**Browser Tab Navigation System - C Programming Project**

Name: Durgam Ganesh Kumar

Regno: 24UG00507

**Problem Statement**

Design and simulate a browser tab navigation system using C programming, based on the concepts of data structures. For each page of the browser, store a unique pageID (integer) and URL (string).

Functionalities to implement:

1. Visit a new page

2. Go back

3. Go forward

4. Show current tab

5.Close current tab

6. Show history

7. Exit

**Approach**

This program simulates a browser using a **doubly linked list**. Each node represents a tab with an ID and name . Operations include opening a new tab, moving to the next tab, and moving to the previous tab. It demonstrates simple navigation in constant time with dynamic memory allocation.

**C Program Code**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef struct Tab {

int pageID;

char url[100];

struct Tab \*next;

struct Tab \*prev;

} Tab;

Tab \*head = NULL, \*tail = NULL, \*current = NULL;

int pageCounter = 1;

Tab\* createTab(char \*url) {

Tab \*newTab = (Tab\*)malloc(sizeof(Tab));

newTab->pageID = pageCounter++;

strcpy(newTab->url, url);

newTab->next = NULL;

newTab->prev = NULL;

return newTab;

}

void visitNewPage(char \*url) {

Tab \*newTab = createTab(url);

if (head == NULL) {

head = tail = current = newTab;

} else {

tail->next = newTab;

newTab->prev = tail;

tail = newTab;

current = newTab;

}

printf("\nVisited New Page -> PageID: %d, URL: %s\n", current->pageID, current->url);

}

void goForward() {

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

current = current->next;

printf("\nMoved Forward -> PageID: %d, URL: %s\n", current->pageID, current->url);

} else {

printf("\nNo forward tab exists!\n");

}

}

void goBack() {

if (current != NULL && current->prev != NULL) {

current = current->prev;

printf("\nMoved Back -> PageID: %d, URL: %s\n", current->pageID, current->url);

} else {

printf("\nNo back tab exists!\n");

}

}

void showCurrent() {

if (current != NULL) {

printf("\nCurrent Tab -> PageID: %d, URL: %s\n", current->pageID, current->url);

} else {

printf("\nNo tab is currently open!\n");

}

}

void closeCurrent() {

if (current == NULL) {

printf("\nNo tab to close!\n");

return;

}

printf("\nClosing Tab -> PageID: %d, URL: %s\n", current->pageID, current->url);

if (current->prev != NULL) current->prev->next = current->next;

else head = current->next;

if (current->next != NULL) current->next->prev = current->prev;

else tail = current->prev;

Tab \*temp = current;

if (current->next != NULL) current = current->next;

else current = current->prev;

free(temp);

}

void showHistory() {

if (head == NULL) {

printf("\nNo history available!\n");

return;

}

printf("\nBrowser History:\n");

Tab \*temp = head;

while (temp != NULL) {

printf("PageID: %d, URL: %s\n", temp->pageID, temp->url);

temp = temp->next;

}

}

int main() {

int choice;

char url[100];

while (1) {

printf("\n=== Browser Tab Navigation ===\n");

printf("1. Visit a New Page\n");

printf("2. Go Back\n");

printf("3. Go Forward\n");

printf("4. Show Current Tab\n");

printf("5. Close Current Tab\n");

printf("6. Show History\n");

printf("7. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

getchar(); // clear input buffer

switch (choice) {

case 1:

printf("Enter URL: ");

fgets(url, sizeof(url), stdin);

url[strcspn(url, "\n")] = 0;

visitNewPage(url);

break;

case 2:

goBack();

break;

case 3:

goForward();

break;

case 4:

showCurrent();

break;

case 5:

closeCurrent();

break;

case 6:

showHistory();

break;

case 7:

printf("\nExiting Browser...\n");

exit(0);

default:

printf("\nInvalid choice! Try again.\n");

}

}

return 0;

}

**Conclusion**

This program demonstrates a simple application of a **doubly linked list** through the management of browser-like tabs. Each tab is represented as a node containing an ID and a name, with pointers to both the previous and next nodes, enabling smooth two-way navigation. Operations such as opening new tabs, moving forward, and moving backward are implemented efficiently with minimal time complexity.

**OUTPUT**

1. **Visit a new page**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

1. **Go back**

A screenshot of a computer

AI-generated content may be incorrect.

1. **Go forward**

A screenshot of a computer

AI-generated content may be incorrect.

1. **Show current tab**

A screenshot of a computer

AI-generated content may be incorrect.

1. **Close current tab**

A screenshot of a computer

AI-generated content may be incorrect.

1. **Show history**

A screenshot of a computer

AI-generated content may be incorrect.

1. **Exit**

**A screenshot of a computer

AI-generated content may be incorrect.**