



# 컴파일러 예측파서구현 과제 #1

---

제출일 2022.03.27

---

학과 소프트웨어학과

학번 2020039009

이름 차현아

담당교수 이재성 교수님

---



## 코드 - 함수원형

```

1  #include<stdio.h>
2
3  char lookahead;
4
5  void match(char token);
6  char nexttoken();
7  void expr();
8  void term();
9  void expr_rest();
10 void term_rest();
11 void factor();

```

line 3 : 예측기호를 의미하는 변수를 전역변수로 선언

line 5 : 현재 예측기호와 함수의 인자로 들어온 토큰이 일치하면 다음 예측기호를 읽는 함수의 원형

line 6 : getchar()을 통해 입력 문자열을 입력하고, 차례대로 한 문자씩 읽어주는 함수 원형

line 7~line 11 : 예측파서 구현을 위해 필요한 함수의 원형들

\* 오른쪽 순환으로 변환한 생성규칙

## 2020039009 - 차현아

$expr \rightarrow term \{ print('3') \} expr\_rest$

$expr\_rest \rightarrow + term \{ print('1') \} expr\_rest$   
 $\quad \quad \quad | - term \{ print('2') \} expr\_rest$   
 $\quad \quad \quad | \epsilon$

$term \rightarrow factor \{ print('6') \} term\_rest$

$term\_rest \rightarrow * factor \{ print('4') \} term\_rest$   
 $\quad \quad \quad | / factor \{ print('5') \} term\_rest$   
 $\quad \quad \quad | \epsilon$

$factor \rightarrow (expr) \{ print('7') \}$   
 $\quad \quad \quad | 숫자(0 \sim 9)$

## 코드 - main

```

13 void main()
14 {
15     printf("입력 :");
16
17     lookahead = nexttoken();
18
19     expr();
20     if (lookahead == '$')
21         printf("Wn");
22     else
23         printf(" errorWn");
24 }

```

line 17 : nexttoken()내의 getchar()를 통해 문자열을 입력받아 한 글자씩 읽어들이

line 19 : expr() 함수 호출

line 20~23 : expr() 함수 호출이 끝나고 돌아왔을 때, 예측기호가 문장의 끝임을 의미하는 '\$'이면 개행문자를 출력하면서 정상 종료 / 그렇지 않으면 error 출력

## 코드 - match 함수

```

26 void match(char token)
27 {
28     if (lookahead == token)
29         lookahead = nexttoken();
30     else
31     {
32         printf("errorWn");
33         exit(1);
34     }
35 }

```

line 28 : 현재 예측기호와 함수의 인자로 들어온 token이 일치하면, 다음 예측기호 읽기

line 30~35 : 예측기호와 토큰이 일치하지 않으면, 에러를 출력하고 프로그램 종료

## 코드 - nexttoken 함수

```

37 char nexttoken()
38 {
39     char c;
40     while (1)
41     {
42         c = getchar();
43         if (c == ' ' || c == '\t' || c == '\n' || c == '\0')
44             continue;
45         return c;
46     }
47 }

```

line 40~45 : 입력받은 문자열을 차례대로 한 문자씩 읽어오는 코드

: line 43의 if문의 조건에 걸리면, 문자를 리턴하지 않고 다음 문자를 읽는다. 조건에 걸리지 않으면 읽은 문자를 리턴한다.

## 코드 - expr 함수

```

48 void expr()
49 {
50     term(); printf("3 "); expr_rest();
51 }

```

line 48 : main에서 첫 번째로 실행되는 함수

line 50 : 가장 먼저 term()을 호출하고, 끝나고 돌아오면 3을 출력한 후 expr\_rest()를 호출

## 코드 - term 함수

```

52 void term()
53 {
54     factor();
55     printf("6 ");
56     term_rest();
57 }

```

line 53~56 : factor()를 호출하고, 끝나고 돌아오면 6을 출력한 후 term\_rest()를 호출한다.

