[자료구조 숙제4 리스트]

2018187 16
컴퓨터공학부
김용현

〈다항식 프로그램〉 - 전체 코드

```cpp
/* 리스트 - 다항식 프로그램 */
#include <iostream>
#include <list>
#include <algorithm>
using namespace std;

int main(int argc, char* argv[]) {
    /* Faster */
    ios::sync_with_stdio(0);
    cin.tie(0); cout.tie(0);

    /* Init */
    list<pair<int, int>> li[3];

    /* Input */
    for (int i = 0; i < 2; i++) {
        int n; cin >> n;
        for (int j = 0; j < n; j++) {
            int coef, degr; cin >> coef >> degr; //계수, 차수
            li[i].push_back(make_pair(coef, degr));
        }
    }

    /* Merge */
    list<pair<int, int>>::iterator i = li[0].begin();
    list<pair<int, int>>::iterator j = li[1].begin();

    while ((i != li[0].end()) && (j != li[1].end())) {
        if (i->second > j->second) {
            li[2].push_back(*i);
            i++;
        }
        else if (i->second < j->second) {
            li[2].push_back(*j);
            j++;
        }
        else {
            li[2].push_back(make_pair(i->first + j->first, i->second));
            i++; j++;
        }
    }
    while (i != li[0].end()) {
        li[2].push_back(*i);
        i++;
    }
    while (j != li[1].end()) {
        li[2].push_back(*j);
        j++;
    }

    /* Output */
    for (int i = 0; i < 3; i++) {
        cout << '(' << li[i].size() << ") = ";
        while (!li[i].empty()) {
            if (li[i].front().first != 0) {
                cout << li[i].front().first << ".0" << ' ';
                cout << "x^" << li[i].front().second << ' ';
                if (li[i].size() != 1)
                    cout << "+ ";
            }

            li[i].pop_front();
        }
        cout << '\n';
    }

    /* Return */
    return 0;
}
```

〈다항식 프로그램〉 - 헤더 파일

```
/* 리스트 - 다항식 프로그램 */
#include <iostream>
#include <list>
#include <algorithm>
using namespace std;
```

#include 〈iostream〉 : C++ 기본 입출력.
#include 〈list〉 : C++ STL List 사용.
- list : Doubly Linked List.
- forward_list : Singly Linked List. (여기서는 사용하지 않음.)
#include 〈algorithm〉 : C++ STL Pair 사용.

〈다항식 프로그램〉 - /* Faster */

```
/* Faster */
    ios::sync_with_stdio(0);
    cin.tie(0); cout.tie(0);
```

#. 싱글 쓰레드 환경에서, printf와 scanf 함수가 이용하는 stdio.h 버퍼와, cin과 cout이 이용하는 iostream의 버퍼의 동기화를 해제함으로써, 입출력 속도를 빠르게 하였다.
#. ios::sync_with_stdio(0)은 stdio.h와 iostream의 동기화를 해제하겠다는 뜻이다.
#. cin.tie(0) 및 cout.tie(0)의 경우, cin과 cout은 서로 연결되어 있어 cin을 쓰면 출력 버퍼를 비우고 입력이 발생한다. 이러한 flush(버퍼를 비우는 과정) 과정도 시간이 소요되기 때문에, cin과 cout의 상호 연결을 끊어주기 위하여 해당 코드를 삽입하는 것이다.

〈다항식 프로그램〉 - /* Init */

```
/* Init */
    list<pair<int, int>> li[3];
    //다항식1 · 2 · 3 저장.
```

#. li[0]에는 입력값으로 들어온 첫 번째 다항식을 저장한다.
#. li[1]에는 입력값으로 들어온 두 번째 다항식을 저장한다.
#. li[2]에는 li[0]과 li[1]의 다항식을 더한 값을 저장한다.

〈다항식 프로그램〉 - /* Input */

```
/* Input */
    for(int i = 0; i < 2; i++) {
        int n; cin >> n;
        for(int j = 0; j < n; j++) {
            int coef, degr; cin >> coef >> degr;//계수, 차수
            li[i].push_back(make_pair(coef, degr));
        }
    }
```

#. 입력값들을 받아 li[0]과 li[1]에 각각 push_back 해준다. (push_back()은 리스트의 뒤쪽으로 삽입이 일어나게 해준다.)
#. make_pair는 반환값으로 pair〈T1, T2〉를 반환한다. 여기서, T1과 T2는 인자로 전달되는 자료형에 의존적이다. (여기서는 〈int, int〉를 인자로 전달 받았으므로, 반환값은 pair〈int, int〉 이다.)

