Data Structures HW #1

정보대학 컴퓨터학과

2015410090 이소라

1. 개발환경

Windows 64-bit, Microsoft Visual Studio 2017에서 실행하였습니다.

2. 소스코드 설명

가독성을 위해 sublime text editor의 캡처 이미지를 첨부하였습니다.

```
1  #define _CRT_SECURE_NO_WARNINGS
2
3  #include <stdio.h>
4
5
6
7  #define MAX_TERMS 101
8  #define COMPARE(x,y)( ((x) < (y)) ? -1 : ((x) == (y)) ? 0 : 1)
9
10 typedef struct {
11     float coef;
12     int expon;
13 } polynomial;
14 // definition of polynomial
15
16
17 polynomial terms[MAX_TERMS];
18 int avail = 0;      // front of available index in terms
19
20 void attach(float coefficient, int exponent) {
21     if (avail >= MAX_TERMS) {
22         printf("Too many terms in the polynomial\n");
23         return;
24     }
25     terms[avail].coef = coefficient;
26     terms[avail++].expon = exponent;
27 }
28
```

라이브러리는 stdio.h만 사용했습니다. Struct polynomial의 멤버변수로 float coef와 int expon을 선언했고 polynomial struct의 배열 Terms와 avail 변수를 전역변수로 선언했습니다. Attach 메소드의 소스코드는 ppt의 것과 같습니다.

```
29  void padd(int startA, int endA, int startB, int endB, int *startD, int *endD) {
30      float coefficient;
31      *startD = avail;
32      while (startA <= endA && startB <= endB) {                          // we have to see from start to end
33          switch (COMPARE(terms[startA].expon, terms[startB].expon)) {
34          case -1:                                                        // A's degree < B's degree
35              attach(terms[startB].coef, terms[startB].expon);            // bigger degree computaion
36              startB++;                                                   // because we have to see next term of B
37              break;
38          case 0:                                                         // A's degree == B's degree
39              coefficient = terms[startA].coef + terms[startB].coef;      // if A or B's coef is null, coefficient is null
40              if (coefficient)                                            // if coefficient is not null
41                  attach(coefficient, terms[startA].expon);               // sum of coef A and B attach to D
42              startA++;                                                   // because we have to see next term of A
43              startB++;                                                   // because we have to see next term of B
44              break;
45          case 1:                                                         // A's degree > B's degree
46              attach(terms[startA].coef, terms[startA].expon);            // bigger degree computaion
47              startA++;                                                   // because we have to see next term of A
48          }
49      }
50      for (; startA <= endA; startA++) {                                  // if we finish seeing B
51          attach(terms[startA].coef, terms[startA].expon);                // remaining terms of A attach to D
52      }
53      for (; startB <= endB; startB++) {                                  // if se finish seeing A
54          attach(terms[startB].coef, terms[startB].expon);                // remaining terms of B attach to D
55      }
56      *endD = avail - 1;
57
58
59  }
60
```

Padd도 ppt의 것과 같습니다. 주석을 달아 설명을 붙여 놓았습니다.

StartD는 endB의 다음 칸인 현재 avail값을 넣어줍니다.

먼저 startA의 차수가 startB의 차수보다 작은 경우는 startB의 것을 그대로 D에 붙여 넣습니다. startA의 차수와 startB의 차수가 같은 경우는 두 항의 계수를 더해서 D에 붙여 넣습니다. 마지막으로 startA의 차수가 startB의 차수보다 큰 경우는 startA의 것을 그대로 D에 붙여 넣습니다.

