자료구조 HW6

B711222 박조은
Hongik University
mrnglory@mail.hongik.ac.kr

November 1, 2019

I. List of Source Files

- hw6
  - hw6.cpp
  - list.h
  - list.cpp
  - makefile
  - input.in

II. hw6

i. hw6.cpp

```cpp
#include <iostream>
#include "list.h"
using namespace std;

int main()
{
    IntList il;
    int input;

    cout << "============ Input ============"
        << endl;

        while (1)
        {
            // -1 을 받을 때 까지 반복
            cin >> input;

            if (input == -1)
                break;

            il.Insert(input);

            cout << il;
        }

    cout << "============ Delete ============"
        << endl;

        input = 0;

        while (1)
        {
            // -1 을 받을 때 까지 반복
            cin >> input;

            if (input == -1)
                break;

            il.Delete(input);

            cout << il;
    }

    // Push_Front 와 Push_Back 은 2회씩.
    cout << "========== Push_Front ==========="
        << endl;

        cin >> input;
        il.Push_Front(input);
        cout << il;

        cin >> input;
        il.Push_Front(input);
        cout << il;

    cout << "============ Push_Back ============"
        << endl;

        cin >> input;
        il.Push_Back(input);
        cout << il;

        cin >> input;
        il.Push_Back(input);
        cout << il;

    return 0;
}
```

ii. list.h

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

```

```
4   struct Node
5   {
6       Node (int d = 0, Node* ptr = NULL) : data(d), link(
            ↪ ptr) {}
7
8       int data;
9       Node* link;
10  };
11
12  struct IntList
13  {
14      IntList()
15      {
16          last = first = NULL;
17      }
18
19      void Push_Back(int); // 리스트 뒤에 삽입 중복 허용
20      void Push_Front(int); // 리스트 앞에 삽입 중복 허용
21      void Insert(int); // 정렬되어있다는 가정 하에 제
            ↪ 위치에 삽입
22      void Delete(int); // 리스트의 원소 삭제
23      Node *first;  // 첫 노드를 가리킴
24      Node *last;  // 마지막 노드를 가리킴
25  };
26
27  ostream& operator << (ostream&, IntList&);
```

iii. list.cpp

```
1   #include <iostream>
2   #include "list.h"
3
4   ostream& operator << (ostream& os, IntList& il)
5   {
6       Node *ptr = il.first;
7
8       while (ptr != NULL)
9       {
10          os << ptr -> data << " ";
11          ptr = ptr -> link;
12      }
13
14      os << endl;
15
16      return os;
17  }
18
19  void IntList::Push_Back(int e)
20  {
21      if (!first)
22          first = last = new Node(e);
23
24      else
25      {
26          last -> link = new Node(e);
27          last = last -> link;
28      }
29  }
30
31  void IntList::Push_Front(int e)
32  {
33      Node * newbie = new Node(e);
34      newbie -> link = NULL;
35
36      if (!first)
37      {
38          first = newbie;
39          last = newbie;
40      }
41
42      else
43      {
44          newbie -> link = first;
45          first = newbie;
46      }
47  }
48
49
50
51  void IntList::Insert(int e)
52  {
53      // Push_Front, Push_Back 사용할 것, 중복허용 안함
54      if (!first)
55      {
56          // 빈 리스트인 경우
57          Push_Front(e);
58      }
59
60      else if (first -> data > e)
61      {
62          // 리스트 맨 앞에 추가
63          Push_Front(e);
64      }
65
66      else if (first -> data != e)
67      {
68          Node * a = first -> link;
69          Node * b = first;
70
71          while (a != NULL)
72          {
73              if (e > (a -> data))
74              {
75                  b = a;
76                  a = a -> link;
77              }
78
79              else if (e < a -> data)
80              {
81                  Node * newbie = new Node(e);
82                  newbie -> link = NULL;
```