〈다항식 프로그램〉 - /* Merge */

```
/* Merge */
    list<pair<int, int>>::iterator i = li[0].begin();
    list<pair<int, int>>::iterator j = li[1].begin();

    while((i != li[0].end()) && (j != li[1].end())) {
        if(i->second > j->second) {
            li[2].push_back(*i);
            i++;
        }
        else if(i->second < j->second) {
            li[2].push_back(*j);
            j++;
        }
        else{
            li[2].push_back(make_pair(i->first + j->first, i->second));
            i++; j++;
        }
    }
    while(i != li[0].end()) {
        li[2].push_back(*i);
        i++;
    }
    while(j != li[1].end()) {
        li[2].push_back(*j);
        j++;
    }
```

#. 두 다항식을 합치는 부분이다. li[0]과 li[1]에 존재하는 값들을 합쳐 하나의 다항식으로 만든 후 li[2]에 저장한다.

#. Logic : li[0] 혹은 li[1]의 iterator가 end() 위치에 도달할 때 까지 while문을 반복한다.

이때, list의 반복자(iterator)는 양방향(Bidirectional) 반복자로, ++연산과 --연산만 지원한다.

[초기]

iterator of li[0] = i	V			
계수(First)	3	2	1	end
차수(Second)	12	8	0	end

iterator of li[1] = j	V				
계수(First)	15	14	2	1	end
차수(Second)	11	3	2	1	end

li[2]

[1회]

iterator of li[0] = i			V		
계수(First)	3		2	1	end
차수(Second)	12		8	0	end

iterator of li[1] = j	V				
계수(First)	15	14	2	1	end
차수(Second)	11	3	2	1	end

li[2]
3
12

[2회]

iterator of li[0] = i			V		
계수(First)	3		2	1	end
차수(Second)	12		8	0	end

iterator of li[1] = j		V			
계수(First)	15	14	2	1	end
차수(Second)	11	3	2	1	end

li[2]	
3	15
12	11

[3회]

iterator of li[0] = i				V	
계수(First)	3		2	1	end
차수(Second)	12		8	0	end

iterator of li[1] = j		V			
계수(First)	15	14	2	1	end
차수(Second)	11	3	2	1	end

li[2]		
3	15	2
12	11	8

[4회]

iterator of li[0] = i				V	
계수(First)	3		2	1	end
차수(Second)	12		8	0	end

iterator of li[1] = j			V		
계수(First)	15	14	2	1	end
차수(Second)	11	3	2	1	end

li[2]			
3	15	2	14
12	11	8	3

[5회]

iterator of li[0] = i				V	
계수(First)	3	2		1	end
차수(Second)	12	8		0	end

iterator of li[1] = j				V	
계수(First)	15	14	2	1	end
차수(Second)	11	3	2	1	end

li[2]					
3	15	2	14	2	
12	11	8	3	2	

[6회]

iterator of li[0] = i				V	
계수(First)	3	2		1	end
차수(Second)	12	8		0	end

iterator of li[1] = j					V
계수(First)	15	14	2	1	end
차수(Second)	11	3	2	1	end

li[2]					
3	15	2	14	2	1
12	11	8	3	2	1

#. [7회차]에 첫 번째 while문을 진입하면, j가 li[1].end()를 가리키게 되므로, 반복 조건문이 false가 되어 다음 while문으로 넘어간다.
#. 아래 2개의 while문은 li[0]과 li[1]에 남아있는 원소들을 li[2]에 넣기 위한 Logic으로, 아래 2개의 while문을 실행하게 되면, li[0]과 li[1]에서 li[2]로 병합되지 못한 원소들이 모두 li[2]로 넘어가게 된다.

<다항식 프로그램> - /* Output */

```
/* Output */
    for(int i = 0; i < 3; i++) {
        cout << '('<< li[i].size() << ") = ";
        while(!li[i].empty()) {
            if(li[i].front().first != 0) {
                cout << li[i].front().first << ".0" << ' ';
                cout << "x^" << li[i].front().second << ' ';
                if(li[i].size() != 1)
                    cout << "+ ";
            }
            li[i].pop_front();
        }
        cout <<'\n';
    }
```

