Compiler

컴파일러

1번 과제물 (수정-1)

2023. 3.20.

소속 : 충북대학교 소프트웨어학과
학번 : 2019038004
학년 : 3
작성자 : 조민우

	수정 내역
초판	초판
v1.1	· Parser 코드 구현부 수정 - instr() 함수가 파싱의 관점에서 직관적으로 보이도록 수정 - match() 함수를 활용하도록 코드 수정

□ 개요

현 위치에서 명령에 따라 동서남북으로 한 단계씩 이동할 수 있는 로봇이 있다고 가정하자. 이 로봇의 명령어 형식이 다음과 같은 문법으로 주어졌을 경우, 이를 파싱하여 처리할 수 있는 예측파서를 C언어로 구현하시오. 단, 왼쪽 순환 문법을 오른쪽 순환 문법으로 변환하여 구현하시오.

〈seq〉 −〉 〈seq〉 〈instr〉 | **begin**
〈instr〉 −〉 **east** | **north** | **west** | **south**

begin 은 현위치를 (0,0)으로 설정하는 명령이며, east는 오른쪽으로 한 칸, north는 위쪽으로 한 칸, west는 왼쪽으로 한 칸, south는 아래쪽으로 한 칸 이동하는 명령이다. 단 구현의 편의상, begin, east, north, west, south는 각각 첫 글자인 b, e, n, w, s로 입력되고 명령의 마지막은 $가 반드시 포함된다. 또 각 명령이 분석(파싱)될 때 해당 명령을 수행하고 위치를 출력하는 action을 포함하시오.

□ 과정

· 기존 CFG를 오른쪽 순환으로 변경
〈seq〉가 start symbol 이라는 것은 문제에 주어져 있지 않으나, 이후 예시들에 b로 시작한다는 것이 주어져 있으므로 〈seq〉가 start symbol 이라는 것은 자명하다고 볼 수 있음.
〈seq〉가 start 심볼이라고 가정할시, 〈seq〉를 통해 만들 수 있는 문장은 begin (〈seq〉)* 로 표현할 수 있음.
즉, begin이 0개 이상 포함되어야 하므로
〈seq〉 -〉 begin 〈instr〉
로 수정할 수 있으며 〈instr〉 은 오른쪽 순환 표현을 위해
〈instr〉 -〉 east 〈instr〉 | north 〈instr〉 | west 〈instr〉 | south 〈instr〉 | ε
으로 수정할 수 있음.

즉, 구현해야 하는 문법은

〈seq〉 -〉 begin 〈instr〉
〈instr〉 -〉 east 〈instr〉 | north 〈instr〉 | west 〈instr〉 | south 〈instr〉 | ε

이며, 실 코드 부에서는 terminal들을 각각 begin:b, east,:e, north:n, west:w, south:s, ε:$ 로 표현하며, 각각의 terminal들에 대해 동작을 prt_pos 함수를 통해 현재 좌표를 표현하도록 함.

· 수도코드를 통한 파서 작성

```
x, y = 0, 0
def prt_pos: print((x, y))
def nexttoken: lookahead = getchar()


def seq:
    if lookahead == 'b':
        prt_pos()
        nexttoken()
        instr()
    else: print(error)


def instr:
    if lookahead in [e, n, w, s, $]:
        if instr_chk() == 1:
            prt_pos()
            nexttoken()
            instr()
        else if 0: print(new_line)
        else: print(error)


def instr_chk:
    if lookahead is [e, n, w, s]:
        do each (x+1, y+1, x-1, y-1)
        return 1
    elif $:
        return 0
    else <- means error :
        return -1


main:
    nexttoken()
    if lookahead == b: seq()
    else: print(error)
```

