

Roy's Linked List documentation

Generated by Doxygen 1.9.2

1 Data Structure Index	1
1.1 Data Structures	1
2 File Index	3
2.1 File List	3
3 Data Structure Documentation	5
3.1 LinkedList Struct Reference	5
3.1.1 Field Documentation	5
3.1.1.1 count	5
3.1.1.2 head	5
3.2 listNode Struct Reference	5
3.2.1 Detailed Description	6
3.2.2 Field Documentation	6
3.2.2.1 data	6
3.2.2.2 next	6
4 File Documentation	7
4.1 LinkedList.c File Reference	7
4.1.1 Function Documentation	7
4.1.1.1 createLList()	7
4.1.1.2 deleteLList()	7
4.1.1.3 deleteNodeFromLLinkedList()	8
4.1.1.4 getNodeDataInLList()	8
4.1.1.5 insertNodeInLList()	9
4.1.1.6 LListLength()	9
4.1.1.7 printLList()	9
4.2 LinkedList.h File Reference	10
4.2.1 Typedef Documentation	10
4.2.1.1 LinkedList_t	10
4.2.1.2 LinkedListPtr_t	10
4.2.1.3 ListNode_t	11
4.2.1.4 ListNodePtr_t	11
4.2.2 Function Documentation	11
4.2.2.1 createLList()	11
4.2.2.2 deleteLList()	11
4.2.2.3 deleteNodeFromLLinkedList()	12
4.2.2.4 getNodeDataInLList()	12
4.2.2.5 insertNodeInLList()	13
4.2.2.6 LListLength()	13
4.2.2.7 printLList()	13
4.3 LinkedList.h	14
4.4 test_LinkedList.c File Reference	14

4.4.1 Function Documentation	14
4.4.1.1 main()	15
Index	17

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

LinkedList	5
listNode	5

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

LinkedList.c	7
LinkedList.h	10
test_LinkedList.c	14

Chapter 3

Data Structure Documentation

3.1 LinkedList Struct Reference

```
#include <LinkedList.h>
```

Data Fields

- [ListNodePtr_t head](#)
- int [count](#)

3.1.1 Field Documentation

3.1.1.1 count

```
int count
```

3.1.1.2 head

```
ListNodePtr\_t head
```

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

- [LinkedList.h](#)

3.2 listNode Struct Reference

```
#include <LinkedList.h>
```

Data Fields

- long [data](#)
- struct [listNode](#) * [next](#)

3.2.1 Detailed Description

[LinkedList.h](#) - Linked List ADT header file

Author

Roy Kravitz (roy.kravitz@pdx.edu)

Date

07-Nov-2022

This is the header file for a Linked list ADT that implements a pointer-based singly linked list

Note

Code is based on SinglyLinkedList.c from Narasimha Karumanchi Data Structures and Algorithms Made Easy, Career Monk Publishers, 2016

The data in this implementation is a single long int. To change the data type change the struct [listNode](#).

3.2.2 Field Documentation

3.2.2.1 data

long data

3.2.2.2 next

struct [listNode](#)* next

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

- [LinkedList.h](#)

Chapter 4

File Documentation

4.1 LinkedList.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include "LinkedList.h"
```

Functions

- [LinkedListPtr_t createLList](#) (void)
- int [LListLength](#) (LinkedListPtr_t L)
- long [getNodeDataInLList](#) (LinkedListPtr_t L, int pos)
- void [insertNodeInLList](#) (LinkedListPtr_t L, long data, int pos)
- void [deleteNodeFromLLinkedList](#) (LinkedListPtr_t L, int pos)
- void [printLList](#) (LinkedListPtr_t L)
- void [deleteLList](#) (LinkedListPtr_t L)

4.1.1 Function Documentation

4.1.1.1 createLList()

```
LinkedListPtr_t createLList (
    void )
```

Creates a new instance of the Linked List

Returns

Pointer to the new Linked List instance if it succeeds. NULL if it fails

4.1.1.2 deleteLList()

```
void deleteLList (
    LinkedListPtr_t L )
```

Deletes all of the nodes in the linked list and then the [LinkedList](#) instance

Parameters

<i>L</i>	is a Pointer to a LinkedList instance
----------	---

Returns

void

4.1.1.3 deleteNodeFromLinkedList()

```
void deleteNodeFromLinkedList (
    LinkedListPtr\_t L,
    int pos )
```

Deletes a new node into the linked list

Parameters

<i>L</i>	is a Pointer to a LinkedList instance
<i>pos</i>	is the position in the list of the node to delete

Returns

void

4.1.1.4 getNodeDataInLList()

