

Course: 자료구조 (5118006)

Professor: 이의종 (email: kongjjagae@cbnu.ac.kr)

TA: 서한빛 (email: hb.seo-isw@cbnu.ac.kr)

< Programming Assignment #1 >

3 April 2024

Due Date: 17 April 2024, 11:59 pm

1. 목표: 배열과 연결리스트를 이용하여 다항식의 생성, 출력, 덧셈 기능을 구현

- 배열: 초기 버전과 개선된 버전 모두 구현

- 초기버전 (3. 배열과 구조 p35-36페이지 참조)

$A = (n, \underline{a_n}, \underline{a_{n-1}}, \dots, \underline{a_1}, \underline{a_0})$	n: degree of A a: n+1 coefficients
예) $A(x) = x^4 + 10x^3 + 3x^2 + 1$: n = 4
$A = (4, 1, 10, 3, 0, 1)$: 6 elements

```
/* d = a + b, 여기서 a, b, d는 다항식이다. */
d = Zero();
while (! IsZero(a) && ! IsZero(b)) do {
    switch COMPARE(Lead_Exp(a), Lead_Exp(b)) {
        case -1:
            d = Attach(d, Coef(b, Lead_Exp(b)), Lead_Exp(b));
            b = Remove(b, Lead_Exp(b));
            break;
        case 0:
            sum = Coef(a, Lead_Exp(a)) + Coef(b, Lead_Exp(b));
            if (sum) {
                Attach(d, sum, Lead_Exp(a));
                a = Remove(a, Lead_Exp(a));
                b = Remove(b, Lead_Exp(b));
            }
            break;
        case 1:
            d = Attach(d, Coef(a, Lead_Exp(a)), Lead_Exp(a));
            a = Remove(a, Lead_Exp(a));
    }
}
a 또는 b의 나머지 항을 d에 삽입한다.
```

■ 개선된버전 (3. 배열과 구조 p37-39페이지 참조)

$$A(x) = 2x^{1000} + 1$$

$$B(x) = x^4 + 10x^3 + 3x^2 + 1$$

계수(coef)가 대부분 0인 경우 메모리 절약

	<i>startA</i>	<i>finishA</i>	<i>startB</i>		<i>finishB</i>	<i>avail</i>
<i>coef</i>	2	1	1	10	3	1
<i>exp</i>	1000	0	4	3	2	0
	0	1	2	3	4	5

```
void padd(int starta, int finisha, int startb, int finishb, int *startd,
int *finishd)
{ /* A(x)와 B(x)를 더하여 D(x)를 생성한다. */
  float coefficient;
  *startd = avail;
  while (starta <= finisha && startb <= finishb)
    switch (COMPARE(terms[starta].expon, terms[startb].expon))
    {
      case -1: /* a의 expon이 b의 expon보다 작은 경우 */
        attach(terms[startb].coef, terms[startb].expon);
        startb++; break;
      case 0: /* 지수가 같은 경우 */
        coefficient = terms[starta].coef + terms[startb].coef;
        if(coefficient) attach(coefficient, terms[starta].expon);
        starta++; startb++; break;
      case 1: /* a의 expon이 b의 expon보다 큰 경우 */
        attach(terms[starta].coef, terms[starta].expon);
        starta++; }
  /* 각 다항식의 exp, coef 끝까지 */
}
```

```
/* A(x)의 나머지 항들을 첨가한다. */
for(; starta <= finisha; starta++)
  attach(terms[starta].coef, terms[starta].expon);
/* B(x)의 나머지 항들을 첨가한다. */
for(; startb <= finishb; startb++)
  attach(terms[startb].coef, terms[startb].expon);
*finishd = avail - 1;
```

< 두 다항식을 더하는 함수 >

```
void attach(float coefficient, int exponent)
{
  /* 새 항을 다항식에 첨가한다. */
  if (avail >= MAX_TERMS) {
    fprintf(stderr, "다항식에 항이 너무 많다.");
    exit(1);
  }
  terms[avail].coef = coefficient;
  terms[avail++].expon = exponent;
}
```

< 새로운 항을 첨가하는 함수 >

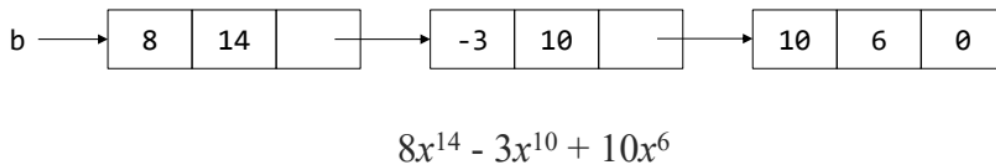
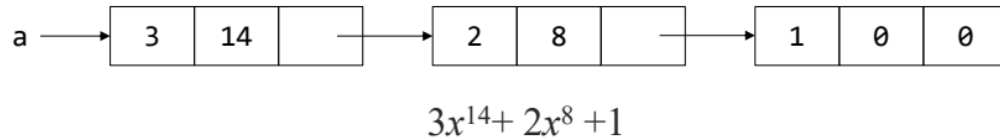
● 연결리스트: (5. 연결리스트 p31-36 참조)

