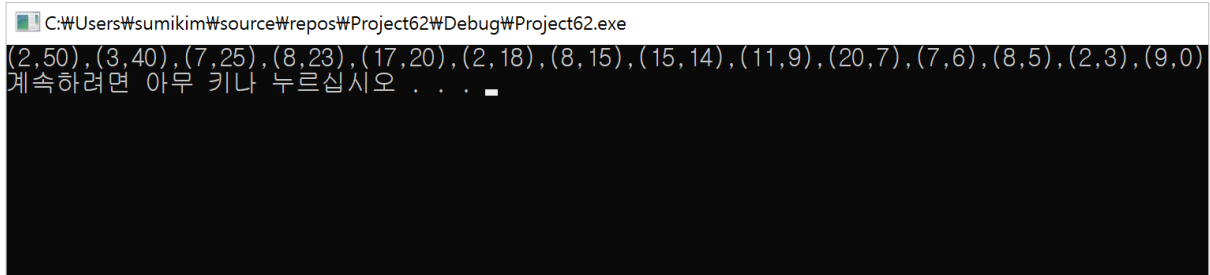


## 오픈랩 과제 #2 : Polynomial

2020.9.29 (화) / 20181202 김수미

### 1) 코드 실행결과



### 2) 코드 내용 설명

```
1  #include<stdio.h>
2  #include<stdlib.h>
3  #define MAX_TERMS 100
4  #define COMPARE(x, y) (((x)<(y)) ? -1 : ((x)==(y)) ? 0 : 1)
5
6  //구조체 polynomial을 선언한다
7  typedef struct {
8      int coef;
9      int expon;
10 } polynomial;
11 polynomial terms[MAX_TERMS];
12 int avail = 0; //값을 추가하기 시작할 수 있는 위치
13
14 //새로운 항(term)을 추가하는 함수 정의
15 void attach(int coefficient, int exponent) {
16     if (avail >= MAX_TERMS) {
17         fprintf(stderr, "다항식에 항이 너무 많습니다.");
18         exit(1);
19     }
20     terms[avail].coef = coefficient;
21     terms[avail++].expon = exponent;
22 }
```

해당 알고리즘은 두 다항식의 합을 계수와 지수를 이용해 계산하고 표현하는 알고리즘이다. 먼저 다항식을 표현하기 위한 구조체 `polynomial` 을 선언해준다. 해당 구조체는 각 항의 `coef`(계수) 정보와 `expon`(지수) 정보를 포함한다. 그 다음 다항식의 항들을 저장하는 `polynomial` 타입 배열 `terms` 를 `MAX_TERMS` 의 크기만큼 선언해준다. `int avail` 은 다항식의 항이 저장되지 않은 배열 `terms`의 index위치를 나타내며, 해당 위치부터 새로운 항의 정보를 저장할 수 있음을 표시하는데 사용된다.

`attach` 함수는 배열 `terms`에 새로운 다항식의 항을 추가하는 함수이다. 다항식  $A(x)$ 와  $B(x)$ 의 합을 계산하고자 할 때, 각 다항식의 항을 저장하고 두 다항식의 합을 계산한 결과의 항들까지 `terms` 배열에 저장하게 되는데, 이 때 이 함수가 사용된다. 만약 배열 `terms` 내부에 다항식  $A(x)$ 와  $B(x)$ 의 합  $D(x)$ 를 수용할 충분한 공간이 없다면 에러 메시지를 출력하고 프로그램을 종료하게 된다.

```

24 int readPoly(int i, FILE* fp) {
25     if (fp == NULL) {
26         fprintf(stderr, "file open error \n");
27         exit(1); }
28     while (!feof(fp)) {
29         fscanf(fp, "%d", &terms[i].coef);
30         fscanf(fp, "%d", &terms[i].expon);
31         i++; }
32     return i;
33 }
34
35 void printPoly() {
36     for (int i = startD; i <= finD; i++) {
37         if (i == finD) { //마지막 항 출력시에는 콤마 출력하지 않는다
38             printf("(%d,%d)\n", terms[i].coef, terms[i].expon);
39             continue;
40         }
41         printf("(%d,%d),", terms[i].coef, terms[i].expon);
42     }
43 }

