

linkedList

0.0.1

Generated by Doxygen 1.8.17



|  |          |
|--|----------|
| <b>1 Class Index</b>                         | <b>1</b> |
| 1.1 Class List                               | 1        |
| <b>2 File Index</b>                          | <b>3</b> |
| 2.1 File List                                | 3        |
| <b>3 Class Documentation</b>                 | <b>5</b> |
| 3.1 DLL Class Reference                      | 5        |
| 3.1.1 Detailed Description                   | 6        |
| 3.1.2 Constructor & Destructor Documentation | 6        |
| 3.1.2.1 DLL()                                | 6        |
| 3.1.3 Member Function Documentation          | 6        |
| 3.1.3.1 addHead()                            | 6        |
| 3.1.3.2 addMiddle()                          | 7        |
| 3.1.3.3 addToTail()                          | 7        |
| 3.1.3.4 get()                                | 8        |
| 3.1.3.5 printList()                          | 8        |
| 3.1.3.6 removeHead()                         | 8        |
| 3.1.3.7 removeTail()                         | 9        |
| 3.1.4 Member Data Documentation              | 9        |
| 3.1.4.1 head                                 | 9        |
| 3.1.4.2 n                                    | 9        |
| 3.2 DLNode Class Reference                   | 9        |
| 3.2.1 Detailed Description                   | 10       |
| 3.2.2 Constructor & Destructor Documentation | 10       |
| 3.2.2.1 DLNode()                             | 10       |
| 3.2.3 Member Data Documentation              | 10       |
| 3.2.3.1 data                                 | 10       |
| 3.2.3.2 nextNode                             | 10       |
| 3.2.3.3 prevNode                             | 11       |
| 3.3 Node Class Reference                     | 11       |
| 3.3.1 Detailed Description                   | 11       |
| 3.3.2 Constructor & Destructor Documentation | 11       |
| 3.3.2.1 Node()                               | 11       |
| 3.3.3 Member Data Documentation              | 12       |
| 3.3.3.1 data                                 | 12       |
| 3.3.3.2 nextNode                             | 12       |
| 3.4 SLL Class Reference                      | 12       |
| 3.4.1 Detailed Description                   | 13       |
| 3.4.2 Constructor & Destructor Documentation | 13       |
| 3.4.2.1 SLL()                                | 13       |
| 3.4.3 Member Function Documentation          | 13       |
| 3.4.3.1 addHead()                            | 13       |

---

|   |           |
|---|-----------|
| 3.4.3.2 addMiddle()   | 14        |
| 3.4.3.3 addToTail()   | 14        |
| 3.4.3.4 get()   | 14        |
| 3.4.3.5 printList()   | 15        |
| 3.4.3.6 removeHead()  | 15        |
| 3.4.3.7 removeTail()  | 16        |
| 3.4.4 Member Data Documentation                                   | 16        |
| 3.4.4.1 head  | 16        |
| 3.4.4.2 n   | 16        |
| 3.4.4.3 tail  | 16        |
| <b>4 File Documentation</b>                                       | <b>17</b> |
| 4.1 /home/addis/linkedList/linkedList/src/main.cpp File Reference | 17        |
| 4.1.1 Detailed Description  | 18        |
| 4.1.2 Function Documentation                                      | 18        |
| 4.1.2.1 main()  | 18        |
| 4.2 /home/addis/linkedList/linkedList/src/SLL.cpp File Reference  | 19        |
| 4.2.1 Function Documentation                                      | 20        |
| 4.2.1.1 main()  | 20        |
| <b>Index</b>  | <b>23</b> |

# Chapter 1

## Class Index

### 1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

|                        |    |
|------------------------|----|
| <a href="#">DLL</a>    | 5  |
| <a href="#">DLNode</a> | 9  |
| <a href="#">Node</a>   | 11 |
| <a href="#">SLL</a>    | 12 |



