**20182647 컴퓨터학부 이형규**

**1. 함수설명**

void start\_interpreter();

- 인터프리터 실행함수

- 메뉴 선택 화면을 띄워주고 메뉴를 선택하는 기능을 가지고 있다.

void add\_minus\_if();

- DEFUN으로 정의한 함수들을 저장하는 변수 defun\_data와 defun\_names 벡터에 MINUS와 IF 정보를 추가시키는 함수

void define\_defun();

- 함수를 새로 정의하는 기능을 가진 함수

void load\_defun();

- defun.txt에서 정의된 함수들을 읽어 defun\_data와 defun\_names에 정보를 저장하는 함수

void interpreter();

- 파일명을 입력받아 한줄 씩 읽어들은 뒤 연산하여 값을 출력하는 함수

void make\_tokens2(vector<string>& tmp1, char\* commands);

bool make\_tokens3(vector<string>& dst, string command);

- 넘겨받은 명령어들을 토큰화 시키는 함수

void converter(vector<string>& old\_v, vector<string>& new\_v, int\* index);

- 토큰화시킨 벡터에서 MINUS와 IF 명령어만 남도록 다른 함수들을 변환시켜주는 함수

bool check\_bracket\_cnt(vector<string>& tokens);

bool check\_error(vector<string>& tokens, int\* index, string command);

- 괄호갯수, 위치, 정의되지 않은 함수, 실수, 잘못된 기호, - 기호의 사용오류를 체크하는 함수

Node\* make\_tree(vector<string>& tokens, int\* index);

Node\* create\_node(string name);

- tokens 벡터를 이용하여 tree제작과 tree제작에 필요한 노드를 생성하는 함수

void print\_tree(Node\* move);

- tree의 구조를 post order로 순회하면서 출력하는 함수

bool is\_character(char a);

bool is\_number(char a);

- 입력받은 문자가 문자(a~z, A~Z)인지 숫자(0~9)인지 판별하는 함수

void calc\_tree(Node\* root);

- tree의 루트를 매개변수로 받아 값을 계산하는 함수

2. 구현 자료구조

vector<string> defun\_names;

- 함수들의 이름을 저장하는 벡터로서, Define\_defun 기능에서 중복되는 기능이 있는지 검사한다

vector<pair<vector<string>, vector<string>>> defun\_data;

- DEFUN들의 대한 정보를 가지고 있다.

- pair의 첫번째 인자는 함수의 프로토타입을 토큰화 시킨 형태로 가지고 있다

- ex) ADD ( x y ) 의 경우 -> ADD , ( , x , y , )와 같이 쪼개져있다.

- pair의 두번째 인자는 함수의 내용을 토큰화 시킨 형태로 가지고 있다.

- ex) ADD 함수의 경우 -> ( , MINUS , x , ( , MINUS , 0 , y , ) , ) , ) 와 같이 쪼개져있다.

- interpreter 함수에서 변환을 할 때 주로 사용한다.

typedef struct Node

{

char\* name;

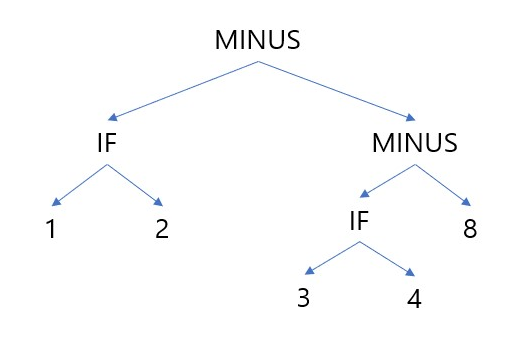
struct Node\* left;

struct Node\* right;

}Node;