□ 결과

· 실제로 작성된 C언어 코드부

```c
#include <stdio.h>
char lookahead;
int x =0;
int y =0;


void match(char token);
void nexttoken();
void seq();
void instr();
void prt_pos();


int main() {
    nexttoken();
    if (lookahead =='b') {
        seq();
    }
    else {
        printf("error\n");
    }
    return 0;
}


void match(char token) {
    if (lookahead == token)
        nexttoken();
    else {
        printf("error\n");
        return;
    }
}


void nexttoken() {
    char c;
    while (1) {
        c = getchar();
        if (c == EOF) {
            lookahead ='$';
            return;
        }
        if (c ==' '|| c =='\t'|| c =='\n'|| c =='\0') continue;
        lookahead = c;
        return;
    }
}
```

```
void seq()
{
    match('b');
    prt_pos();
    if (lookahead =='e'|| lookahead =='n'|| lookahead =='w'|| lookahead =='s'|| lookahead
=='$') {
        instr();
    }
    else {
        printf("error");
    }
}


void instr()
{
    if (lookahead =='e') {
        match('e');
        x +=1;
        prt_pos();
        instr();
    }
    else if (lookahead =='n') {
        match('n');
        y +=1;
        prt_pos();
        instr();
    }
    else if (lookahead =='w') {
        match('w');
        x -=1;
        prt_pos();
        instr();
    }
    else if (lookahead =='s') {
        match('s');
        y -=1;
        prt_pos();
        instr();
    }
    else if (lookahead =='$') {
        printf("\n");
    }
    else {
        printf("error");
    }
}


void prt_pos()
{
    printf("(%d, %d) ", x, y);
}
```

· 해당 파싱이 적절한지 판단하기 위해 적절한 문자열을 생성하기 위해 작성한 파이썬 코드

```
import random
import sys
sys.stdout =open("./result.txt", 'w')
moves ="enws"
for tries in range(5):
    print(tries+1, ":", end='')
    result ="b"
    for i in range(98):
        result += moves[random.randrange(4)]
    result +="$"
    print(result)
```

· 해당 코드로 생성된 문법이 올바른 100자리 문자열 5종

```
1:bnnseewwwswnwneswnseeswseeswwnennenwensneennnnnsssesesneseswnsewssesssssennneeweseenesewwewesswswwnw$
2:bnseeesnswwnssenswewnwsnwnewsseeewnwweennwnwnwnwnnsesnwsnwnwnsennenwssenweesnnewnsneesnnwwswsene$
3:benwwsneenssewwwweweensnwennwwnneswswwnweeesseewwnnensesnewswswwnswssnwssnsnnseewwnwnwsnsnnswnswnwe$
4:bwnenneenewewweensseweewwessnnewnnwsnswweesessnnnwwnseseswwweeewseeewwwwsnewssssneswsnesweewwewenew$
5:bennwennswnenwwnnwneesnswwenssesnnennesnssnnnseesssseewenwnnnnweswnewwsnwsnwwseewewesnwswwwnswnww$
```

· 해당 파서가 제대로 동작하는지 확인하기위한 구현 기능상 동일한 파이썬 코드 작성

```
import sys

x, y =0, 0

def prt_pos():
    print("({}, {})".format(x, y), end=" ")

text = sys.stdin.readline().rstrip()

if text[0] !="b":
    print("error")
else:
    i =1
    prt_pos()
    while i < len(text):
        if text[i] =='e':
            x +=1
            prt_pos()
        elif text[i] =='n':
            y +=1
            prt_pos()
        elif text[i] =='w':
            x -=1
            prt_pos()
        elif text[i] =='s':
            y -=1
            prt_pos()
        elif text[i] =='$':
            print()
            break
        else:
            print("error")
            break
        i +=1
```

· 해당 파이썬 코드는 기획에서 의도한 바와 같이 동작할 수 밖에 없다.

