

# Polynomial Array

중앙대학교 소프트웨어학과 20186889 권용한

## ●소스코드 설명

```
void load_term(polynomial_term A, int n) //항 배열 방식으로 작성된 A(x)를 Terms 변수로 옮기기 위해 작성함
{
    for (int i = 0; i < n; i++)
    {
        Terms[i].exp = A.term[i].exp;
        Terms[i].coef = A.term[i].coef;
        avail++; //Terms에 값이 저장되면 전역변수 avail도 그에 따라서 숫자가 증가해야한다. 따라서 값을 저장한 이후 avail도 값을 증가시켜줌
    }
};
```

load term 함수로 항배열 방식으로 작성된 A(x)를 Terms에 저장하기 위한 함수이다. Terms에 A(x)와 B(x)를 저장하여 이를 padd함수를 통해 계산하기 위해 작성

Terms에 요소들을 저장하기 때문에 avail번째 요소들에 값이 저장되고 있으므로 값이 저장될 때 마다 avail을 추가해준다.

```
void load_coefficient(polynomial_coefficient B, int n) //계수 배열 방식으로 작성된 B(x)를 항배열 방식으로 작성된 Terms에 옮기고 그 방식도 계수배열에서 항배열 방식으로 변환하여 저장함.
{
    for (int i = 0; i < B.degree+1; i++)
    {
        if (B.coef[i] != 0) //항배열방식으로 Terms에 저장하기 위해 계수배열방식에서 0으로 저장된 부분을 제외하기 위해 씬
        {
            Terms[n].coef = B.coef[i];
            Terms[n].exp = B.degree - i; //계수배열방식에서는 항배열방식과는 달리 가장 높은 차수의 지수만 저장하기 때문에 저장해야할 항이 동일한다면 그 항의 지수를 계산하기 위해 가장 높은 차수의 지수 - 반복횟수
            avail++; //Terms에 값이 저장되면 전역변수 avail도 그에 따라서 숫자가 증가해야한다. 따라서 값을 저장한 이후 avail도 값을 증가시켜줌
        }
    }
};
```

load coefficient함수로 계수배열 방식으로 작성된 B(x)를 Terms에 저장하기 위한 함수

Terms가 항배열 방식으로 작성되었기 때문에 계수배열 방식으로 작성된 B(x)에서 항배열 방식에 필요한 요소들을 구해내는 과정이 포함되어있다.

항배열방식의 경우 계수가 0이 아닌 경우 그 계수와 지수를 저장하지 않기 때문에 if(B.coef[i]!=0) 이 문장을 통해 0이 아닌 값만을 선택한다. 0인 요소들을 Terms에 저장하지 않기 때문에 isZero(b)함수나 LeapExp(b),Coef(),Remove()함수의 경우 만일 원래 계수배열방식으로 작성된 B(x)였다면 별도의 계산 과정이나 함수가 필요로 되었겠지만, B(x)의 값들을 Terms에 저장한 나의 코드 같은 경우 별도의 계산과정이나 함수 없이 항배열 방식의 계산 과정과 동일한 방법으로 요소들을 구할 수 있다.

항배열방식의 경우 가장 높은 차수의 지수만 저장하고 계수가 0인 부분도 저장하여 배열의 몇 번째 요소인지 안다면 그때의 지수를 알 수 있지만 이를 항배열방식에 저장하기 위한 숫자정보가 구조체에 존재하지 않

는다. 따라서 계수가 0이 아닌 지점의 지수를 구하기 위해선 그때의 지수값을 계산할 수 밖에 없는데, 그 값이 가장높은 차수-반복횟수로 계산되기 때문에 그 값을 Terms의 exp요소에 저장한다.

Terms에 요소들을 저장하기 때문에 avail번째 요소들에 값이 저장되고 있으므로 값이 저장될 때 마다 avail을 추가해준다.

```
void attach(float coefficient, int exponent)//새로운 항을 추가하는 함수
{
    if (avail >= MAX_TERMS) { //Terms가 정적으로 할당되었기 때문에 기준치 이상으로 메모리를 요구한다면 이를 거부
        printf("다항식에 항이 너무 많다.");
        exit(1);
    }
    Terms[avail].coef = coefficient;
    Terms[avail++].exp = exponent; //값이 저장된 이후에 그 다음 Terms 요소에 값을 저장해야하므로 avail값을 증가시킴
};
```

