

CPE112: Linked List

Data Structure - Lab 3

Linked List - Display the List

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <stdlib.h>
4  #include <stdbool.h>
5
6  struct node {
7      int data;
8      int key;
9      struct node *next;
10 };
11
12 struct node *head = NULL;
13 struct node *current = NULL;
```

```
15 //display the list
16 void printList() {
17     struct node *ptr = head;
18     printf("\n[ ");
19
20     //start from the beginning
21     while(ptr != NULL) {
22         printf("(%d,%d) ", ptr->key, ptr->data);
23         ptr = ptr->next;
24     }
25
26     printf(" ]");
27 }
28
```

Linked List - Insert First

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <stdlib.h>
4  #include <stdbool.h>
5
6  struct node {
7      int data;
8      int key;
9      struct node *next;
10 };
11
12 struct node *head = NULL;
13 struct node *current = NULL;
```

```
29 //insert link at the first location
30 void insertFirst(int key, int data) {
31     //create a link
32     struct node *link = (struct node*) malloc(sizeof(struct node));
33
34     link->key = key;
35     link->data = data;
36
37     //point it to old first node
38     link->next = head;
39
40     //point first to new first node
41     head = link;
42 }
43
```

Linked List - main function

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <stdlib.h>
4  #include <stdbool.h>
5
6  struct node {
7      int data;
8      int key;
9      struct node *next;
10 };
11
12 struct node *head = NULL;
13 struct node *current = NULL;
```

```
184 void main() {
185     insertFirst(1,10);
186     insertFirst(2,20);
187     insertFirst(3,30);
188     insertFirst(4,1);
189     insertFirst(5,40);
190     insertFirst(6,56);
191
192     printf("Original List: ");
193
194     //print list
195     printList();
196 }
197
```

OUTPUT

/tmp/HaAzoRmVZg.o

Original List:

[(6,56) (5,40) (4,1) (3,30) (2,20) (1,10)]

Linked List - Delete First

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <stdlib.h>
4  #include <stdbool.h>
5
6  struct node {
7      int data;
8      int key;
9      struct node *next;
10 };
11
12 struct node *head = NULL;
13 struct node *current = NULL;
```

```
44 //delete first item
45 struct node* deleteFirst() {
46
47     //save reference to first link
48     struct node *tempLink = head;
49
50     //mark next to first link as first
51     head = head->next;
52
53     //return the deleted link
54     return tempLink;
55 }
56
```


Linked List - isEmpty and length()

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <stdlib.h>
4  #include <stdbool.h>
5
6  struct node {
7      int data;
8      int key;
9      struct node *next;
10 };
11
12 struct node *head = NULL;
13 struct node *current = NULL;
```

```
57 //is list empty
58 bool isEmpty() {
59     return head == NULL;
60 }
61
62 int length() {
63     int length = 0;
64     struct node *current;
65
66     for(current = head; current != NULL; current = current->next) {
67         length++;
68     }
69
70     return length;
71 }
```

Linked List - main function

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <stdlib.h>
4  #include <stdbool.h>
5
6  struct node {
7      int data;
8      int key;
9      struct node *next;
10 };
11
12 struct node *head = NULL;
13 struct node *current = NULL;
```

```
184 void main() {
185     insertFirst(1,10);
186     insertFirst(2,20);
187     insertFirst(3,30);
188     insertFirst(4,1);
189     insertFirst(5,40);
190     insertFirst(6,56);
191
192     printf("Original List: ");
193
194     //print list
195     printList();
196
197     while(!isEmpty()) {
198         struct node *temp = deleteFirst();
199         printf("\nDeleted value:");
200         printf("(%d,%d) ",temp->key,temp->data);
201     }
202
203     printf("\nList after deleting all items: ");
204     printList();
205 }
```

