LinkedLists

Generated by Doxygen 1.8.11

Contents

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 Doubly Struct Reference

Doubly type of linked list.

Data Fields

- int data

  An integer variable for storage the data in linked list.
- struct Doubly * next

  The link to next element.
- struct Doubly * prev

  The link to previous element.

3.1.1 Detailed Description

Doubly type of linked list.

Structure for doubly type of linked lists.

3.1.2 Field Documentation

3.1.2.1 int data

An integer variable for storage the data in linked list.

3.1.2.2 struct Doubly * next

The link to next element.

3.1.2.3 struct Doubly * prev

The link to previous element.

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

- C:/Users/Dori-PC/Desktop/LinkedLists/functions.c
- C:/Users/Dori-PC/Desktop/LinkedLists/main.c

3.2 Singly Struct Reference

Singly type of linked list.

Data Fields

- int data

  An integer variable for storage the data in linked list.
- struct Singly * next

  The link to next element.

3.2.1 Detailed Description

Singly type of linked list.

Structure for singly type of linked lists.

3.2.2 Field Documentation

3.2.2.1 int data

An integer variable for storage the data in linked list.

3.2.2.2 struct Singly * next

The link to next element.

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

- C:/Users/Dori-PC/Desktop/LinkedLists/functions.c
- C:/Users/Dori-PC/Desktop/LinkedLists/main.c

Chapter 4

File Documentation

4.1 C:/Users/Dori-PC/Desktop/LinkedLists/doubly_functions.h File Reference

C library for operations with doubly linked lists.

Typedefs

- typedef struct Doubly Doubly

Functions

- void doubly_print_list (Doubly *head, FILE *f)
  Print a doubly linked list.
- int doubly_pop_pos (Doubly *head, int pos)
  Return an element from a given position.
- Doubly * doubly_init_emptylist ()
  Return a pointer of type doubly linked list.
- void doubly_push_first (Doubly *head, int val)
  Add a new element on the first position of list.
- void doubly_push_last (Doubly *head, int val)
  Add a new element on the last position of list.
- void doubly_push_pos (Doubly *head, int pos, int val)
  Add a new element on the specified position of list.
- void doubly_delete_pos (Doubly *head, int pos)
  Delete an element on the specified position of list.
- int doubly_length_list (Doubly *head)
  Return the length of list.
- void doubly_append_lists (Doubly *head_list1, Doubly *head_list2)
  Append two doubly lists.

4.1.1 Detailed Description

C library for operations with doubly linked lists.

Implements operations as initialisation of an empty list, adding a value at the beginning and at the end, inserting an item at a specified position, removing an item at a specified position, computing the length of a list and appending two lists for doubly linked lists.

4.1.2 Typedef Documentation

4.1.2.1 typedef struct Doubly Doubly

4.1.3 Function Documentation

4.1.3.1 void doubly_append_lists (Doubly * *head_list1,* Doubly * *head_list2*)

Append two doubly lists.

Parameters

**head_list1*	pointer to the first element of the list 1.
**head_list2*	pointer to the first element of the list 2.

With a "current" node go the end of list.
The "current" node will point to the first element of second list.
The first element of second list will point to the last element of first list.

4.1.3.2 void doubly_delete_pos (Doubly * *head,* int *pos*)

Delete an element on the specified position of list.

Parameters

**head*	pointer to the first element of the list.
pos	represent the position where the value will be deleted.

Creates and allocates memory for a deleted-node.
With a "current" node go through the list to position "pos".
The deleted-node will be "current" next element.
"Current" node will point to deleted-node next element.
The element after deleted-node will point to current.
Free the deleted-node memory.

4.1.3.3 Doubly* doubly_init_emptylist ( )

Return a pointer of type doubly linked list.

Returns

Return a pointer of type doubly linked list.

Creates and allocates memory for a new node.
The list will be empty and the node will point to NULL.

4.1.3.4 int doubly_length_list (Doubly * *head*)

Return the length of list.

Parameters

∗*head*	pointer to the first element of the list.

Returns

The length of the linked list

With a "current" node go through the list until the end.
Count each element from list and return the number of elements.