```
83
84                      newbie -> link = b -> link;
85                      b -> link = newbie;
86
87                      break;
88                  }
89
90                  else if (e == a -> data)
91                  {
92                      break;
93                  }
94              }
95
96              if (a == NULL)
97              {
98                  Push_Back(e);
99              }
100         }
101     }
102
103     void IntList::Delete(int e)
104     {
105         if (first -> data == e)
106         {
107             // 첫 번째 데이터를 삭제할 경우
108             Node * del_ptr = first;
109             first = first -> link;
110             delete del_ptr;
111         }
112
113         else
114         {
115             Node * a = first -> link;
116             Node * b = first;
117
118             while (a != NULL)
119             {
120                 if (e == a -> data)
121                 {
122                     b -> link = a -> link;
123                     delete a;
124                     a = b;
125
126                     break;
127                 }
128
129                 b = a;
130                 a = a -> link;
131             }
132
133             if (a == NULL)
134             {
135                 last = b;
136             }
137         }
138     }
```

```
/*
 * line 51 - 101: 데이터 입력 매개변수 e 를 node 의 data 로써 받아와, 각 조건에 맞게 Insert 를 구현한 함수이다.
 * first 인 경우, first 의 data 가 e 보다 큰 경우, first 의 data 가 e 와 다를 경우로 조건을 나누며, 후자의 경우 안에서는 계속해서 리스트의 node data 들과 e 의 대소관계를 따져나가며 data 가 오름차순으로 적절히 정렬되도록 구현하였다.
 * 입력값 e 에 대하여 각 node 들 사이에서의 대소관계를 앞 뒤로 따지기 위해서 line 115 - 116 과 같이 first 와 first -> link 를 가리키는 node type pointer a 와 b 를 생성하였고, 이는 while 문을 통해 한칸씩 뒤로 이동한다.
 * 한편, Push_Front, Push_Back 함수를 각 조건에 맞는 부분에 사용하였으며, data 의 중복을 허용하지 않게 구현하였다.
 * line 103 - 138: list 의 node 를 입력 매개변수 e 에 대하여 각 조건에 맞게 delete 를 구현한 함수이다.
 * line 108: 매개변수 e 를 data 로 가지는 node 를 삭제할 때, 해당 node 를 가리키는 pointer 를 생성한다.
 * line 109: first -> data == e 의 경우이므로, first = first -> link 라 한다.
 * line 110: 그제서야 삭제하고자 하는 node 를 del_ptr 을 이용하여 지울 수 있다.
 * line 113 - 137: first -> data != e 의 경우 delete 구현하는 코드이며, 이는 동일 조건의 Insert 함수 구현과 마찬가지로, first 와 first -> link 를 가리키는 node type pointer a, b 생성 및 while 문 통해 a != NULL 일 때까지 한 칸씩 갱신하여 반복한다.
 * 다만 delete 는 대소관계를 따질 필요 없이 입력받은 값이 해당 리스트 node data 중 일치하는지의 여부만 판단하면 되기 때문에, e == a -> data 의 경우만 조사한다.

*/
```

iv. Results

iv.1 makefile

```
1  hw6: hw6.o list.o
2      g++ -o hw6 hw6.o list.o
3  hw6.o list.o: list.h
```

iv.2 input.in

```
1  1 5 4 4 3 2
2  -1
3  1 3 2 2 5
4  -1
5  1 2
6  3 4
```

iv.3 compile

```
1  [B711222@localhost hw6d]$ ./hw6 < input.in
2  ============ Input =============
3  1
4  1 5
5  1 4 5
6  1 4 5
7  1 3 4 5
8  1 2 3 4 5
9  ============= Delete ============
10 2 3 4 5
11 2 4 5
12 4 5
13 4 5
14 4
15 ========== Push_Front ===========
16 1 4
17 2 1 4
18 ========== Push_Back ===========
19 2 1 4 3
20 2 1 4 3 4
```

/* 어려웠던 점
 * 결과값이 조금씩 흐트러져서 나왔었는데, Push_Front 와 Delete 함수에서 last 에 대한 정의를 해주지 않았기 때문이라는 것을 캐치해야했던 점이 어려웠다.

*/