HANYANG UNIVERSITY
DIVISION OF COMPUTER SCIENCE

Data Structures

박영준 교수님

Lab2:LinkedList fixed

Linked List

Prev

```c
#include <stdio.h>

#define TRUE 1
#define FALSE 0

typedef int DATATYPE;

typedef struct {
    DATATYPE data;
    struct Node *next;
} Node;

typedef struct
{
    Node *Head;
    Node *Cur;
    Node *Tail;
    int NumofData;
} LinkedList;

void InitList(LinkedList *list);
void Insert(LinkedList *list, DATATYPE data);

int PosHead(LinkedList *list, DATATYPE *data);
int PosNext(LinkedList *list, DATATYPE *data);

DATATYPE Remove(LinkedList *list);
int RetCount(LinkedList *list);
```

```c
void InitList(LinkedList *list)
{
    list->Head = (Node*)malloc(sizeof(Node));
    list->Head = NULL;
    list->Tail = NULL;
    list->Cur = NULL;
    list->NumofData = 0;
}

void Insert(LinkedList *list, DATATYPE data)
{
    Node *temp = (Node*)malloc(sizeof(Node));
    temp->data = data;
    temp->next = NULL;

    if(list->Head == NULL & list->Tail == NULL)
    {
        list->Head = temp;
    }
    else
    {
        list->Tail->next = temp;
    }

    list->Tail = temp;

    list->NumofData++;
}
```

Linked List

Prev

```
124 int PosHead(LinkedList *list, DATATYPE *data)
125 {
126     if(list->Head == NULL)
127     {
128         return FALSE;
129     }
130
131     list->Tail = list->Head;
132     list->Cur = list->Head;
133
134     *data = list->Cur->data;
135     return TRUE;
136 }
137
138 int PosNext(LinkedList *list, DATATYPE *data)
139 {
140     if(list->Cur->next == NULL)
141     {
142         return FALSE;
143     }
144
145     list->Tail = list->Cur;
146     list->Cur = list->Cur->next;
147
148     *data = list->Cur->data;
149     return TRUE;
150 }
```

```
152 DATATYPE Remove(LinkedList *list)
153 {
154     Node *temp = list->Cur;
155     DATATYPE tdata = temp->data;
156
157     list->Tail->next = list->Cur->next;
158     list->Cur = list->Tail;
159
160     free(temp);
161     list->NumofData--;
162     return tdata;
163 }
164
165 int RetCount(LinkedList *list)
166 {
167     return list->NumofData;
168 }
169
170
```

Linked List

Prev

```
30 int main(int argc, char *argv[])
31 {
32     LinkedList list;
33     int data;
34
35     InitList(&list);
36
37     //save 5 data
38     Insert(&list, 12);
39     Insert(&list, 24);
40     Insert(&list, 45);
41     Insert(&list, 24);
42     Insert(&list, 33);
43
44     //print all
45     printf("Num of datas %d\n", RetCount(&list));
46
47     if(PosHead(&list, &data))
48     {
49         printf("%d ", data);
50
51
52         while(PosNext(&list, &data))
53         {
54             printf("%d ", data);
55         }
56     }
57     printf("\n");
58     printf("\n");
59
```

```
60     //serch 24 and delete
61     int target = 24;
62     if(PosHead(&list, &data))
63     {
64         if(data == target)
65         {
66             Remove(&list);
67         }
68
69         while(PosNext(&list, &data))
70         {
71             if(data == target)
72             {
73                 Remove(&list);
74             }
75         }
76     }
77
78     //print all
79     printf("Num of datas %d\n", RetCount(&list));
80
81     if(PosHead(&list, &data))
82     {
83         printf("%d ", data);
84
85         while(PosNext(&list, &data))
86         {
87             printf("%d ", data);
88         }
89     }
90     printf("\n");
91
92     return 0;
93 }
94
```

Linked List

Fixed

```c
#include <stdio.h>

#define TRUE 1
#define FALSE 0

typedef int DATATYPE;

typedef struct Node {
    DATATYPE data;
    struct Node *next;
} Node;

typedef struct
{
    Node *Head;
    Node *Cur;
    Node *Tail;
    int NumofData;
} LinkedList;

void InitList(LinkedList *list);
void Insert(LinkedList *list, DATATYPE data);

int PosHead(LinkedList *list, DATATYPE *data);
int PosNext(LinkedList *list, DATATYPE *data);

DATATYPE Remove(LinkedList *list);
int RetCount(LinkedList *list);
```

```c
void InitList(LinkedList *list)
{
    list->Head = (Node*)malloc(sizeof(Node));
    list->Head->next = NULL;
    list->Tail = list->Head;
    list->Cur = NULL;
    list->NumofData = 0;
}

void Insert(LinkedList *list, DATATYPE data)
{
    Node *temp = (Node*)malloc(sizeof(Node));
    temp->data = data;
    temp->next = NULL;

    list->Tail->next = temp;
    list->Tail = temp;

    list->NumofData++;
}
```

Linked List

Fixed

```
116 int PosHead(LinkedList *list, DATATYPE *data)
117 {
118     if(list->Head->next == NULL)
119     {
120         return FALSE;
121     }
122
123     list->Tail = list->Head;
124     list->Cur = list->Head->next;
125
126     *data = list->Cur->data;
127     return TRUE;
128 }
129
130 int PosNext(LinkedList *list, DATATYPE *data)
131 {
132     if(list->Cur->next == NULL)
133     {
134         return FALSE;
135     }
136
137     list->Tail = list->Cur;
138     list->Cur = list->Cur->next;
139
140     *data = list->Cur->data;
141     return TRUE;
142 }
```

```
152 DATATYPE Remove(LinkedList *list)
153 {
154     Node *temp = list->Cur;
155     DATATYPE tdata = temp->data;
156
157     list->Tail->next = list->Cur->next;
158     list->Cur = list->Tail;
159
160     free(temp);
161     list->NumofData--;
162     return tdata;
163 }
164
165 int RetCount(LinkedList *list)
166 {
167     return list->NumofData;
168 }
```

Linked List

Fixed

```
30 int main(int argc, char *argv[])
31 {
32     LinkedList list;
33     int data;
34
35     InitList(&list);
36
37     //save 5 data
38     Insert(&list, 12);
39     Insert(&list, 24);
40     Insert(&list, 45);
41     Insert(&list, 24);
42     Insert(&list, 33);
43
44     //print all
45     printf("Num of datas %d\n", RetCount(&list));
46
47     if(PosHead(&list, &data))
48     {
49         printf("%d ", data);
50
51
52         while(PosNext(&list, &data))
53         {
54             printf("%d ", data);
55         }
56     }
57     printf("\n");
58     printf("\n");
59
60     //serch 24 and delete
61     int target = 24;
62     if(PosHead(&list, &data))
63     {
64         if(data == target)
65         {
66             Remove(&list);
67         }
68
69         while(PosNext(&list, &data))
70         {
71             if(data == target)
72             {
73                 Remove(&list);
74             }
75         }
76     }
77
78     //print all
79     printf("Num of datas %d\n", RetCount(&list));
80
81     if(PosHead(&list, &data))
82     {
83         printf("%d ", data);
84
85         while(PosNext(&list, &data))
86         {
87             printf("%d ", data);
88         }
89     }
90     printf("\n");
91
92     return 0;
93 }
```