## 코드 - factor 함수

```
58 void factor()
59 {
60     if (lookahead == '(')
61     {
62         match('(');
63         expr();
64         match(')');
65         printf("7 ");
66     }
67     else if (lookahead >= 48 && lookahead <= 57)
68     {
69         match(lookahead);
70     }
71     else
72     {
73         printf("error");
74         exit(1);
75     }
76 }
```

line 60~66 : 현재 예측기호가 '('라면, 위에서 정의한 문법대로 match('(')를 호출하고 다음 예측기호를 읽어들인다. 후에 expr()함수를 호출하고, 끝나고 돌아와 match(')')를 호출한다. match함수 내에서 예측기호와 토큰이 일치한다면 다음 예측기호를 읽어온 후, 7을 출력한다.

line 67~70 : 현재 예측기호가 0~9중의 숫자라면, (0과 9의 아스키코드 값 : 48,57) match함수를 호출하여 다음 예측기호를 읽어들인다.

line 71~75 : 현재 예측기호가 숫자도 아니고, '('도 아니라면 에러를 출력하고 프로그램을 종료한다. 그 이유는 위에서 정의한 생성규칙에서 factor의 first는 '(' 또는 숫자이기 때문이다.

## 코드 - expr\_rest 함수

```

77 void expr_rest()
78 {
79     if (lookahead == '+')
80     {
81         match('+');
82         term();
83         printf("1 ");
84         expr_rest();
85     }
86     else if (lookahead == '-')
87     {
88         match('-');
89         term();
90         printf("2 ");
91         expr_rest();
92     }
93 }

```

## 코드 - term\_rest 함수

```

94 void term_rest()
95 {
96     if (lookahead == '*')
97     {
98         match('*');
99         factor();
100         printf("4 ");
101         term_rest();
102     }
103     else if (lookahead == '/')
104     {
105         match('/');
106         factor();
107         printf("5 ");
108         term_rest();
109     }
110 }

```

line 79~85 : 현재 예측기호가 '+'라면, match('+')를 호출하여 다음 예측기호를 읽어들이고 term()을 호출하고 끝나고 돌아오면 1을 출력한다. 후에 expr\_rest()를 호출한다.

line 96~102 : 현재 예측기호가 '\*'라면, match('\*')를 호출하여 다음 예측기호를 읽어들이고 factor()을 호출하고 끝나고 돌아오면 4를 출력한다. 후에 term\_rest()를 호출한다.

**실행결과 1 : 입력 -  $1+2*3$** Microsoft Visual Studio 디버그 콘솔입력 :  $1+2*3$ 

6 3 6 4 1

C:\Users\User\source\repos\Project7\Debug  
이 창을 닫으려면 아무 키나 누르세요...**실행결과 2 : 입력 -  $4*(9-2)$** Microsoft Visual Studio 디버그 콘솔입력 :  $4*(9-2)$ 

6 6 3 6 2 7 4 3

C:\Users\User\source\repos\Project7\Debug  
이 창을 닫으려면 아무 키나 누르세요...**실행결과 3 : 입력 -  $4/2*5$** Microsoft Visual Studio 디버그 콘솔입력 :  $4/2*5$ 

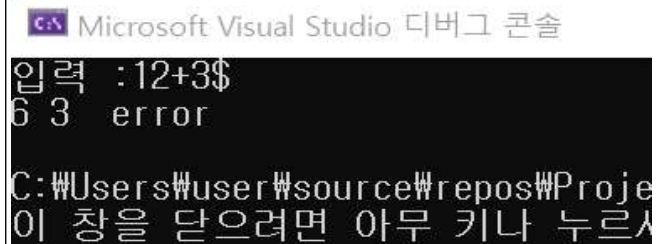
6 5 4 3

C:\Users\User\source\repos\Project7\Debug  
이 창을 닫으려면 아무 키나 누르세요...**실행결과 4 : 입력 -  $5*-2$** Microsoft Visual Studio 디버그 콘솔입력 :  $5*-2$ 

6 error

C:\Users\User\source\repos\Project7\Debug  
이 창을 닫으려면 아무 키나 누르세요...