□

· 각 번호 수에 대해 입력된 결과 값 확인

	제작된 예측 파서(C언어)	검증을 위한 기능상 동일한 프로그램(파이썬)
1	(0, 0) (0, 1) (0, 2) (0, 1) (1, 1) (2, 1) (1, 1) (0, 1) (-1, 1) (-1, 0) (-2, 0) (-2, 1) (-3, 1) (-3, 2) (-2, 2) (-2, 1) (-3, 1) (-3, 2) (-3, 1) (-2, 1) (-1, 1) (-1, 0) (-2, 0) (-2, -1) (-1, -1) (0, -1) (0, -2) (-1, -2) (-2, -2) (-2, -1) (-1, -1) (-1, 0) (-1, 1) (0, 1) (0, 2) (-1, 2) (0, 2) (0, 3) (0, 2) (0, 3) (1, 3) (2, 3) (2, 4) (2, 5) (2, 6) (2, 7) (2, 8) (2, 7) (2, 6) (2, 5) (3, 5) (3, 4) (4, 4) (4, 3) (4, 4) (5, 4) (5, 3) (6, 3) (6, 2) (5, 2) (5, 3) (5, 2) (6, 2) (5, 2) (5, 1) (5, 0) (6, 0) (6, -1) (6, -2) (6, -3) (6, -4) (7, -4) (7, -3) (7, -2) (7, -1) (8, -1) (9, -1) (8, -1) (9, -1) (9, -2) (10, -2) (11, -2) (11, -1) (12, -1) (12, -2) (13, -2) (12, -2) (11, -2) (12, -2) (11, -2) (12, -2) (12, -3) (12, -4) (11, -4) (11, -5) (10, -5) (9, -5) (9, -4) (8, -4)	(0, 0) (0, 1) (0, 2) (0, 1) (1, 1) (2, 1) (1, 1) (0, 1) (-1, 1) (-1, 0) (-2, 0) (-2, 1) (-3, 1) (-3, 2) (-2, 2) (-2, 1) (-3, 1) (-3, 2) (-3, 1) (-2, 1) (-1, 1) (-1, 0) (-2, 0) (-2, -1) (-1, -1) (0, -1) (0, -2) (-1, -2) (-2, -2) (-2, -1) (-1, -1) (-1, 0) (-1, 1) (0, 1) (0, 2) (-1, 2) (0, 2) (0, 3) (0, 2) (0, 3) (1, 3) (2, 3) (2, 4) (2, 5) (2, 6) (2, 7) (2, 8) (2, 7) (2, 6) (2, 5) (3, 5) (3, 4) (4, 4) (4, 3) (4, 4) (5, 4) (5, 3) (6, 3) (6, 2) (5, 2) (5, 3) (5, 2) (6, 2) (5, 2) (5, 1) (5, 0) (6, 0) (6, -1) (6, -2) (6, -3) (6, -4) (7, -4) (7, -3) (7, -2) (7, -1) (8, -1) (9, -1) (8, -1) (9, -1) (9, -2) (10, -2) (11, -2) (11, -1) (12, -1) (12, -2) (13, -2) (12, -2) (11, -2) (12, -2) (11, -2) (12, -2) (12, -3) (12, -4) (11, -4) (11, -5) (10, -5) (9, -5) (9, -4) (8, -4)
2	(0, 0) (0, 1) (0, 0) (1, 0) (2, 0) (3, 0) (3, -1) (3, 0) (3, -1) (2, -1) (1, -1) (1, 0) (1, -1) (1, -2) (2, -2) (2, -1)(2, -2) (1, -2) (2, -2) (1, -2) (1, -1) (0, -1) (0, -2) (0, -1) (-1, -1) (-1, 0) (0, 0) (-1, 0) (-1, -1) (-1, -2) (0, -2) (1, -2) (2, -2) (1, -2) (1, -1) (0, -1) (-1, -1) (-2, -1) (-1, -1) (0, -1) (0, 