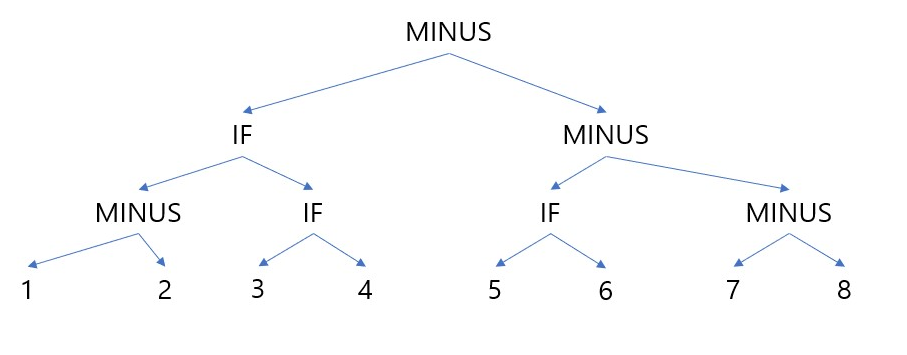
- 직접적인 연산을 위해 tree를 구성하는 노드의 구조체이다.

**(MINUS (IF 1 2) (MINUS (IF 3 4) 8)의 경우**

****

탐색 순서 : 1->2->IF->3->4->IF->8->MINUS->MINUS 순으로 탐색

**(MINUS (IF (MINUS 1 2) (IF 3 4)) (MINUS (IF 5 6) (MINUS 7 8)))의 경우**



탐색 순서 : 1->2->MINUS->3->4->IF->IF->5->6->IF->7->8->MINUS->MINUS->MINUS

- 정수일 경우 스택에 저장 IF나MINUS일 경우 스택에서 pop연산을 2번해서 연산 후 다시 스택에 저장. 트리 순회를 마치고 나서 스택에 남아있는 정수 하나가 계산 결과

- 2번과제 보고서(본인이 직접 작성)를 인용했습니다.

**converter() 함수 내부 자료구조**

**- (IF/THEN/ELSE 1 2 3)의 경우 (old\_v) (old\_v는 lex별로 쪼개진 vector임)**

- converter()함수의 인자로 old\_v와 변환이 완료된 벡터인 new\_v를 인자로 넘겨주고, old\_v를 차례대로 검사하면서 new\_v에 push\_back해준다.

- old\_v[index]가 문자가 아닌 경우 new\_v에 넣어주고, 문자일 경우 MINUS나 IF면 new\_v에 넣어준다. 나머지 문자들의 경우는 defun\_data의 첫번째 데이터의 처음과 비교하여 같은 함수를 찾는다. 그리고 두번째 데이터를 vector<string> exp 에 저장하여 함수의 내용을 저장해둔다.

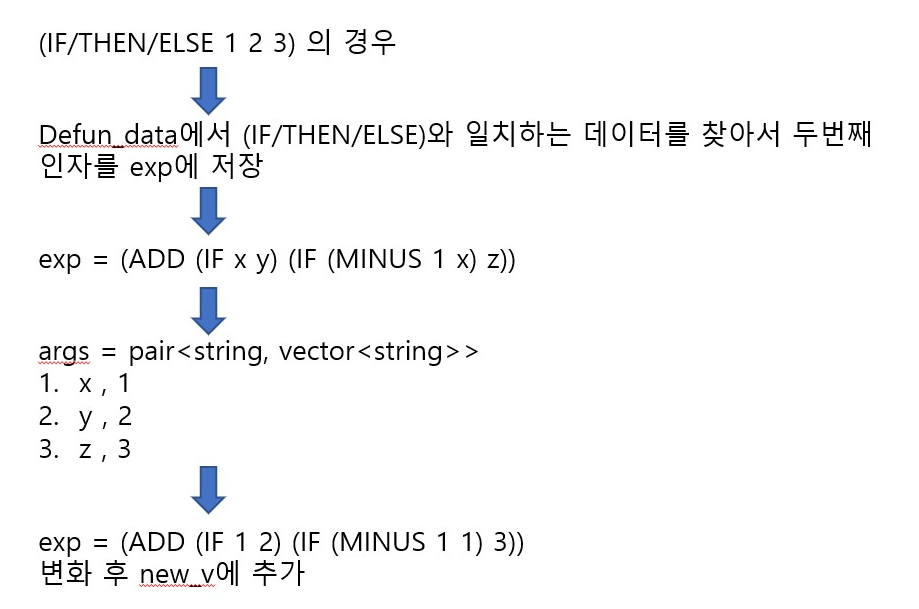
- 그리고 첫번째 데이터의 “(“와 “)”의 위치를 저장한 뒤 매개변수의 개수를 세어준다.

