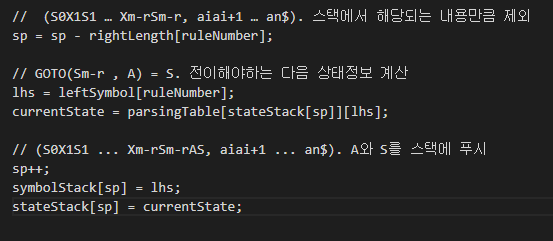
**우파스와 실험결과 분석**

먼저 pgs를 실행시켜 주어진 문법(MiniC.gr)를 입력값으로 주어 MiniC.lst와 MiniC.tbl을 생성한다. MiniC.lst의 terminal에 대응되도록 Scan할때의 tsymbol 열거와 tokenName 배열을 수정해야한다. MiniC.lst에는 생성규칙번호와 심볼의 상태에 따른 레퍼런스가 들어있다. MiniC.tbl에는 문법을 기반으로 생성된 parsingTable이 들어가있다.

코드를 살펴보자. parser 파트에서 사용되는 stateStack은 LR Parser에서 사용되는 상태들을 저장하는 스택이고, symbolStack은 심볼들을 저장하는 스택이다. parsingTable의 가로(row)는 상태들의 개수, 세로(column)은 terminal+nonterminal+$이다. parsingTable의 원소는 entry라고 표현되며 이 entry가 양수면 shift action이 수행되고 음수면 reduce action이 수행된다

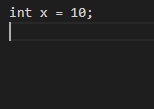
shift action에선 symbol과 state를 스택에 shift한다. 즉, symbolStack과 stateStack에 각각 symbol과 state를 저장하고 다음 token을 읽어온다.

reduce action에선 entry가 음수이다. 그럼 일단 reduce action을 한다는거다. 이 entry에 해당하는 lhs와 entry에 해당되는만큼의 내용을 스택에서 pop해야한다. 이 “entry에 해당되는 만큼의 내용의 길이”를 의미하는 것이 rightLength배열이다. 즉, sp = sp-rightLength(ruleNumber); 의 의미는 reduce action에서 (S0X1S1 … Xm-rSm-r, aiai+1 … an$)를 수행한 것이다.그리곤 현재 state를 계산하여 다시 stack에 넣기 위해 currentState와 이를 위한 lhs를 계산한다(GOTO(Sm-r , A) = S). chapter 8 8번쨰 슬라이드에 해당하는 내용이다. 정리하자면 다음과 같다



이때 ruleNumber는 reduce할 때 생성규칙을 가리키는 인덱스역할을 하게 되므로 양수가 되어야한다. 따라서 ruleNumber = -entry를 해야한다.

쉽게 이해하기 위하여 다음과 같은 내용만을 포함하는 test.mc파일을 실험에 사용했다. (기존의 test.mc파일을 손으로 하나하나 분석하기엔 길어져서 임의로 바꾸었다.)

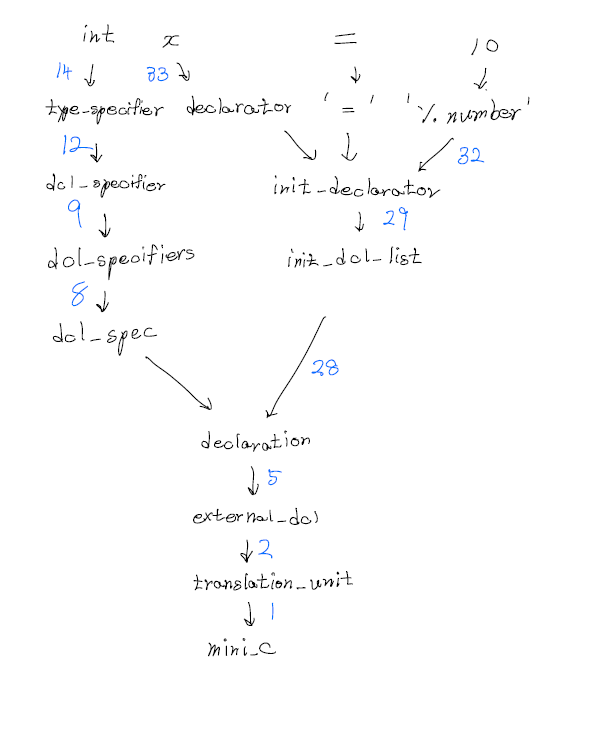
 test.mc

Scanner의 결과와 Parser의 결과는 다음과 같다.

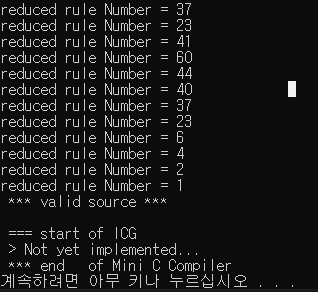


valid source라 출력되는 것을 보아 올바르게 파싱한 모양이다. start of Scanner는 토큰을 만든것이고, start of Parser는 우파스이다. 우파스이므로 출력된 ruleNumber는 MiniC.gr 문법의 생성규칙번호이다. 처음의 ruleNumber가 14라는건 처음의 entry가 -14라는 거고 reduce action이 일어났다는거다. 이때 rightLength[14]에 해당하는건 1이고 leftSymbol[14]에 해당하는건 99이다. 스택에서 1만큼 제외되고 99에 해당하는 lhs를 다시 계산해 스택에 푸시되는 것이다.

결과창을 참고하여 수기로 분석을 해보았다. 올바르게 분석됬음을 볼 수 있다.



이번엔 bubble.mc로 실험해보자



역시 토큰이 생성되고 우파스가 출력된다.

**소감 및 의견**

헤더파일과 parser.cpp파일, scanner.cpp파일을 분리하려했으나 LNK에러가 나서 그냥 parser.c와 main만 분리했다.