4.1.3.5 int doubly_pop_pos (Doubly ∗ *head,* int *pos*)

Return an element from a given position.

Parameters

∗*head*	pointer to the first element of the list.
pos	represent the position where displays the value.

Returns

The value of element from position "pos".

With a "current" node go through the list to position "pos".
Return the value of "current" node.

4.1.3.6 void doubly_print_list (Doubly ∗ *head,* FILE ∗ *f*)

Print a doubly linked list.

Parameters

∗*head*	pointer to the first element of the list.
∗*f*	pointer to the file for output results.

If the list is not empty, create a new node which will go through the list from the beginning.
Print each element until the list ends.

4.1.3.7 void doubly_push_first (Doubly ∗ *head,* int *val*)

Add a new element on the first position of list.

Parameters

∗*head*	pointer to the first element of the list.
val	represent the value that will be added.

Creates and allocates memory for a new node.
Gives value to the new node.
The new node will point to second element and to the head of list.
The head of list will point to the new node.

4.1.3.8 void doubly_push_last (Doubly * *head*, int *val*)

Add a new element on the last position of list.

Parameters

**head*	pointer to the first element of the list.
val	represent the value that will be added.

Creates and allocates memory for a new node.
Gives value to the new node.
With a "current" node go through the list until the end.
The "current" node will point to the new node.
The new node will point to "current" node and to NULL.

4.1.3.9 void doubly_push_pos (Doubly * *head*, int *pos*, int *val*)

Add a new element on the specified position of list.

Parameters

**head*	pointer to the first element of the list.
val	represent the value that will be added.
pos	represent the position where the value will be added.

Creates and allocates memory for a new node.
With a "current" node go through the list to position "pos".
The new node will point to the "current" next element and to "current" node.
The "current" node will point to the new node. Gives value to the new node.

4.2 C:/Users/Dori-PC/Desktop/LinkedLists/functions.c File Reference

C library implementation for operations with singly and doubly linked lists.

```
#include <stdio.h>
#include <stdlib.h>
#include "singly_functions.h"
#include "doubly_functions.h"
```

Data Structures

- struct Singly

  Singly type of linked list.
- struct Doubly

  Doubly type of linked list.

Functions

- void singly_print_list (Singly *head, FILE *f)

  Print a singly linked list.
- int singly_pop_pos (Singly *head, int pos)

  Return an element from a given position.
- Singly * singly_init_emptylist ()

  Return a pointer of type singly linked list.
- void singly_push_first (Singly *head, int val)

  Add a new element on the first position of list.
- void singly_push_last (Singly *head, int val)

  Add a new element on the last position of list.
- void singly_push_pos (Singly *head, int pos, int val)

  Add a new element on the specified position of list.
- void singly_delete_pos (Singly *head, int pos)

  Delete an element on the specified position of list.
- int singly_length_list (Singly *head)

  Return the length of list.
- void singly_append_lists (Singly *head_list1, Singly *head_list2)

  Append two singly lists.
- void doubly_print_list (Doubly *head, FILE *f)

  Print a doubly linked list.
- int doubly_pop_pos (Doubly *head, int pos)

  Return an element from a given position.
- Doubly * doubly_init_emptylist ()

  Return a pointer of type doubly linked list.
- void doubly_push_first (Doubly *head, int val)

  Add a new element on the first position of list.
- void doubly_push_last (Doubly *head, int val)

  Add a new element on the last position of list.
- void doubly_push_pos (Doubly *head, int pos, int val)

  Add a new element on the specified position of list.
- void doubly_delete_pos (Doubly *head, int pos)

  Delete an element on the specified position of list.
- int doubly_length_list (Doubly *head)

  Return the length of list.
- void doubly_append_lists (Doubly *head_list1, Doubly *head_list2)

  Append two doubly lists.

4.2.1 Detailed Description

C library implementation for operations with singly and doubly linked lists.

Implements operations as initialisation of an empty list, adding a value at the beginning and at the end, inserting an item at a specified position, removing an item at a specified position, computing the length of a list and appending two lists for singly and doubly linked lists.

4.2.2 Function Documentation

4.2.2.1 void doubly_append_lists (Doubly * *head_list1,* Doubly * *head_list2*)

Append two doubly lists.

