Roy's Linked List documentation

Generated by Doxygen 1.9.2

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

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

L	is a Pointer to a LinkedList instance

Returns

void

4.1.1.3 deleteNodeFromLLinkedList()

```
void deleteNodeFromLLinkedList (
            LinkedListPtr_t L,
            int pos )
```

Deletes a new node into the linked list

Parameters

L	is a Pointer to a LinkedList instance
pos	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

L	is a Pointer to a LinkedList instance
pos	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

L	is a Pointer to a LinkedList instance
data	is the data item to put into the ndw node
pos	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

L	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

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

L	is a Pointer to a LinkedList instance

Returns

void

4.2.2.3 deleteNodeFromLLinkedList()

```
void deleteNodeFromLLinkedList (
            LinkedListPtr_t L,
            int pos )
```

Deletes a new node into the linked list

Parameters

L	is a Pointer to a LinkedList instance
pos	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

L	is a Pointer to a LinkedList instance
pos	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

L	is a Pointer to a LinkedList instance
data	is the data item to put into the ndw node
pos	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

L	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

L	is a Pointer to a LinkedList instance

Returns

void

4.3 LinkedList.h

Go to the documentation of this file.

```
#ifndef _LINKEDLIST_H_
#define _LINKEDLIST_H_

// include required header files
#include <stddef.h>
#include <limits.h>
#include <malloc.h>

// define the struct that contains a node in the linked list
typedef struct listNode {
    long data;  // data for the node
    struct listNode *next;  // next pointer for the node
} ListNode_t, *ListNodePtr_t;

// define the struct that contains an instance of the Linked List
typedef struct LinkedList {
    ListNodePtr_t head;        // pointer to the head node of the list
    int count;             // number of elements on the list
} LinkedList_t, *LinkedListPtr_t;


// function prototypes
LinkedListPtr_t createLList(void);
int LListLength(LinkedListPtr_t L);
void insertNodeInLList(LinkedListPtr_t L, long data, int pos);
long getNodeDataInLList(LinkedListPtr_t L, int pos);
void deleteNodeFromLLinkedList(LinkedListPtr_t L, int pos);
void printLList(LinkedListPtr_t L);
void deleteLList(LinkedListPtr_t L);

#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 if 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