## Chapter 2

# File Index

### 2.1 File List

Here is a list of all files with brief descriptions:

|  |                    |
|--|--------------------|
| /home/addis/linkedList/linkedList/src/ <a href="#">main.cpp</a>          |                    |
| This is a test of CMake, doxygen, and GitHub . . . . .                   | <a href="#">17</a> |
| /home/addis/linkedList/linkedList/src/ <a href="#">SLL.cpp</a> . . . . . | <a href="#">19</a> |



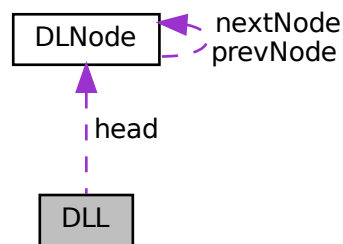


## Chapter 3

# Class Documentation

### 3.1 DLL Class Reference

Collaboration diagram for DLL:



#### Public Member Functions

- `DLL ()`
- `bool addHead (int d)`
- `bool addToTail (int d)`
- `bool removeTail (int d)`
- `int get (int ii)`
- `bool addMiddle (int ii, int d)`
- `bool removeHead (int &d)`
- `void printList ()`

#### Public Attributes

- `DLNode * head`
- `int n`

### 3.1.1 Detailed Description

Definition at line 28 of file main.cpp.

### 3.1.2 Constructor & Destructor Documentation

#### 3.1.2.1 DLL()

```
DLL::DLL ( ) [inline]
```

Definition at line 33 of file main.cpp.

```
33     {  
34         head = NULL;  
35         n = 0;  
36     }
```

### 3.1.3 Member Function Documentation

#### 3.1.3.1 addHead()

```
bool DLL::addHead (  
    int d ) [inline]
```

Definition at line 38 of file main.cpp.

```
38     {  
39         DLNode* newNode = new DLNode(d);  
40         if (n == 0) { // the list is empty  
41             head = newNode;  
42         }  
43         else {  
44             newNode->nextNode = head;  
45             head->prevNode = newNode;  
46             head = newNode;  
47         }  
48     }  
49     n++;  
50     return(true);  
51 }  
52 }
```

## 3.1.3.2 addMiddle()

```
bool DLL::addMiddle (
    int ii,
    int d ) [inline]
```

Definition at line 113 of file main.cpp.

```
113         {
114             DLNode* curNode;
115             DLNode* newNode = new DLNode(d);
116             if (head == NULL) { // the list is empty
117                 return(false);
118             }
119             else if (ii >= n) {
120                 cout << "ERROR: Asked for node beyond tail" << endl;
121                 return(false);
122             }
123             else if (ii < 0) {
124                 cout << "ERROR: Asked for negative index" << endl;
125                 return(false);
126             }
127             else {
128                 curNode = head;
129                 // traverse list to desired node
130                 for (int jj = 0; jj < ii; jj++) {
131                     curNode = curNode->nextNode;
132                 }
133                 newNode->nextNode = curNode->nextNode;
134                 newNode->prevNode = curNode;
135                 curNode->nextNode = newNode;
136                 if (newNode->nextNode != NULL) {
137                     newNode->nextNode->prevNode = newNode;
138                 }
139                 n++;
140                 return(true);
141             }
142         }
```

## 3.1.3.3 addToTail()

```
bool DLL::addToTail (
    int d ) [inline]
```

Definition at line 54 of file main.cpp.

```
54         {
55             DLNode* newNode = new DLNode(d);
56             if (n == 0) { // the list is empty
57                 head = newNode;
58             }
59             else {
60                 DLNode* value = head;
61                 while (value->nextNode != NULL) {
62                     value = value->nextNode;
63                 }
64                 newNode->prevNode = value;
65                 newNode->nextNode = NULL;
66                 value->nextNode = newNode;
67             }
68             n++;
69             return(true);
70         }
```