Parameters

**head_list1*	pointer to the first element of the list 1.
**head_list2*	pointer to the first element of the list 2.

With a "current" node go the end of list.
The "current" node will point to the first element of second list.
The first element of second list will point to the last element of first list.

4.2.2.2 void doubly_delete_pos (Doubly * *head,* int *pos*)

Delete an element on the specified position of list.

Parameters

**head*	pointer to the first element of the list.
pos	represent the position where the value will be deleted.

Creates and allocates memory for a deleted-node.
With a "current" node go through the list to position "pos".
The deleted-node will be "current" next element.
"Current" node will point to deleted-node next element.
The element after deleted-node will point to current.
Free the deleted-node memory.

4.2.2.3 Doubly* doubly_init_emptylist ( )

Return a pointer of type doubly linked list.

Returns

Return a pointer of type doubly linked list.

Creates and allocates memory for a new node.
The list will be empty and the node will point to NULL.

4.2.2.4 int doubly_length_list (Doubly * *head*)

Return the length of list.

Parameters

∗*head*	pointer to the first element of the list.

Returns

The length of the linked list

With a "current" node go through the list until the end.
Count each element from list and return the number of elements.

4.2.2.5 int doubly_pop_pos (Doubly ∗ *head,* int *pos*)

Return an element from a given position.

Parameters

∗*head*	pointer to the first element of the list.
pos	represent the position where displays the value.

Returns

The value of element from position "pos".

With a "current" node go through the list to position "pos".
Return the value of "current" node.

4.2.2.6 void doubly_print_list (Doubly ∗ *head,* FILE ∗ *f*)

Print a doubly linked list.

Parameters

∗*head*	pointer to the first element of the list.
∗*f*	pointer to the file for output results.

If the list is not empty, create a new node which will go through the list from the beginning.
Print each element until the list ends.

4.2.2.7 void doubly_push_first (Doubly ∗ *head,* int *val*)

Add a new element on the first position of list.

Parameters

∗*head*	pointer to the first element of the list.
val	represent the value that will be added.

Creates and allocates memory for a new node.
Gives value to the new node.
The new node will point to second element and to the head of list.
The head of list will point to the new node.

4.2.2.8 void doubly_push_last (Doubly * *head,* int *val*)

Add a new element on the last position of list.

Parameters

**head*	pointer to the first element of the list.
val	represent the value that will be added.

Creates and allocates memory for a new node.
Gives value to the new node.
With a "current" node go through the list until the end.
The "current" node will point to the new node.
The new node will point to "current" node and to NULL.

4.2.2.9 void doubly_push_pos (Doubly * *head,* int *pos,* int *val*)

Add a new element on the specified position of list.

Parameters

**head*	pointer to the first element of the list.
val	represent the value that will be added.
pos	represent the position where the value will be added.

Creates and allocates memory for a new node.
With a "current" node go through the list to position "pos".
The new node will point to the "current" next element and to "current" node.
The "current" node will point to the new node. Gives value to the new node.

4.2.2.10 void singly_append_lists (Singly * *head_list1,* Singly * *head_list2*)

Append two singly lists.

Parameters

**head_list1*	pointer to the first element of the list 1.
**head_list2*	pointer to the first element of the list 2.

With a "current" node go to the end of list.
The "current" node will point to the first element of second list.

4.2.2.11 void singly_delete_pos (Singly * head, int pos)

Delete an element on the specified position of list.

Parameters

**head*	pointer to the first element of the list.
pos	represent the position where the value will be deleted.

Creates and allocates memory for a deleted-node.
With a "current" node go through the list to position "pos".
The deleted-node will be "current" next element.
"Current" node will point to deleted-node next element.
Free the deleted-node memory.

4.2.2.12 Singly* singly_init_emptylist ( )

Return a pointer of type singly linked list.

Returns

Return a pointer of type singly linked list.

Creates and allocates memory for a new node.
The list will be empty and the node will point to NULL.

4.2.2.13 int singly_length_list (Singly * head)

Return the length of list.

Parameters

**head*	pointer to the first element of the list.

Returns

The length of the linked list.

With a "current" node go through the list until the end.
Count each element from list and return the number of elements.

