HW2 보고서

1. 구현 자료구조 및 함수 설명

1.1 전역변수

```
vector<string> input;
stack<string> st;
vector<int> error_msg;
string error_message[5] = { "- 기호가 여러개 존재합니다.",
                            "정수로 입력하여 주세요",
                            "undefined" ,
                            "(의 위치가 잘못되었습니다.",
                            ")의 위치가 잘못되었습니다." };
```

1.2 Tree 구조체

```
typedef struct TreeNode {
  string node;
  TreeNode* LNode;
  TreeNode* RNode;
}TreeNode;
```

binary-tree 구조체이다. 자신의 노드(string), 왼쪽 자식 포인터, 오른쪽 자식 포인터로 구성되어있다.

1.3 함수

1.3.0 `int main() {}`

맨 처음 실행되는 main 함수이다. while문으로 무한 루프를 돌려 command를 입력받을 수 있도록 하였다.

1.3.1 `void command() {}`

사용자가 맨 처음 명령을 입력받을 때 필요한 함수이다.

switch-case문으로 1을 입력하였을 때 fileload()를, 2를 입력하였을 때 interactive()를, 3을 입력하였을 때 exit(1)을 할 수 있도록 하였다.

1.3.3 `void fileload() {}`

사용자가 작성한 파일이 같은 디렉토리에 있을 경우, 불러와서 한 줄 씩 읽는다. 만약 파일이 발견되지 않았을 경우, "파일 읽기 오류"를 출력하며 다시 command부터 입력하도록 한다.

getline()으로 한 줄 씩 입력 받으며, 이것은 char line[]에 저장하여, operate함수로 넘겨준다.

1.3.2 `void interactive() {}`

사용자가 직접 문장을 입력할 경우이다. getline()으로 한 줄을 입력 받으며, 이것은 char line[]에 저장하여, operate함수로 넘겨준다.

1.3.4 `void operate(char line[]) {}`

operate()함수는 크게 parse(), addNode(), postorder()로 구성되어있다. 세 함수를 실행시키고, 최종적으로 결과값을 출력해낸다.

error메세지를 출력하는 부분도 구성되어있다. error_check_num()이나 error_check_str()함수에서 에러가 발견되면, 해당 error메세지를 출력해주고 계산 없이 리턴한다.

1.3.5 `void parse(char line[]) {}`

line[]으로 입력줄을 받았으면, 이것을 괄호, 띄어쓰기로 파싱하여 전역변수인 input vector에 넣어준다.

예를들어, 사용자가 (IF 3 1)을 입력하였다면, input vector에 차례대로 (, IF, 3, 1,) 이 string 타입으로 들어가게 된다.

1.3.6 `TreeNode* addNode(int start, int end) {}`

파싱이 끝났다면, 최초에 addNode()가 operate()에서 불려졌을 때, 0과 input.size() -1 을 인자로 전달하여 실행이 된다. 이는 input 벡터의 첫번째 값, 마지막 값을 나타낸다.

이 함수는 재귀적으로 구성되어 있으며, 점차 start와 end의 범위를 좁혀 나간다. 재귀를 부르는 조건은 '(' 괄호의 갯수와 ')' 괄호의 갯수가 같을 때, 그 위치를 인자로 넣어주어 부른다. 또한, 왼쪽 오른쪽을 구분하여 넣는다.

함수가 실행될 때 마다 newNode 를 만들어 자신에게 값을 넣고, start와 end의 차이가 4라면 (예 : (IF 3 2)의 경우, start = 0, end = 4로 차이가 4이다.) 왼쪽, 오른쪽을 각각 삽입하도록 한다.

또한, 노드를 삽입할 때 마다 숫자와 문자열이 올바르게 들어가는지 error_check_str(), error_check_num() 함수로 검사한다.

1.3.7 `void postorder(TreeNode* ptr) {}`

addNode()로 만들어진 binary tree를 postorder로 읽어 스택에 저장한다. 재귀적으로 left를 먼저 방문해주고, right를 방문한다. stack에 저장시, MINUS나 IF를 만나면 pop()연산을 해 주어, calculate함수로 결과를 계산한다. 그리고 그 결과를 다시 스택에 push()해준다. 최종적으로, 스택에는 결과값만 남게 된다.

