과제 #1. Recursive-descent parser

소프트웨어학부 20180325 임성현

1. 과제 내용

- Syntax Graph에 기반하여, 간단한 수식을 검사하는Recursive-descent parser 프로그램을 작성한다.

2. 문제 해결 방법

- 교수님 1강 강의교안 12 슬라이드를 참고하였다.

- enum형 NULL을 NONE으로 변경하였다.

- get\_token() 함수를 구현하였다.

- 입력받은 공백 문자를 무시한다.

- ‘+’, ‘\*’, ‘(’, ‘)’ 등의 수식을 검사한다.

- NUMBER 값을 입력받을 때, int number\_value에 ASCII 코드 값을 char에서 int로 변환하였다.

- expression(), term(), factor() 함수를 순회할 때 마다, return 되어 나온 결과값을 int result에 저장한다.

- term() 함수에서 123+, 123\* 등의 연산자 뒤에 피연산자가 없는 경우를 에러처리 하였다.

- factor() 함수에서 여러자릿 수가 입력되었을 때를 구현하였다.

- 출력 결과에 대한 아이디어는 [여기](https://c11.kr/1boru)를 참고하였다.

3. 실행 결과

- 테스트 케이스를 통과하는 경우

텍스트, 화면, 스크린샷이(가) 표시된 사진

자동 생성된 설명

- 테스트 케이스를 통과하지 못하는 경우



4. 결론

- 과제 명세에 맞게, *여러자릿수의 정수, 소괄호, +,\* 등을 포함하는 여러가지의 수식이 잘 처리되는지 실험* 하였다.

- 해당 과제를 통해 Recursive-descent parser 작동 원리를 알게 되었다.

5. 문제점

- 3(100+20) 같은 수식은 계산할 수 없다.

- float형 수식은 계산할 수 없다.

- int형 범위를 넘어가는 범위의 수식은 계산할 수 없다. 실험 결과 overflow, underflow가 발생했다.

6. 원시 프로그램 (소스코드)

#include <ctype.h> //isdigit()

#include <stdio.h>

#include <stdlib.h> //error 처리 시, exit()

int term();

int factor();

void error();

// NULL은 예악어이므로 NONE으로 대체함

enum { NONE, PLUS, STAR, NUMBER, LPAREN, RPAREN, END } token;

int number\_value = 0; // 숫자 값

void get\_token() {

int ch = getchar();

// 빈칸 입력 처리

while (ch == ' ')

ch = getchar();

switch (ch) {

case '+':

printf("+");

token = PLUS;

break;

case '\*':

printf("\*");

token = STAR;

break;

case '(':

printf("(");

token = LPAREN;

break;

case ')':

printf(")");

token = RPAREN;

break;

case '\n':

case EOF:

token = END;

break;

default:

// if (ch <= '9' && ch >= '0')을 isdigit()로 구현

if (isdigit(ch)) {

printf("%c", ch);

token = NUMBER;

number\_value = ch - '0'; // char 타입을 int로 변환

} else {

token = NONE;

}

break;

}

}

int expression() {

int result = term();

while (token == PLUS) {

get\_token();

result += term();

}

return result;

}

int term() {

int result = factor();

while (token == STAR) {

get\_token();

// 연산자 입력 후, 별도 숫자가 입력되지 않은 경우

// i.e. 123+ , 123\* 등 의 경우

if (token == NONE) {

error();

} else

result \*= factor();

}

return result;

}

int factor() {

int result;

if (token == NUMBER) {

result = number\_value;

get\_token();

// 일의 자리, 십의 자리, 백의 자리, ... 등의 자리수를 계산

// i.e. 234를 입력받은 경우.

// i) main()에서 2 입력 받음, token = NUMBER;

// ii) factor()에서 result = 2; token = NUMBER;

// iii) while()에서 result = 2 \* 10; result += 3;

// iv) 결과적으로 result = 23;

// v) 위 과정을 반복

while (token == NUMBER) {

result \*= 10;

result += number\_value;

get\_token();

}

} else if (token == LPAREN) {

get\_token();

result = expression();

if (token == RPAREN)

get\_token();

else

error();

} else

error();

return result;

}

int main() {

printf("입력하는 수식 : ");

get\_token();

int result = expression();

if (token != END)

error();

else

printf(" = %d\n", result); // 계산 결과 출력

}

void error() {

printf("\n에러가 발생하였습니다.");

exit(1);

}