2020년 1학기

프로그래밍언어 과제2 : Toy Language [Version1]

컴퓨터학부

20162481 안승훈

1. 개요

- 프로그래밍을 이용해 Toy Language의 interpreter를 작성한다.

- Toy Language의 Interpreter 의 구현을 통해 Parser 및 interpretive 언어에 대한 이해를 높인다.

- Stack를 이용하여 interpreter의 역할을 수행한다.

2. 구현 방법

- Visual Studio 프로그램과 C++언어를 이용하여 설계하였다.

- 파일 로드와 스택을 이용하여 인터프리팅을 할 수 있도록 구현하였다.

- 사용자로부터 파일의 이름을 입력 받아 파일을 열어볼 수 있도록 설계하였다.

- 스택을 사용하여 결과 값을 계산할 수 있도록 하였다.

3. 구현한 자료구조 및 함수 설명

- C++을 이용하여 프로그램을 작성하였다

* 저장을 위한 자료구조

stack <char> opst;

- 연산자를 저장하기 위한 스택 자료구조이다.

- c++ STL로 제공되는 Stack 라이브러리를 이용하였다.

stack <int> inst;

- 상수를 저장하기 위한 스택 자료구조이다.

- C++ STL로 제공되는 stack 라이브러리를 이용하였다.

* Main 함수를 비롯한 다른 함수

**< int main(void) >**

**int** main(**void**) {

**int** choice;

**while** (1) {

choice = menu();

**switch**(choice) {

**case** 1 :

file\_load();

**break**;

**case** 2 :

get\_input();

**break**;

**case** 3:

printf("프로그램을 종료합니다.\n");

exit(0);

**break**;

}

}

**return** 0;

}

- main함수에서는 menu () 함수를 통해 사용자가 입력하는 번호에 따라, 파일 로드, 인터액티브 모드, 프로그램 종료의 기능을 분기할 수 있도록 한다.

- switch-case 문을 이용하여 각각의 선택에 따라 분기하도록 하였다.

**< int menu(void), void print\_menu(void) >**

**int** menu(**void**) {

**char** input\_c = 0;

**int** input\_i = 0;

print\_menu();

**while** (1) {

scanf("%c", &input\_c);

getchar();

**if** (input\_c == '1' || input\_c == '2' || input\_c == '3') {

input\_i = input\_c - '0';

**return** input\_i;

}

**else** {

printf("다시 입력하세요\n");

printf("메뉴를 선택하세요 >> ");

}

}

}

**void** print\_menu(**void**) {

printf("=========================\n");

printf("1. File Load\n");

printf("2. Interactive Mode\n");

printf("3. Exit\n");

printf("=========================\n");

printf("Select Menu >> ");

}

- menu 함수는 사용자로부터 메뉴의 선택을 입력받는 함수이다.

- print\_menu() 함수를 통해 사용자의 선택지를 출력하여 준 후, 사용자로부터 번호를 입력받는다.

- 사용자로부터 입력받은 것이 올바른 선택지가 아닐 경우, 사용자로부터 올바른 입력이 이루어질 때까지 재입력을 받을 수 있도록 설계하였다.

- 입력이 제대로 된 경우, 사용자의 선택을 리턴한다.

**< void file\_load(void) >**

**void** file\_load(**void**) {

**char** fname[BUF\_SIZE];

**char** buf[BUF\_SIZE + 1];

**int** res;

**FILE** \*fp = NULL;

memset(fname, 0, BUF\_SIZE);

memset(buf, 0, BUF\_SIZE);

**while** (1) {

printf("파일명을 입력하세요 >> ");

scanf("%s", fname);

getchar();

**if** ((fp = fopen(fname, "r")) == NULL) {

fprintf(stderr, "파일 오픈에 실패했습니다\n");

return;

}

**else** **break**;

}

printf("파일 내용은 \n");

printf("------------------------------------\n");

**while**(fgets(buf, BUF\_SIZE, fp) != NULL) {

printf("%s\n", buf);

}

printf("------------------------------------\n");

printf("입니다.\n");

rewind(fp);

**while** (fgets(buf, BUF\_SIZE, fp) != NULL) {

res = check(buf);

**if** (res > 0) {

res = calcul(buf);

print\_res(res);

}

}

**return**;

}

- file\_load 함수는 사용자로부터 파일의 이름을 입력받아, 파일을 오픈하고, 파일의 내용의 문장을 계산하여 결과값을 출력하는 함수이다.