1.3.8 `string calculate(string op, string a, string b) {}`

MINUS일 경우, a-b를 리턴해준다. IF일 경우, a가 양수이면 b를 리턴하고, 그렇지 않으면 0을 리턴한다.

1.3.9 `void error_check_str(string check) {}`

문자열이 IF나 MINUS가 아니면 error_msg 벡터에 1을 push_back()한다.

1.3.10 `void error_check_num(string check) {}`

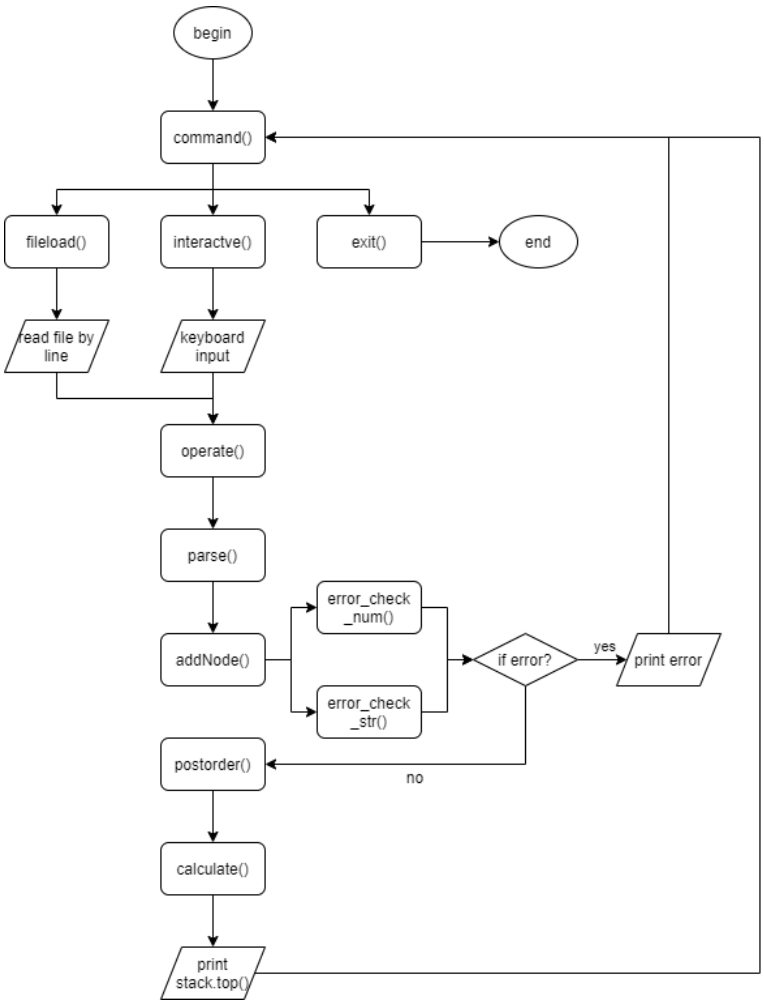
숫자가 실수이거나 -기호가 여러개 있을 경우, 해당하는 에러메세지 인덱스를 err_msg 벡터에 push_back()한다.

1.3.11 `bool isInt(string str) {}`

정수형인지 아닌지 검사하는 부가적인 함수이다. addNode에서 노드 삽입시 사용된다.