- vector<pair<string, vector<string>>> args 에다가 첫번째 인자로는 정의한 함수의 매개변수를 저장하고 두번째 인자로는 매개변수에 대입한 벡터를 저장한다.

- exp 에다가 args를 이용하여 매개변수 값들을 실제 값으로 변경시켜준다

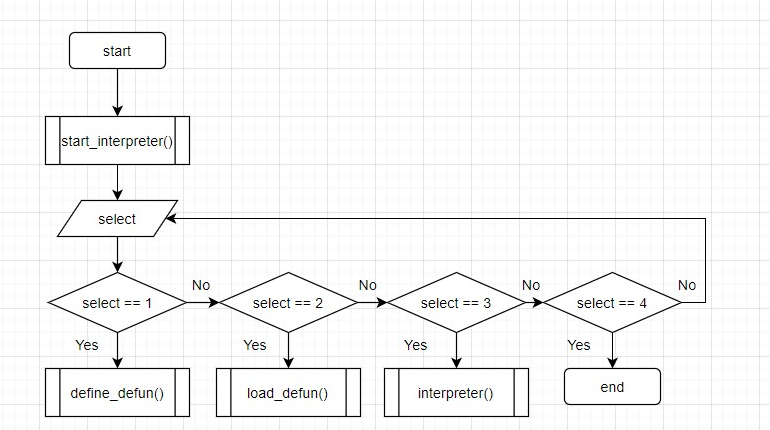
- 함수 변환이 끝나면 new\_v에다가 exp를 추가한다.

- 위 과정을 new\_v에 MINUS와 IF만 남을때까지 진행한다.

