Name :- Om Thanage

Roll no:- 16010123217 Batch :- C3

Code:

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

struct node {

int data;

struct node\* next;

};

struct node\* head = NULL;

//Function to insert before node

void insertBefore(int n, int value) {

struct node\* newnode = (struct node\*)malloc(sizeof(struct node));

newnode->data = n;

newnode->next = NULL;

struct node\* curr = head;

struct node\* prev = NULL;

//if there is no element in linked list

if (head == NULL || head->data == value) {

newnode->next = head;

head = newnode;

return;

}

while (curr != NULL && curr->data != value) {

prev = curr;

curr = curr->next;

}

if (curr != NULL) {

prev->next = newnode;

newnode->next = curr;

} else {

printf("Value %d not found in the list.\n", value); //Value not found condition

free(newnode); //free the allocated memory

}

}

//function for inserting at end of linked list

void insertEnd(int n) {

struct node\* newnode = (struct node\*)malloc(sizeof(struct node));

newnode->data = n;

newnode->next = NULL;

if (head == NULL) {

head = newnode;

} else {

struct node\* curr = head;

while (curr->next != NULL) {

curr = curr->next;

}

curr->next = newnode;

}

}

//Function for deleting after node

void deleteAfter(int value) {

//Checking if list is empty

if (head == NULL) {

printf("List is empty\n");

return;

}

struct node\* curr = head;

while (curr != NULL && curr->data != value) {

curr = curr->next;

}

if (curr != NULL && curr->next != NULL) {

struct node\* temp = curr->next;

curr->next = curr->next->next;

printf("%d deleted from the list.\n", temp->data);

free(temp);

} else {

printf("Value not found or no node to delete after.\n");

}

}

void deleteFirst() {

//Checking if list is empty

if (head == NULL) {

printf("List is empty\n");

return;

}

struct node\* temp = head;

head = head->next;

printf("%d is the first node deleted\n", temp->data);

free(temp);

}

//Displaying the list

void display() {

if (head == NULL) {

printf("List is empty\n");

return;

}

struct node\* curr = head;

while (curr != NULL) {

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

curr = curr->next;

}

printf("NULL\n");

}

//Creating the list

void createList() {

int n, value;

printf("Enter the number of nodes: ");

scanf("%d", &n);

//Correcting the user if he enters negative value

if (n <= 0) {

printf("The number of nodes must be positive.\n");

return;

}

for (int i = 0; i < n; i++) {

printf("Enter value for node %d: ", i + 1);

scanf("%d", &value);

insertEnd(value);

}

}

int main() {

printf("Welcome to linked list operations\n");

createList();

bool over = false;

//Menu-driven program

while (!over) {

printf("\nEnter your choice:\n");

printf("1. Insert before node\n"); printf("2. Insert at end\n");

printf("3. Delete after node\n");

printf("4. Delete first node\n");

printf("5. Display\n");

printf("6. Exit\n");

//Getting choice from user

int choice;

scanf("%d", &choice);

switch (choice) {

case 1:

{

int a, value;

printf("Enter the number to be inserted: ");

scanf("%d", &a);

printf("Enter the value to be inserted before: ");

scanf("%d", &value);

insertBefore(a, value);

}

break;

case 2:

{

int b;

printf("Enter the number to be inserted at the end: ");

scanf("%d", &b);

insertEnd(b);

}

break;

case 3:

{

int value;

printf("Enter the value after which you want to delete: ");

scanf("%d", &value);

deleteAfter(value);

}

break;

case 4:

deleteFirst();

break;

case 5:

display();

break;

case 6:

printf("Exiting the program\n");

over = true;

break;

default:

printf("Invalid choice, please try again.\n");

break;

}

}

return 0;

}

Output:





