자료구조 (특별)

Chap 02. 연결 리스트 (다항식) - 실습

2019년 하계 특별 학기

컴퓨터과학과
민경하

Contents

2. 연결 리스트 (다항식)

2.1 소개

2.2 자료 구조 1 (배열)

2.3 자료 구조 2 (연결 리스트)

2.4 기본 연산

2.5 다항식 연산

2.1 소개

- 다항식 (Polynomial)
  - 항(term)의 합
  - 각 항은 다음의 같이 구성됨: $a\,x^e$
    - a: 계수 (coefficient)
    - x: 변수 (variable)
    - e: 지수 (exponent)

$$A(x) = 3x^{20} + 2x^5 + 4$$
$$B(x) = x^4 + 10x^3 + 1$$

$$A(x) = \sum a_i x^i$$
$$B(x) = \sum b_i x^i$$

2.2 자료 구조1 (배열)

- 다항식을 표현하는 자료구조: 배열
  - *n*차 다항식

$$a_n x^n + a_{n-1} x^{n-1} + a_{n-2} x^{n-2} + \cdots + a_1 x + a_0$$

  - *n+1* 크기의 배열

a_n	a_{n-1}	a_{n-2}	……	a_i	……	a_3	a_2	a_1	a_0

2.2 자료 구조1 (배열)

- 다항식을 표현하는 자료구조: 배열
  - degree: 다항식의 최고차 항의 지수
  - *coef: 각 항의 계수

```
class polynomial {
        int degree;
        int *coef;
};
```

2.2 자료 구조1 (배열)

- 다항식을 표현하는 자료구조: 배열
  - 예)

$$A(x) = 3x^{20} + 2x^3 + 4$$

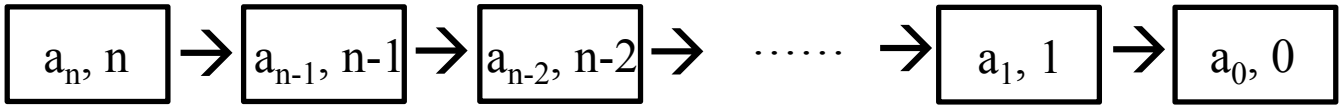
3	0	2	0	0	4
x^{20}	……	x^3	x^2	x^1	x^0

```
a.degree = 20;
a.coef = (int *) calloc ( a.degree + 1, sizeof(int));
for ( i = 0; i <= n; i++ )
       a.coef[i] = 0;
a.coef[n-20] = 3;
a.coef[n-3] = 2;
a.coef[n] = 4;
```

2.3 자료구조 2 (연결 리스트)

- 다항식을 표현하는 자료구조: 연결 리스트
  - n 차 다항식

$$a_n x^n + a_{n-1} x^{n-1} + a_{n-2} x^{n-2} + \cdots + a_1 x + a_0$$



2.3 자료구조 2 (연결 리스트)

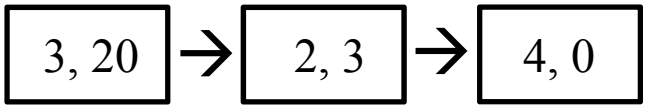
- 다항식을 표현하는 자료구조: 연결 리스트
  - coef: 항의 계수
  - exp: 항의 지수

```
class term {
        int coef;
        int exp;

        term *next;
};

class polynomial {
        int degree;
        term *head;
};
```

2.3 자료구조 2 (연결 리스트)

- 다항식을 표현하는 자료구조: 연결 리스트
  - 예제

$$A(x) = 3x^{20} + 2x^3 + 4$$



2.4 연산

- 기본 연산
  (1) Zero ( ): 0인 다항식을 생성 (초기화)
  (2) IsZero ( ): 0인 다항식인지 확인
  (3) coef (e): 지수가 e인 항의 계수인 coef를 리턴
  (4) LeadExp ( ): 최고 다항식의 지수를 리턴
  (5) Append (e, c): c x^e 항을 다항식의 마지막에 추가
  (6) Insert (e, c): c x^e 항을 다항식의 적절한 위치에 삽입
    - 같은 지수의 항이 있으면 적절히 처리할 것

- 고급 연산
  – 더하기, 곱하기, 출력하기

2.4 연산 (1): Zero ( )

- “0”인 다항식을 생성
  - _n을 0으로 초기화

```
void zero(int _n) {
        n = _n;
        head = (term *)malloc(sizeof(term));
        head->coef = 0;
}
```

2.4 연산 (2): IsZero ( )

- 다항식이 “0”이면 True를 리턴

```
bool is_zero() {
        return (head->coef == 0);
}
```

2.4 연산 (3): coef ( )