#. 문제에 주어진 양식에 맞게 결과값을 출력한다.

```cpp
#include <iostream>
#include <list>
#include <string>
using namespace std;

int main(int argc, char* argv[]) {
    /* Faster */
    ios::sync_with_stdio(0);
    cin.tie(0); cout.tie(0);

    /* Init */
    list<string> li;
    list<string>::iterator it;

    /* Loop */
    while (true) {
        /* Init */
        char c; cin >> c;

        switch (c) {
        /* i */
        case 'i': {
            /* Input */
            int n; cin >> n;
            string s; cin.ignore(); getline(cin, s);

            /* Insert */
            it = li.begin();
            for (int i = 0; i < n; i++) it++;
            li.insert(it, s);

            /* Break */
            break;
        }
        /* d */
        case 'd': {
            /* Input */
            int n; cin >> n;

            /* Delete */
            it = li.begin();
            for (int i = 0; i < n; i++) it++;
            li.erase(it);

            /* Break */
            break;
        }
        /* r */
        case 'r': {
            /* Input */
            int n; cin >> n;
            string s; cin.ignore(); getline(cin, s);

            /* Replace */
            it = li.begin();
            for (int i = 0; i < n; i++) it++;
            *it = s;

            /* Break */
            break;
        }
        /* f */
        case 'f': {
            /* Input */
            string prev; cin.ignore(); getline(cin, prev);
            string next; getline(cin, next);

            /* Find */
            for (auto& i : li)
                if (i == prev) i = next;

            /* Break */
            break;
        }
        /* p */
        case 'p': {
            /* Output */
            int line = 0;
            for (auto i : li)
                cout << line++ << ':' << ' ' << i << '\n';
            cout << "EOF\n";

            /* Break */
            break;
        }
        /* q */
        case 'q' :
            return 0;
        }
    }
}
```

<라인 편집기>

/* Header */
#include <iostream> : C++ 기본 입출력.
#include <list> : C++ STL List 사용.
 - list : Doubly Linked List.
 - forward_list : Singly Linked List. (여기서는 사용하지 않음.)
#include <string> : C++ STL String 사용.

/* Faster */
#. 싱글 쓰레드 환경에서, printf와 scanf 함수가 이용하는 stdio.h 버퍼와, cin과 cout이 이용하는 iostream의 버퍼의 동기화를 해제함으로써, 입출력 속도를 빠르게 하였다.
#. ios::sync_with_stdio(0)은 stdio.h와 iostream의 동기화를 해제하겠다는 뜻이다.
#. cin.tie(0) 및 cout.tie(0)의 경우, cin과 cout은 서로 연결되어 있어 cin을 쓰면 출력 버퍼를 비우고 입력이 발생한다. 이러한 flush(버퍼를 비우는 과정) 과정도 시간이 소요되기 때문에, cin과 cout의 상호 연결을 끊어주기 위하여 해당 코드를 삽입하는 것이다.

/* Init */
#. list li를 선언하고, 해당 list의 iterator(반복자) it를 선언한다. 이때, list의 iterator는 양방향(Bidirectional) 반복자로, ++연산과 --연산만 지원한다.

/* Loop */
 /* i */
 #. 입력으로 들어온 문자열(s)을, 입력으로 들어온 정수(n)값의 위치에 삽입한다.
 #. iterator를 li.begin() 즉, 맨 처음 위치로 설정한 뒤, for문을 통해 n번째 위치로 iterator의 위치를 옮긴 후, 해당 위치에 insert() 메소드를 통해 값을 삽입한다.
 #. cin.ignore()를 통해, 정수값을 입력받은 후 생긴 개행문자('₩n')값을 버퍼에서 지운다.

 /* d */
 #. n번째 위치에 해당하는 값을 삭제한다.
 #. iterator를 li.begin() 즉, 맨 처음 위치로 설정한 뒤, for문을 통해 n번째 위치로 iterator의 위치를 옮긴 후, 해당 위치에 존재하는 값을 erase() 메소드를 통해 삭제한다.

 /* r */
 #. 입력으로 들어온 문자열(s)을, 입력으로 들어온 정수(n)값의 위치에 있는 문자열과 교체한다.
 #. iterator를 li.begin() 즉, 맨 처음 위치로 설정한 뒤, for문을 통해 n번째 위치로 iterator의 위치를 옮긴 후, 해당 위치에 존재하는 문자열을 *연산자를 통해 접근한 후, 해당 값을 변경한다.

 /* f */
 #. 입력으로 들어온 첫 번째 문자열(prev)를 list에서 찾아, 입력으로 들어온 두 번째 문자열(next)로 변경한다.
 #. 범위 기반 for문을 통해 list에 존재하는 모든 원소들을 탐색한 후, 만약 해당 문자열 값이 prev와 같다면, 해당 값을 next로 변경해준다.

 /* p */
 #. 출력 양식에 맞게 값을 출력한다.

 /* q */
 #. 프로그램을 종료한다.