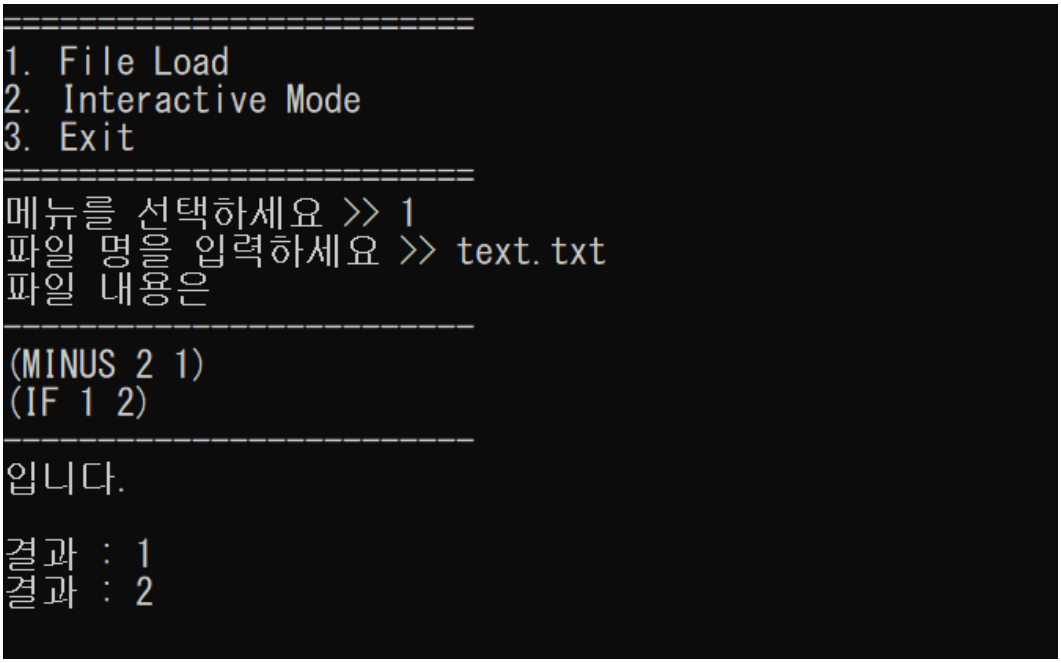
---

2. 흐름도

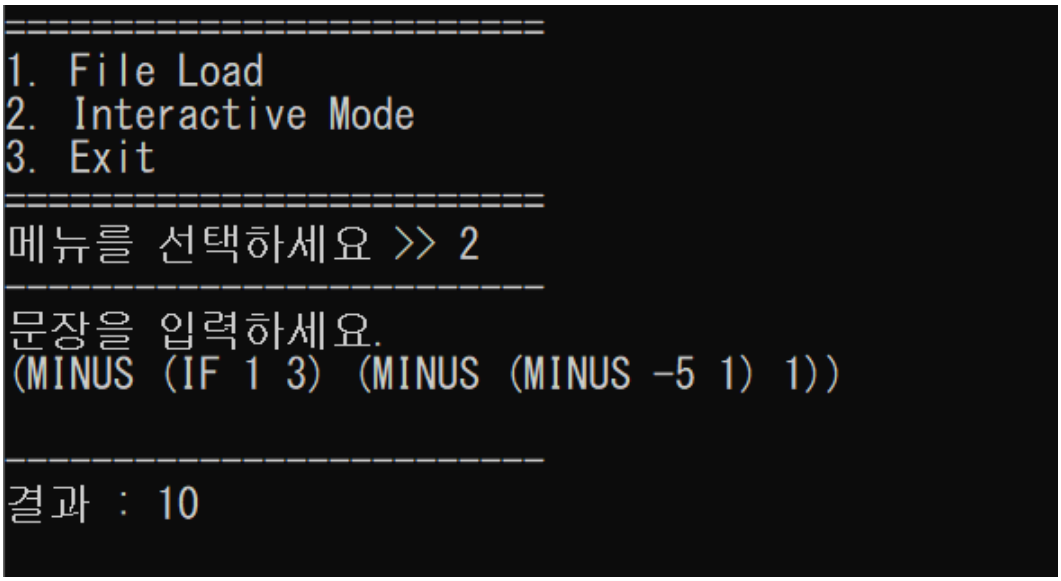