붙여 넣은 후에는 그 식의 start값을 증가시켜 다음 항으로 startA가 endA가 되거나 startB가 endB가 될 때까지 같은 일을 반복합니다. 그런 후에 end에 start가 도달하지 못한 식의 남은 항들은 그대로 더해줍니다.

endD는 현재 avail값의 바로 전 칸이므로 avail – 1을 넣어줍니다.

```c
int main() {

    int startA, endA, startB, endB;             // indices in terms
    int startD, endD;                           // index of answer
    char c;                                     // tool of detecting '\n'

    printf("첫번째 다항식을 입력하세요.\n");
    startA = 0;                                 // A starts at 0
    endA = startA;                              // At first, endA is same with startA

    while(1) {

        scanf("%f", &terms[avail].coef);
        scanf("%d", &terms[avail].expon);
        endA++;                                 // if input is not null, endA becomes endA + 1
        avail++;                                // because avail slot is full, avail has to be added by one
        if ( (c = fgetc(stdin)) == '\n') {      // if input is '\n', we have to stop while loof
            break;
        }
    }
    endA--;                                     // at last, endA has to be subtracted by 1

    printf("두번째 다항식을 입력하세요.\n");

    startB = avail;                             // B starts at avail
    endB = startB;                              // At first, endB is same with startB

    while (1) {
        scanf("%f", &terms[avail].coef);
        scanf("%d", &terms[avail].expon);
        endB++;
        avail++;
        if ((c = fgetc(stdin)) == '\n') {
            break;
        }
    }
    endB--;

```

Main 함수를 살펴보겠습니다.

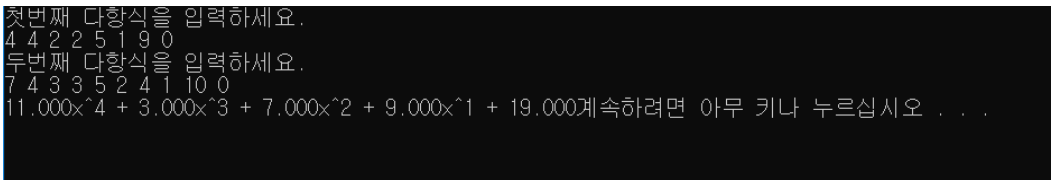
첫번째 다항식은 startA가 0입니다. endA는 startA부터 시작합니다. 입력 받으며 endA를 하나씩 늘려주고 사용가능한 가장 왼쪽 칸인 avail도 함께 늘려줍니다. 반복해서 입력 받다가 '₩n'이 나오면 while문을 종료합니다. 그리고 루프의 마지막에 avail과 함께 늘어난 endA에서 1을 빼줍니다.

두번째 다항식은 startB가 현재 avail입니다. endB도 startB부터 시작합니다. 입력은 첫번째 다항식과 동일합니다.

```
101     padd(startA, endA, startB, endB, &startD, &endD);
102
103
104
105
106
107     int indexD = startD;
108
109     while (1) {                                          // print D
110         if (terms[indexD].expon != 0)
111             printf("%.3fx^%d", terms[indexD].coef, terms[indexD].expon);
112         else if (terms[indexD].expon == 0)
113             printf("%.3f", terms[indexD].coef);
114
115         if (indexD == endD) {                            // end of print at endD
116             break;
117         }
118         indexD++;
119         printf(" + ");
120     }
121
122     system("pause");
123
124 }
```

위에서 선언해준 padd 함수에 파라미터를 넣어 실행합니다. indexD라는 변수를 만들어 startD 값을 대입해줍니다. 이 값을 1씩 늘려 startD부터 endD까지의 항을 출력합니다. endD에 도달한 경우 while문을 종료합니다.

3. 결과 출력



테스트케이스를 출력한 화면입니다.

4. 문제 해결

1) Expon이 0이 아닌 항으로 입력이 끝나는 경우

➔ 입력 버퍼 stdin에 들은 '₩n'을 감지했을 때 입력 while문을 종료하도록 했습니다.

2) endA와 endB의 값이 avail과 함께 움직이게 되어 while 문을 빠져나왔을 때 빈 칸을 가리킴

➔ while문을 빠져나온 직후 1씩 감소시켜 주었습니다.