자료구조응용
12. 환형리스트 (10점)

2022.4.13.

1. 다음과 같이 **헤더노드를 가진 단일연결 환형리스트(singly linked circular list with header node)**를 이용한 다항식 더하기 프로그램을 작성하라. 단, 프로그램 수행 중 더 이상 사용하지 않는 노드를 저장하는 **체인 형태의 노드풀(node pool or available space list)**을 같이 유지한다. (10점)

[실행 순서]

① 입력파일("a.txt," "b.txt")로부터 데이터를 입력받아서 두 개의 **다항식 a, b**를 각각 **헤더노드를 가진 단일연결 환형리스트** 형태로 구현하고 **last 포인터**를 유지한다.
※ inputPolyCL 2회 호출

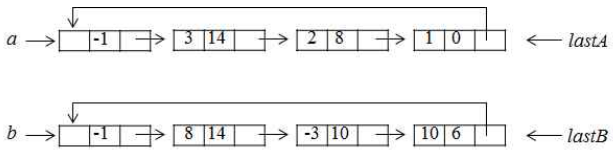
- 각 파일의 첫 줄이 'a'이면 지수에 대해 오름차순, 'd'이면 내림차순으로 입력된다.
- **오름차순으로 입력되면 각 노드는 환형리스트의 첫 노드로** 삽입되어야 하며, **내림차순으로 입력되면 각 노드는 환형리스트의 마지막 노드로** 추가되어야 한다.

$a = 3x^{14} + 2x^8 + 1,\ b = 8x^{14} - 3x^{10} + 10x^6$ 에 대한 입력 예

a.txt	b.txt
a	d
1 0	8 14
2 8	-3 10
3 14	10 6

or

a.txt	b.txt
d	a
3 14	10 6
2 8	-3 10
1 0	8 14



< 헤더노드를 가진 단일연결 환형리스트 형태의 다항식 생성하기 >

② a, b 두 다항식의 정보를 출력한다. ※ printCList
③ a+b의 결과를 c에 저장하는 다항식 더하기를 수행한다. ※ cpadd
④ 다항식 c를 출력한다. ※printCList
⑤ 다항식 a, b, c를 ***avail*** 에 반납한다. ※ cerase
⑥ avail을 삭제한다. ※ erase

[구현세부사항]

① 다항식 및 노드풀 정의

```
typedef struct polyNode *polyPointer;
typedef struct polyNode {
            int coef;
            int expon;
            polyPointer link;
}polyNode;

/* 헤더노드를 가진 다항식 (단일연결 환형리스트) */
polyPointer a = NULL, b = NULL, c = NULL;

/* 다항식 a와 b에 대한 last 포인터 */
polyPointer lastA, lastB;

/* 노드풀 (체인) */
polyPointer avail = NULL;
```

② **다항식 a, b, c는 "헤더노드를 가진 단일연결 환형리스트"**로 구현하며 추가적으로 마지막 노드에 대한 포인터를 각각 가진다.

③ **노드풀 avail은 "체인"**으로 구현한다.

[함수원형]

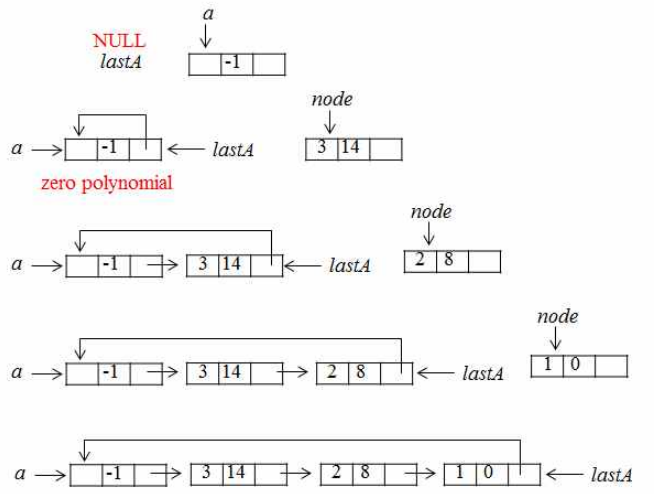
※ 다음은 참고사항임! 각자 더 효율적인 방법이 있다면 그대로 구현하면 됨!