### 3.1.3.4 get()

```
int DLL::get (
    int ii ) [inline]
```

Definition at line 90 of file main.cpp.

```
90     {
91         DLNode* curNode;
92         if (head == NULL) { // the list is empty
93             return(-999999);
94         }
95         else if (ii >= n) {
96             cout << "ERROR: Asked for node beyond tail" << endl;
97             return(-999998);
98         }
99         else if (ii < 0) {
100             cout << "ERROR: Asked for negative index" << endl;
101             return(-999997);
102         }
103         else {
104             curNode = head;
105             // traverse list to desired node
106             for (int jj = 0; jj < ii; jj++) {
107                 curNode = curNode->nextNode;
108             }
109             return(curNode->data);
110         }
111     }
```

### 3.1.3.5 printList()

```
void DLL::printList ( ) [inline]
```

Definition at line 162 of file main.cpp.

```
162     {
163         DLNode* curNode;
164         if (head == NULL) { // the list is empty
165             cout << "Empty list" << endl;
166         }
167         else { // the list is not empty
168             curNode = head; // start at the beginning
169             while (curNode->nextNode != NULL) {
170                 cout << curNode->data << " -> ";
171                 curNode = curNode->nextNode; // update to next node
172             }
173             cout << curNode->data;
174             cout << endl;
175         }
176     }
```

### 3.1.3.6 removeHead()

```
bool DLL::removeHead (
    int & d ) [inline]
```

Definition at line 144 of file main.cpp.

```
144     {
145         int val;
146         DLNode* old; // save off the old node
147         if (head != NULL) {
148             val = head->data; // collect the data from node to be removed
149             old = head; // save off pointer to node we are removing
150             head = head->nextNode; // update head to new node
151             head->prevNode = NULL;
152             delete old; // release the memory from the removed node
153             n--; // decrement n to show shorter list
154             d = val;
155             return(true);
156         }
157         else { //list is empty
158             return(false);
159         }
160     }
```

### 3.1.3.7 removeTail()

```
bool DLL::removeTail (
    int d ) [inline]
```

Definition at line 72 of file main.cpp.

```
72     {
73         DLNode* var = head->nextNode;
74
75         if (head->nextNode == NULL) {
76             return(false);
77         }
78         while (var->nextNode->nextNode != NULL) {
79             var = var->nextNode;
80         }
81         d = var->nextNode->data;
82         delete var->nextNode;
83         var->nextNode = NULL;
84
85         n--;
86         return(true);
87     }
88 }
```

## 3.1.4 Member Data Documentation

### 3.1.4.1 head

```
DLNode* DLL::head
```

Definition at line 30 of file main.cpp.

### 3.1.4.2 n

```
int DLL::n
```

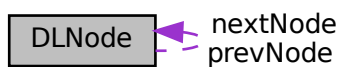
Definition at line 31 of file main.cpp.

The documentation for this class was generated from the following file:

- </home/addis/linkedlist/linked-list/src/main.cpp>

## 3.2 DLNode Class Reference

Collaboration diagram for DLNode:



## Public Member Functions

- [DLNode](#) (int d)

## Public Attributes

- int [data](#)
- [DLNode](#) \* [nextNode](#)
- [DLNode](#) \* [prevNode](#)

### 3.2.1 Detailed Description

Definition at line 15 of file main.cpp.

### 3.2.2 Constructor & Destructor Documentation

#### 3.2.2.1 DLNode()

```
DLNode::DLNode (  
    int d ) [inline]
```

Definition at line 21 of file main.cpp.

```
21     {  
22         data = d;  
23         nextNode = NULL;  
24         prevNode = NULL;  
25     }
```

### 3.2.3 Member Data Documentation

#### 3.2.3.1 data

```
int DLNode::data
```

Definition at line 17 of file main.cpp.

#### 3.2.3.2 nextNode

```
DLNode* DLNode::nextNode
```

Definition at line 18 of file main.cpp.

### 3.2.3.3 prevNode

`DLNode*` `DLNode::prevNode`

Definition at line 19 of file `main.cpp`.