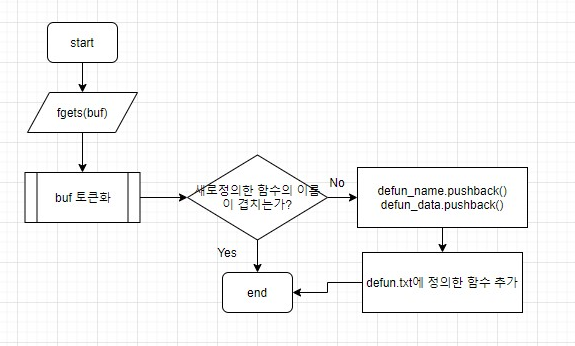


**3. 흐름도**

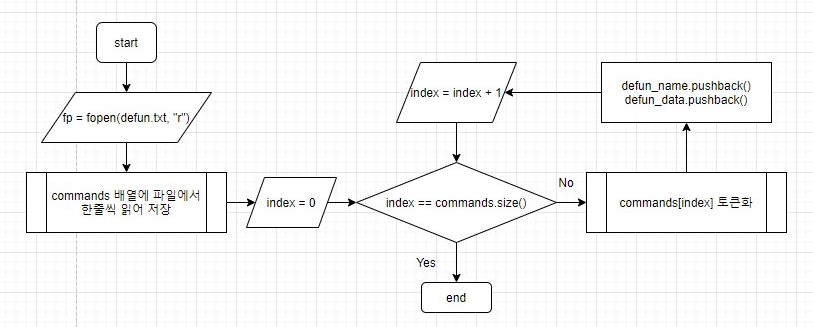
**- main 함수**

****

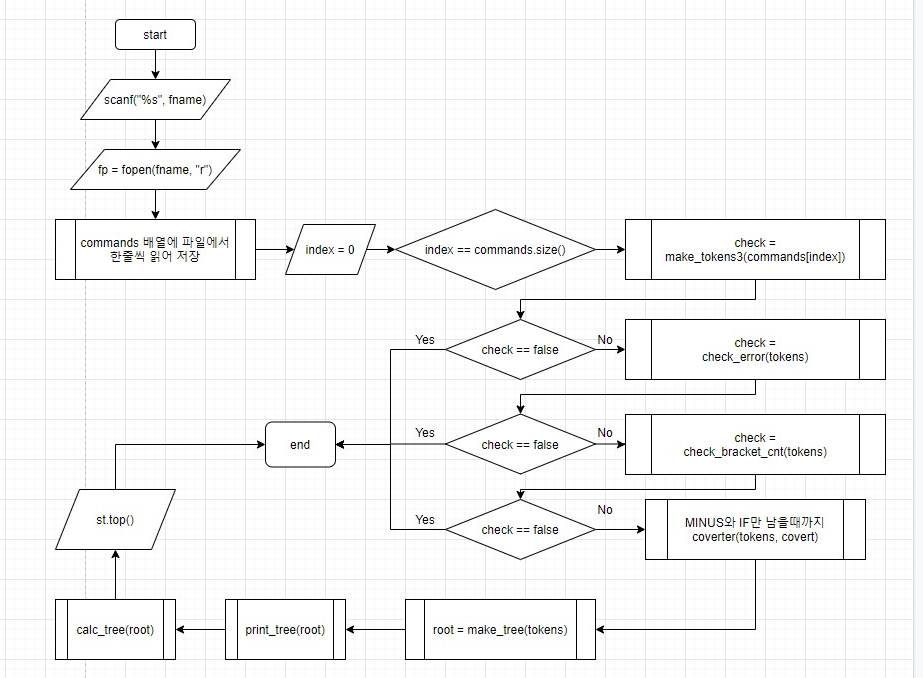
**- define\_defun()**

****

**- load\_defun()**

****

**- interpreter()**

