

DOUBLY LINKED LIST LAB 7 PGM

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

typedef struct Node{

int data;

struct Node \*next;

struct Node \*prev;

} node;

node\* head = NULL;

int count = 0;

void insert(int data, int position);

void delete(int element);

void display();

int main(){

int data, choice, pos;

printf("1. Insert\n2. Delete\n3. Exit\nChoice: ");

scanf("%d", &choice);

while(choice != 3){

if (choice == 1){

printf("Enter data and position: ");

scanf("%d%d", &data, &pos);

insert(data, pos);

printf("Count: %d\n", count);

} else if (choice == 2){

printf("Enter element: ");

scanf("%d", &pos);

delete(pos);

printf("Count: %d\n", count);

}

display();

printf("Enter choice: ");

scanf("%d", &choice);

}

return 0;

}

void insert(int data, int position){

if (position == 0){

node\* new\_node = malloc(sizeof(node));

new\_node->data = data;

new\_node->next = head;

new\_node->prev = NULL;

if (head != NULL) head->prev = new\_node;

head = new\_node;

count++;

return;

} else if (position == count){

node\* new\_node = malloc(sizeof(node));

new\_node->data = data;

new\_node->next = NULL;

node\* temp = head;

while(temp->next != NULL)

temp = temp->next;

temp->next = new\_node;

new\_node->prev = temp;

count++;

return;

} else if (position > count || position < 0){

printf("Unable to insert at given position\n");

return;

} else {

node\* temp = head;

for(int i = 0; i < position-1; i++)

temp = temp->next;

node\* new\_node = malloc(sizeof(node));

new\_node->data = data;

new\_node->next = temp->next;

new\_node->prev = temp;

temp->next->prev = new\_node;

temp->next = new\_node;

count++;

return;

}

}

void delete(int element){

int position = 0; node \*temp = head;

if (head == NULL){

printf("List is empty, cannot delete"); return;

}

for(;position < count; temp=temp->next, position++)

if (temp->data == element) break;

if (temp == NULL){

printf("Element does not exist in list"); return;

}

if (position == 0){

node\* temp = head;

temp = temp->next;

temp->prev = NULL;

free(head);

head = temp;

count--;

return;

} else if (position == count-1){

node\* temp = head;

for(int i = 1; i < count-1; i++)

temp = temp->next;

node\* temp1 = temp->next;

temp->next = NULL;

free(temp1);

count--;

return;

} else if (position > count || position < 0){

printf("Unable to delete at position\n");

return;

} else {

node\* temp = head;

for(int i = 0; i < position; i++)

temp = temp->next;

temp->next->prev = temp->prev;

temp->prev->next = temp->next;

free(temp);

count--;

return;

}

}

void display(){

node\* temp = head;

printf("Linked List: ");

while (temp->next != NULL){

printf("%d ", temp->data);

temp = temp->next;

}

printf("%d ", temp->data);

printf("\n");

}