- 파일 오픈에 실패할 경우, 에러 메시지를 출력한 후 초기화면으로 복귀한다.

- 파일 오픈에 성공한 경우, 파일의 내용을 한 줄씩 읽어와 사용자에게 출력한 후, 각각의 문장 별로 check() 함수, calcul 함수를 통해 결과 값을 계산한 후, print\_res() 함수를 통해 사용자에게 결과를 출력한다.

- check() 함수의 return 값을 이용하여 문장이 올바른지를 확인한다.

- 문장이 올바른 경우에 calcul() 함수를 통해 결과 값을 도출한 후, print\_res() 함수를 이용하 결과 값을 출력한다.

**< int check( char buf[ ] ) >**

**int** check(**char** buf[]) {

**int** lcnt = 0, rcnt = 0;

**int** len = strlen(buf);

*//괄호 갯수 체크*

**for** (**int** i = 0; i < len; i++) {

**if** (buf[i] == '(')

lcnt++;

**else** **if** (buf[i] == ')')

rcnt++;

}

**if** (lcnt > rcnt) {

printf("')' 갯수가 부족합니다.\n");

**return** -1;

}

**else** **if** (lcnt < rcnt) {

printf("'(' 갯수가 부족합니다.\n");

**return** -1;

}

*// -가 여러개 쓰인 경우 체크*

**for** (**int** i = 0; i < len; i++) {

**if** (buf[i] == '-') {

**if** (buf[i + 1] == '-') {

printf("'-'사용이 잘못되었습니다.\n");

**return** -1;

}

}

}

**return** 1;

}

- check() 함수는 사용자로부터 입력받거나, 또는 파일에서 읽어온 문장이 올바른 형식인지 확인하는 함수이다.

- 괄호 개수를 비교하여, 괄호가 부족할 경우 에러 메시지를 출력하고, -1을 리턴한다.

- ( – ) 기호가 2개 이상 사용된 경우, 사용 오류가 있음을 출력하고 -1을 리턴한다.

- 위의 오류가 없는 경우 1을 리턴한다.

**< int calcul( buf [ ] ) >**

**int** calcul(**char** buf[]) {

**int** cur, x = 0;

**int** len = strlen(buf);

**char** tmp[10];

memset(tmp, 0, 10);

**for** (cur = 0; cur < len; ) {

**if** (buf[cur] == '(') {

cur++;

**continue**;

}

**else** **if** (buf[cur] == 'M') {

cur += 5;

opst.push('M');

}

**else** **if** (buf[cur] == 'I') {

cur += 2;

opst.push('I');

}

**else** **if** (buf[cur] == '-') {

x = 0;

memset(tmp, 0, 10);

tmp[x++] = buf[cur++];

**while** ('0' <= buf[cur] && buf[cur] <= '9')

tmp[x++] = buf[cur++];

inst.push(atoi(tmp));

}

**else** **if** ('0' <= buf[cur] && buf[cur] <= '9') {

x = 0;

memset(tmp, 0, 10);

**while** ('0' <= buf[cur] && buf[cur] <= '9')

tmp[x++] = buf[cur++];

inst.push(atoi(tmp));

}

**else** **if** (buf[cur] == ' ' || buf[cur] == '\n') {

cur++;

**continue**;

}

**else** **if** (buf[cur] == ')') {

**char** op = opst.top();

opst.pop();

**int** b = inst.top();

inst.pop();

**int** a = inst.top();

inst.pop();

**if** (op == 'M')

inst.push(a - b);

**else** **if** (op == 'I') {

**if** (a <= 0)

inst.push(0);

**else**

inst.push(b);

}

cur++;

}

}

**int** res = inst.top();

inst.pop();

**return** res;

}

- calcul() 함수는 연산자 stack, 상수 stack을 이용하여 계산을 진행하는 함수이다.

- 여는 괄호 ( ‘(‘ ) 가 나올 경우 그 다음부터 진행한다.