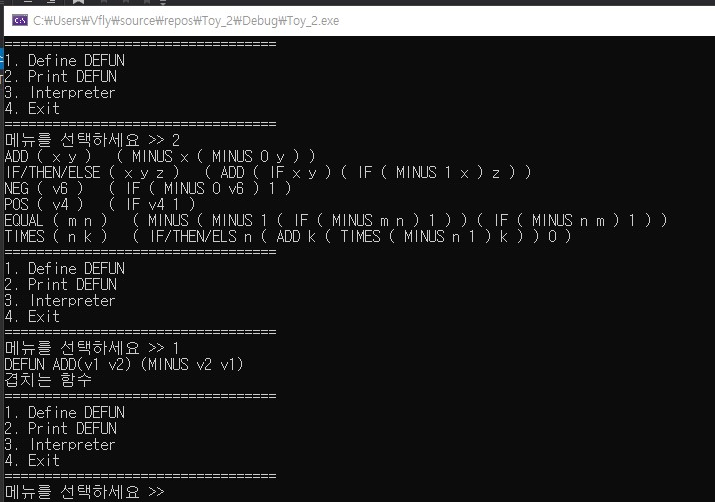
****

**4. 실행 결과**

**- Print DEFUN 시**

****

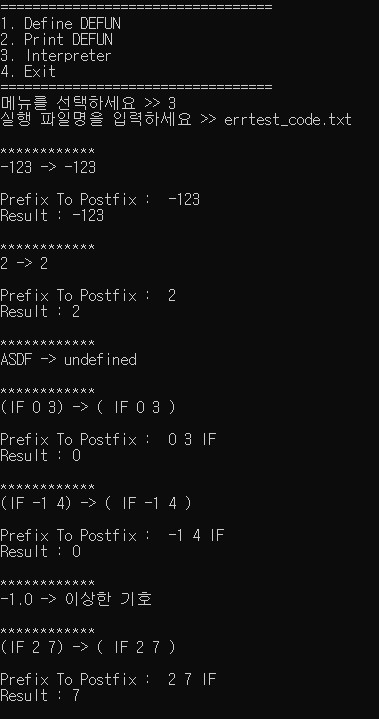
**- Define Defun시 겹치는 함수가 존재할경우**

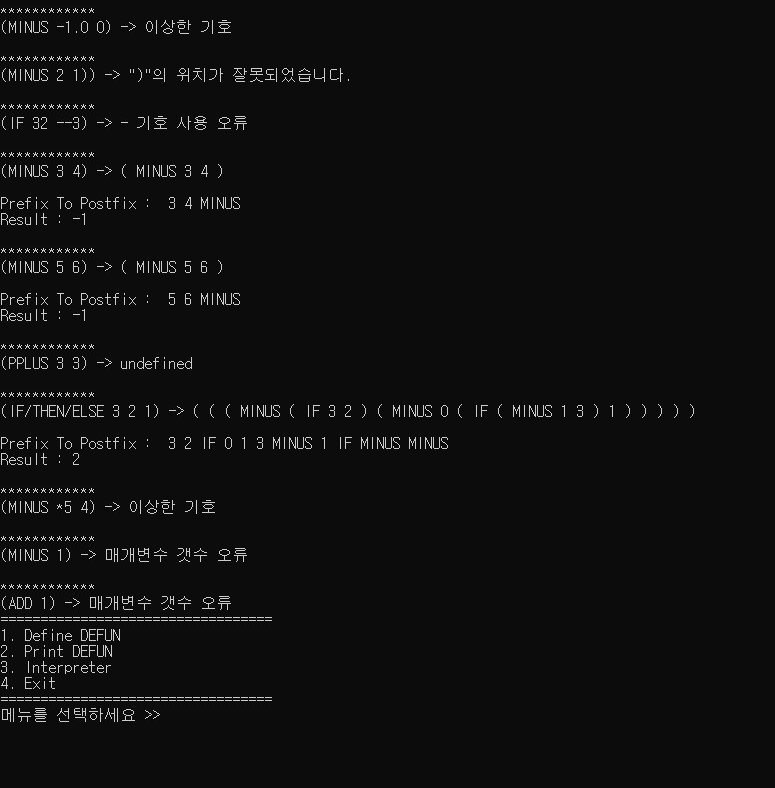
****

**- 정상적인 Define 후 Print DEFUN 시**

