Assignment #1: Adrian Kacmarcik

Generated by Doxygen 1.8.13

Contents

1 File Index

1.1 File List

Here is a list of all documented files with brief descriptions:

cachelist.c **Description: Linked lists operstions with cacheing**	**1**

2 File Documentation

2.1 cachelist.c File Reference

Description: Linked lists operstions with cacheing.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "cachelist.h"
```

Functions

- static cl_node ∗ make_node (int value, const char ∗label)

  Make a new node based on inputs.
- cl_node ∗ cl_add_end (cl_node ∗list, int value, const char ∗label)

  Add a node to the end of |list|.
- cl_node ∗ cl_add_front (cl_node ∗list, int value, const char ∗label)

  Add a new node to the front of |list|.
- cl_node ∗ cl_remove (cl_node ∗list, int search_value)

  Search |list| for a match and delete it.
- cl_node ∗ cl_insert_before (cl_node ∗list, int search_value, int value, const char ∗label)

  Add a new node before the matching value.
- void cl_insert_after (cl_node ∗list, int search_value, int value, const char ∗label)

  Add new node after matching value.
- cl_node ∗ cl_find (cl_node ∗list, int search_value, bool cache, int ∗compares)

  Find a node in |list|, and if |cache| then move to the beginning if found.
- void cl_destroy (cl_node ∗list)

  Deletes and frees all of |list|.
- void cl_dump (const cl_node ∗list)

  print out the list

2.1.1 Detailed Description

Description: Linked lists operstions with cacheing.

Author

Adrian T.P. Kacmarcik

DP email a.kacmarcik@digipen.edu

course CS180

section A

assignment 1

Date

2020/01/23

cl_add_end: Add a node to the end of the list contained with the inputted list pointer

cl_add_front: Add a node to the front of the list that is contained in the pointer passed in (this will change the head pointer to point to the new beginning of the list)

cl_remove: Search the list contained with the head pointer for the inputted search value and removes the forst one that it finds from the list

cl_insert_before: Inserts a new node with the inputted value and label before the matching value. If no matching value is found, then nothing is done.

cl_insert_after: Inserts a new node with the inputted value and label after the matching value. If no matching value is found, the nothings is done.

cl_find: Searches through the list looking for the first matching value, counting the number of comparisons it took to find the matching node. If cacheing is true then the found node will be moved to the beginning of the list to simmulate cacheing.

cl_destroy: Frees the memory in the linked list that the head pointer is pointing to.

cl_dump: Prints out the list to the console.

2.1.2 Function Documentation

2.1.2.1 cl_add_end()

```
cl_node* cl_add_end (
             cl_node * list,
             int value,
             const char * label )
```

Add a node to the end of |list|.

Parameters

list	Pointer to the start of the linked list.
value	Value to store in the new node.
label	Label for the new node.

Returns

The pointer to the head of the list.

Definition at line 87 of file cachelist.c.

References make_node().

```
{
  // copy of the start of the list
  cl_node *listLocation = list;
  cl_node *newNode = make_node(value, label); // node to add to |list|

  // check if the list exists
  if (list)
  {
    // find the end of the list
    while (listLocation->next) {
      listLocation = listLocation->next;
    }
     listLocation->next = newNode; // add |newNode| to end of |list|

     return list; // return pointer to start of list
   }
   else
   {
     list = newNode; // set the head of the list to be the new node
     return newNode; // return the head of the new list
   }
 }
```

2.1.2.2 cl_add_front()

```
cl_node* cl_add_front (
              cl_node * list,
              int value,
              const char * label )
```

Add a new node to the front of |list|.

Parameters

list	Head pointer.
value	Value for new node.
label	String for the new node.

Returns

The pointer to the head of |list|.

Definition at line 118 of file cachelist.c.

References make_node().

```
 {
   cl_node *newNode = make_node(value, label); // node to add to |list|

   newNode->next = list; // point next to be the old start
   list = newNode;       // update the head pointer

   return newNode; // return new head pointer
 }
```

2.1.2.3 cl_destroy()

```
void cl_destroy (
            cl_node * list )
```

Deletes and frees all of |list|.

Parameters

list	Pointer to the head of the list to free.

Definition at line 294 of file cachelist.c.

```
{
  cl_node *listLocation = list; // temp storeage of location in |list|

  // loop through |list| freeing all of the nodes
  while (list)
  {
    listLocation = list; // store the current location in |listLocation|
    list = list->next;   // move |list| to be the next location
    free(listLocation);  // free |listLocation|
  }
}
```

2.1.2.4 cl_dump()

```
void cl_dump (
            const cl_node * list )
```

print out the list

Parameters

list	Pointer to the head of the list to walkk through.

Definition at line 311 of file cachelist.c.

```
{
  printf("==================\n"); // print seperator

  // loop through |list|
  while (list)
  {
    printf("%4i: %s\n", list->value, list->label); // print formatted
    list = list->next;                             // next node in |list|
  }
}
```

2.1.2.5 cl_find()

```
cl_node* cl_find (
             cl_node * list,
             int search_value,
             bool cache,
             int * compares )
```

Find a node in |list|, and if |cache| then move to the beginning if found.

Parameters

list	Pointer to the head of the list
search_value	Value to search for in |list|
cache	Weather we should be cacheing
compares	Pointer to store the number of comparasons

Returns

The pointer to the head of the list.

Definition at line 254 of file cachelist.c.

