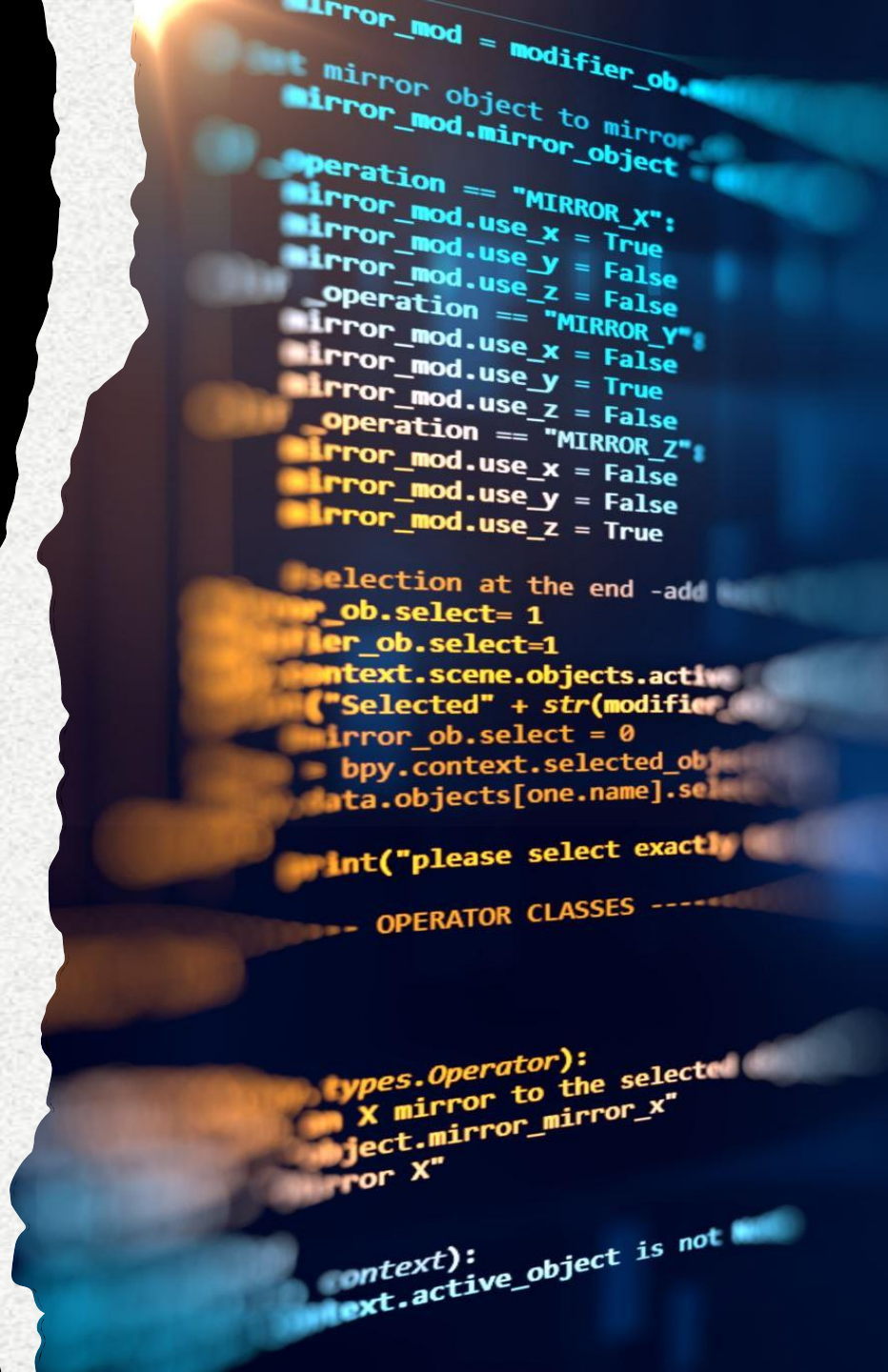
Title: Web Browser History Management System

Subtitle: Data Structures Project –Doubly Linked List Implementation

SUBMITTED BY: RIDHI

UID: 24BCD10027



Introduction

A simple browser history management system simulating the navigation of web pages.

Allows users to:

- Visit new pages .
- Go back to previously visited pages.
- Move forward in the history .
- View the complete browsing history.

System Features

VISIT NEW PAGE:

ADD NEW URLS TO THE BROWSING HISTORY.

•GO BACK: NAVIGATE TO THE PREVIOUS PAGE IN HISTORY.

•GO FORWARD: MOVE FORWARD TO THE NEXT PAGE IF AVAILABLE.

• EXIT: CLOSE THE APPLICATION GRACEFULLY.

Technologies Used

- Programming Language: C
- Data Structure: Doubly Linked List (for managing history) .
- Memory Management: Dynamic memory allocation using malloc() .
- I/O Operations: printf(), scanf(), and getchar() for user interaction .

Why Use Doubly Linked List for Browser History?

Doubly linked list

✅ **Doubly Linked List Advantages**

Allows **two-way navigation** (Back & Forward).

Each node stores:

- **URL**
- **Pointer to previous page**
- **Pointer to next page**

Dynamic memory: Easily add/remove pages.

Matches real browser behavior

Other data structures

❌ **Why Not Other Data Structures?**

🧱 **Stack**

- LIFO (Last In, First Out)
- Good for **backward**, but not for **forward** navigation.