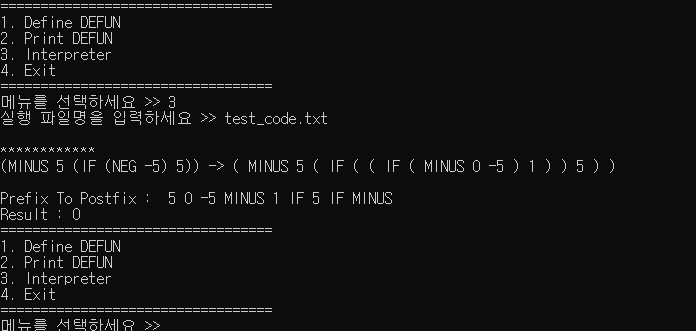
****

**- 올려주신 errtest\_code.txt 실행 시**

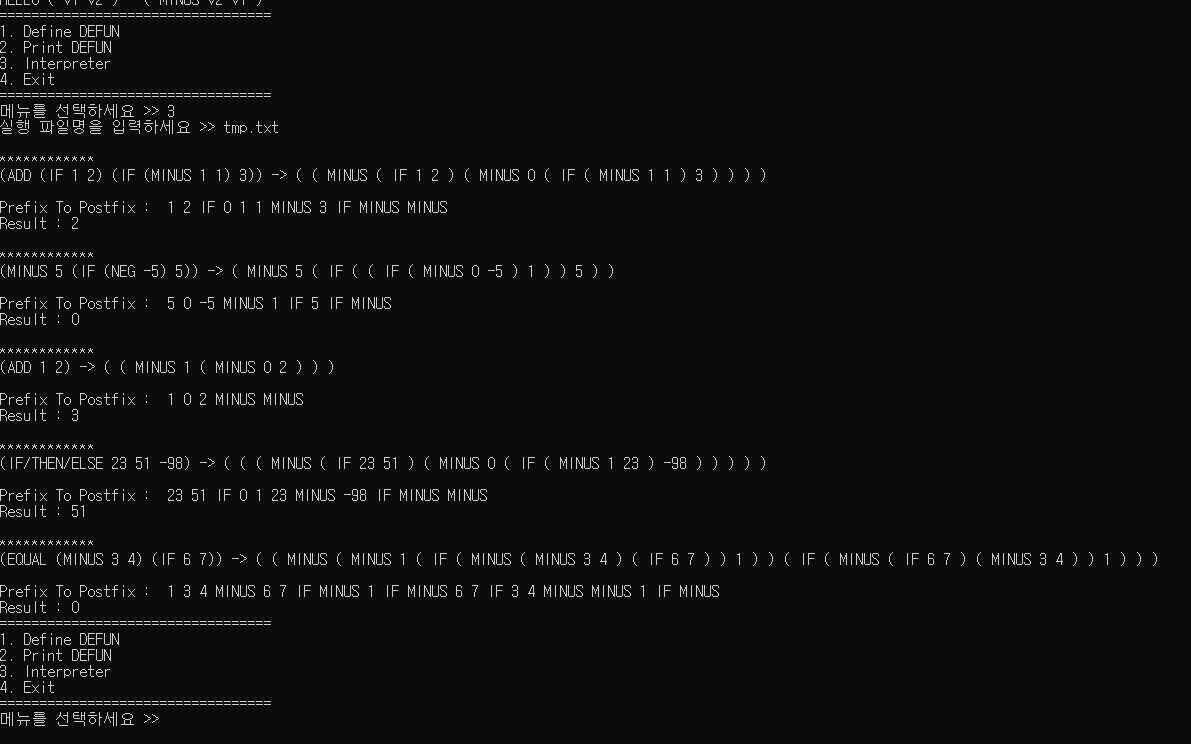
****

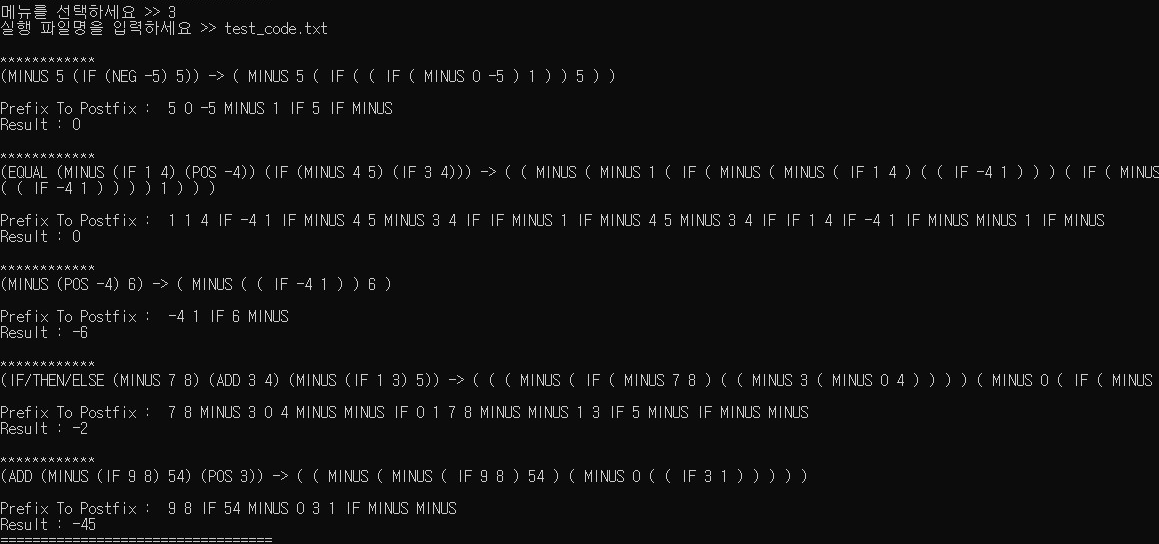
****

**- 올려주신 test\_code.txt 실행 시**

****

**- 추가 예제**

****

****