3. 실행 화면

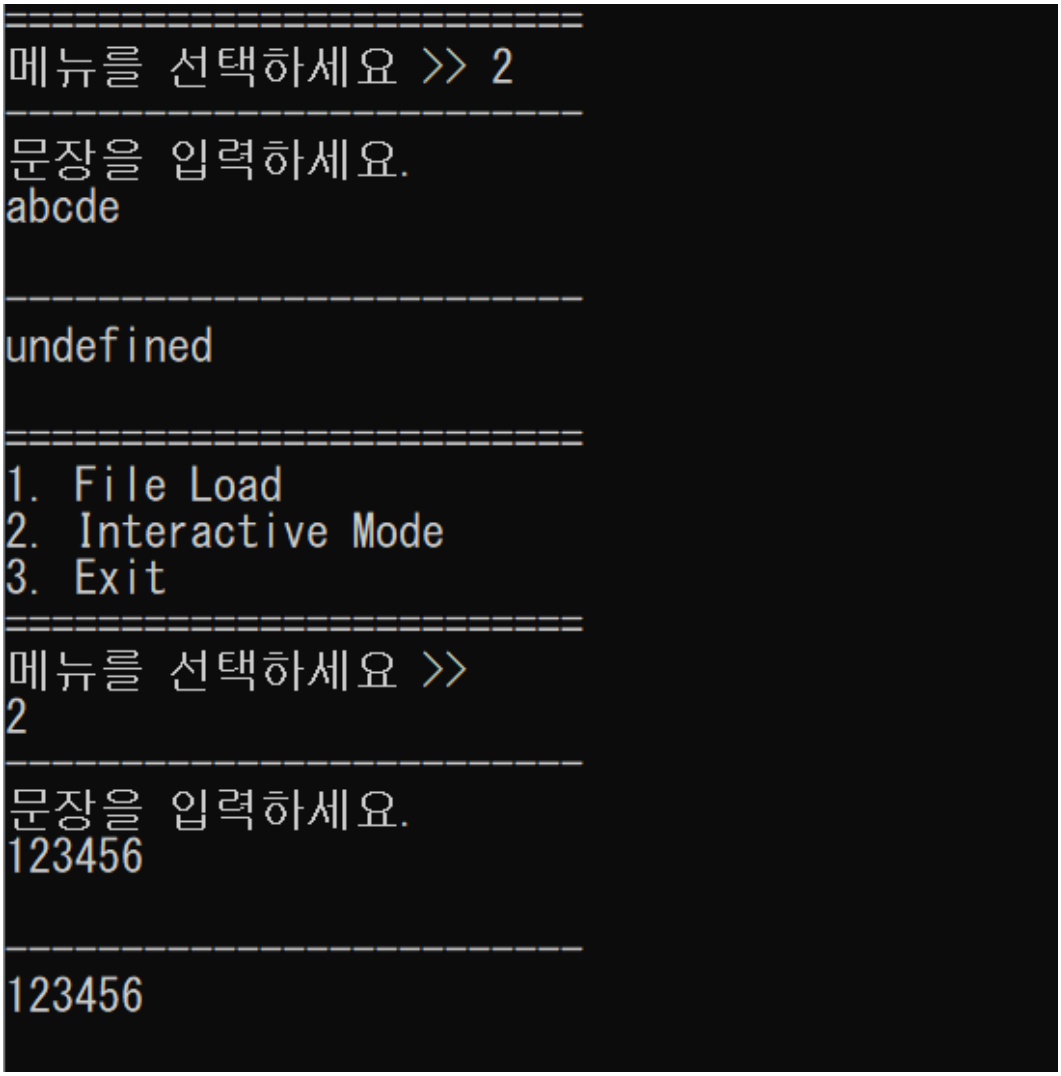
3.1 파일에서 입력받기(간단한 구문)