```
{
  cl_node *listLocation = list;     // current location in |list|
  cl_node *prevListLocation = list; // previous location in |lsit|
  *compares = 0;                    // set the number of coimpares to |0|

  // loop through |list|
  while (listLocation)
  {
    (*compares)++; // incrament the number of compares

    // check if we have found a matching value
    if (listLocation->value == search_value)
    {
      // check if we are cacheing
      if (cache)
      {
        // check if we have moved form the start
        if (prevListLocation != listLocation)
        {
          prevListLocation->next = listLocation->next; // skip over curr loc
          listLocation->next = list;                   // add beggining
          list = listLocation;                         // point head to curr
        }
      }
      // return pointer to the start of the list
      return list;
    }
    // if we havent found anything then go to next node
    prevListLocation = listLocation;
    listLocation = listLocation->next;
  }
  // if there was nothing to match in the list then return the head
  return list;
}
```

2.1.2.6 cl_insert_after()

```
void cl_insert_after (
              cl_node * list,
              int search_value,
              int value,
              const char * label )
```

Add new node after matching value.

Parameters

list	Pointer to the start of the list.
search_value	Value to match
value	Value of the new node
label	String for the new node

Definition at line 223 of file cachelist.c.

References make_node().

```
{
  cl_node *listLocation = list; // current location in |list|

  // loop through |list|
  while (listLocation)
  {
    // check if we found a match
    if (listLocation->value == search_value)
    {
      cl_node *newNode = make_node(value, label); // node to add to |list|

      // insert node in |list|
      newNode->next = listLocation->next;
      listLocation->next = newNode;

      return; // escape with first match found
    }
    listLocation = listLocation->next; // next node
  }
}
```

2.1.2.7 cl_insert_before()

```
cl_node* cl_insert_before (
              cl_node * list,
              int search_value,
              int value,
              const char * label )
```

Add a new node before the matching value.

Parameters

list	Pointer to the start of the list.
search_value	Value to search |list| for.
value	Value for new node.
label	String for the new node.

Returns

The pointer to the head of the list.

Definition at line 178 of file cachelist.c.

References make_node().

```
{
  cl_node *listLocation = list;     // current location in |list|
  cl_node *prevListLocation = list; // previous location in |list|

  // loop through |list|
  while (listLocation)
  {
    // check if ew have found a match
    if (listLocation->value == search_value)
    {
      cl_node *newNode = make_node(value, label); // create new node
      newNode->next = listLocation;               // point next to curr

      //check if we are at the start of the list
      if (prevListLocation == listLocation)
      {
        // new node is at the beggining change the head pointer
        list = newNode;
        return newNode; // return new head pointer
      }
      else
      {
        // set the previous node to be pointing at the new one
        prevListLocation->next = newNode;

        return list; // return the head pointer
      }
    }
    // move to the next node
    prevListLocation = listLocation;
    listLocation = listLocation->next;
  }
  // if none are found return the head pointer
  return list;
}
```

2.1.2.8 cl_remove()

```
cl_node* cl_remove (
              cl_node * list,
              int search_value )
```

Search |list| for a match and delete it.

Parameters

list	Head pointer of the list.
search_value	Value to search |list| for.

Returns

The head pointer.

Definition at line 134 of file cachelist.c.

```
{
  cl_node *listLocation = list;     // current location in |list|
  cl_node *prevListLocation = list; // previous location in |list|

  // loop through |list|
  while (listLocation)
  {
    // check for match
    if (listLocation->value == search_value)
    {
      // check for beggingingof |list|
      if (prevListLocation == listLocation)
      {
        // move head pointer to next node
        cl_node *temp = listLocation->next;
        free(listLocation); // free the head pointer
        list = temp;        // change the head pointer to new one
        return temp;        // return new head pointer
      }
      else
      {
        // move the pointers to skip over |listLocation|
        prevListLocation->next = listLocation->next;
        free(listLocation); // free |listLocaiton|
        return list;        // return head pointer
      }
    }
    // move to next node
    prevListLocation = listLocation;
    listLocation = listLocation->next;
  }
  // if nothing found return head pointer
  return list;
}
```

2.1.2.9 make_node()

```
static cl_node* make_node (
                int value,
                const char * label )   [static]
```

Make a new node based on inputs.

Parameters

value	Number to be stored in the newNodes's |value|.
label	String to be stored in |label|.

Returns

The pointer to the new node.

Definition at line 57 of file cachelist.c.

Referenced by cl_add_end(), cl_add_front(), cl_insert_after(), and cl_insert_before().

```
{
  // malloc the space for a new node
  cl_node *node = (cl_node *)malloc(sizeof(cl_node));

  // check if malloc succeeded
  if (!node)
  {
    printf("Can't allocate new node.\n"); // print that it failed
    exit(1);                              // exit the code early
  }

  // set the values of |node|
  node->value = value;
  node->next = NULL;

  /* Be sure not to overwrite memory */
  strncpy(node->label, label, LABEL_SIZE - 1); // copy the label in |node|
  node->label[LABEL_SIZE - 1] = 0;             // set end |NULL|

  return node; // return the pointer to the new node
}
```

Index