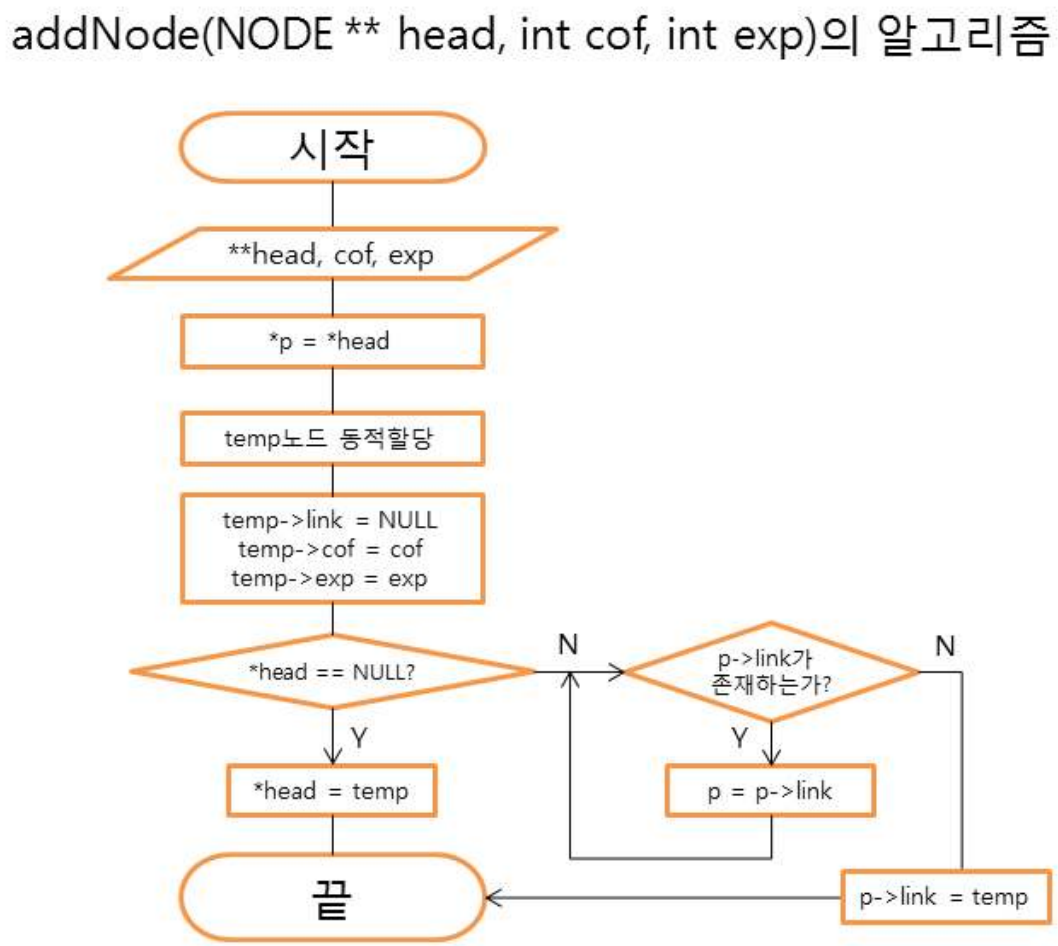
4. 연결리스트를 이용한 다항식 구현

21410785 황 희

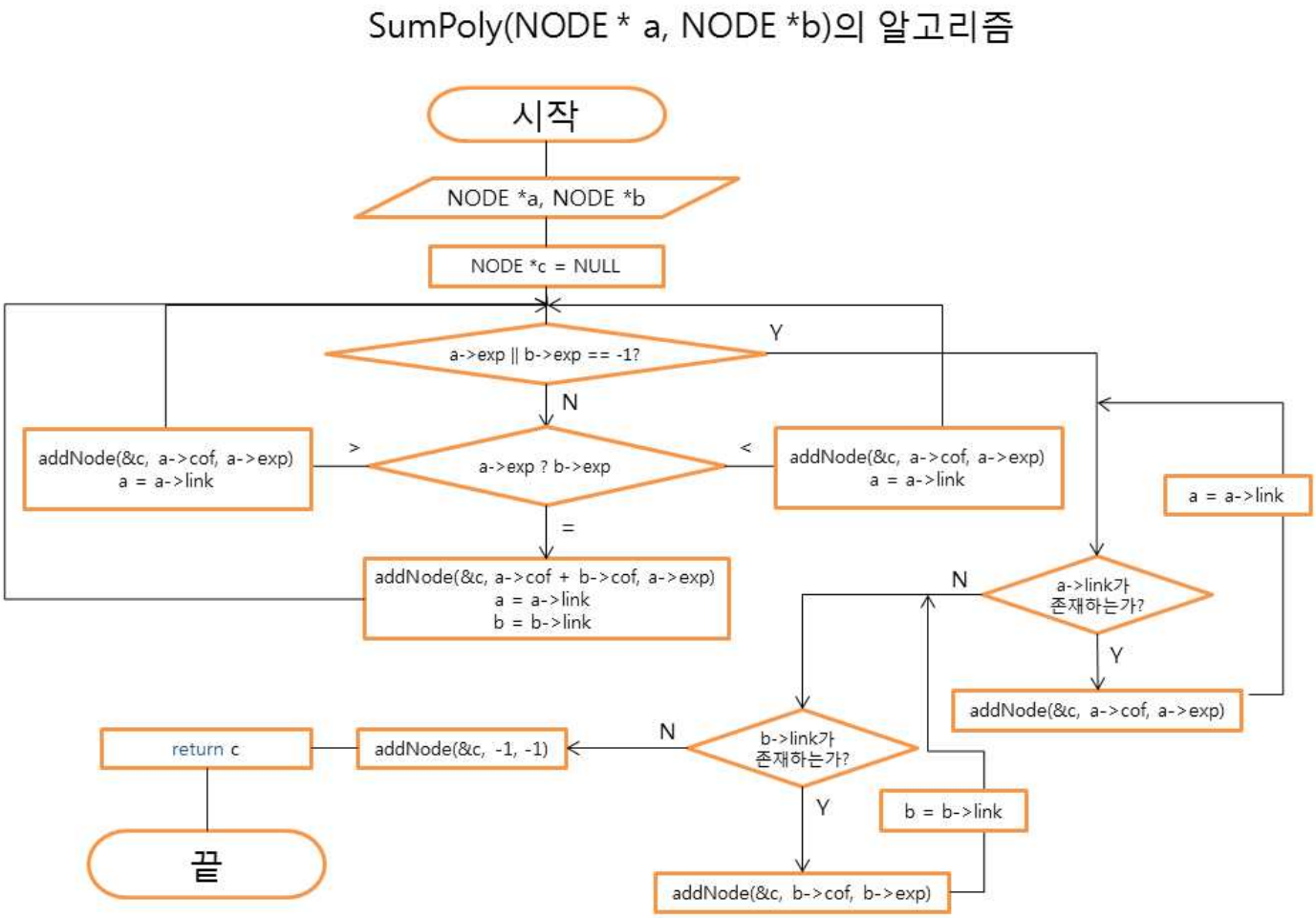
1. 개요

- addNode(NODE **head, int cof, int exp)의 알고리즘



addNode함수는 현재 구현되어있는 노드의 제일 끝자리에 노드를 추가해주는 함수이다. 먼저, 매개변수로 노드를 추가할 리스트의 주소(&head)와 추가할 지수와 계수를 입력받는다. 그다음 리스트의 현재 위치를 알려줄 노드포인터 p를 설정하고 head의 위치를 *p에 대입해준다 그다음 임시노드 temp를 동적으로 생성하고, 지수와 계수, 링크를 입력하여 추가될 노드를 생성한다. 만약 리스트의 머리부분이 NULL값이라면 리스트가 구현되어있지 않다는 뜻이므로 temp를 리스트의 제일 처음 노드로 지정해주는 작업을 하고 끝낸다. 만약 리스트가 구현되어있는 상태라면 리스트의 현재 위치를 가리키는 p의 다음노드가 존재하는지 알아보고 존재한다면 p가 다음노드를 가리키도록 p를 p->link로 바꿔준다 이 작업을 계속 반복하다가 p가 더 이상 p->link가 존재하지 않는 마지막 노드에 도착한다면 temp를 마지막노드에 연결시켜주고 함수를 종료한다.