- ‘MINUS’ 가 나올 경우, 연산자 stack에 ‘M’을 push하고, ‘IF’가 나올 경우, 연산자 stack에 ‘I’를 push한다.

- 상수가 나올 경우, 상수를 상수 stack에 push한다.

- 닫는 괄호가 나올 경우, 상수 stack 상위 2개의 상수와 연산자 stack의 top을 확인하여 알맞는 연산을 진행한다. 이후 연산 결과 값을 다시 상수 stack에 삽입한다.

- 문장의 끝에 도달한 경우, 상수 스택에 마지막으로 남아있는 수를 return한다. 이 수가 최종 결과값이 된다.

**< void print\_res(int res) >**

**void** print\_res(**int** res)

{

printf("결과 : %d\n", res);

**return**;

}

- print\_res() 함수는 결과값을 인자로 받아 결과 값을 출력하는 함수이다.

**< void get\_input(void) >**

**void** get\_input(**void**)

{

**char** input[BUF\_SIZE];

memset(input, 0, BUF\_SIZE);

printf("문장을 입력하세요\n");

printf("--------------------------------\n");

fgets(input, BUF\_SIZE, stdin);

printf("\n--------------------------------\n");

**int** res = check(input);

**if** (res > 0) {

res = calcul(input);

print\_res(res);

printf("-----------------------------------\n");

}

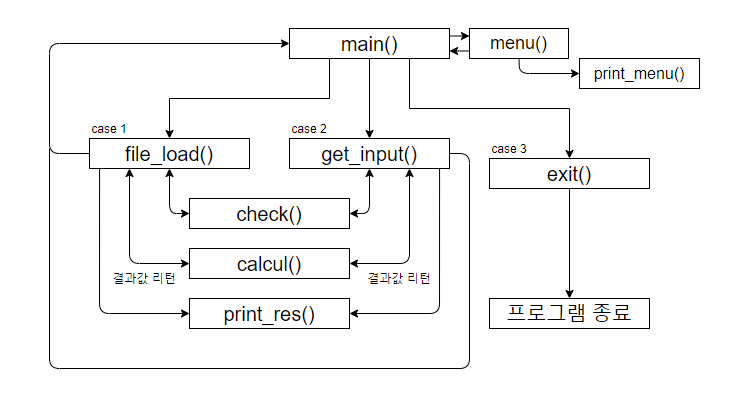
**return**;

}

- get\_input 함수는 사용자가 2번 선택지를 선택했을 때, 문장을 입력받아 결과 값을 계산하는 함수이다.

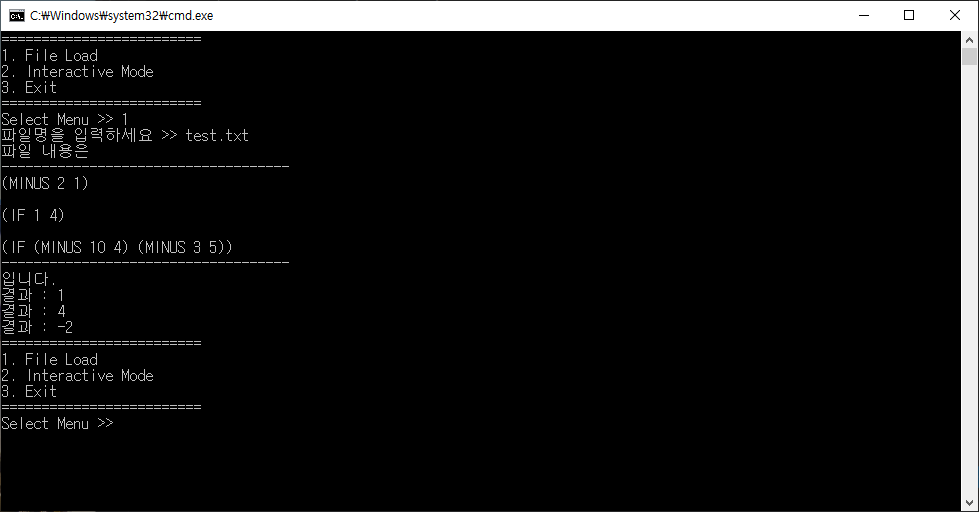
- 사용자로부터 문장을 입력받아, check 함수를 통해 올바른 문장인지를 판단한 후에, calcul 함수를 통해 결과를 계산하고 출력한다.

