과제 #2. Recursive-descent parser 2

소프트웨어학부 20180325 임성현

1. 과제 내용

- 과제 #1에 기반하여, 정수 연산 기능 뿐 아니라 실수 연산이 되게 구현했다.

- 또한, 사칙연산(‘+’, ‘-’, ‘\*’, ‘/’)이 되도록 구현했다.

2. 문제 해결 방법

- 세부 명세는 강의교안 1-19 슬라이드를 참고하였다.

- typedef struct {…} number로 정수와 실수를 다뤘다. 이후, term(),factor(), expression() 의 반환값을 int 에서 number 로 수정하였다. 또한, term(),factor(), expression() 에서 INT, FLT 에 따라 연산되도록 수정하였다.

- enum {…} token; 에서 MINUS, DIV 을 추가하여 사칙연산을 구현하였다.

- 정수와 실수가 입력되는 경우 다음과 같이 처리하였다.

printf("Warning: 정수와 실수 연산이 입력되었습니다.\n");

3. 실행 결과

텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명- 테스트 케이스를 통과하는 경우

텍스트이(가) 표시된 사진

자동 생성된 설명

- 테스트 케이스를 통과하지 못하는 경우

텍스트이(가) 표시된 사진

자동 생성된 설명

4. 결론

- 과제 명세에 맞게, *사칙연산과 정수 및 실수 연산이 잘 처리 되는지 실험* 하였다.

5. 문제점

- 입력 값으로 1/10/100이 들어오는 경우, 정수만 입력받았으나 계산 도중 실수로 처리되기 때문에 정수와 실수가 입력된 것 처럼 warning이 뜬다.

6. 원시 프로그램 (소스코드)

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

#include <stdio.h>

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

// 정수와 실수 처리

typedef enum { INT, FLT } kind;

typedef struct {

kind t;

union {

int i;

float f;

} val;

} number;

number term();

number factor();

void error();

// NULL은 예악어이므로 NONE으로 대체함

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

number number\_value; // 숫자 값

// 정수와 실수를 받았을 때 1씩 증가한다. 해당 카운터를

// 활용해 warning 메시지를 출력할 지 결정한다.

// 그렇지 않고 exp(), term(), factor() 별로 warning을 출력하면 여러번 출력된다.

// i.e.

// 입력: 10+1.1+100

// 출력결과

// Warning: 정수와 실수 연산이 입력되었습니다.

// Warning: 정수와 실수 연산이 입력되었습니다.

int diff\_type\_cnt = 0;

void get\_token() {

static int ch = ' ';

ch = getchar();

// 빈칸 입력 처리

while (ch == ' ')

ch = getchar();

switch (ch) {

case '+':

// printf("+");

token = PLUS;

break;

case '-':

// printf("-");`

token = MINUS;

break;

case '\*':

// printf("\*");

token = STAR;

break;

case '/':

// printf("/");

token = DIV;

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)) {

// 여러 자리 수를 처리하는 코드

// 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

int num = ch - '0';

// printf("%c", ch);

// 소수점이 있는 경우 1로 변경됨

int is\_float = 0;

// 실수 처리도 가능하게 함

while (isdigit(ch = getchar()) || ch == '.') {

if (ch == '.') {

is\_float = 1;

} else {

num = num \* 10 + (ch - '0');

if (is\_float) {

is\_float \*= 10;

}

}

}

// getchar()로 읽어온 문자를 다시 stdin에 넣어줌

// i.e. 111+222 입력 시 111+ 까지 입력되는 걸 한번 무르게 해 111 만

// 입력되도록 함.

ungetc(ch, stdin);

token = NUMBER;

// 정수와 실수, 각 경우에 따라 type과 값을 입력받음

if (is\_float) {

number\_value.t = FLT;

number\_value.val.f = num / (float)is\_float;

} else {

number\_value.t = INT;

number\_value.val.i = num;

}

} else {

token = NONE;

}

break;

}

}

number expression() {

number result = term();

// while(PLUS){} while(MINUS)시 -연산 후 +연산이 되지 않는다.

while ((token == PLUS) || (token == MINUS)) {

if (token == PLUS) {

get\_token();

number next = term();