The documentation for this class was generated from the following file:

- `/home/addis/linkedList/linkedList/src/main.cpp`

## 3.3 Node Class Reference

Collaboration diagram for Node:



### Public Member Functions

- `Node` (`int d`)

### Public Attributes

- `int data`
- `Node *` `nextNode`

### 3.3.1 Detailed Description

Definition at line 4 of file `SLL.cpp`.

### 3.3.2 Constructor & Destructor Documentation

#### 3.3.2.1 Node()

```
Node::Node (  
    int d ) [inline]
```

Definition at line 9 of file `SLL.cpp`.

```
9      {  
10         data = d;  
11         nextNode = NULL;  
12     }
```

### 3.3.3 Member Data Documentation

#### 3.3.3.1 data

```
int Node::data
```

Definition at line 6 of file SLL.cpp.

#### 3.3.3.2 nextNode

```
Node* Node::nextNode
```

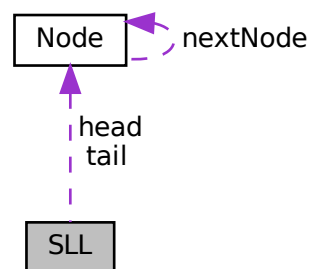
Definition at line 7 of file SLL.cpp.

The documentation for this class was generated from the following file:

- [/home/addis/linkedList/linkedList/src/SLL.cpp](#)

## 3.4 SLL Class Reference

Collaboration diagram for SLL:



### Public Member Functions

- [SLL](#) ()
- bool [addHead](#) (int d)
- bool [addToTail](#) (int d)
- bool [removeTail](#) (int d)
- int [get](#) (int ii)
- bool [addMiddle](#) (int ii, int d)
- bool [removeHead](#) (int &d)
- void [printList](#) ()



## Public Attributes

- `Node * head`
- `Node * tail`
- `int n`

### 3.4.1 Detailed Description

Definition at line 15 of file SLL.cpp.

### 3.4.2 Constructor & Destructor Documentation

#### 3.4.2.1 SLL()

```
SLL::SLL ( ) [inline]
```

Definition at line 21 of file SLL.cpp.

```
21     {
22         head = NULL;
23         tail = NULL;
24         n = 0;
25     }
```

### 3.4.3 Member Function Documentation

#### 3.4.3.1 addHead()

```
bool SLL::addHead (
    int d ) [inline]
```

Definition at line 27 of file SLL.cpp.

```
27     {
28         Node* newNode = new Node(d);
29         if (n == 0) { // the list is empty
30             head = newNode;
31         }
32         else {
33             newNode->nextNode = head;
34             head = newNode;
35         }
36         n++;
37         return(true);
38     }
```

### 3.4.3.2 addMiddle()

```
bool SLL::addMiddle (
    int ii,
    int d ) [inline]
```

Definition at line 91 of file SLL.cpp.

```
91     {
92         Node* curNode;
93         Node* newNode = new Node(d);
94         if(head == NULL) { // the list is empty
95             return(false);
96         } else if(ii >= n) {
97             cout << "ERROR: Asked for node beyond tail" << endl;
98             return(false);
99         } else if(ii < 0) {
100             cout << "ERROR: Asked for negative index" << endl;
101             return(false);
102         } else {
103             curNode = head;
104             // traverse list to desired node
105             for(int jj = 0; jj < ii; jj++) {
106                 curNode = curNode->nextNode;
107             }
108             // At this point curNode points to the node we want to add after
109             newNode->nextNode = curNode->nextNode;
110             curNode->nextNode = newNode;
111             n++;
112             return(true);
113         }
114     }
```

### 3.4.3.3 addToTail()

```
bool SLL::addToTail (
    int d ) [inline]
```

Definition at line 40 of file SLL.cpp.

```
40     {
41         Node* newNode = new Node(d);
42         if(n == 0) { // the list is empty
43             head = newNode;
44             tail = newNode;
45         } else {
46             tail->nextNode = newNode; // update the last node's next node to newNode
47             tail = newNode; // update the tail pointer to newNode
48         }
49         n++;
50         return(true);
51     }
```

