Class Demo Singly Linked List

0.1.0

Generated by Doxygen 1.8.17

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Class Documentation

3.1 Node Class Reference

Collaboration diagram for Node:

Public Member Functions

- Node (int d)

Public Attributes

- int data
- Node * nextNode

3.1.1 Detailed Description

Definition at line 14 of file main.cpp.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 Node()

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

Constructor

Definition at line 22 of file main.cpp.

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

3.1.3 Member Data Documentation

3.1.3.1 data

```
int Node::data
```

Definition at line 16 of file main.cpp.

3.1.3.2 nextNode

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

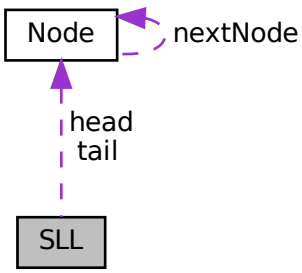
Definition at line 17 of file main.cpp.

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

- /home/drseth/CPTR227/20210208-SLLClassDemo/src/main.cpp

3.2 SLL Class Reference

Collaboration diagram for SLL:



Public Member Functions

- SLL ()
- bool addToTail (int d)
- int get (int ii)
- bool addMiddle (int ii, int d)
- bool removeHead (int &d)
- int length ()
- void printList ()

Public Attributes

- Node * head
- Node * tail
- int n

3.2.1 Detailed Description

Definition at line 28 of file main.cpp.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 SLL()

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

Constructor

Definition at line 37 of file main.cpp.

```
37           {
38         head = NULL;
39         tail = NULL;
40         n = 0;
41     }
```

3.2.3 Member Function Documentation

3.2.3.1 addMiddle()

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

Adds node after the iith node

Parameters

ii	the node to insert after
d	the data in the new node

Returns

true if successful

Definition at line 91 of file main.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      }
```

3.2.3.2 addToTail()

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

Adds node to tail of list

Definition at line 46 of file main.cpp.

```
46                          {
47          Node* newNode = new Node(d);
48          if(n == 0) { // the list is empty
49              head = newNode;
50              tail = newNode;
51          } else {
52              tail->nextNode = newNode; // update the last node's next node to newNode
53              tail = newNode; // update the tail pointer to newNode
54          }
55          n++;
56          return(true);
57      }
```

3.2.3.3 get()

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

Returns the data from the iith node

Parameters

ii	the number of the node to collect data from

Definition at line 64 of file main.cpp.

```
64                  {
65          Node* curNode;
66          if(head == NULL) { // the list is empty
67              return(-999999);
68          } else if(ii >= n) {
69              cout « "ERROR: Asked for node beyond tail" « endl;
70              return(-999998);
71          } else if(ii < 0) {
72              cout « "ERROR: Asked for negative index" « endl;
73              return(-999997);
74          } else {
75              curNode = head;
76              // traverse list to desired node
77              for(int jj = 0; jj < ii; jj++) {
78                  curNode = curNode->nextNode;
79              }
80              return(curNode->data);
81          }
82      }
```

3.2.3.4 length()

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

Returns the length of the list

Definition at line 141 of file main.cpp.

```
141                 {
142          return(n);
143      }
```

3.2.3.5 printList()

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

Prints the list to stdout

Definition at line 148 of file main.cpp.

```
148                     {
149          Node* curNode;
150          if(head == NULL) { // the list is empty
151              cout « "Empty list" « endl;
152          } else { // the list is not empty
153              curNode = head; // start at the beginning
154              while(curNode->nextNode != NULL){
155                  cout « curNode->data « " -> ";
156                  curNode = curNode->nextNode; // update to next node
157              }
158              cout « curNode->data;
159              cout « endl;
160          }
161      }
```

3.2.3.6 removeHead()

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

Removes the head node and returns the data value from the removed node

Parameters

d	pointer to integer to return value

Returns

true if successful

Definition at line 122 of file main.cpp.

```
122                             {
123         int val;
124         Node* old; // save off the old node
125         if(head != NULL) {
126             val = head->data; // collect the data from node to be removed
127             old = head; // save off pointer to node we are removing
128             head = head->nextNode; // update head to new node
129             delete old; // release the memory from the removed node
130             n--; // decrement n to show shorter list
131             d = val;
132             return(true);
133         } else { //list is empty
134             return(false);
135         }
136     }
```

3.2.4 Member Data Documentation

3.2.4.1 head

```
Node* SLL::head
```

Definition at line 30 of file main.cpp.

3.2.4.2 n

```
int SLL::n
```

Definition at line 32 of file main.cpp.

3.2.4.3 tail

```
Node* SLL::tail
```

Definition at line 31 of file main.cpp.

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

- /home/drseth/CPTR227/20210208-SLLClassDemo/src/main.cpp

Chapter 4

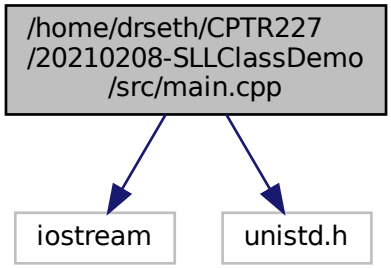
File Documentation

4.1 /home/drseth/CPTR227/20210208-SLLClassDemo/src/main.cpp File Reference

This is a demo of making a singly linked list.

```
#include <iostream>
#include <unistd.h>
```

Include dependency graph for main.cpp:



Classes

- class Node
- class SLL

Functions

- int main (int, char **)

4.1.1 Detailed Description

This is a demo of making a singly linked list.

Based on ODS book examples

Author

Seth McNeill

Date

2021 February 08

4.1.2 Function Documentation

4.1.2.1 main()

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

Definition at line 164 of file main.cpp.

```
164                         {
165     SLL myList;
166     int retData; // for data from remove
167     long int nTimes = 10000000; // number of items to work with
168 
169     cout « "Process ID: " « getpid() « endl;
170     cout « "Enter a number and press enter to continue";
171     cin » retData;
172 
173     for(long int ii = 0; ii < nTimes; ii++) {
174         myList.addToTail(1);
175     }
176 
177     cout « "Create list " « myList.length() « " long" « endl;
178     cout « "Enter a number and press enter to continue: ";
179     cin » retData;
180 //    cout « "get(1) = " « myList.get(1) « endl;
181 
182 //    myList.addMiddle(3,10);
183 
184     for(int ii = 0; ii < nTimes; ii++) {
185         myList.removeHead(retData);
186     }
187     cout « "List is now " « myList.length() « " long" « endl;
188     cout « "Enter a number and press enter to continue: ";
189     cin » retData;
190 }
```

Index