4. 함수 흐름도



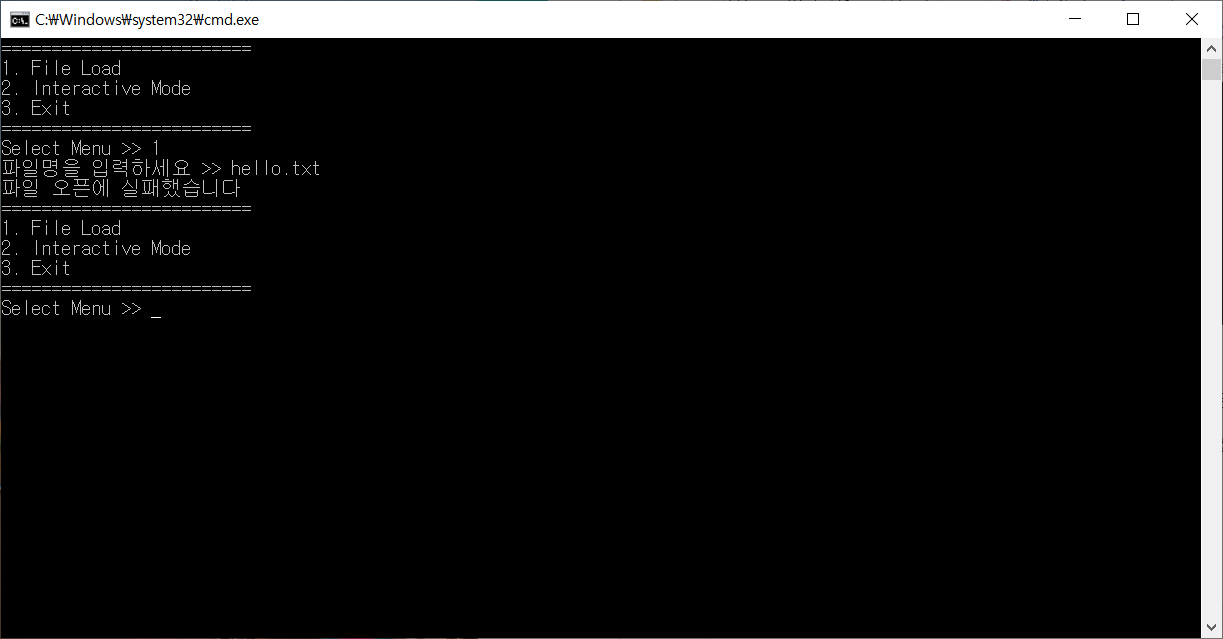
- 프로그램에서 사용한 함수들의 흐름도를 flow chart 형식으로 작성하였다.