0) (0, 1) (-1, 1) (-1, 2) (-1, 3) (-2,3) (-2, 4) (-3, 4) (-4, 4) (-4, 5) (-5, 5) (-5, 6) (-5, 7) (-5, 6) (-4, 6) (-4, 5) (-4, 6) (-5, 6) (-5, 5) (-5, 6) (-6,6) (-6, 7) (-7, 7) (-7, 8) (-7, 7) (-6, 7) (-6, 8) (-6, 9) (-5, 9) (-5, 10) (-6, 10) (-6, 9) (-6, 8) (-5, 8) (-5, 9) (-6, 9) (-5, 9) (-4, 9) (-4, 8) (-4, 9) (-4, 10) (-3, 10) (-4, 10) (-4, 11) (-4, 10) (-4, 11) (-3, 11) (-2, 11) (-2, 10) (2, 11) (-2, 12) (-3, 12) (-4, 12) (-4, 11) (-5, 11) (-5, 10) (-4, 10) (-4, 11) (-3, 11)	(0, 0) (0, 1) (0, 0) (1, 0) (2, 0) (3, 0) (3, -1) (3, 0) (3, -1) (2, -1) (1, -1) (1, 0) (1, -1) (1, -2) (2, -2) (2, -1) (2, -2) (1, -2) (2, -2) (1, -2) (1, -1) (0, -1) (0, -2) (0, -1) (-1, -1) (-1, 0) (0, 0) (-1, 0) (-1, -1) (-1, -2) (0, -2) (1, -2) (2, -2) (1, -2) (1, -1) (0, -1) (-1, -1) (-2, -1) (-1, -1) (0, -1) (0, 0) (0, 1) (-1, 1) (-1, 2) (-1, 3) (-2, 3) (-2, 4) (-3, 4) (-4, 4) (-4, 5) (-5, 5) (-5, 6) (-5, 7) (-5, 6) (-4, 6) (-4, 5) (-4, 6) (-5, 6) (-5, 5) (-5, 6) (-6, 6) (-6, 7) (-7, 7) (-7, 8) (-7, 7) (-6, 7) (-6, 8) (-6, 9) (-5, 9) (-5, 10) (-6, 10) (-6, 9) (-6, 8) (-5, 8) (-5, 9) (-6, 9) (-5, 9) (-4, 9) (-4, 8) (-4, 9) (-4, 10) (-3, 10) (-4, 10) (-4, 11) (-4, 10) (-4, 11) (-3, 11) (-2, 11) (-2, 10) (-2, 11) (-2, 12) (-3, 12) (-4, 12) (-4, 11) (-5, 11) (-5, 10) (-4, 10) (-4, 11) (-3, 11)
3	(0, 0) (1, 0) (1, 1) (0, 1) (-1, 1) (-1, 0) (-1, 1) (0, 1) (1, 1) (1, 2) (1, 1) (1, 0) (2, 0) (1, 0) (0, 0) (-1, 0) (-2,0) (-1, 0) (-2, 0) (-1, 0) (0, 0) (0, 1) (0, 0) (0, 1) (-1, 1) (0, 1) (0, 2) (0, 3) (-1, 3) (-2, 3) (-2, 4) (-2, 5) (-1, 5) (-1, 4) (-2, 4) (-2, 3) (-3, 3) (-4, 3) (-4, 4) (-5, 4) (-6, 4) (-5, 4) (-4, 4) (-4, 3) (-4, 2) (-3, 2) (-2, 2) (-3, 2) (-4, 2) (-4, 3) (-4, 4) (-3, 4) (-3, 5) (-3, 4) (-2, 4) (-2, 3) (-2, 4) (-1, 4) (-2, 4) (-2, 3) (-3, 3) (-3, 2) (-4, 2) (-5, 2) (-5, 3) (-5, 2) (-6, 2) (-6, 1) (-6, 0) (-6, 1) (-7, 1) (-7, 0) (-7, -1) (-7, 0) (-7, -1) (-7, 0) (-7, 1) (-7, 0) (-6, 0) (-5, 0) (-6, 