4.2.2.14 int singly_pop_pos (Singly * head, int pos)

Return an element from a given position.

Parameters

∗*head*	pointer to the first element of the list.
pos	represent the position from where displays the value.

Returns

The value of element from position "pos".

With a "current" node go through the list to position "pos".
Return the value of "current" node.

4.2.2.15 void singly_print_list (Singly ∗ *head,* FILE ∗ *f*)

Print a singly linked list.

Parameters

∗*head*	pointer to the first element of the list.
∗*f*	pointer to the file for output results.

If the list is not empty, create a new node which will go through the list from the beginning.
Print each element until the list ends.

4.2.2.16 void singly_push_first (Singly ∗ *head,* int *val*)

Add a new element on the first position of list.

Parameters

∗*head*	pointer to the first element of the list.
val	represent the value that will be added.

Creates and allocates memory for a new node.
Gives value to the new node.
The new node will point to second element and the head of list will point to the new node.

4.2.2.17 void singly_push_last (Singly ∗ *head,* int *val*)

Add a new element on the last position of list.

Parameters

∗*head*	pointer to the first element of the list.
val	represent the value that will be added.

Creates and allocates memory for a new node.
Gives value to the new node.
With a "current" node go through the list until the end.
The "current" node will point to the new node.
The new node will point to NULL.

4.2.2.18 void singly_push_pos (Singly * *head*, int *pos*, int *val*)

Add a new element on the specified position of list.

Parameters

**head*	pointer to the first element of the list.
val	represent the value that will be added.
pos	represent the position where the value will be added.

Creates and allocates memory for a new node.
With a "current" node go through the list to position "pos".
The new node will point to the "current" next element.
The "current" node will point to the new node. Gives value to the new node.