```
long getNodeDataInLList (
    LinkedListPtr\_t L,
    int pos )
```

Returns the data from a selected node

Parameters

<i>L</i>	is a Pointer to a LinkedList instance
<i>pos</i>	is the position in the list to insert the item

Returns

the data from the selected node as a long int

4.1.1.5 insertNodeInLList()

```
void insertNodeInLList (
    LinkedListPtr_t L,
    long data,
    int pos )
```

Inserts a new node into the linked list

Parameters

<i>L</i>	is a Pointer to a LinkedList instance
<i>data</i>	is the data item to put into the ndw node
<i>pos</i>	is the position in the list to insert the item

Returns

void

4.1.1.6 LListLength()

```
int LListLength (
    LinkedListPtr_t L )
```

Returns the number of items in the list

Parameters

<i>L</i>	is a Pointer to a LinkedList instance
----------	---

Returns

Returns the number of nodes in the linked list

4.1.1.7 printLList()

```
void printLList (
    LinkedListPtr_t L )
```

Prints all of the data items in the Linked List

Parameters

<i>L</i>	is a Pointer to a LinkedList instance
----------	---

Returns

void

4.2 LinkedList.h File Reference

```
#include <stddef.h>
#include <limits.h>
#include <malloc.h>
```

Data Structures

- struct [listNode](#)
- struct [LinkedList](#)

Typedefs

- typedef struct [listNode](#) [ListNode_t](#)
- typedef struct [listNode](#) * [ListNodePtr_t](#)
- typedef struct [LinkedList](#) [LinkedList_t](#)
- typedef struct [LinkedList](#) * [LinkedListPtr_t](#)

Functions

- [LinkedListPtr_t](#) [createLLList](#) (void)
- int [LLListLength](#) ([LinkedListPtr_t](#) L)
- void [insertNodeInLLList](#) ([LinkedListPtr_t](#) L, long data, int pos)
- long [getNodeDataInLLList](#) ([LinkedListPtr_t](#) L, int pos)
- void [deleteNodeFromLLinkedList](#) ([LinkedListPtr_t](#) L, int pos)
- void [printLLList](#) ([LinkedListPtr_t](#) L)
- void [deleteLLList](#) ([LinkedListPtr_t](#) L)

4.2.1 Typedef Documentation

4.2.1.1 [LinkedList_t](#)

```
typedef struct LinkedList LinkedList\_t
```

4.2.1.2 [LinkedListPtr_t](#)

```
typedef struct LinkedList * LinkedListPtr\_t
```

4.2.1.3 ListNode_t

```
typedef struct listNode ListNode_t
```

[LinkedList.h](#) - Linked List ADT header file

Author

Roy Kravitz (roy.kravitz@pdx.edu)

Date

07-Nov-2022

This is the header file for a Linked list ADT that implements a pointer-based singly linked list

Note

Code is based on SinglyLinkedList.c from Narasimha Karumanchi Data Structures and Algorithms Made Easy, Career Monk Publishers, 2016

The data in this implementation is a single long int. To change the data type change the struct [listNode](#).

4.2.1.4 ListNodePtr_t

```
typedef struct listNode * ListNodePtr_t
```

4.2.2 Function Documentation

4.2.2.1 createLList()

```
LinkedListPtr_t createLList (  
    void )
```

Creates a new instance of the Linked List

Returns

Pointer to the new Linked List instance if it succeeds. NULL if it fails

4.2.2.2 deleteLList()

```
void deleteLList (  
    LinkedListPtr_t L )
```

Deletes all of the nodes in the linked list and then the [LinkedList](#) instance

Parameters

<i>L</i>	is a Pointer to a LinkedList instance
----------	---

Returns

void

4.2.2.3 deleteNodeFromLinkedList()

```
void deleteNodeFromLinkedList (
    LinkedListPtr\_t L,
    int pos )
```

Deletes a new node into the linked list

Parameters

<i>L</i>	is a Pointer to a LinkedList instance
<i>pos</i>	is the position in the list of the node to delete

Returns

void

4.2.2.4 getNodeDataInLList()

```
long getNodeDataInLList (
    LinkedListPtr\_t L,
    int pos )
```

Returns the data from a selected node

Parameters

<i>L</i>	is a Pointer to a LinkedList instance
<i>pos</i>	is the position in the list to insert the item

Returns

the data from the selected node as a long int

4.2.2.5 insertNodeInLList()

```
void insertNodeInLList (
    LinkedListPtr_t L,
    long data,
    int pos )
```

Inserts a new node into the linked list

Parameters

<i>L</i>	is a Pointer to a LinkedList instance
<i>data</i>	is the data item to put into the ndw node
<i>pos</i>	is the position in the list to insert the item

Returns

void

4.2.2.6 LListLength()