0) (-7, 0) (-7, 1) (-8, 1) (-8, 2) (-9, 2) (-9, 1) (-9, 2) (-9, 1) (-9, 2) (-9, 3) (-9, 2) (-10, 2) (-10, 3) (-10, 2) (-11, 2) (-11, 3) (-12, 3) (-11, 3)	(0, 0) (1, 0) (1, 1) (0, 1) (-1, 1) (-1, 0) (-1, 1) (0, 1) (1, 1) (1, 2) (1, 1) (1, 0) (2, 0) (1, 0) (0, 0) (-1, 0) (-2, 0) (-1, 0) (-2, 0) (-1, 0) (0, 0) (0, 1) (0, 0) (0, 1) (-1, 1) (0, 1) (0, 2) (0, 3) (-1, 3) (-2, 3) (-2, 4) (-2, 5) (-1, 5) (-1, 4) (-2, 4) (-2, 3) (-3, 3) (-4, 3) (-4, 4) (-5, 4) (-6, 4) (-5, 4) (-4, 4) (-4, 3) (-4, 2) (-3, 2) (-2, 2) (-3, 2) (-4, 2) (-4, 3) (-4, 4) (-3, 4) (-3, 5) (-3, 4) (-2, 4) (-2, 3) (-2, 4) (-1, 4) (-2, 4) (-2, 3) (-3, 3) (-3, 2) (-4, 2) (-5, 2) (-5, 3) (-5, 2) (-6, 2) (-6, 1) (-6, 0) (-6, 1) (-7, 1) (-7, 0) (-7, -1) (-7, 0) (-7, -1) (-7, 0) (-7, 1) (-7, 0) (-6, 0) (-5, 0) (-6, 0) (-7, 0) (-7, 1) (-8, 1) (-8, 2) (-9, 2) (-9, 1) (-9, 2) (-9, 1) (-9, 2) (-9, 3) (-9, 2) (-10, 2) (-10, 3) (-10, 2) (-11, 2) (-11, 3) (-12, 3) (-11, 3)
4	(0, 0) (-1, 0) (-1, 1) (0, 1) (0, 2) (0, 3) (1, 3) (2, 3) (2, 4) (3, 4) (2, 4) (3, 4) (2, 4) (1, 4) (2, 4) (3, 4) (3, 5)(3, 4) (3, 3) (4, 3) (3, 3) (4, 3) (5, 3) (4, 3) (3, 3) (4, 3) (4, 2) (4, 1) (4, 2) (4, 3) (5, 3) (4, 3) (4, 4) (4, 5) (3, 5) (3, 4) (3, 5) (3, 4) (2, 4) (1, 4) (2, 4) (3, 4) (3, 3) (4, 3) (4, 2) (4, 1) (4, 2) (4, 3) (4, 4) (3, 4) (2, 4) (2, 5) (2, 4) (3, 4) (3, 3) (4, 3) (4, 2) (3, 2) (2, 2) (1, 2) (2, 2) (3, 2) (4, 2) (3, 2) (3, 1) (4, 1) (5, 1) (6, 1) (5, 1) (4, 1) (3, 1) (2, 1) (2, 0) (2, 1) (3, 1) (2, 1) (2, 0) (2, -1) (2, -2) (2, -3) (2, -2) (3, -2) (3, -3) (2, -3) (2,-4) (2, -3) (3, -3) (3, -4) (2, -4) (3, -4) (4, -4) (3, -4) (2, -4) (3, -4) (2, -4) (3, -4) (3, -3) (4, -3) (3, -3)	(0, 0) (-1, 0) (-1, 1) (0, 1) (0, 2) (0, 3) (1, 3) (2, 3) (2, 4) (3, 4) (2, 4) (3, 4) (2, 4) (1, 4) (2, 4) (3, 4) (3, 5) (3, 4) (3, 3) (4, 3) (3, 3) (4, 3) (5, 3) (4, 3) (3, 3) (4, 3) (4, 2) (4, 1) (4, 2) (4, 3) (5, 3) (4, 3) (4, 4) (4, 5) (3, 5) (3, 4) (3, 5) (3, 4) (2, 4) (1, 