① **void inputPolyCL(char *filename, polyPointer *header, polyPointer *last);**

- 하나의 파일입력으로부터 **"헤더노드를 가진 단일연결 환형리스트"**로 구현된 하나의 다항식을 생성한다. 제일 먼저 헤더노드만으로 구성된 **제로 다항식**을 생성한 후, 파일입력으로부터 하나씩 노드를 생성 한 후 리스트에 추가한다.

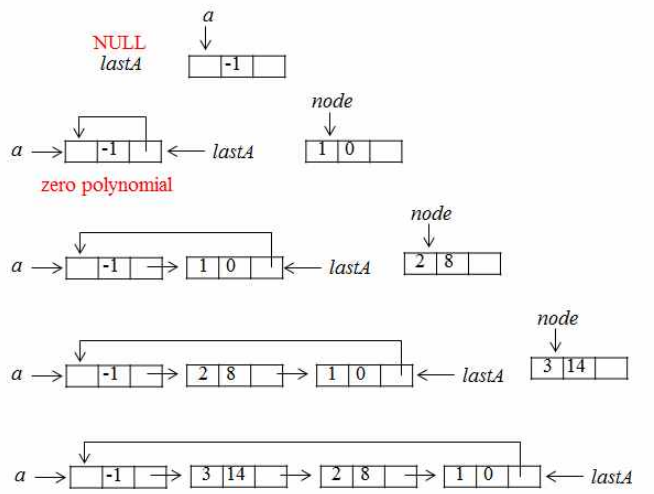
- 이때, **파일 입력형식이 'd'이면 insertLastCL 함수**를 호출하여 리스트의 마지막 노드로 추가하며, **'a'이면 insertFrontCL 함수**를 호출하여 리스트의 첫 노드로 추가한다.

- 각 노드를 동적으로 생성하기 위해 **getNode() 함수**를 호출한다.



※ 각 노드 추가 시 insertLastCL 함수를 호출한다.

(a) 내림차순 입력으로부터 생성



※ 각 노드 추가 시 insertFrontCL 함수를 호출한다.

※ 첫 노드 (1, 0) 삽입 시 lastA를 변경하고 이후는 lastA의 변경 없음

(b) 오름차순 입력으로부터 생성

< 헤더노드를 가진 단일연결 환형리스트 생성과정($a = 3x^{14} + 2x^8 + 1$) >

② void insertFrontCL(polyPointer *last, polyPointer node);

- 헤더노드를 가진 단일연결 환형리스트의 첫 노드로 추가한다.

Program 4.18. insertFront 함수 수정. 헤더노드 다음에 추가함.

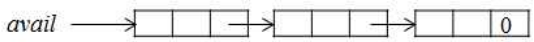
③ void insertLastCL(polyPointer *last, polyPointer node);
- 헤더노드를 가진 단일연결 환형리스트의 마지막에 노드를 추가한다.
- Program 4.18. insertFront 함수의 else 블록 마지막에 *last = node; 추가로 구현됨

④ void printCList(polyPointer header);
- 헤더노드를 가진 단일연결 환형리스트로 표현된 다항식을 출력한다.

⑤ void attach(float coefficient, int exponent, polyPointer *ptr);
- (coefficient, exponent) 값을 가지는 다항식 노드를 새로 생성한 후, *ptr이 가리키는 마지막 노드 뒤에 추가하기. 노드 추가 후 *ptr은 새 노드로 수정된다.
- Program 4.10. attach 함수에서 MALLOC 대신 **getNode()를 사용하여 새 노드를 생성**하도록 수정.