- 지수가 exp인 항의 계수를 리턴
  - 찾는 지수의 항이 없으면 0을 리턴

```
int coef(int exp)
{
        term *curr;
        for (curr = head; curr != NULL; curr = curr->next) {
                if (curr->exp == exp)
                        return curr->coef;
        }
        return 0;
}
```

2.4 연산 (4): LeadExp ( )

- 최고차 항의 지수를 리턴 → n차 방정식?

```
int LeadExp() {
	if (is_zero()) {
		printf("Zero polynomial\n");
		return 0;
	}

	return head->exp;
}
```

2.4 연산 (5): Append ( )

- 항을 다항식의 가장 뒤에 추가하는 연산

```
void append (int _coef, int _exp) {
// 예외 처리

// 마지막 항으로 이동

// 새로운 항을 만들어서 첨부
}
```

2.4 연산 (7): insert ( )

- 지수와 계수가 **exp**와 **coef**인 항을 추가하는 연산
  - 다항식이 **exp**의 내림차순으로 정렬된 성질을 유지하면서 추가
  - 동일한 **exp**인 항이 있으면 **coef**를 더함
  - 더해진 **coef**가 **0**이면 항을 다항식에서 제거

2.4 연산 (6): insert ( )

- 지수와 계수가 **exp**와 **coef**인 항을 추가하는 연산

```
void insert(int coef, int exp) {
// 예외 처리

// 삽입할 위치를 결정
      // exp와 같은 지수를 가진 항이 있는 경우

      // exp와 같은 지수를 가진 항이 없는 경우

// 삽입할 위치가 마지막인 경우
```

2.5 다항식 연산 (1): Add ( )

- 두 다항식의 합을 구하는 연산
  - 두 연결 리스트를 앞에서부터 비교하여 처리
  - merge ( )와 비슷한 과정으로 구현

  - append ( ) 함수를 이용

2.5 다항식 연산 (1): Add ( )

- 두 다항식의 합을 구하는 연산

```
void add (Poly pA, poly pB)
{
    currA ← pA.head;
    currB ← pB.head;

    while ( currA != NULL && curb != NULL ) {
    // currA의 지수와 currB의 지수가 같으면

    // currA의 지수가 currB의 지수보다 크면

    // currA의 지수가 currB의 지수보다 작으면

    }

    // 나머지 처리
}
```

2.5 다항식 연산 (2): Multiply ( )

- 두 다항식의 곱을 구하는 연산
  - 두 연결 리스트의 모든 항을 비교하여 처리
  - sparse matrix의 곱과 비슷
  - insert ( ) 함수를 이용

2.5 다항식 연산 (2): Multiply ( )

- 두 다항식의 곱을 구하는 연산

```
void multiply (Poly pA, poly pB)
{
    for (currA = pA.head; currA != NULL; currA = currA->next ) {
        for ( currB = pB.head; currb != NULL; currB = currB->next ) {
        // currA와 currB를 곱해서 insert할 것
        }
    }
}
```

2.5 다항식 연산 (3): print ( )

- 다항식을 출력하는 연산
  - (3, 2) → (2, 0) ➔ 3x^2 + 2
  - (1, 3) → (1, 1) → x^3 + 1
  - (-1, 2) → -x^2

2.5 다항식 연산 (3): print ( )

- 다항식을 출력하는 연산
  - 고려할 사항
    - coef가 1인가?
    - coef가 음수인가?
    - coef가 -1인가?
    - 출력할 항이 첫번째 항인가?
    - exp가 0인가?
    - 또? ……

요구 사항

- 두 개의 다항식을 다음과 같은 파일을 통해서 입력받는다.

  - 첫번째 다항식: 4 개의 항, -x^4 + 3x^3 + 6x^2 + x
  - 두번째 다항식: 3개의 항, 2x^5 -6 x^2 -36

- 두 다항식의 합과 곱을 구하는 프로그램을 작성한다.

프로그램 구조

- pmain.cpp
  - main 함수 포함
- poly.h
  - term, poly에 대한 class 정의
  - 다항식 관련 함수의 prototype 선언
- poly.cpp
  - poly.h에서 선언된 함수 정의

main ( ) 함수

```
int main()
{
        int i;
        poly pA, pB, pC, pD;

        FILE *fp = fopen("poly.txt", "r+t");
        pA.load(fp);
        pB.load(fp);

        printf("poly A is.....................................\n");
        pA.print();

        printf("\npoly B is.....................................\n");
        pB.print();

        pC.zero(0);
        printf("\npoly C is.....................................\n");
        pC.print();

        pC.add(pA, pB);
        printf("\nAddition.....................................\n");
        pC.print();
```

main ( ) 함수

```
        pD.zero(0);
        printf("\npoly D is.....................................\n");
        pD.print();

        pD.multiply(pA, pB);
        printf("\nMultiplication.....................................\n");
        pD.print();

        fclose(fp);
        system("pause");
        return 0;
}
```