4) (2, 4) (3, 4) (3, 3) (4, 3) (4, 2) (4, 1) (4, 2) (4, 3) (4, 4) (3, 4) (2, 4) (2, 5) (2, 4) (3, 4) (3, 3) (4, 3) (4, 2) (3, 2) (2, 2) (1, 2) (2, 2) (3, 2) (4, 2) (3, 2) (3, 1) (4, 1) (5, 1) (6, 1) (5, 1) (4, 1) (3, 1) (2, 1) (2, 0) (2, 1) (3, 1) (2, 1) (2, 0) (2, -1) (2, -2) (2, -3) (2, -2) (3, -2) (3, -3) (2, -3) (2, -4) (2, -3) (3, -3) (3, -4) (2, -4) (3, -4) (4, -4) (3, -4) (2, -4) (3, -4) (2, -4) (3, -4) (3, -3) (4, -3) (3, -3)
5	(0, 0) (1, 0) (1, 1) (1, 2) (0, 2) (1, 2) (1, 3) (1, 4) (1, 3) (0, 3) (0, 4) (1, 4) (1, 5) (0, 5) (-1, 5) (-1, 6) (-1, 7) (-2, 7) (-2, 8) (-1, 8) (0, 8) (0, 7) (0, 8) (0, 7) (-1, 7) (-2, 7) (-1, 7) (-1, 8) (-1, 7) (-1, 6) (0, 6) (0, 5) (0,6) (0, 7) (1, 7) (1, 8) (1, 9) (2, 9) (2, 8) (2, 9) (2, 8) (2, 7) (2, 8) (2, 9) (2, 10) (2, 9) (3, 9) (4, 9) (4, 8) (4,7) (4, 6) (5, 6) (6, 6) (5, 6) (6, 6) (6, 7) (5, 7) (5, 8) (5, 9) (5, 10) (5, 11) (4, 11) (5, 11) (5, 10) (4, 10) (4, 11) (5, 11) (4, 11) (3, 11) (3, 10) (3, 11) (2, 11) (2, 10) (2, 11) (1, 11) (0, 11) (0, 10) (1, 10) (2, 10) (1, 10) (2, 10) (1, 10) (2, 10) (2, 9) (1, 9) (1, 8) (1, 9) (1, 10) (0, 10) (0, 9) (-1, 9) (-2, 9) (-3, 9) (-3, 10) (-3, 9) (-4, 9) (4, 10) (-5, 10) (-6, 10)	(0, 0) (1, 0) (1, 1) (1, 2) (0, 2) (1, 2) (1, 3) (1, 4) (1, 3) (0, 3) (0, 4) (1, 4) (1, 5) (0, 5) (-1, 5) (-1, 6) (-1, 7) (-2, 7) (-2, 8) (-1, 8) (0, 8) (0, 7) (0, 8) (0, 7) (-1, 7) (-2, 7) (-1, 7) (-1, 8) (-1, 7) (-1, 6) (0, 6) (0, 5) (0, 6) (0, 7) (1, 7) (1, 8) (1, 9) (2, 9) (2, 8) (2, 9) (2, 8) (2, 7) (2, 8) (2, 9) (2, 10) (2, 9) (3, 9) (4, 9) (4, 8) (4, 7) (4, 6) (5, 6) (6, 6) (5, 6) (6, 6) (6, 7) (5, 7) (5, 8) (5, 9) (5, 10) (5, 11) (4, 11) (5, 11) (5, 10) (4, 10) (4, 11) (5, 11) (4, 11) (3, 11) (3, 10) (3, 11) (2, 11) (2, 10) (2, 11) (1, 11) (0, 11) (0, 10) (1, 10) (2, 10) (1, 10) (2, 10) (1, 10) (2, 10) (2, 9) (1, 9) (1, 8) (1, 9) (1, 10) (0, 10) (0, 9) (-1, 9) (-2, 9) (-3, 9) (-3, 10) (-3, 9) (-4, 9) (-4, 10) (-5, 10) (-6, 10)

