Programming Language hw2

B811056 노이진

April 2021

1 hw2_1

lex는 정의절, 규칙절, 사용자의 서브루틴절 3가지로 이루어져 있는데 정의절은 C 코드 선언 부이기 때문에 사용할 전처리기나 변수를 적고, 규칙절은 입력된 문자에서 매칭되는 문자열의 패턴이 나타났을 때 해당하는 동작을 실행하기 때문에 필요한 패턴을 설정한다. 그리고 서브루틴절에서는 사용자가 원하는 서브루틴을 정의할 수 있다.

따라서 우선 정의절에는 전처리문과, love의 개수를 셀 count라는 변수를 초기화해줬다. 그리고 규칙절에는 love를 l이 대문자일 경우와 소문자일 경우 모두 count 될 수 있도록 [Ll]을 이용해 패턴을 만들어줬다. grouping은 꼭 해줄 필요는 없지만 해주었다. 그리고 나머지는 |으로 한줄로 쓴 후 ;로 처리해 아무일도 하지 않도록 하였다.

서브루틴절의 main함수에서는 yylex를 이용해 txt를 스캔, 분석하고 love을 개수가 다 세어진 count를 출력하도록 하였다. 그리고 yywrap함수를 써서 txt파일을 다 읽을 경우 종료되도록 하였다.

2 hw2_2

정의절에는 1번과 마찬가지로 전처리문 #include⟨stdio.h⟩를 넣어줬다. 하지만 이번에는 규칙과 맞을 경우 바로 출력을 할 것이므로 변수는 만들지 않았다.

규칙절에는 python과제 때처럼 정규표현식을 만들었다. 대신 앞 뒤에 ∧와 $를 추가해 줄의 맨 시작부터 끝까지 정규표현식과 매치되는지 확인하도록 했다. 규칙과 맞을 경우 yytext에 그 문자열이 저장되므로 printf함수를 사용해 yytext를 출력하도록 하였다. 그리고 . |\n을 ;로 처리해 나머지는 출력되지 않도록 했다. 마지막으로 서브루틴절의 main함수에서는 yylex함수를 호출하고, yywrap함수에서는 yylex함수가 txt를 읽다가 EOP일 경우 종료될 수 있게 했다.

3 hw2_3

정의절에서는 전처리문를 먼저 넣어주고 그 다음부터는 preprocessor, octal number, negative decimal number, positive decimal number, operator, comment, '=', '{', '}', wordcase1, wordcase2, word, mark를 조건에 맞게 count할 변수를 만들어 모두 0으로 초기화해주었다.

규칙절에서는 가장 먼저 주석의 패턴부터 설정했다. 규칙절에서는 먼저 작성되는 규칙이 먼저 선택되기 때문에 주석 내의 문자들이 다른 패턴과 먼저 일치되어 count되는 경우를 막기 위해서다. 주석의 종류는 2가지이므로 두가지를 다른 정규표현식으로 나눠서 설정했다. "\\"는 주석의 내용까지 함께 포함해 하나의 주석으로 카운트 할 것이기 때문에 .*을 붙여 주석에 어떤 문자가 나오든지 0번 이상나올 수 있고 개행되면 주석이 끝나도록 설정했다. "* *\"는 주석이 여러 줄일 경우 사용하는 주석이므로 *뒤에 [\n]넣어 개행이 들어갈 수 있도록 하고, 어떤 문자든지 0번 이상 들어갈 수 있도록 .*을 그 앞뒤로 넣어주었다. 그리고 전체를 그룹화해서 그 전체의 그룹이 0번 이상 반복할 수 있도록 해 주었다. 그리고 마지막 부분에 *\을 넣어주어서 주석을 완성해줬다. 그리고 그 규칙이 맞을 경우 주석을 세는 변수를 +1하여 count해주었다.

그 다음에는 전처리문 두가지를 주석과 마찬가지로 .*을 사용하고 개행처리를 해서 전처리문 한 줄이 패턴과 맞을 경우 count하도록 하였다.

다음은 8진수의 개수를 count했는데 8진수는 0으로 시작하고 숫자는 7까지 사용되므로, 0뒤에 0부터 7까지의 수가 0번 이상 사용될 수 있도록 0[0-7]*을 사용했다. 패턴과 맞으면 +1을 하도록 해서 count했다.

음수 10진수를 count할 때는 10진수는 0부터 9까지의 숫자가 1번 이상 이용되므로 8진수와 마찬가지로 [0-9]를 사용하고 대신 1번 이상이므로 *대신 +를 사용하였다. 대신 음수이므로 앞쪽에 -를 붙여주었다.

양수 10진수는 음수와 마찬가지의 조건에서 -만 빼면 되므로 [0-9]+로 표현하였다. 그리고 패턴이 맞으면 양수 10진수를 세는 변수에 +1을 해줘서 count했다.

그리고 연산자들을 모두 |을 이용해 count시켜줬다. 차례대로 산술연산자, 논리연산자, 관계연산자. 증감연산자, 콤마연산자, 참조연산자, 포인터연산자를 count해주었고, 대신 그 모양이 같을 경우에는 중복적으로 표현하진 않았다.(ex.*)

'=','{','}'는 모두 단순한 기호이기 때문에 쌍따옴표를 이용하여 그 기호가 나올 경우에 count해주었다.

wordcase1은 p가 딱 2번만 들어간 단어의 개수이기 때문에 p가 3개가 나올 경우를 배제하는 것이 가장 중요하다고 생각했다. 그래서 "pp"의 앞이나 뒤에 올수 있는 알파벳 범위를 p를 빼고 앞 뒤로 나누었다. []을 이용해서 해주었는데 [a-oA-O][q-zQ-Z]이렇게 p의 바로 앞 알파벳인 o와 바로 뒤 알파벳인 q를 범위의 기준으로 하여 범위를 설정해 주었다. 또 이렇게 나눈 알파벳을 |를 사이에 넣어 두 범위 중 하나 알파벳이 선택되도록 하였다. 그리고 이것을 그룹화한 후 *을 이용해 0번 이상 반복되도록 설정해 pp의 앞 뒤에 적어주었다. 마지막으로 그러한 조건에 맞을 경우 count했다.

wordcase2는 e로 시작하고 m으로 끝나는 단어이기 때문에 그냥 e와 m을 정규 표현식의 앞과 끝에 배치했다. 대신 다 글자 사이에 어떤 알파벳이라도 0개 이상 들어갈 수 있도록 []과 *을 이용했다. 그리고 조건이 match되면 count되도록 하였다.

word는 위의 조건을 모두 만족하지 않는 나머지 단어를 뜻하고 규칙은 위쪽부터 행해지므로 다른 단어들의 아래쪽에 배치했고 이때 []와 +를 이용해 알파벳이 한개라도 있을 경우 word로 생각해 count하도록 하였다.

마지막으로 모든 조건에 만족하지 않는 나머지들을 .|\n을 사용해 count했는데, 모든 조건에 만족하지 않아야하므로 가장 아래쪽에 배치했다.

사용자의 루틴절에서는 최종으로 main함수에서 yylex()를 먼저 사용해 규칙들을 체크하고, 모든 count된 변수들을 print하도록 했다. 그리고 1번,2번과 마찬가지로 yywrap()을 이용해 종료될 수 있도록 했다.