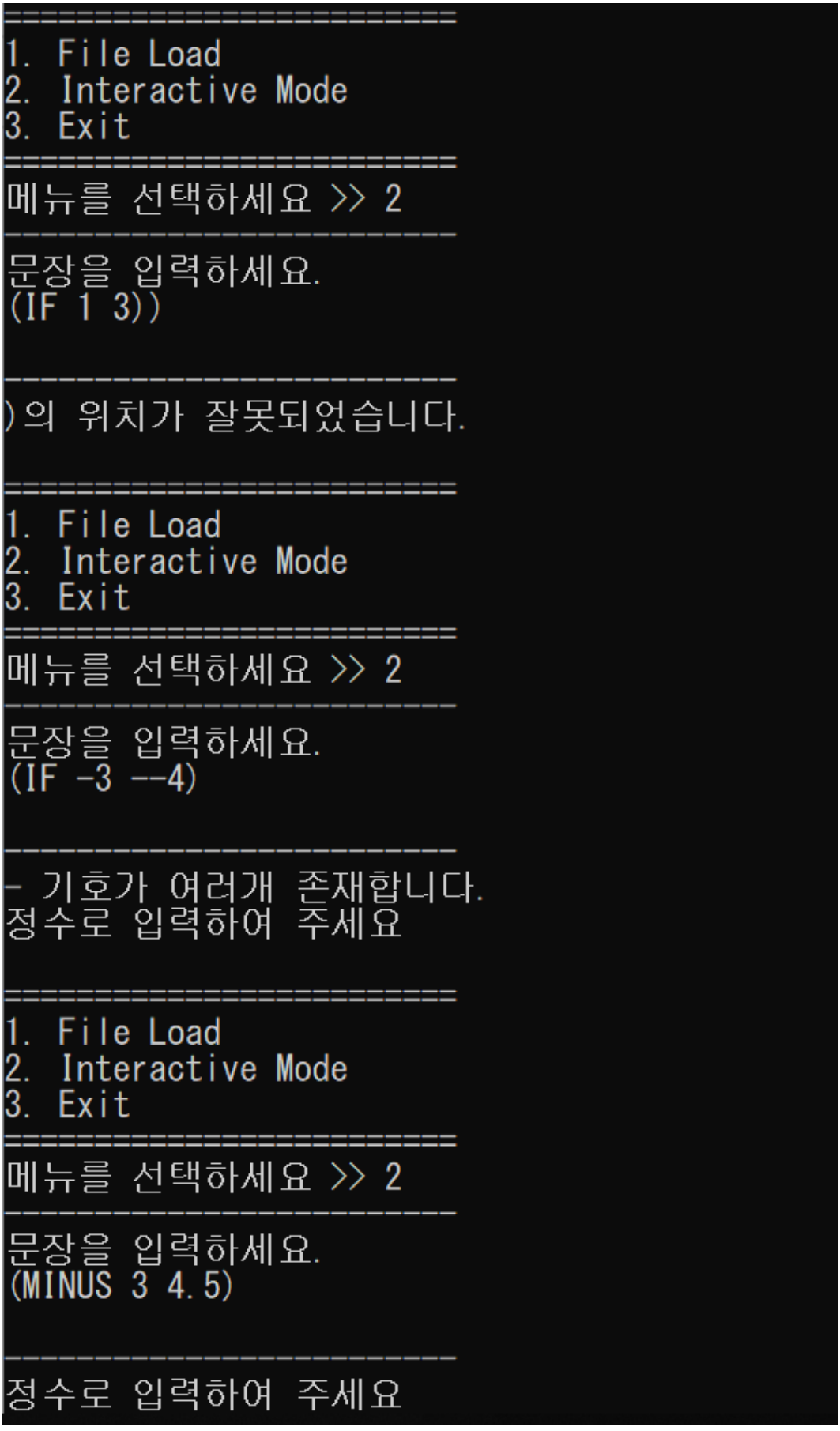
3.2 사용자가 입력받기 (복잡한 구문)



3.3 예외처리1 (숫자, IF, MINUS를 제외한 문자열)



3.4 예외처리2 (괄호 갯수, -기호, 실수)


==========
1. File Load
2. Interactive Mode
3. Exit
==========
메뉴를 선택하세요 >> 2

문장을 입력하세요.
(IF 1 3))

)의 위치가 잘못되었습니다.
==========
1. File Load
2. Interactive Mode
3. Exit
==========
메뉴를 선택하세요 >> 2

문장을 입력하세요.
(IF -3 --4)

- 기호가 여러개 존재합니다.
정수로 입력하여 주세요
==========
1. File Load
2. Interactive Mode
3. Exit
==========
메뉴를 선택하세요 >> 2

문장을 입력하세요.
(MINUS 3 4.5)

정수로 입력하여 주세요

3.4 프로그램 종료