### 3.4.3.4 get()

```
int SLL::get (
    int ii ) [inline]
```

Definition at line 71 of file SLL.cpp.

```
71     {
72         Node* curNode;
73         if(head == NULL) { // the list is empty
74             return(-999999);
75         } else if(ii >= n) {
76             cout << "ERROR: Asked for node beyond tail" << endl;
77             return(-999998);
78         } else if(ii < 0) {
```

```

79         cout << "ERROR: Asked for negative index" << endl;
80         return(-999997);
81     } else {
82         curNode = head;
83         // traverse list to desired node
84         for(int jj = 0; jj < ii; jj++) {
85             curNode = curNode->nextNode;
86         }
87         return(curNode->data);
88     }
89 }

```

### 3.4.3.5 printList()

```
void SLL::printList ( ) [inline]
```

Definition at line 132 of file SLL.cpp.

```

132     {
133         Node* curNode;
134         if(head == NULL) { // the list is empty
135             cout << "Empty list" << endl;
136         } else { // the list is not empty
137             curNode = head; // start at the beginning
138             while(curNode->nextNode != NULL){
139                 cout << curNode->data << " -> ";
140                 curNode = curNode->nextNode; // update to next node
141             }
142             cout << curNode->data;
143             cout << endl;
144         }
145     }

```

### 3.4.3.6 removeHead()

```
bool SLL::removeHead (
    int & d ) [inline]
```

Definition at line 116 of file SLL.cpp.

```

116     {
117         int val;
118         Node* old; // save off the old node
119         if(head != NULL) {
120             val = head->data; // collect the data from node to be removed
121             old = head; // save off pointer to node we are removing
122             head = head->nextNode; // update head to new node
123             delete old; // release the memory from the removed node
124             n--; // decrement n to show shorter list
125             d = val;
126             return(true);
127         } else { //list is empty
128             return(false);
129         }
130     }

```

### 3.4.3.7 removeTail()

```
bool SLL::removeTail (
    int d ) [inline]
```

Definition at line 53 of file SLL.cpp.

```
53     {
54         Node* var = head->nextNode;
55
56         if (head->nextNode == NULL) {
57             return(false);
58         }
59         while (var->nextNode->nextNode != NULL) {
60             var = var->nextNode;
61         }
62         d = var->nextNode->data;
63         delete var->nextNode;
64         var->nextNode = NULL;
65
66         n--;
67         return(true);
68     }
69 }
```

## 3.4.4 Member Data Documentation

### 3.4.4.1 head

`Node*` SLL::head

Definition at line 17 of file SLL.cpp.

### 3.4.4.2 n

`int` SLL::n

Definition at line 19 of file SLL.cpp.

### 3.4.4.3 tail

`Node*` SLL::tail

Definition at line 18 of file SLL.cpp.

The documentation for this class was generated from the following file:

- [/home/addis/linkedList/linkedList/src/SLL.cpp](#)

## Chapter 4

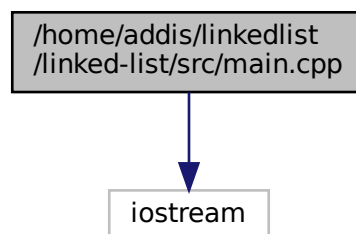
# File Documentation

### 4.1 /home/addis/linkedList/linkedList/src/main.cpp File Reference

This is a test of CMake, doxygen, and GitHub.

```
#include <iostream>
```

Include dependency graph for main.cpp:



#### Classes

- class [DLNode](#)
- class [DLL](#)

#### Functions

- int [main](#) (int, char \*\*)

### 4.1.1 Detailed Description

This is a test of CMake, doxygen, and GitHub.

This is the long brief at the top of [main.cpp](#).