```
int LListLength (
    LinkedListPtr_t L )
```

Returns the number of items in the list

Parameters

<i>L</i>	is a Pointer to a LinkedList instance
----------	---

Returns

Returns the number of nodes in the linked list

4.2.2.7 printLList()

```
void printLList (
    LinkedListPtr_t L )
```

Prints all of the data items in the Linked List

Parameters

<i>L</i>	is a Pointer to a LinkedList instance
----------	---

Returns

void

4.3 LinkedList.h

[Go to the documentation of this file.](#)

```
1
17 #ifndef _LINKEDLIST_H_
18 #define _LINKEDLIST_H_
19
20 // include required header files
21 #include <stddef.h>
22 #include <limits.h>
23 #include <malloc.h>
24
25 // define the struct that contains a node in the linked list
26 typedef struct listNode {
27     long data; // data for the node
28     struct listNode *next; // next pointer for the node
29 } ListNode_t, *ListNodePtr_t;
30
31 // define the struct that contains an instance of the Linked List
32 typedef struct LinkedList {
33     ListNodePtr_t head; // pointer to the head node of the list
34     int count; // number of elements on the list
35 } LinkedList_t, *LinkedListPtr_t;
36
37
38 // function prototypes
39 LinkedListPtr_t createLList(void);
40 int LListLength(LinkedListPtr_t L);
41 void insertNodeInLList(LinkedListPtr_t L, long data, int pos);
42 long getNodeDataInLList(LinkedListPtr_t L, int pos);
43 void deleteNodeFromLList(LinkedListPtr_t L, int pos);
44 void printLList(LinkedListPtr_t L);
45 void deleteLList(LinkedListPtr_t L);
46
47 #endif
```

4.4 test_LinkedList.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include "LinkedList.h"
```

Functions

- int [main](#) ()

4.4.1 Function Documentation

4.4.1.1 main()

```
int main ( )
```

[test_LinkedList.c](#) - Test program for the Linked List ADT

Author

Roy Kravitz (roy.kravitz@pdx.edu)

Date

07-Nov-2022

This is the source code file for a test program for a Link List ADT. Although it shouldn't matter as long as the API doesn't change, this test is based on Karumanchi's SinglyLinkedList example.

Note

Code is based on DynamicStack.c from Narasimha Karumanchi Data Structures and Algorithms Made Easy, Career Monk Publishers, 2016

Index

- count
 - LinkedList, [5](#)
- createLList
 - LinkedList.c, [7](#)
 - LinkedList.h, [11](#)
- data
 - listNode, [6](#)
- deleteLList
 - LinkedList.c, [7](#)
 - LinkedList.h, [11](#)
- deleteNodeFromLLinkedList
 - LinkedList.c, [8](#)
 - LinkedList.h, [12](#)
- getNodeDataInLList
 - LinkedList.c, [8](#)
 - LinkedList.h, [12](#)
- head
 - LinkedList, [5](#)
- insertNodeInLList
 - LinkedList.c, [8](#)
 - LinkedList.h, [12](#)
- LinkedList, [5](#)
 - count, [5](#)
 - head, [5](#)
- LinkedList.c, [7](#)
 - createLList, [7](#)
 - deleteLList, [7](#)
 - deleteNodeFromLLinkedList, [8](#)
 - getNodeDataInLList, [8](#)
 - insertNodeInLList, [8](#)
 - LListLength, [9](#)
 - printLList, [9](#)
- LinkedList.h, [10](#)
 - createLList, [11](#)
 - deleteLList, [11](#)
 - deleteNodeFromLLinkedList, [12](#)
 - getNodeDataInLList, [12](#)
 - insertNodeInLList, [12](#)
 - LinkedList_t, [10](#)
 - LinkedListPtr_t, [10](#)
 - ListNode_t, [10](#)
 - ListNodePtr_t, [11](#)
 - LListLength, [13](#)
 - printLList, [13](#)
- LinkedList_t
 - LinkedList.h, [10](#)
- LinkedListPtr_t
 - LinkedList.h, [10](#)
- listNode, [5](#)
 - data, [6](#)
 - next, [6](#)
- ListNode_t
 - LinkedList.h, [10](#)
- ListNodePtr_t
 - LinkedList.h, [11](#)
- LListLength
 - LinkedList.c, [9](#)
 - LinkedList.h, [13](#)
- main
 - test_LinkedList.c, [14](#)
- next
 - listNode, [6](#)
- printLList
 - LinkedList.c, [9](#)
 - LinkedList.h, [13](#)
- test_LinkedList.c, [14](#)
 - main, [14](#)