⑥ void erase(polyPointer *ptr);
- ptr이 가리키는 체인을 삭제하는 함수이다. 체인의 처음부터 따라가며 모든 노드에 대한 메모리해제를 수행한다 (Program 4.11)

⑦ polyPointer getNode(void);
- 체인으로 구현된 노드풀로부터 노드를 하나 가져온다. (Program 4.12)



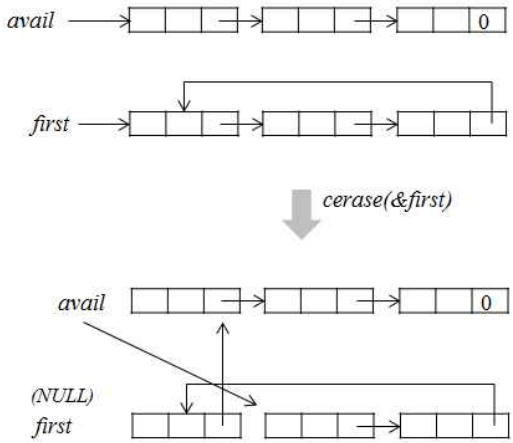
< 체인으로 구현된 노드풀 >

⑧ void retNode(polyPointer node);
- 더 이상 사용하지 않는 노드 하나를 노드풀에 반환한다. (Program 4.13)

⑨ void cerase(polyPointer *ptr);
- ptr이 가리키는 다항식(단일연결 환형리스트)을 노드풀에 반환하여 삭제한다. 환형 연결리스트의 각 노드에 대한 메모리 해제를 통한 삭제가 아님을 주의하라. (Program 4.14)

```
void cerase(polyPointer *ptr)
{/* erase the circular list pointed to by ptr */
   polyPointer temp;
   if (*ptr) {
      temp = (*ptr)→link;
      (*ptr)→link = avail;
      avail = temp;
      *ptr = NULL;
   }
}
```

Program 4.14: Erasing a circular list



< 단일연결 환형리스트를 노드풀에 반납하기 >

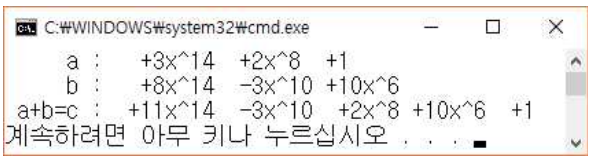
⑩ polyPointer cpadd(polyPointer a, polyPointer b);

- "헤더노드를 가진 단일연결 환형리스트"로 표현된 두 다항식 a, b에 대해 더하기 연산을 수행하여 그 결과 다항식을 반환한다.

```
polyPointer cpadd(polyPointer a, polyPointer b)
{/* polynomials a and b are singly linked circular lists
    with a header node. Return a polynomial which is
    the sum of a and b */
  polyPointer startA, c, lastC;
  int sum, done = FALSE;
  startA = a;          /* record start of a */
  a = a→link;          /* skip header node for a and b*/
  b = b→link;
  c = getNode();       /* get a header node for sum */
  c→expon = -1; lastC = c;
  do {
     switch (COMPARE(a→expon, b→expon)) {
        case -1: /* a→expon < b→expon */
              attach(b→coef,b→expon,&lastC);
              b = b→link;
              break;
        case 0:  /* a→expon = b→expon */
              if (startA == a)  done = TRUE;
              else {
                 sum = a→coef + b→coef;
                 if (sum) attach(sum,a→expon,&lastC);
                 a = a→link; b = b→link;
              }
              break;
        case 1:  /* a→expon > b→expon */
              attach(a→coef,a→expon,&lastC);
              a = a→link;
     }
  } while (!done);
  lastC→link = c;
  return c;
}
```

Program 4.15: Adding two polynomials represented as circular lists with header nodes

[실행예]



■ 제출 형식

- 솔루션 이름 : DS 12
- 프로젝트 이름 : 1
- 각 소스파일에 주석처리

  "학번 이름"

  "본인은 이 소스파일을 다른 사람의 소스를 복사하지 않고 직접 작성하였습니다."
- 실행화면을 캡쳐한 보고서를 작성 후 pdf 파일로 변환하여 솔루션 폴더에 포함
- 솔루션 정리 메뉴를 수행 후 전체 솔루션을 "학번.zip"으로 압축하여 제출

■ 주의

- 소스 복사로는 실력향상을 기대할 수 없습니다!!!

- 1차 마감 (LMS 과제 마감일) : 수업일 자정

- 2차 마감 (LMS 과제 열람기한) : 수업 익일 자정(만점의 50%, 반올림)

- 1, 2차 마감 이외의 제출은 허용하지 않습니다. (이메일 제출 불가!)