4.3 C:/Users/Dori-PC/Desktop/LinkedLists/main.c File Reference

Libraries 2. : A library for linked lists.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "singly_functions.h"
#include "doubly_functions.h"
```

Data Structures

- struct Singly
  Singly type of linked list.
- struct Doubly
  Doubly type of linked list.

Functions

- void singly_test (FILE *f)
  Test generator for singly type structure.
- void doubly_test (FILE *f)
  Test generator for doubly type structure.
- int main ()
  Main function.

4.3.1 Detailed Description

Libraries 2. : A library for linked lists.

4.3.2 Function Documentation

4.3.2.1 void doubly_test (FILE * f)

Test generator for doubly type structure.

Parameters

*f	pointer to the file for output results.

Returns

The test results.

length : a variable for length of list, default at 2000 elements.

val : a variable to enter values in list.

pos : a variable for positions in lists.

Create two lists using doubly_push_last and doubly_push_first functions with length value of "length".

Print the initial list and the list after we delete the element.
Print the element from position "pos" (generated with C random function), before and after we delete the element.

Add the a new element "val" on position "pos", both generated with C random function.
Print the element from position "pos".

Print the length of the list and the list.

Append the two list, print the result and lengh of appended lists.

4.3.2.2 int main ( )

Main function.

Function call tests generator for singly and doubly linked lists giving tests with operation imported from "singly_↩ functions.h" and "doubly_functions.h".
Opens a text file for writing in appending mode. If it does not exist, then a new file is created. The program will start appending content in the existing file content.

Intializes random number generator.

Uses the singly_test and doubly_test, tests generator function.

4.3.2.3 void singly_test (FILE * f)

Test generator for singly type structure.

Parameters

*f	pointer to the file for output results.

Returns

The test results.

length : a variable for length of list, default at 2000 elements.

val : a variable to enter values in list.

pos : a variable for positions in lists.

Create two lists using singly_push_last and singly_push_first functions with length value of "length".

Print the initial list and the list after we delete the element.
Print the element from position "pos" (generated with C random function), before and after we delete the element.

Add the a new element "val" on position "pos", both generated with C random function.
Print the element from position "pos".

Print the length of the list and the list.

Append the two list, print the result and lengh of appended lists.

4.4 C:/Users/Dori-PC/Desktop/LinkedLists/singly_functions.h File Reference

C library for operations with singly linked lists.

Typedefs

- typedef struct Singly Singly

Functions

- void singly_print_list (Singly ∗head, FILE ∗f)

  Print a singly linked list.
- int singly_pop_pos (Singly ∗head, int pos)

  Return an element from a given position.
- Singly ∗ singly_init_emptylist ()

  Return a pointer of type singly linked list.
- void singly_push_first (Singly ∗head, int val)

  Add a new element on the first position of list.
- void singly_push_last (Singly ∗head, int val)

  Add a new element on the last position of list.
- void singly_push_pos (Singly ∗head, int pos, int val)

  Add a new element on the specified position of list.
- void singly_delete_pos (Singly ∗head, int pos)

  Delete an element on the specified position of list.
- int singly_length_list (Singly ∗head)

  Return the length of list.
- void singly_append_lists (Singly ∗head_list1, Singly ∗head_list2)

  Append two singly lists.

4.4.1 Detailed Description

C library for operations with singly linked lists.

Implements operations as initialisation of an empty list, adding a value at the beginning and at the end, inserting an item at a specified position, removing an item at a specified position, computing the length of a list and appending two lists for singly linked lists.

4.4.2 Typedef Documentation

4.4.2.1 typedef struct Singly Singly

4.4.3 Function Documentation

4.4.3.1 void singly_append_lists (Singly ∗ *head_list1,* Singly ∗ *head_list2*)

Append two singly lists.

Parameters

∗*head_list1*	pointer to the first element of the list 1.
∗*head_list2*	pointer to the first element of the list 2.

With a "current" node go to the end of list.
The "current" node will point to the first element of second list.

4.4.3.2 void singly_delete_pos (Singly * *head,* int *pos*)

Delete an element on the specified position of list.

Parameters

**head*	pointer to the first element of the list.
pos	represent the position where the value will be deleted.

Creates and allocates memory for a deleted-node.
With a "current" node go through the list to position "pos".
The deleted-node will be "current" next element.
"Current" node will point to deleted-node next element.
Free the deleted-node memory.

4.4.3.3 Singly* singly_init_emptylist ( )

Return a pointer of type singly linked list.

Returns

Return a pointer of type singly linked list.

Creates and allocates memory for a new node.
The list will be empty and the node will point to NULL.

4.4.3.4 int singly_length_list (Singly * *head*)

Return the length of list.

Parameters

**head*	pointer to the first element of the list.

Returns

The length of the linked list.

With a "current" node go through the list until the end.
Count each element from list and return the number of elements.

4.4.3.5 int singly_pop_pos (Singly * *head,* int *pos*)

Return an element from a given position.

Parameters

∗*head*	pointer to the first element of the list.
pos	represent the position from where displays the value.

Returns

The value of element from position "pos".

With a "current" node go through the list to position "pos".
Return the value of "current" node.

4.4.3.6 void singly_print_list (Singly ∗ *head,* FILE ∗ *f*)

Print a singly linked list.

Parameters

∗*head*	pointer to the first element of the list.
∗*f*	pointer to the file for output results.

If the list is not empty, create a new node which will go through the list from the beginning.
Print each element until the list ends.

4.4.3.7 void singly_push_first (Singly ∗ *head,* int *val*)

Add a new element on the first position of list.

Parameters

∗*head*	pointer to the first element of the list.
val	represent the value that will be added.

Creates and allocates memory for a new node.
Gives value to the new node.
The new node will point to second element and the head of list will point to the new node.

4.4.3.8 void singly_push_last (Singly ∗ *head,* int *val*)

Add a new element on the last position of list.

Parameters

∗*head*	pointer to the first element of the list.
val	represent the value that will be added.

Creates and allocates memory for a new node.
Gives value to the new node.
With a "current" node go through the list until the end.
The "current" node will point to the new node.
The new node will point to NULL.

4.4.3.9 void singly_push_pos (Singly * *head*, int *pos*, int *val*)

Add a new element on the specified position of list.

Parameters

**head*	pointer to the first element of the list.
val	represent the value that will be added.
pos	represent the position where the value will be added.

Creates and allocates memory for a new node.
With a "current" node go through the list to position "pos".
The new node will point to the "current" next element.
The "current" node will point to the new node. Gives value to the new node.

Index