- sumPoly(NODE *a, NODE * b)의 알고리즘



sumPoly함수는 다항식 a과 다항식 b의 합을 반환해주는 함수이다. 먼저 a다항식의 주소와 b노드의 주소를 받는다. 그리고 합을 저장할 노드의 포인터 c를 NULL로 정의, 선언해준다. 그리고 a나 b가 가리키는 노드의 지수가 -1인지 확인 후 -1이 아니라면 a가 가리키는 노드의 지수와 b가 가리키는 노드의 지수의 크기를 비교한다. 그래서 지수가 더 높은 노드부터 c에 addNode해주고, 만약 a와 b의 지수가 같다면 두 계수의 합과 a의 지수를 c에 addNode해준다. 이를 a나 b의 지수가 -1이 아닐 때까지 반복하다가 a혹은 b에서 지수가 -1이 나온다면 a혹은 b에 남은 식이 있을지도 모르니 a와 b의 남은 노드를 모드 c에 addNode해준다. 그다음 마지막 노드임을 표시해주는 지수와 계수가 -1인 노드를 c의 마지막에 추가해준다. 마지막으로, 다항식이 합해진 연결리스트의 첫 번째 노드를 가리키는 노드포인터 c를 반환해 줌으로 함수를 끝낸다.

2. 본문

- 코드(설명은 주석으로 대체)

```
#include<stdio.h>
#include<stdlib.h>
typedef struct node // 연결리스트를 구현할 노드
{
	int cof; // 계수
	int exp; // 지수
	struct node * link; // 다음 노드를 가리킬 노드포인터
}NODE;

void addNode(NODE ** head, int cof, int exp) // 노드를 추가하는 함수
{
	NODE * p = *head; // head의 주소값을 p에 저장
	NODE * temp = (NODE*)malloc(sizeof(NODE));
	// 연결리스트 끝에 연결시킬 temp노드 생성
	temp->link = NULL; // temp노드가 리스트의 제일 끝임을 알려줌
	temp->cof = cof; // temp노드의 계수를 입력받은 계수로 지정
	temp->exp = exp; // temp노드의 지수를 입력받은 지수로 지정
	if (*head == NULL) // 노드가 하나도 없다면 (제일 처음노드생성)
	{
		*head = temp; // head가 가리키는 값을 temp로 지정
		return; // 끝
	}
	// 노드가 이미 있다면
	while (p->link) p = p->link;
	// p가 가리키는 노드가 있다면 가리키는 노드가 없을때까지 노드를 이동
	p->link = temp; // 마지막 노드 p가 temp를 가리키도록 설정
}

void showList(NODE * head) // 다항식을 보여주는 함수
{
	NODE * p = head; // head의 주소값을 p에 대입
	while (p) // p가 존재하는 동안(연결 리스트가 존재하는동안)
	{
		if (p->exp == -1) // p가 가르치는 노드의 지수가 -1이면
			break; // 빠져나감
		printf("%dx^%d%s", p->cof, p->exp, p->link ? "+" : ""); // 3x^2->2x^1이런 형식으로 출력
		p = p->link; // p를 다음노드로 이동
	}
}

NODE * SumPoly(NODE * a, NODE * b) // 다항식 a와 b를 합하는 함수
{ // 매개변수 : a의 head, b의 head
	NODE * c = NULL; // 노드 포인터 c생성

	while (a->exp != -1 && b->exp != -1)
	{// a의 지수와 b의 지수가 -1이 아닌동안
		if (a->exp > b->exp) // a의 지수가 b의 지수보다 크면
		{
```

```
                        addNode(&c, a->cof, a->exp);
                        // c의 뒤에 a의 지수와 계수로 이루어진 노드 추가
                        a = a->link; // a를 다음노드로 이동
                }
                else if (a->exp < b->exp) // b의 지수가 a의 지수보다 크면
                {
                        addNode(&c, b->cof, b->exp);
                        // c의 뒤에 b의 지수와 계수로 이루어진 노드 추가
                        b = b->link; // b를 다음노드로 이동
                }
                else // a와 b의 지수가 같다면
                {
