

#include <cstddef>

#include "linkedlist.h"

#include <iostream>

#include <stdio.h>

#include <stdlib.h>

/////////// function prototype /////////////

void Node::operation\_select(){

printf("\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("select your operation (e.g., 2)\n");

printf("1- Display link list\n");

printf("2- Insert\_at\_head\n");

printf("3- Insert\_at\_tail\n");

printf("4- Delete\_at\_head\n");

printf("5- Delete\_at\_tail\n");

printf("6- Delete\_with\_val\n");

printf("7- Find\_element\n");

printf("8- Count\_element\n");

printf("9- Destroy\_list\n");

printf("10- Exit\n");

printf("Which operation? ");

}

int Node::display(){

Node\* curr\_node = this;

printf("\n\n");

while (curr\_node != NULL) {

printf(" %d --->", curr\_node->data);

curr\_node = curr\_node->next;

}

printf("\*NULL\*\n\n");

return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

insert\_at\_head (int val)

this function adds the val into the head of the

linked list and returns head pointer

Example:

linked list: 10 --->20 --> 30 --> NULL

insert\_at\_head (42)

linked list: 42 ---> 10 --->20 ---> 30 ---> NULL

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Node\* Node::insert\_at\_head(int val){

//create a link

Node \*link = (Node \*) malloc(sizeof(Node));

link->data = val;

//point it to old first node

link->next = this;

return link;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

insert\_at\_tail (int val)

this function adds the val into the tail of the

linked list and returns head pointer

Example:

linked list: 10 --->20 --> 30 --> NULL

insert\_at\_tail (42)

linked list: 10 --->20 ---> 30 ---> 42 ---> NULL

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Node\* Node::insert\_at\_tail(int val){

/\*if the list was empty\*/

if(this == NULL) {

//create a link

Node \*link = (Node \*) malloc(sizeof(Node));

link->data = val;

link->next = NULL;

return link;

}

Node \*current = this;

while (current->next != NULL) {

current = current->next;

}

/\* now we can add a new variable \*/

current->next = (Node \*) malloc(sizeof(Node));

current->next->data = val;

current->next->next = NULL;

return this;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

delete\_at\_head()

this function deletes the value in the head of the

linked list and returns head pointer

Example:

linked list: 10 --->20 --> 30 --> NULL

delete\_at\_head()

linked list: 20 ---> 30 ---> NULL

Note: if the list is empty print out an appropriate message

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Node\* Node::delete\_at\_head(){

if (this == NULL) {

printf("the list is empty");

return this;

}

//return next to first link as first

return this->next;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

delete\_at\_tail()

this function deletes the value from the tail of the

linked list and returns head pointer

Example:

linked list: 10 --->20 --> 30 --> NULL

delete\_at\_tail()

linked list: 10 ---> 20 ---> NULL

Note: if the list is empty print out an appropriate message

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Node\* Node::delete\_at\_tail(){

if (this == NULL) {

printf("the list is empty");

return this;

}

/\* if there is only one item in the list, remove it \*/

if (this->next == NULL) {

free(this);

return NULL;

}

/\* get to the last node in the list \*/

Node \*current = this;

while (current->next->next != NULL) {

current = current->next;

}

/\* now current points to the last item of the list, so let's remove current->next \*/

free(current->next);

current->next = NULL;

return this;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

delete\_with\_val(int val)

this function deletes the node with the selected value

and returns head pointer

Example:

linked list: 10 --->20 --> 30 --> NULL

delete\_with\_val(20)

linked list: 10 ---> 30 ---> NULL

Note: if the list is empty or the value is not in the

linked list print out an appropriate message

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Node\* Node::delete\_with\_val(int val){

//start from the first link

Node \*head = this;

Node \*current = head;

Node \*previous = NULL;

//if list is empty

if(head == NULL) {

printf("Not Found! the value is not in the linked list!");

return NULL;

}

//navigate through list

while(current->data != val) {

//if it is last node

if(current->next == NULL) {

printf("Not Found! the value is not in the linked list!");

return head;

} else {

//store reference to current link

previous = current;

//move to next link

current = current->next;

}

}

//found a match, update the link

if(current == head) {

//change first to point to next link

head = head->next;

} else {

//bypass the current link

previous->next = current->next;

}

return head;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

find\_element(int val)

this function finds the node with the selected value

Example:

linked list: 10 --->20 --> 30 --> NULL

find\_element(20)

output is : Found! the value is node number 2

find\_element(42)

output is: Not Found! the value is not in the linked list!

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Node::find\_element(int val){

//start from the first link

Node \*current = this;

int counter = 1;

//if list is empty

if(this == NULL) {

printf("Not Found! the value is not in the linked list!");

return;

}

//navigate through list

while(current->data != val) {

//if it is last node

if(current->next == NULL) {

printf("Not Found! the value is not in the linked list!");

return;

} else {

//go to next link

current = current->next;

counter++;

}

}

//if data found, return the current Link

printf("Found! the value is node number %i", counter);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

count\_element()

this function counts the number of node in the linked list

Example:

linked list: 10 --->20 --> 30 --> NULL

count\_element()

output is : 3 elements

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Node::count\_element(){

//start from the first link

Node \*current = this;

int counter = 1;

//if list is empty

if(this == NULL) {

printf("0 element");

return;

}

//navigate through list

while(current->next != NULL) {

//if it is last node

current = current->next;

counter++;

}

//if data found, return the current Link

printf("%i elements", counter);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

destroy\_list()

this function removes all the nodes from the linked list

Example:

linked list: 10 --->20 --> 30 --> NULL

destroy\_list()

linked list: NULL

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Node\* Node::destroy\_list(){

if(this == NULL) {

return this;

}

if(this->next != NULL){

this->next->destroy\_list();

this->next = NULL;

free(this);

}

return NULL;

}