5. 실행 결과

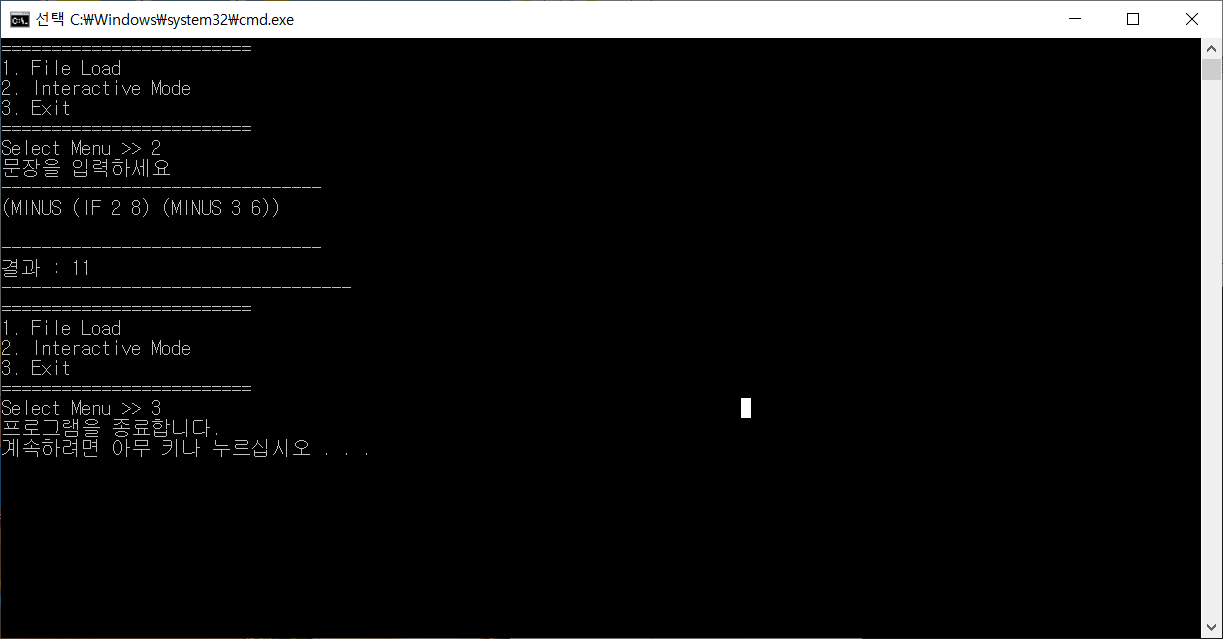


- 1번 기능은 파일 로드 기능의 실행 화면이다.  
 - test.txt 파일에 미리 작성해둔 파일의 내용을 불러와 출력하는 것을 볼 수 있다.

- 파일의 각 줄에 대해서 계산 결과를 출력하는 것을 알 수 있다.



- 없는 파일을 오픈하려는 경우, 오류 메시지를 출력한 후, 초기 화면으로 되돌아 오게 된다.

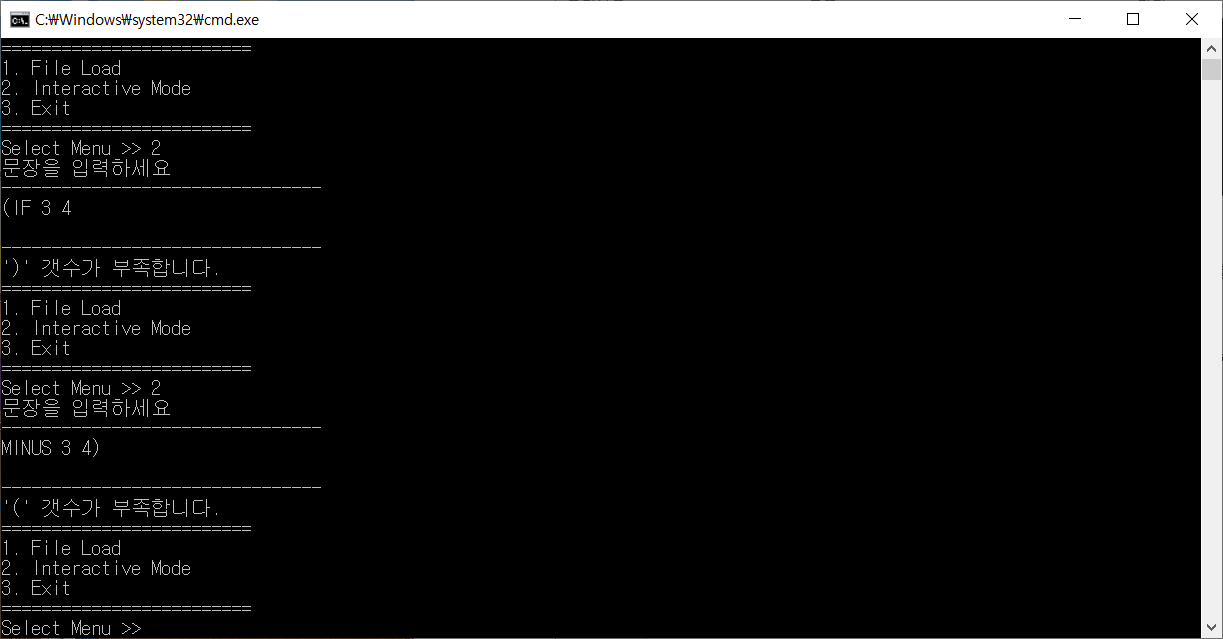


- 2번 기능을 선택한 실행 화면이다.

