfg.txt tokens.txt” 로 실행한다면 automaton이 생성됩니다.

사진을 통해서 좀 더 자세한 실행 방법을 설명드리겠습니다.

텍스트, 스크린샷, 소프트웨어, 컴퓨터이(가) 표시된 사진

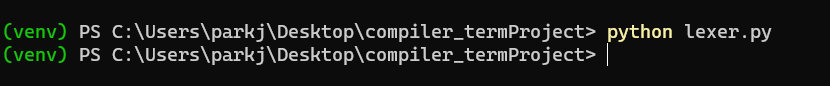
자동 생성된 설명

프로젝트 디렉토리에서 powershell을 열어줍니다.

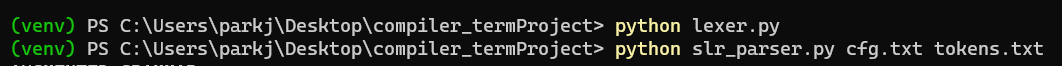
텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

Python의 virtualenv를 실행합니다.



lexer.py를 실행해서 코드를 token화 합니다. 만약 token을 넣고 싶다면 이 작업은 건너뛰고 tokens.txt만 수정하여 사용하면 됩니다.



명령어를 실행하여 parsing 합니다.

3. 코드 구성.

우선 코드는 세 파트로 나뉘어집니다. 코드를 받아서 lexing을 하는 lexer.py와 파싱을 하는데 필요한 cfg를 받아서 해석하는 grammar.py 그리고 parser를 구축하는 slr\_parser.py로 나뉘어집니다.

lexer.py에는 token으로 바꾸기 위해 token에 대한 정의를 써놨습니다. 그리고 input\_code.txt.를 받으면 한 줄의 token으로 나열되어 tokens.txt를 output으로 냅니다.

grammar.py에는 CFG를 해석하는 코드를 넣었습니다. 텍스트 형태로 제공되기 때문에 텍스트를 파싱하는 과정을 포함해서 CFG를 잘 해석하게끔 합니다.

slr\_parser에는 주어진 CFG를 grammar를 import해서 parser를 구축하도록 합니다. 이 코드에는 first\_follow set을 구성하여 나타내도록 하는 함수가 있고 SLRParser class 안에는 parsing table을 만드는 construct\_table 함수, automaton을 만들어주는 generate\_automaton 함수, 그리고 가장 중요한 LR\_Parser를 만들어주는 LR\_Parser 함수가 있습니다. LR\_Parser 함수에 대한 설명을 하자면 stack을 사용하여 parsing을 진행하도록 합니다. Parsing table을 구현하는 코드도 여기에 사용한 stack을 재사용하여 만들었습니다. 특히, reduce를 하면 부모 노드가 되도록 하는 알고리즘을 사용하였습니다.

4. 몇가지 간단한 예제

먼저 가장 간단한 예제부터 보여드리겠습니다.

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

Input\_code.txt에 내용을 작성합니다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

Python lexer.py를 실행하면 다음과 같은 tokens.txt가 생성됩니다.

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

명령어를 실행하면 다음과 같은 parsing 과정과 parse tree를 생성하게 됩니다.