attach함수로 padd함수에서 계산되거나 그대로 넘어가는 경우 사용하는 함수이다. Terms함수 중 비어있는 공간 중 가장 앞에 있는 요소, 즉 avail번째 요소에 입력받은 coefficient와 exponent를 저장하고 함수 끝에 avail을 추가하여 그 이전의 avail번째 요소가 입력되어 더 이상 avail하지 않음을 나타낸다.

```
void padd(int startA, int finishA, int startB, int finishB, int *startD, int *finishD)
{
    int coefficient; //지수가 같을 경우 그 값을 더하여 저장할 변수 선언
    *startD = avail;
    while (startA <= finishA && startB <= finishB) {
        switch (COMPARE(Terms[startA].exp, Terms[startB].exp)) {
            case -1: //B식의 지수가 클 경우
                attach(Terms[startB].coef, Terms[startB].exp); //B식의 지수와 계수를 Terms의 avail번째에 저장
                startB++;
                break;
            case 0: //두 식의 지수가 같을 경우
                coefficient = Terms[startA].coef + Terms[startB].coef;
                if (coefficient) {
                    attach(coefficient, Terms[startA].exp);
                }
                startA++;
                startB++;
                break;
            case 1: //A식의 지수가 클 경우
                attach(Terms[startA].coef, Terms[startA].exp); //A식의 지수와 계수를 Terms의 avail번째에 저장
                startA++;
        }
    }
    for (; startA <= finishA; startA++) //while문 이후 A식에 남은 값이 있을 경우 남은 값들을 Terms에 저장
        attach(Terms[startA].coef, Terms[startA].exp);
    for (; startB <= finishB; startB++) //while문 이후 B식에 남은 값이 있을 경우 남은 값들을 Terms에 저장
        attach(Terms[startB].coef, Terms[startB].exp);
    *finishD = avail;
};
```

padd함수로 Terms에 저장된 A(x)와 B(x) 다항식을 더하는 함수이다.

!IsZero(a)함수의 경우 while(startA<=finishA&&startB<=finishB)에서 startA<=finishA로 작성되었으며, LeadExp(a)같은 경우 Terms[startA].exp로 대체하였다. Coef(a,LeadExp(a))는 Terms[startA].coef로, Remove(a,LeadExp(a))는 startA++;문장으로 하였다. B(x)함수의 경우도 Terms에 항배열 방식으로 저장해 두었기 때문에 같은 방식으로 계산을 할 수 있다.

```

printf("Enter A(x):\n");
printf("계수가 0이 아닌 항의 수?:");
scanf_s("%d", &routine_num_term);
if (routine_num_term == 0)
{
    return 0; //A(x)에서 0이 아닌 항의 갯수가 0이면 프로그램을 종료
}
for (int i = 0; i < routine_num_term; i++)
{
    printf("지수?:");
    scanf_s("%d", &A.term[i].exp);
    printf("계수?:");
    scanf_s("%d", &A.term[i].coef);
}

```

A(x)함수 입력 과정으로 routine\_num\_term 변수에 반복해야 할 횟수를 저장하고 for문을 통해서 지수와 계수를 입력받을 수 있다.

pdf에 적혀있는 5. 이상의 1~4 과정을 앞서와 다른 A(x)와 B(x)에 대해 반복하되, A(x)의 입력 과정에서 A(x)의 항의 개수로 0이 입력되면 프로그램을 종료한다. 이 문장에 따라, 계수가 0이 아닌 항의 개수가 0으로 입력된다면 프로그램을 종료하도록 return 0; 하도록 한다.

```

printf("Enter B(x):\n");
printf("계수가 0이 아닌 항의 수?:");
scanf_s("%d", &routine_num_coefficient);
for (int i = 0; i < routine_num_coefficient; i++)
{
    printf("지수?:");
    scanf_s("%d", &temperate_degree); //계수배열방식

    if (i == 0) //내림차순으로 입력된다면 가장 먼저 입력되는 지수가 가장 높은 차수의 지수이므로 가장 처음의 경우 선택
    {
        B.degree = temperate_degree; //가장 먼저 입력되는 지수가 가장 높은 차수의 지수이므로 그 지수를 저장
        B.coef = (int*)malloc((B.degree + 1) * sizeof(int)); //입력된 가장 큰 지수+1만큼 B(x)의 계수를 저장할 메모리 동적할당
    }
    else
    {
        int sub_degree = pre_temperate_degree - temperate_degree - 1; //계수배열방식에서 전의 지수와 그 다음 지수의 차이만큼 나머지 공간에 0이 저장되어야하므로 그 차이를 계산
        if (sub_degree != 0) {
            for (int j = 0; j < sub_degree; j++) //나머지 공간에 0을 저장
            {
                B.coef[j] = 0;
            }
            n++; //B.coef의 n번째 요소와 그 이후 요소들에 접근하기 위해 n의 값을 증가시킴
        }
    }
    printf("계수?:");
    scanf_s("%d", &B.coef[n]);

    pre_temperate_degree = temperate_degree;
    n++;

    if (i == routine_num_coefficient - 1) //마지막 항의 지수가 0이 아니라면 그 이후부분에 0을 저장하지 않고 프로그램이 마무리되므로 이를 해결하기 위해 작성함
    {
        for (; n <= B.degree; )
        {
            B.coef[n] = 0; //나머지 부분에 0으로 초기화
            n++;
        }
    }
}