                        addNode(&c, (a->cof + b->cof), a->exp);
                        // c의 뒤에 (a계수+b계수)와 a의 지수로 이루어진 노드 추가
                        a = a->link; // a를 다음노드로 이동
                        b = b->link; // b를 다음노드로 이동
                }
        }
        for (; a; a = a->link) { // a가 남은 노드가 있으면( b의 노드는 모두 사용한 상황)
                if (a->exp == -1) // a가 가리키는 노드의 지수가 -1이라면
                        break; // for문 종료
                addNode(&c, a->cof, a->exp); // c에 a의 지수와 계수로 이루어진 노드추가
        }
        for (; b; b = b->link) {// b가 남은 노드가 있으면( a의 노드는 모두 사용한 상황)
                if (b->exp == -1) //b가 가리키는 노드의 지수가 -1이라면
                        break; // for문 종료
                addNode(&c, b->cof, b->exp); // c에 b의 지수와 계수로 이루어진 노드 추가
        }
        addNode(&c, -1, -1);
// c 다항식의 제일 마지막임을 표시하기 위한 지수 -1, 계수 -1인 노드 추가
        return c; // c노드포인터 리턴
}


int main()
{
        //다항식 a -> ax^2 + bx^1 + cx^0 + (-1x^-1)
        //다항식 b -> a'x^2 + b'x^1 + c'x^0 + (-1x^-1)
        //다항식 c -> 다항식 a + 다항식 b
        NODE * a = NULL, *b = NULL, *c = NULL;
        int tempExp, tempCof; // 지수와 계수를 입력받을 때 사용할 임시 변수
        printf("a : \n");
        while (1) //tempExp가 -1이 되기전까지
        { // 다항식을 입력받는다 지수->계수 순
                printf("exp : "); scanf("%d", &tempExp);
                printf("cof : "); scanf("%d", &tempCof);
                addNode(&a, tempCof, tempExp); // 입력받은 지수와 계수로 노드 추가
                if (tempExp == -1) // 지수에 -1 입력시
                        break; // while문 종료

        }
```

```
	printf("b : \n");
	while (1) // tempExp가 -1이 되기전까지
	{ // 다항식을 입력받는다 지수->계수 순
		printf("exp : "); scanf("%d", &tempExp);
		printf("cof : "); scanf("%d", &tempCof);
		addNode(&b, tempCof, tempExp); // 입력받은 지수와 계수로 노드 추가
		if (tempExp == -1) // 지수에 -1 입력시
			break; //while문 종료
	}

	c = SumPoly(a, b); // 다항식 a와 b를 더함
	showList(c); // 최종 결과 다항식 c를 출력

}
```

- 결과창

먼저 다항식 a의 지수와 계수를 내림차순으로 입력하고, 마지막에 지수와 계수를 -1로 지정해주고, 다항식 b도 a와 같이 입력해준다. 그러면 다항식 a와 b는 addNode로 인해 연결리스트가 생성되고, sumPoly에 의해 합쳐진 다항식 c가 출력된다.

3. 결론

연결리스트를 선언할 때, 포인터형식을 사용하는 이유는 함수에서 사용하기 편리하기 위함이다. 노드를 추가하거나 노드를 삭제할 때 주소를 이용하여 주소값을 참조할 때 더 편리하게 사용할 수 있다. addNode에서는 매개변수로 노드포인터의 주소를 받는데, 매개변수로 오는 노드포인터의 위치는 변경하지 않고 함수 내에서만 사용하는 노드포인터 p만을 옮김으로써 리스트의 끝까지 가서 동적 할당한 temp노드를 head리스트에 연결하기 위함임을 알 수 있다. 또한 연결리스트를 사용함으로써 배열을 이용한 다항식과는 달리 메모리를 덜 차지한다는 부분에서 장점이 있다.

2017-12-10

황 희