-〉 전부 동일함을 확인

· 이전 문장 생성 코드로 생성된 문법이 올바른 99자리 문자열에 잘못된 규칙을 추가한 5종

```
1:bweesesenswnseswwwswnwnwssnwensnswsnseeswnswssesesesswwwensnwsewwbswnsnwsnewensewwnewsnswnwwswsenssn$
-> 임의의 위치에 b를 새로 삽입
2:bneeswwnswwenseenwesnwnwnwnewwwnesnesswsewsswaennewsnwennseseenwsnwnsnnssseeneewwsnewweweewnwesnsse$
-> 임의의 위치에 올바르지 않은문자 a를 새로 삽입
3:wnsnseeessnewnwwnwwssnnnwwnewswwssswwenwnesesensnewwwsseseenssnwnwnwewsnensnsewweewnsnnewenewewwsww$
-> b로 시작하지 않도록 문자열을 조정
4:bewnewnsnwsneenwsswewewnwwwsseenwnwwnenswwsennwnwwnn$enssnnwnsseeenwwnssennnnsewenssneeeswesnesswse$
-> 문자열의 중간에 $를 집어넣어 문장을 강제종료
5:b$seeweseswnnsnsssseswnweeewwsnssesseeensennnewnnewwwsennwssewweewweewwsenssnsswewwwenеneweensswwnes$
-> 문자열의 시작 직후에 $를 집어넣어 문장을 강제종료
```