🏞️ **Queue**

- FIFO (First In, First Out)
- Only supports **one-way movement**.

🧮 **Array**

- Fixed size or hard to resize
- **Slow insert/delete**
- Managing indexes is complex.

System Design

- Node Structure: Represents each visited URL with pointers to the previous and next nodes.
- Current Pointer: Tracks the current page the user is viewing.
- Navigation Logic: Handles forward and backward navigation efficiently.

WORKING:

Visiting a Page: Creates a new node and links it to the current node.

•Going Back: Moves the current pointer to the previous node.

•Going Forward: Moves to the next node if available.

•Displaying History: Traverses from the first node to the current node.

CREATING STRUCTURE OF A DOUBLE LINKED LIST

```
typedef struct Node {
    char url[100];
    struct Node* prev;
    struct Node* next;
} Node;
```

CREATING NODE:

```c
Node* createNode(char* url) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    if (!newNode) {
        printf("Memory allocation failed!\n");
        exit(1);
    }
    strcpy(newNode->url, url);
    newNode->prev = newNode->next = NULL;
    return newNode;
}
```

FUNCTION TO VISIT A PAGE

```c
void visitPage(char* url) {
    Node* newNode = createNode(url);
    if (current) {
        current->next = newNode;
        newNode->prev = current;
    }
    current = newNode;
    printf("Visited: %s\n", current->url);
}
```

FUNCTION TO MOVE BACK

```
void goBack() {
    if (current && current->prev) {
        current = current->prev;
        printf("Back to: %s\n", current->url);
    } else {
        printf("No previous page!\n");
    }
}
```

FUNCTION TO MOVE FORWARD

```
void goForward() {
    if (current && current->next) {
        current = current->next;
        printf("Forward to: %s\n", current->url);
    } else {
        printf("No forward page!\n");
    }
}
```

DISPLAY HISTORY

```
void displayHistory() {
    if (!current) {
        printf("No browsing history!\n");
        return;
    }
    Node* temp = current;
    while (temp->prev) temp = temp->prev;
    printf("\nBrowsing History:\n");
    while (temp) {
        printf("%s\n", temp->url);
        temp = temp->next;
    }
}
```

```
        switch (choice) {
            case 1:
                printf("Enter URL: ");
                scanf("%s", url);
                visitPage(url);
                break;
            case 2:
                goBack();
                break;
            case 3:
                goForward();
                break;
            case 4:
                displayHistory();
                break;
            case 5:
                printf("Exiting...\n");
                return 0;
            default:
                printf("Invalid choice! Try again.\n");
        }
    }
    return 0;
}
```

```
int main() {
    int choice;
    char url[100];
    while (1) {
        printf("\n1. Visit New Page\n");
printf("2. Go Back\n");
printf("3. Go Forward\n");
printf("4. Show History\n");
printf("5. Exit\n");
printf("Enter your choice: ");
scanf("%d", &choice);
        getchar();
```

MAIN FUNCTION

```
1. Visit New Page
2. Go Back
3. Go Forward
4. Show History
5. Exit
Enter your choice: 1
Enter URL: google.com
Visited: google.com

1. Visit New Page
2. Go Back
3. Go Forward
4. Show History
5. Exit
Enter your choice: 1
Enter URL: instagram.com
Visited: instagram.com

1. Visit New Page
2. Go Back
3. Go Forward
4. Show History
5. Exit
Enter your choice: 1
Enter URL: facebook.com
Visited: facebook.com

1. Visit New Page
2. Go Back
3. Go Forward
4. Show History
5. Exit
Enter your choice: 2
Back to: instagram.com
```

```
1. Visit New Page
2. Go Back
3. Go Forward
4. Show History
5. Exit
Enter your choice: 3
Forward to: facebook.com

1. Visit New Page
2. Go Back
3. Go Forward
4. Show History
5. Exit
Enter your choice: 4

Browsing History:
google.com
instagram.com
facebook.com

1. Visit New Page
2. Go Back
3. Go Forward
4. Show History
5. Exit
Enter your choice: 5
Exiting...
PS C:\Users\lenovo\OneDrive\Desktop\DSA C>
```

OUTPUT

EXAMPLE FLOWCHART

User Actions:

- Visit: google.com → facebook.com → twitter.com
- Go Back → Displays facebook.com
- Go Forward → Returns to twitter.com
- Show History → Displays all visited page

Case Study : Online Shopping Cart System

Introduction

An Online Shopping Cart System allows users to add, view, and manage products before checking out. Using the provided C code, which manages browser history with a doubly linked list, we can adapt it to create a basic shopping cart system.

Key Features:

- **Add Items to Cart**: Similar to visiting new pages, products are added as new nodes.
- **Remove Items**: Like navigating back, items can be removed from the cart.
- **View Cart**: Displays all items, akin to showing browsing history.
- **Checkout Process**: Finalizes the purchase and clears the cart.

SUMMARY

01

This project demonstrates web browser history management using a **doubly linked list in C**, allowing users to visit pages, navigate back and forward, and view history.

02

It enables **bidirectional navigation**, simulating browser back and forward functionality with efficient memory handling using **dynamic allocation** in C.

03

It highlights key concept of **pointer manipulation and** shows real-world relevance through a **shopping cart case study**.