Linked List - main function

```
//delete first item
struct node* deleteFirst() {

    //save reference to first link
    struct node *tempLink = head;

    //mark next to first link as first
    head = head->next;

    //return the deleted link
    return tempLink;
}
```

```
197 while(!isEmpty()) {
198     struct node *temp = deleteFirst();
199     printf("\nDeleted value:");
200     printf("(%d,%d) ",temp->key,temp->data);
201 }
202
203 printf("\nList after deleting all items: ");
204 printList();
205 }
```

OUTPUT

Original List:

[(6,56) (5,40) (4,1) (3,30) (2,20) (1,10)]

Deleted value:(6,56)

Deleted value:(5,40)

Deleted value:(4,1)

Deleted value:(3,30)

Deleted value:(2,20)

Deleted value:(1,10)

List after deleting all items:

[]

Linked List - Find

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <stdlib.h>
4  #include <stdbool.h>
5
6  struct node {
7      int data;
8      int key;
9      struct node *next;
10 };
11
12 struct node *head = NULL;
13 struct node *current = NULL;
```

```
73 //find a link with given key
74 struct node* find(int key) {
75
76     //start from the first link
77     struct node* current = head;
78
79     //if list is empty
80     if(head == NULL) {
81         return NULL;
82     }
83
84     //navigate through list
85     while(current->key != key) {
86
87         //if it is last node
88         if(current->next == NULL) {
89             return NULL;
90         } else {
91             //go to next link
92             current = current->next;
93         }
94     }
95
96     //if data found, return the current Link
97     return current;
98 }
```

Linked List - Delete a Link with a Given Key

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <stdlib.h>
4  #include <stdbool.h>
5
6  struct node {
7      int data;
8      int key;
9      struct node *next;
10 };
11
12 struct node *head = NULL;
13 struct node *current = NULL;
```

```
100 //delete a link with given key
101 struct node* delete(int key) {
102
103     //start from the first link
104     struct node* current = head;
105     struct node* previous = NULL;
106
107     //if list is empty
108     if(head == NULL) {
109         return NULL;
110     }
111
112     //navigate through list
113     while(current->key != key) {
114
115         //if it is last node
116         if(current->next == NULL) {
117             return NULL;
118         } else {
119             //store reference to current link
120             previous = current;
121             //move to next link
122             current = current->next;
123         }
124     }
125
```

```
125
126     //found a match, update the link
127     if(current == head) {
128         //change first to point to next link
129         head = head->next;
130     } else {
131         //bypass the current link
132         previous->next = current->next;
133     }
134
135     return current;
136 }
137
```

Linked List - Delete a Link with a Given Key

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <stdlib.h>
4  #include <stdbool.h>
5
6  struct node {
7      int data;
8      int key;
9      struct node *next;
10 };
11
12 struct node *head = NULL;
13 struct node *current = NULL;
```

```
struct node *foundLink = find(4);

if(foundLink != NULL) {
    printf("Element found: ");
    printf("(%d,%d) ", foundLink->key, foundLink->data);
    printf("\n");
} else {
    printf("Element not found.");
}

delete(4);
printf("List after deleting an item: ");
printList();
printf("\n");
foundLink = find(4);

if(foundLink != NULL) {
    printf("Element found: ");
    printf("(%d,%d) ", foundLink->key, foundLink->data);
    printf("\n");
} else {
    printf("Element not found.");
}
```


Linked List - Delete a Link with a Given Key

OUTPUT

```
Element found: (4,1)
List after deleting an item:
[ (6,56) (5,40) (3,30) (2,20) (1,10) ]
Element not found.
```

```
struct node *foundLink = find(4);

if(foundLink != NULL) {
    printf("Element found: ");
    printf("(%d,%d) ", foundLink->key, foundLink->data);
    printf("\n");
} else {
    printf("Element not found.");
}

delete(4);
printf("List after deleting an item: ");
printList();
printf("\n");
foundLink = find(4);

if(foundLink != NULL) {
    printf("Element found: ");
    printf("(%d,%d) ", foundLink->key, foundLink->data);
    printf("\n");
} else {
    printf("Element not found.");
}
```