```

B(x)함수를 입력받는 과정으로 routine\_num\_coefficient 변수에 반복해야 할 횟수를 저장하여 for문을 작성하였다.

B(x)의 경우 지수를 입력받고 그 이후에 B(x)에 필요한 만큼(B.degree+1) 동적할당을 하였기 때문에, 동적할당을 하기 전에는 B.coef에 어떠한 주소도 입력되어있지 않아서 B.coef[i]에 값을 저장할 수 없다. 그래서 불

가피하게 pdf에 적힌 예시와는 반대로 지수부터 입력을 받고 그 이후에 계수를 입력받는 방식으로 코드를 작성하였다.

또한 for문을 지수+1만큼 반복하지 않고, routine\_num\_coefficient에 저장된 만큼만 반복하였기 때문에 B.coef에 값을 따로 입력받지 않는 요소들의 경우 초기화가 되어있지 않다. 이 부분을 초기화 하기 위해선 초기화가 필요한 부분의 크기나 정확한 번지수를 알아야 하는데 이를 위해서 이전에 입력받은 지수를 저장하는 pre\_temperate\_degree, 그리고 현재 입력된 지수를 저장하는 temperate\_degree, 그리고 이 둘의 차이를 저장하는 sub\_degree변수를 이용하여 지수를 입력받으면 sub\_degree를 계산하여 그 나머지 부분을 0으로 초기화시킨 후 계수를 입력받아 b.coef[]를 저장하였다. 또한 여기까지만 코드를 작성한다면 만일 지수가 0인 지점에 계수가 저장되는 것이 아니라면 마지막 요소에 값이 저장된 후, 그 다음 요소들 부터는 값이 초기화 되지 않는 문제가 발생한다. 때문에 이러한 문제를 해결하고자 if(i==routine\_num\_coefficient-1){ for{~}~ }부분을 작성하여 그 뒷부분도 마저 0으로 초기화를 시킨다.

```
for (int i = startD; i < finishD; i++)//D(x)를 출력
{
    printf("D[%d].coef = %d ", i - routine_num_coefficient - routine_num_term, Terms[i].coef);
    printf("D[%d].exp = %d\n", i-routine_num_coefficient-routine_num_term, Terms[i].exp);
}
```