## 실행결과 5 : 입력 - 12+3\$



```

Microsoft Visual Studio 디버그 콘솔
입력 :12+3$
6 3 error
C:\Users\User\source\repos\Project\
이 창을 닫으려면 아무 키나 누르세요

```

: expr() -> term() 3 expr\_rest()

term() -> factor() 6 term\_rest()

factor() -> 현재 예측기호(1)가 숫자이므로 match함수를 이용하여 다음 예측기호를 읽는다

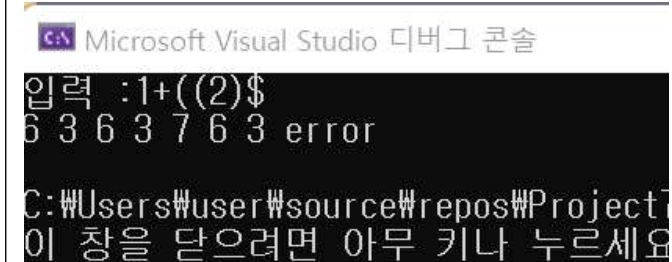
-> factor() 끝나고 **6출력**, 현재 예측기호는 2

-> term\_rest()호출-> 조건문과 일치하는 예측기호가 없으므로 pass

-> **3 출력** -> expr\_rest()호출-> 조건문과 일치하는 예측기호가 없으므로 pass

-> expr()가 끝나고 main으로 돌아왔지만, 현재 예측기호가 '\$'이 아니므로 **에러 출력**

## 실행결과 6 : 입력 - 1+((2)\$



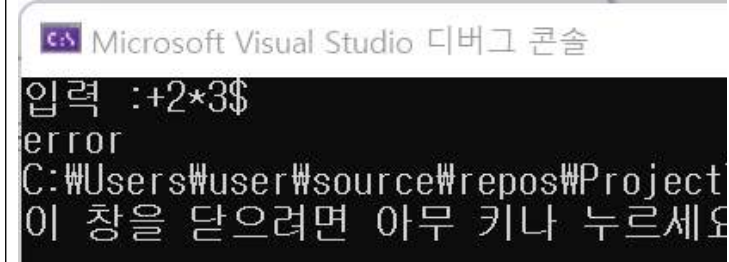
```

Microsoft Visual Studio 디버그 콘솔
입력 :1+((2)$
6 3 6 3 7 6 3 error
C:\Users\User\source\repos\Project\
이 창을 닫으려면 아무 키나 누르세요

```

: '('가 두 번연속 나오는 경우 에러출력

## 실행결과 7 : 입력 - +2\*3\$



```

Microsoft Visual Studio 디버그 콘솔
입력 :+2*3$
error
C:\Users\User\source\repos\Project\
이 창을 닫으려면 아무 키나 누르세요

```

: 연산기호로 시작하는 경우 에러 출력



## 실행결과 8 : 입력 - 2-3\*1+\$

```

Microsoft Visual Studio 디버그 콘솔
입력 :2-3*1+$
6 3 6 4 2 error
C:\Users\user\source\repos\Project7\De
이 창을 닫으려면 아무 키나 누르세요..

```

: 연산기호로 끝나는 경우 에러 출력

: expr-> term 3 expr\_rest

term-> factor 6 term\_rest

factor-> 숫자 -> 다음 예측기호 읽기, 다음 예측기호는 '-' -> 6 출력

term\_rest -> 일치하는 예측기호가 없으므로 pass -> 3 출력

expr\_rest -> - term 2 expr\_rest

: 예측기호 일치-> 다음 예측기호 읽기, 다음 예측기호는 3

term -> factor 6 term\_rest

factor-> 숫자 -> 다음 예측기호 읽기, 다음 예측기호는 '\*' -> 6 출력

term\_rest -> \* factor 4 term\_rest

: 예측기호 일치 -> 다음 예측기호 읽기, 다음 예측기호는 1

factor -> 숫자 -> 다음 예측기호 읽기, 다음 예측기호는 '+' -> 4 출력

term\_rest -> 일치하는 예측기호가 없으므로 pass -> 2출력

expr\_rest -> + term 1 expr\_rest

: 예측기호 일치, 다음 예측기호 읽기, 다음 예측기호는 '\$'

term -> factor 6 term\_rest

factor -> 일치하는 예측기호가 없으므로 에러를 출력하고 종료