```

readPoly 함수와 printPoly 함수는 각각 다항식을 txt 파일로부터 입력받아 생성하고 출력하는 역할을 한다.

```

45 //두 다항식의 덧셈을 수행하는 함수 정의 : A(x) + B(x) = D(x)
46 void padd(int startA, int finA, int startB, int finB, int *startD, int *finD) {
47     int coefficient;
48     *startD = avail;
49
50     while (startA <= finA && startB <= finB)
51     {
52         switch (COMPARE(terms[startA].expon, terms[startB].expon)) {
53             case -1: //A_expon < B_expon
54                 attach(terms[startB].coef, terms[startB].expon);
55                 startB++;
56                 break;
57             case 0: //A_expon = B_expon
58                 coefficient = terms[startA].coef + terms[startB].coef;
59                 if (coefficient) attach(coefficient, terms[startA].expon);
60                 startA++; startB++;
61                 break;
62             case 1: //A_expon > B_expon
63                 attach(terms[startA].coef, terms[startA].expon);
64                 startA++;
65                 break;
66         }
67     }
68
69     //A(x)의 나머지 항들을 추가한다
70     for (; startA <= finA; startA++)
71         attach(terms[startA].coef, terms[startA].expon);
72
73     //B(x)의 나머지 항들을 추가한다.
74     for (; startB <= finB; startB++)
75         attach(terms[startB].coef, terms[startB].expon);
76
77     *finD = avail - 1;
78 }

```

padd 함수는 두 다항식의 덧셈을 수행하는 함수이다. 두 다항식 A(x)와 B(x)의 합을 계산한 결과인 D(x)를 저장하며, A(x) 와 B(x), D(x)의 항들은 모두 배열 terms에 저장된다. A(x) 와 B(x)의 항들의 지수를 차례대로

비교해가며 A의 지수가 더 큰 경우, B의 지수가 더 큰 경우, 두 항의 지수가 같은 경우의 세가지 케이스로 분류하여 계산을 수행한다. 항을 비교하는 와중에 두 다항식 중 어느 하나의 항들이 모두 소진되는 경우에는 남은 항들을 새롭게 계산하여 형성되던 다항식  $D(x)$ 의 뒤쪽에 붙이면 된다. 이는 배열 `terms`에 이어서 저장하면 된다.

```
80 int main() {
81     FILE* fp;
82     int i = 0;
83     int A_len, B_len;
84
85     //A.txt를 입력받아 배열 terms에 저장
86     fp = fopen("A.txt", "r");
87     i = readPoly(i, fp);
88     fclose(fp);
89     A_len = i;
90
91     //B.txt를 입력받아 배열 terms에 이어서 저장
92     fp = fopen("B.txt", "r");
93     i = readPoly(i, fp);
94     fclose(fp);
95     B_len = i - A_len;
96
97     //다항식 덧셈 함수 call
98     int startD, finD;
99     avail = A_len + B_len;
100    padd(0, A_len-1, A_len, avail-1, &startD, &finD);
101
102    //덧셈 결과 출력
103    printPoly()
104
105    system("pause");
106    return 0;
107 }
```

위 함수는 `main` 함수로, `readPoly` 함수를 이용해 A.txt 와 B.txt 파일을 입력 받아 다항식 A와 다항식 B의 각 항의 지수와 계수 정보를 배열 `terms`에 저장하고, `padd` 함수를 호출해 두 다항식의 합을 계산 후 `printPoly` 함수로 그 결과를 출력하는 과정을 담고 있다. 각 과정에 대한 설명은 코드에 주석으로 달아 두었다.

이 코드에서 가장 dominant 한 시간복잡도를 가지는 함수 `padd` 의 Worst Case 시간복잡도는 다항식  $A(x)$ 와  $B(x)$ 의 0이 아닌 항의 개수를 각각  $m$ 과  $n$ 이라고 할 때,  $O(m+n)$  이 된다.