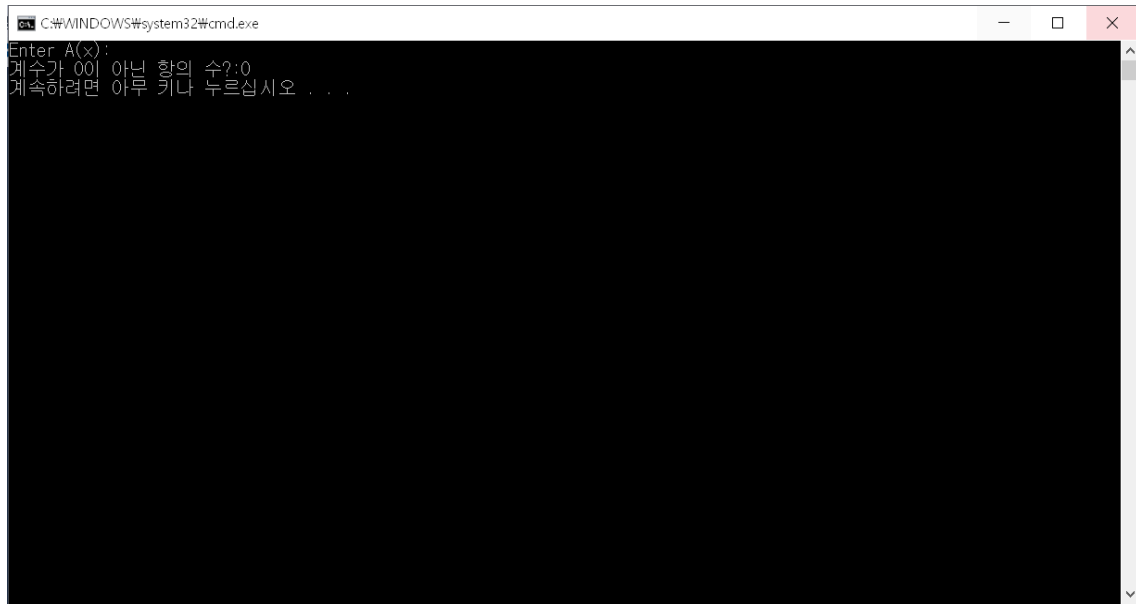
D(x)를 출력하는 부분으로 Terms에 입력된 부분중 padd함수와 그에 속한 attach함수를 통해 계산되고 추가된 부분의 시작점을 startD와 끝점을 finishD로 저장해 두어 그 범위 만큼을 출력한다. 이때 startD와 finishD는 Terms에서 추가된 부분이기 때문에 첫 지점의 값이 0이 아니다. 이를 출력할 때 D의 요소로 보이도록 하기 위해서는 그 이전의 값들을 빼면 되는데, 그 값은 A(x)와 B(x)중 계수가 0이 아닌 항의 개수의 합이기 때문에 i(startA)-routine\_num\_coefficient(B(x)중 계수가 0이 아닌 항의 개수)-routine\_num\_term(A(x)중 계수가 0이 아닌 항의 개수)를 통해 D의 몇 번째 요소인지 계산할 수 있다.

## ●출력 결과

```
C:\WINDOWS\system32\cmd.exe
Enter A(x):
계수가 0이 아닌 항의 수?:3
지수?:500
계수?:2
지수?:20
계수?:-7
지수?:0
계수?:19
Enter B(x):
계수가 0이 아닌 항의 수?:2
지수?:10
계수?:45
지수?:5
계수?:6
A[0].coef=2    A[0].exp=500
A[1].coef=-7   A[1].exp=20
A[2].coef=19   A[2].exp=0
B.degree=10
B.coef[0] = 45
B.coef[5] = 6
D[0].coef = 2   D[0].exp = 500
D[1].coef = -7  D[1].exp = 20
D[2].coef = 45  D[2].exp = 10
D[3].coef = 6   D[3].exp = 5
D[4].coef = 19  D[4].exp = 0
```

pdf 파일에 있는 예시를 입력하였을 때 출력되는 값

A(x)와 D(x)의 경우 항배열 방식으로 작성되었기 때문에 계수와 지수가 동시에 표현되도록 출력되고, B(x)의 경우 계수배열 방식으로 작성되었기 때문에 가장 높은 차수의 지수인 B.degree가 출력되고, 그 이후 항들을 출력되도록 하였다. 이때 계수가 0인 지점을 출력하지 않기 때문에 계수가 0이 아닌 것만 출력되고, 그때의 지수는 B.coef의 몇 번째 요소인지와 B.degree를 이용하여 계산할 수 있다. 이때의 경우 B.degree=10이고, B.coef[0]=45 이기 때문에 이때의 지수는 10이고 계수가 45, B.coef[5]=6의 경우 지수는 5(10-5), 계수는 6임을 계산할 수 있다.



```
C:\WINDOWS\system32\cmd.exe
Enter A(x):
계수가 0이 아닌 항의 수? 0
계속하려면 아무 키나 누르십시오...
```

A(x)에 계수가 0이 아닌 항의 수에 0이 입력되면 프로그램을 종료함

(5. 이상의 1~4 과정을 앞서와 다른 A(x)와 B(x)에 대해 반복하되, A(x)의 입력 과정에서 A(x)의 항의 개수로 0이 입력되면 프로그램을 종료한다.) 이 문장을 적용하여 0이 입력되면 프로그램이 종료되도록함.