#### Author

Seth McNeill

#### Date

1/28/2021

### 4.1.2 Function Documentation

#### 4.1.2.1 main()

```
int main (
    int ,
    char ** )
```

Definition at line 179 of file main.cpp.

```
179         {
180             DLL myList;
181             int retData; // for data from remove
182
183             myList.printList();
184             myList.addToTail(1);
185             myList.printList();
186             myList.addToTail(2);
187             myList.printList();
188             myList.addToTail(3);
189             myList.printList();
190             myList.addToTail(4);
191             myList.printList();
192             myList.addToTail(5);
193             myList.printList();
194
195             cout << "get(0) = " << myList.get(0) << endl;
196             cout << "get(1) = " << myList.get(1) << endl;
197             cout << "get(4) = " << myList.get(4) << endl;
198             cout << "get(5) = " << myList.get(5) << endl;
199             cout << "get(7) = " << myList.get(7) << endl;
200             cout << "get(-3) = " << myList.get(-3) << endl;
201
202             myList.addMiddle(3, 10);
203             myList.printList();
204             myList.addMiddle(3, 11);
205             myList.printList();
206             myList.addMiddle(6, 12);
207             myList.printList();
208             myList.addMiddle(0, 13);
209             myList.printList();
210
211             myList.printList();
212             if (myList.addHead(retData))
213                 cout << "addedtohead " << retData << endl;
214             else
215                 cout << "list was empty" << endl;
216
217             if (myList.addToTail(retData))
218                 cout << "RemovedTail " << retData << endl;
219             else
220                 cout << "list was empty" << endl;
221         }
```

```

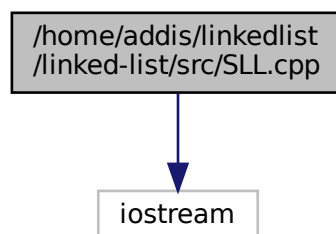
222     myList.printList();
223     if (myList.removeHead(retData))
224         cout << "Removed " << retData << endl;
225     else
226         cout << "list was empty" << endl;
227     myList.printList();
228     if (myList.removeTail(retData))
229         cout << "Removed Tail " << retData << endl;
230
231     myList.printList();
232     if (myList.removeHead(retData))
233         cout << "Removed " << retData << endl;
234     else
235         cout << "list was empty" << endl;
236     myList.printList();
237     if (myList.removeHead(retData))
238         cout << "Removed " << retData << endl;
239     else
240         cout << "list was empty" << endl;
241     myList.printList();
242     if (myList.removeHead(retData))
243         cout << "Removed " << retData << endl;
244     else
245         cout << "list was empty" << endl;
246     myList.printList();
247     if (myList.removeHead(retData))
248         cout << "Removed " << retData << endl;
249     else
250         cout << "list was empty" << endl;
251     myList.printList();
252     if (myList.removeHead(retData))
253         cout << "Removed " << retData << endl;
254     else
255         cout << "list was empty" << endl;
256     myList.printList();
257     if (myList.removeHead(retData))
258         cout << "Removed " << retData << endl;
259     else
260         cout << "list was empty" << endl;
261     myList.printList();
262     if (myList.removeHead(retData))
263         cout << "Removed " << retData << endl;
264     else
265         cout << "list was empty" << endl;
266     myList.printList();
267 }

```

## 4.2 /home/addis/linkedList/linkedList/src/SLL.cpp File Reference

```
#include <iostream>
```

Include dependency graph for SLL.cpp:



### Classes

- class [Node](#)
- class [SLL](#)

## Functions

- int [main](#) (int, char \*\*)