- 사용자로부터 직접 문장을 입력받아, 입력받은 문장의 결과 값을 계산한다.

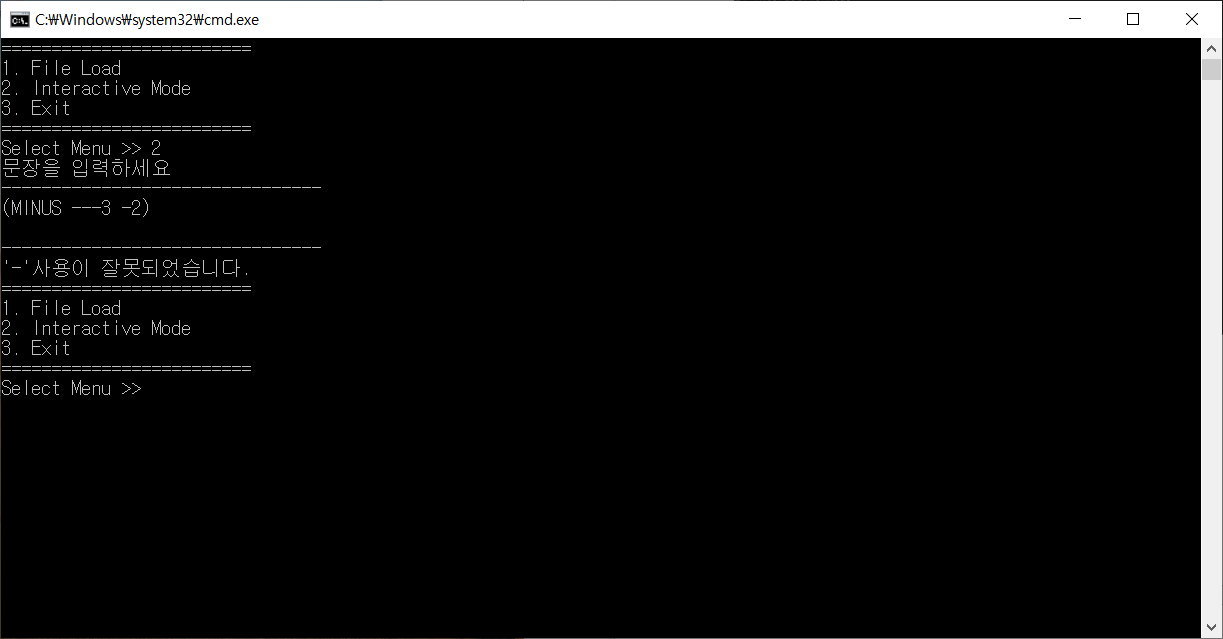
- IF와 MINUS가 여러 번 중첩되어도 원활하게 계산결과가 나오는 것을 볼 수 있다.

- 이어서 3번 기능, 프로그램 종료를 선택할 경우, “프로그램을 종료합니다” 문구를 출력한 후 프로그램을 종료한다.



- 사용자가 입력한 문장에 괄호가 부족한 경우 오류 메시지를 출력한다.

- 오류 출력 후 초기 화면으로 돌아오게 된다.



- ( - )기호의 사용이 잘못되었을 경우, 오류 메시지를 출력한다.

- 오류 메시지 출력 후, 초기 화면으로 돌아오도록 설계하였다.

6. 전체 소스코드

#include <cstdio>

#include <iostream>

#include <cstring>

#include <cstdlib>

#include <stack>

#include <algorithm>

#define BUF\_SIZE 1024

#define TOKEN\_CNT 100

#define TOKEN\_LEN 10

using namespace std;

void print\_menu(void);

int menu(void);

void file\_load(void);

int calcul(char buf[]);

int check(char buf[]);

void print\_res(int res);

void get\_input(void);

stack <char> opst;

stack <int> inst;

int main(void) {

int choice;

while (1) {

choice = menu();

switch(choice) {

case 1 :

file\_load();

break;

case 2 :

get\_input();

break;

case 3:

printf("프로그램을 종료합니다.\n");

exit(0);

break;

}

}

return 0;