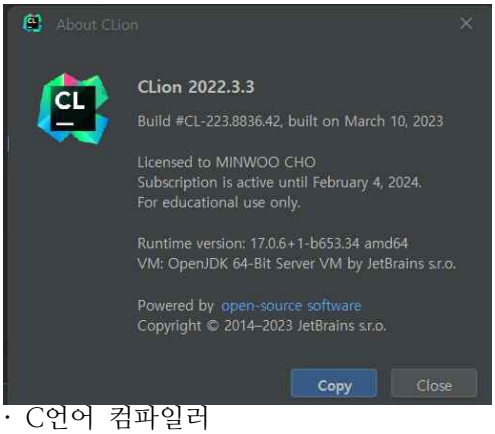
· 각 번호 수에 대해 입력된 결과 값 확인

	제작된 예측 파서(C언어)	검증을 위한 기능상 동일한 프로그램(파이썬)
1	(0, 0) (-1, 0) (0, 0) (1, 0) (1, -1) (2, -1) (2, -2) (3, -2) (3, -1) (3, -2) (2, -2) (2, -1) (2, -2) (3, -2) (3, -3) (2,-3) (1, -3) (0, -3) (0, -4) (-1, -4) (-1, -3) (-2, -3) (-2, -2) (-3, -2) (-3, -3) (-3, -4) (-3, -3) (-4, -3) (-3, -3) (-3, -2) (-3, -3) (-3, -2) (-3, -3) (-4, -3) (-4, -4) (-4, -3) (-4, -4) (-3, -4) (-2, -4) (-2, -5) (-3, -5) (-3, -4) (-3,-5) (-4, -5) (-4, -6) (-4, -7) (-3, -7) (-3, -8) (-2, -8) (-2, -9) (-1, -9) (-1, -10) (-2, -10) (-3, -10) (-4, -10) (-3, -10) (-3, -9) (-3, -10) (-3, -9) (-4, -9) (-4, -10) (-3, -10) (-4, -10) (-5, -10) error	(0, 0) (-1, 0) (0, 0) (1, 0) (1, -1) (2, -1) (2, -2) (3, -2) (3, -1) (3, -2) (2, -2) (2, -1) (2, -2) (3, -2) (3, -3) (2, -3) (1, -3) (0, -3) (0, -4) (-1, -4) (-1, -3) (-2, -3) (-2, -2) (-3, -2) (-3, -3) (-3, -4) (-3, -3) (-4, -3) (-3, -3) (-3, -2) (-3, -3) (-3, -2) (-3, -3) (-4, -3) (-4, -4) (-4, -3) (-4, -4) (-3, -4) (-2, -4) (-2, -5) (-3, -5) (-3, -4) (-3, -5) (-4, -5) (-4, -6) (-4, -7) (-3, -7) (-3, -8) (-2, -8) (-2, -9) (-1, -9) (-1, -10) (-2, -10) (-3, -10) (-4, -10) (-3, -10) (-3, -9) (-3, -10) (-3, -9) (-4, -9) (-4, -10) (-3, -10) (-4, -10) (-5, -10) error
2	(0, 0) (0, 1) (1, 1) (2, 1) (2, 0) (1, 0) (0, 0) (0, 1) (0, 0) (-1, 0) (-2, 0) (-1, 0) (-1, 1) (-1, 0) (0, 0) (1, 0) (1,1) (0, 1) (1, 1) (1, 0) (1, 1) (0, 1) (0, 2) (-1, 2) (-1, 3) (-2, 3) (-2, 4) (-1, 4) (-2, 4) (-3, 4) (-4, 4) (-4, 5) (3, 5) (-3, 4) (-3, 5) (-2, 5) (-2, 4) (-2, 3) (-3, 3) (-3, 2) (-2, 2) (-3, 2) (-3, 1) (-3, 0) (-4, 0) error	(0, 0) (0, 1) (1, 1) (2, 1) (2, 0) (1, 0) (0, 0) (0, 1) (0, 0) (-1, 0) (-2, 0) (-1, 0) (-1, 1) (-1, 0) (0, 0) (1, 0) (1, 1) (0, 1) (1, 1) (1, 0) (1, 1) (0, 1) (0, 2) (-1, 2) (-1, 3) (-2, 3) (-2, 4) (-1, 4) (-2, 4) (-3, 4) (-4, 4) (-4, 5) (-3, 5) (-3, 4) (-3, 5) (-2, 5) (-2, 4) (-2, 3) (-3, 3) (-3, 2) (-2, 2) (-3, 2) (-3, 1) (-3, 0) (-4, 0) error
3	error	error
4	(0, 0) (1, 0) (0, 0) (0, 1) (1, 1) (0, 1) (0, 2) (0, 1) (0, 2) (-1, 2) (-1, 1) (-1, 2) (0, 2) (1, 2) (1, 3) (0, 3) (0, 2) (0, 1) (-1, 1) (0, 1) (-1, 1) (0, 1) (-1, 1) (-1, 2) (-2, 2) (-3, 2) (-4, 2) (-4, 1) (-4, 0) (-3, 0) (-2, 0) (-2, 1) (-3, 1) (-3, 2) (-4, 2) (-5, 2) (-5, 3) (-4, 3) (-4, 4) (-4, 3) (-5, 3) (-6, 3) (-6, 2) (-5, 2) (-5, 3) (-5, 4) (-6, 4) (-6, 5) (-7, 5) (-8, 5) (-8, 6) (-8, 7)	(0, 0) (1, 0) (0, 0) (0, 1) (1, 1) (0, 1) (0, 2) (0, 1) (0, 2) (-1, 2) (-1, 1) (-1, 2) (0, 2) (1, 2) (1, 3) (0, 3) (0, 2) (0, 1) (-1, 1) (0, 1) (-1, 1) (0, 1) (-1, 1) (-1, 2) (-2, 2) (-3, 2) (-4, 2) (-4, 1) (-4, 0) (-3, 0) (-2, 0) (-2, 1) (-3, 1) (-3, 2) (-4, 2) (-5, 2) (-5, 3) (-4, 3) (-4, 4) (-4, 3) (-5, 3) (-6, 3) (-6, 2) (-5, 2) (-5, 3) (-5, 4) (-6, 4) (-6, 5) (-7, 5) (-8, 5) (-8, 6) (-8, 7)
5	(0, 0)	(0, 0)

-> 전부 동일함을 확인

□ 실행환경

· C언어 IDE



· C언어 컴파일러

MinGW w64 9.0

· 파이썬 IDE

· 파이썬 컴파일러

Python 3.11