### 4.2.1 Function Documentation

#### 4.2.1.1 main()

```
int main (
    int ,
    char ** )
```

Definition at line 148 of file SLL.cpp.

```
148         {
149     SLL myList;
150     int retData; // for data from remove
151
152     myList.printList();
153     myList.addToTail(1);
154     myList.printList();
155     myList.addToTail(2);
156     myList.printList();
157     myList.addToTail(3);
158     myList.printList();
159     myList.addToTail(4);
160     myList.printList();
161     myList.addToTail(5);
162     myList.printList();
163
164     cout << "get (0) = " << myList.get(0) << endl;
165     cout << "get (1) = " << myList.get(1) << endl;
166     cout << "get (4) = " << myList.get(4) << endl;
167     cout << "get (5) = " << myList.get(5) << endl;
168     cout << "get (7) = " << myList.get(7) << endl;
169     cout << "get (-3) = " << myList.get(-3) << endl;
170
171     myList.addMiddle(3,10);
172     myList.printList();
173     myList.addMiddle(3,11);
174     myList.printList();
175     myList.addMiddle(6,12);
176     myList.printList();
177     myList.addMiddle(0,13);
178     myList.printList();
179
180
181     if(myList.removeHead(retData))
182         cout << "Removed " << retData << endl;
183     else
184         cout << "list was empty" << endl;
185     myList.printList();
186     if(myList.removeHead(retData))
187         cout << "Removed " << retData << endl;
188     else
189         cout << "list was empty" << endl;
190     myList.printList();
191     if(myList.removeHead(retData))
192         cout << "Removed " << retData << endl;
193     else
194         cout << "list was empty" << endl;
195     myList.printList();
196     if(myList.removeHead(retData))
197         cout << "Removed " << retData << endl;
198     else
199         cout << "list was empty" << endl;
200     myList.printList();
201     if(myList.removeHead(retData))
202         cout << "Removed " << retData << endl;
203     else
204         cout << "list was empty" << endl;
205     myList.printList();
206     if(myList.removeHead(retData))
207         cout << "Removed " << retData << endl;
```



```
208     else
209         cout << "list was empty" << endl;
210     myList.printList();
211     if(myList.removeHead(retData))
212         cout << "Removed " << retData << endl;
213     else
214         cout << "list was empty" << endl;
215     myList.printList();
216     if(myList.removeHead(retData))
217         cout << "Removed " << retData << endl;
218     else
219         cout << "list was empty" << endl;
220     myList.printList();
221 }
```



# Index

/home/addis/linkedList/linkedList/src/SLL.cpp, [19](#)  
/home/addis/linkedList/linkedList/src/main.cpp, [17](#)

addHead

DLL, [6](#)  
SLL, [13](#)

addMiddle

DLL, [6](#)  
SLL, [13](#)

addToTail

DLL, [7](#)  
SLL, [14](#)

data

DLNode, [10](#)  
Node, [12](#)

DLL, [5](#)

addHead, [6](#)  
addMiddle, [6](#)  
addToTail, [7](#)  
DLL, [6](#)  
get, [7](#)  
head, [9](#)  
n, [9](#)  
printList, [8](#)  
removeHead, [8](#)  
removeTail, [8](#)

DLNode, [9](#)

data, [10](#)  
DLNode, [10](#)  
nextNode, [10](#)  
prevNode, [10](#)

get

DLL, [7](#)  
SLL, [14](#)

head

DLL, [9](#)  
SLL, [16](#)

main

main.cpp, [18](#)  
SLL.cpp, [20](#)

main.cpp

main, [18](#)

n

DLL, [9](#)  
SLL, [16](#)

nextNode

DLNode, [10](#)

Node, [12](#)

Node, [11](#)

data, [12](#)  
nextNode, [12](#)  
Node, [11](#)

prevNode

DLNode, [10](#)

printList

DLL, [8](#)  
SLL, [15](#)

removeHead

DLL, [8](#)  
SLL, [15](#)

removeTail

DLL, [8](#)  
SLL, [15](#)

SLL, [12](#)

addHead, [13](#)  
addMiddle, [13](#)  
addToTail, [14](#)  
get, [14](#)  
head, [16](#)  
n, [16](#)  
printList, [15](#)  
removeHead, [15](#)  
removeTail, [15](#)  
SLL, [13](#)  
tail, [16](#)

SLL.cpp

main, [20](#)

tail

SLL, [16](#)