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 16 of file main.cpp.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 Node()

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

Constructor

Definition at line 24 of file main.cpp.

```
24                  {
25          data = d;
26          nextNode = NULL;
27      }
```

3.1.3 Member Data Documentation

3.1.3.1 data

```
int Node::data
```

Definition at line 18 of file main.cpp.

3.1.3.2 nextNode

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

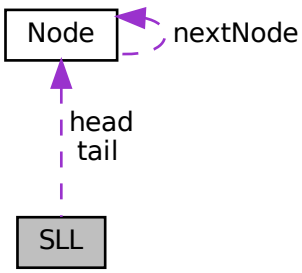
Definition at line 19 of file main.cpp.

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 30 of file main.cpp.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 SLL()

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

Constructor

Definition at line 39 of file main.cpp.

```
39           {
40         head = NULL;
41         tail = NULL;
42         n = 0;
43     }
```

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 93 of file main.cpp.

```
93                                  {
94          Node* curNode;
95          Node* newNode = new Node(d);
96          if(head == NULL) { // the list is empty
97              return(false);
98          } else if(ii >= n) {
99              cout « "ERROR: Asked for node beyond tail" « endl;
100              return(false);
101          } else if(ii < 0) {
102              cout « "ERROR: Asked for negative index" « endl;
103              return(false);
104          } else {
105              curNode = head;
106              // traverse list to desired node
107              for(int jj = 0; jj < ii; jj++) {
108                  curNode = curNode->nextNode;
109              }
110              // At this point curNode points to the node we want to add after
111              newNode->nextNode = curNode->nextNode;
112              curNode->nextNode = newNode;
113              n++;
114              return(true);
115          }
116      }
```

3.2.3.2 addToTail()

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

Adds node to tail of list

Definition at line 48 of file main.cpp.

```
48                          {
49          Node* newNode = new Node(d);
50          if(n == 0) { // the list is empty
51              head = newNode;
52              tail = newNode;
53          } else {
54              tail->nextNode = newNode; // update the last node's next node to newNode
55              tail = newNode; // update the tail pointer to newNode
56          }
57          n++;
58          return(true);
59      }
```

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 66 of file main.cpp.

```
66                      {
67          Node* curNode;
68          if(head == NULL) { // the list is empty
69              return(-999999);
70          } else if(ii >= n) {
71              cout « "ERROR: Asked for node beyond tail" « endl;
72              return(-999998);
73          } else if(ii < 0) {
74              cout « "ERROR: Asked for negative index" « endl;
75              return(-999997);
76          } else {
77              curNode = head;
78              // traverse list to desired node
79              for(int jj = 0; jj < ii; jj++) {
80                  curNode = curNode->nextNode;
81              }
82              return(curNode->data);
83          }
84      }
```

3.2.3.4 length()

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

returns length of list

Definition at line 143 of file main.cpp.

```
143                     {
144          return(n);
145      }
```

3.2.3.5 printList()

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

Prints the list to stdout

Definition at line 150 of file main.cpp.

```
150                       {
151          Node* curNode;
152          if(head == NULL) { // the list is empty
153              cout « "Empty list" « endl;
154          } else { // the list is not empty
155              curNode = head; // start at the beginning
156              while(curNode->nextNode != NULL){
157                  cout « curNode->data « " -> ";
158                  curNode = curNode->nextNode; // update to next node
159              }
160              cout « curNode->data;
161              cout « endl;
162          }
163      }
```

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 124 of file main.cpp.

```
124                             {
125         int val;
126         Node* old; // save off the old node
127         if(head != NULL) {
128             val = head->data; // collect the data from node to be removed
129             old = head; // save off pointer to node we are removing
130             head = head->nextNode; // update head to new node
131             delete old; // release the memory from the removed node
132             n--; // decrement n to show shorter list
133             d = val;
134             return(true);
135         } else { //list is empty
136             return(false);
137         }
138     }
```

3.2.4 Member Data Documentation

3.2.4.1 head

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

Definition at line 32 of file main.cpp.

3.2.4.2 n

```
int SLL::n
```

Definition at line 34 of file main.cpp.

3.2.4.3 tail

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

Definition at line 33 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

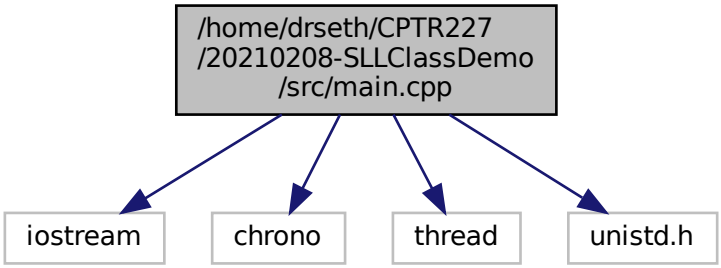
File Documentation

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 <chrono>
#include <thread>
#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 166 of file main.cpp.

```
166                       {
167     SLL myList;
168     int retData; // for data from remove
169     int nTimes = 1000000;
170 
171     cout « "Process ID: " « getpid() « endl;
172     cout « "Enter number and press enter to continue";
173     cin » retData;
174 
175     for(int ii = 0; ii < nTimes; ii++) {
176         myList.addToTail(ii);
177         //this_thread::sleep_for(chrono::microseconds(1));
178     }
179 
180     cout « "Finished adding elements, Enter number and press enter to continue";
181     cin » retData;
182 
183 //    cout « "get(1) = " « myList.get(1) « endl;
184 
185 //    myList.addMiddle(myList.length()/2,10);
186 
187     for(int ii = 0; ii < nTimes; ii++) {
188         myList.removeHead(retData);
189         //this_thread::sleep_for(chrono::microseconds(1));
190     }
191     cout « "Removed all elements. Enter number and press enter to continue";
192     cin » retData;
193 }
```

Index