// 정수와 정수 계산

if (result.t == INT && next.t == INT) {

result.val.i += next.val.i;

}

// 실수와 실수 계산

else if (result.t == FLT && next.t == FLT) {

result.val.f += next.val.f;

}

// 정수와 실수 또는 실수와 정수 계산

else {

diff\_type\_cnt++;

if (result.t == INT && next.t == FLT) {

result.val.f = result.val.i + next.val.f;

// 기존 result 값이 int 였으나 계산 후 float 으로 형 변환

result.t = FLT;

} else if (result.t == FLT && next.t == INT) {

result.val.f += next.val.i;

}

}

} else if (token == MINUS) {

get\_token();

number next = term();

// 정수와 정수 계산

if (result.t == INT && next.t == INT) {

result.val.i -= next.val.i;

}

// 실수와 실수 계산

else if (result.t == FLT && next.t == FLT) {

result.val.f -= next.val.f;

}

// 정수와 실수 또는 실수와 정수 계산

else {

diff\_type\_cnt++;

if (result.t == INT && next.t == FLT) {

result.val.f = result.val.i - next.val.f;

// 기존 result 값이 int 였으나 계산 후 float 으로 형 변환

result.t = FLT;

} else if (result.t == FLT && next.t == INT) {

result.val.f -= next.val.i;

}

}

}

}

return result;

}

number term() {

number result = factor();

// 곱셈 연산

while ((token == STAR) || (token == DIV)) {

if (token == STAR) {

get\_token();

number next = factor();

// 정수와 정수 계산

if (result.t == INT && next.t == INT) {

result.val.i \*= next.val.i;

// 실수와 실수 계산

} else if (result.t == FLT && next.t == FLT) {

result.val.f \*= next.val.f;

}

// 정수와 실수 또는 실수와 정수 계산

else {

diff\_type\_cnt++;

if (result.t == INT && next.t == FLT) {

result.val.f = result.val.i \* next.val.f;

// 기존 result 값이 int 였으나 계산 후 float 으로 형 변환

result.t = FLT;

} else if (result.t == FLT && next.t == INT) {

result.val.f \*= next.val.i;

}

}

}

// 나눗셈 연산

if (token == DIV) {

get\_token();

number next = factor();

// 정수와 정수 계산

if (result.t == INT && next.t == INT) {

// 계산 결과가 실수가 나오는 경우를 처리

// i.e. 10/10 = 1 은 1.000000이 나오면 안 됨

// i.e. 10/7 = 1.428571 은 1이 나오면 안 됨

// 계산 결과가 정수

if ((result.val.i / (float)next.val.i) == (result.val.i / next.val.i)) {

result.val.i /= next.val.i;

} else { //계산 결과가 실수

result.val.f = result.val.i / (float)next.val.i;

result.t = FLT;

}

}

// 실수와 실수 계산

else if (result.t == FLT && next.t == FLT) {

result.val.f /= next.val.f;

}

// 정수와 실수 또는 실수와 정수 계산

else {

diff\_type\_cnt++;

if (result.t == INT && next.t == FLT) {

result.val.f = result.val.i / next.val.f;

// 기존 result 값이 int 였으나 계산 후 float 으로 형 변환

result.t = FLT;

} else if (result.t == FLT && next.t == INT) {

result.val.f /= next.val.i;

}

}

}

}

return result;

}

number factor() {

number result;

// 재귀가 끝나면 factor() -> term() -> expression() 순으로 실행된다.

// INT, FLT 타입에 따라 값을 연산해야 하므로 factor(), term(), expression()

// 함수 마다 number.t (type)에 따라, number.val.i or number.val.f 연산을

// 진행했다.

if (token == NUMBER) {

result.t = number\_value.t;

result.val = number\_value.val;

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();

number result = expression();

if (token != END)

error();

else {

if (diff\_type\_cnt > 0)

printf("Warning: 정수와 실수 연산이 입력되었습니다.\n");

if (result.t == INT)

printf(" = %d\n", result.val.i);

else

printf(" = %f\n", result.val.f);

}

}

void error() {

printf("\n에러가 발생하였습니다.");

exit(1);

}