- $a = 3x^{14} + 2x^8 + 1$

- $b = 8x^{14} - 3x^{10} + 10x^6$

coef	expon	link
------	-------	------

polyNode



```

polyPointer padd(polyPointer a, polyPointer b) /* a와 b가 합산된 다항식을 반환 */
{
    polyPointer c, rear, temp;
    int sum;
    MALLOC(rear, sizeof(*rear)); /* node를 연결하기위한 임시 node 생성 */
    c = rear;
    while(a && b)
        switch(COMPARE(a->expon, b->expon)) {
            case -1: /* a->expon < b->expon */
                attach(b->coef, b->expon, &rear);
                b = b->link;
                break;
            case 0: /* a->expon = b->expon */
                sum = a->coef + b->coef;
                if(sum) attach(sum, a->expon, &rear);
                a = a->link; b = b->link; break;
            case 1: /* a->expon > b->expon */
                attach(a->coef, a->expon, &rear);
                a = a->link;
        }
    /* 리스트 a와 리스트 b의 나머지를 복사 */
    for(; a; a=a->link) attach(a->coef, a->expon, &rear);
    for(; b; b=b->link) attach(b->coef, b->expon, &rear);
    rear->link = NULL;
    /* 필요 없는 초기 노드를 삭제 */
    temp = c; c = c->link; free(temp); /* node를 연결하기위한 임시 node 제거 */
    return c;
}
for loop 전에 if a == NULL or b == NULL
검사 하지 않음 (code simplicity)

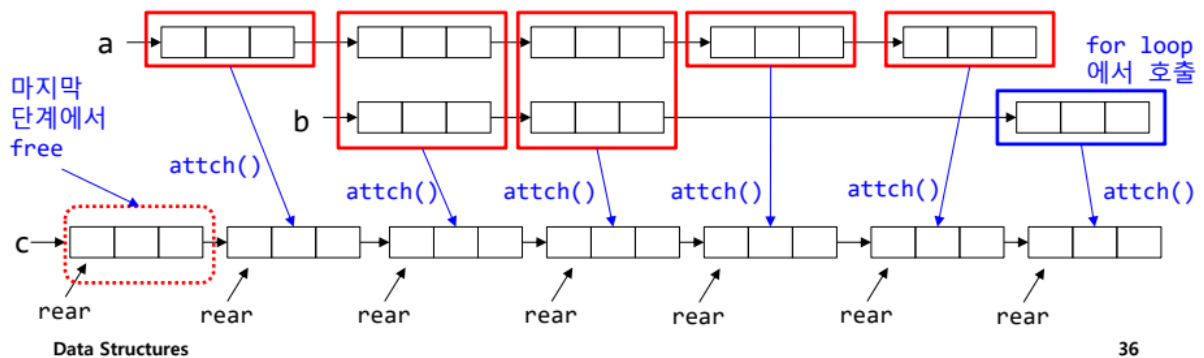
```

```

void attach(float coefficient, int exponent, polyPointer *ptr)
{ /* coef=coefficient 이고 expon=exponent인 새로운 노드를 생성하고,
   그것을 ptr에 의해 참조되는 노드에 첨가한다. ptr을 갱신하여
   이 새로운 노드를 참조하도록 한다. */

    polyPointer temp;
    MALLOC(temp, sizeof(*temp));
    temp->coef = coefficient;
    temp->expon = exponent;
    (*ptr)->link = temp;
    *ptr = temp;
}

```



36

3. 요구사항

코드는 다음과 같은 요구사항을 만족해야 함

- Execution file name: poly.exe
- Execute the program with three arguments: input file name, output file name

■ Example:

./poly.exe input.txt output.txt

- Please ensure that the executable file, input.txt, and output.txt are in the same folder.

If they are in different folders (such as input.txt being in a separate data folder), you will not receive a score.

- Input file format (.txt)

[number_of_exponents]Wt[number_of_exponents]Wn

[coef]Wt[exp]Wn

[coef]Wt[exp]Wn

[coef]Wt[exp]Wn

[coef]Wt[exp]Wn

[coef]Wt[exp]Wn

...

가장 첫줄은 두 다항식의 지수(exponents)의 수를 의미

두번째 줄부터는 다항식의 계수와 지수를 의미

지수는 정렬이 안되어 있음 → 정렬 필수!

■ Example: $f_1(x) = 2x^4 + 1x$, $f_2(x) = 4x^3 + 3x^2 + 1x$

2 3

2 4

1 2

4 3

3 2

1 1

- Output file format (.txt)

[poly1]Wn

[poly2]Wn

[poly1] + [poly2]Wn

- 수행시간1Wt수행시간2Wt수행시간Wn

[poly]내 x의 계수는 x^{power} 로 표현, + 앞뒤에 space로 공백줌

- Example: $f_1(x) = 2x^4 + 1x$, $f_2(x) = 4x^3 + 3x^2 + 1x$

$2x^4 + 1x^1$ Wn

$4x^3 + 3x^2 + 1x^1$ Wn

$2x^4 + 4x^3 + 2x^1$ Wn

1Wt2Wt3Wn

- **Note: Please make sure to match the output format!**
If the format is not correct, you can't get any score.

4. Submission

- **eCampus** 를 통해 제출
 - 보고서
 - Pdf 포맷으로 제출할 것
 - 가이드라인
 - ✓ 코드에 대한 설명
 - ✓ TA의 컴퓨터에서 실행하기 위한 방법 (컴파일러 버전 등등)
(e.g., screenshot) (*Important!!*)
 - 코드
 - 실행파일 (.exe)
 - 소스코드 (.c)

5. Penalty

- Late submission
 - 1-day delay: 75%
 - 2-days delay: 50%
 - Delay more than 3-days: 0%
- Requirements unsatisfied
 - Significant penalty up to 30% will be given when the requirements are not satisfied