}

void get\_input(void) {

char input[BUF\_SIZE];

memset(input, 0, BUF\_SIZE);

printf("문장을 입력하세요\n");

printf("--------------------------------\n");

fgets(input, BUF\_SIZE, stdin);

printf("\n--------------------------------\n");

int res = check(input);

if (res > 0) {

res = calcul(input);

print\_res(res);

printf("-----------------------------------\n");

}

return;

}

void file\_load(void) {

char fname[BUF\_SIZE];

char buf[BUF\_SIZE + 1];

int res;

FILE \*fp = NULL;

memset(fname, 0, BUF\_SIZE);

memset(buf, 0, BUF\_SIZE);

while (1) {

printf("파일명을 입력하세요 >> ");

scanf("%s", fname);

getchar();

if ((fp = fopen(fname, "r")) == NULL) {

fprintf(stderr, "파일 오픈에 실패했습니다\n");

return;

}

else

break;

}

printf("파일 내용은 \n");

printf("------------------------------------\n");

while(fgets(buf, BUF\_SIZE, fp) != NULL) {

printf("%s\n", buf);

}

printf("------------------------------------\n");

printf("입니다.\n");

rewind(fp);

while (fgets(buf, BUF\_SIZE, fp) != NULL) {

res = check(buf);

if (res > 0) {

res = calcul(buf); //해야됨

print\_res(res);

}

}

return;

}

void print\_res(int res) {

printf("결과 : %d\n", res);

return;

}

int check(char buf[]) {

int lcnt = 0, rcnt = 0;

int len = strlen(buf);

//괄호 갯수 체크

for (int i = 0; i < len; i++) {

if (buf[i] == '(')

lcnt++;

else if (buf[i] == ')')

rcnt++;

}

if (lcnt > rcnt) {

printf("')' 갯수가 부족합니다.\n");

return -1;

}

else if (lcnt < rcnt) {

printf("'(' 갯수가 부족합니다.\n");

return -1;

}

// -가 여러개 쓰인 경우 체크

for (int i = 0; i < len; i++) {

if (buf[i] == '-') {

if (buf[i + 1] == '-') {

printf("'-'사용이 잘못되었습니다.\n");

return -1;

}

}

}

return 1;

}

int calcul(char buf[]) { // 계산하는 함수

int cur, x = 0;

int len = strlen(buf);

char tmp[10];

memset(tmp, 0, 10);

for (cur = 0; cur < len; ) {

if (buf[cur] == '(') {

cur++;

continue;

}

else if (buf[cur] == 'M') {

cur += 5;

opst.push('M');

}

else if (buf[cur] == 'I') {

cur += 2;

opst.push('I');

}

else if (buf[cur] == '-') {

x = 0;

memset(tmp, 0, 10);

tmp[x++] = buf[cur++];

while ('0' <= buf[cur] && buf[cur] <= '9')

tmp[x++] = buf[cur++];

inst.push(atoi(tmp));

}

else if ('0' <= buf[cur] && buf[cur] <= '9') {

x = 0;

memset(tmp, 0, 10);

while ('0' <= buf[cur] && buf[cur] <= '9')

tmp[x++] = buf[cur++];

inst.push(atoi(tmp));

}

else if (buf[cur] == ' ' || buf[cur] == '\n') {

cur++;

continue;

}

else if (buf[cur] == ')') {

char op = opst.top();

opst.pop();

int b = inst.top();

inst.pop();

int a = inst.top();

inst.pop();

if (op == 'M')

inst.push(a - b);

else if (op == 'I') {

if (a <= 0)

inst.push(0);

else

inst.push(b);

}

cur++;

}

}

int res = inst.top();

inst.pop();

return res;

}

int menu(void) {

char input\_c = 0;

int input\_i = 0;

print\_menu();

while (1) {

scanf("%c", &input\_c);

getchar();

if (input\_c == '1' || input\_c == '2' || input\_c == '3') {

input\_i = input\_c - '0';

return input\_i;

}

else {

printf("다시 입력하세요\n");

printf("메뉴를 선택하세요 >> ");

}

}

}

void print\_menu(void) {

printf("=========================\n");

printf("1. File Load\n");

printf("2. Interactive Mode\n");

printf("3. Exit\n");

printf("=========================\n");

printf("Select Menu >> ");

}