TOPIC NO.: 10

TITLE: Circular Linked List

Program Code:

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node \*next;

};

struct Node \*createNode(int data) {

struct Node \*newNode = (struct Node \*)malloc(sizeof(struct Node));

if (newNode == NULL) {

printf("Memory allocation failed!\n");

exit(1);

}

newNode->data = data;

newNode->next = NULL;

return newNode;

}

void insertAtBeginning(struct Node \*\*head, int data) {

struct Node \*newNode = createNode(data);

if (\*head == NULL) {

newNode->next = newNode;

\*head = newNode;

} else {

struct Node \*temp = \*head;

while (temp->next != \*head) {

temp = temp->next;

}

temp->next = newNode;

newNode->next = \*head;

\*head = newNode;

}

}

void insertAtEnd(struct Node \*\*head, int data) {

struct Node \*newNode = createNode(data);

if (\*head == NULL) {

newNode->next = newNode;

\*head = newNode;

} else {

struct Node \*temp = \*head;

DATE: 20.03.24

while (temp->next != \*head) {

temp = temp->next;

}

temp->next = newNode;

newNode->next = \*head;

}

}

void insertAtMid(struct Node \*\*head, int data, int position) {

if (position <= 0) {

printf("Invalid Position!\n");

return;

}

if (position == 1) {

insertAtBeginning(head, data);

return;

}

struct Node \*newNode = createNode(data);

struct Node \*temp = \*head;

for (int i = 1; i < position - 1 && temp != NULL; i++) {

temp = temp->next;

}

if (temp == NULL) {

printf("position out of range!\n");

return;

}

newNode->next = temp->next;

temp->next = newNode;

}

void deleteAtBeginning(struct Node \*\*head) {

if (\*head == NULL) {

printf("List is empty!\n");

return;

}

if ((\*head)->next == \*head) {

free(\*head);

\*head = NULL;

} else {

struct Node \*temp = \*head;

while (temp->next != \*head) {

temp = temp->next;

}

temp->next = (\*head)->next;

struct Node \*delNode = \*head;

\*head = (\*head)->next;

free(delNode);

}

}

void deleteAtEnd(struct Node \*\*head) {

if (\*head == NULL) {

printf("List is empty!\n");

return;

}

if ((\*head)->next == \*head) {

free(\*head);

\*head = NULL;

} else {

struct Node \*temp = \*head;

struct Node \*prev = NULL;

while (temp->next != \*head) {

prev = temp;

temp = temp->next;

}

prev->next = \*head;

free(temp);

}

}

void deleteByValue(struct Node \*\*head, int value) {

if (\*head == NULL) {

printf("List is empty!\n");

return;

}

struct Node \*temp = \*head;

struct Node \*prev = NULL;

do {

if (temp->data == value) {

if (prev == NULL) {

deleteAtBeginning(head);

return;

} else {

prev->next = temp->next;

free(temp);

return;

}

}

prev = temp;

temp = temp->next;

} while (temp != \*head);

printf("Value not found in the list!\n");

}

int search(struct Node \*head, int value) {

if (head == NULL) {

printf("List is empty!\n");

return 0;

}

struct Node \*temp = head;

int position = 1;

do {

if (temp->data == value) {

printf("%d found at position %d.\n", value, position);

return 1;

}

position++;

temp = temp->next;

} while (temp != head);

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

return 0;

}

void display(struct Node \*head) {

if (head == NULL) {

printf("List is empty!\n");

return;

}

struct Node \*temp = head;

do {

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

temp = temp->next;

} while (temp != head);

printf("(Head)\n");

}

int main() {

struct Node \*head = NULL;

int choice, data, position, value;

while (1)

{

printf("\n\n1. Insert at beginning\n");

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

printf("3. Insert at mid\n");

printf("4. Delete at beginning\n");

printf("5. Delete at end\n");

printf("6. Delete by value\n");

printf("7. Search\n");

printf("8. Display\n");

printf("9. Exit\n");

printf("Enter your choice : ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter data to insert at beginning : ");

scanf("%d", &data);

insertAtBeginning(&head, data);

break;

case 2:

printf("Enter data to insert at end : ");

scanf("%d", &data);

insertAtEnd(&head, data);

break;

case 3:

printf("Enter data to insert : ");

scanf("%d", &data);

printf("Enter position to insert at : ");

scanf("%d", &position);

insertAtMid(&head, data, position);

break;

case 4:

deleteAtBeginning(&head);

break;

case 5:

deleteAtEnd(&head);

break;

case 6:

printf("Enter value to delete: ");

scanf("%d", &value);

deleteByValue(&head, value);

break;

case 7:

printf("Enter value to search : ");

scanf("%d", &value);

search(head, value);

break;

case 8:

printf("Linked list : ");

display(head);

break;

case 9:

exit(0);

break;

default:

printf("Invalid choice! Please enter a valid option.\n");

}

}

return 0;

}

Output:

1. Insert at beginning

2. Insert at end

3. Insert at mid

4. Delete at beginning

5. Delete at end

6. Delete by value

7. Search

8. Display

9. Exit

Enter your choice : 1

Enter data to insert at beginning : 5

1. Insert at beginning

2. Insert at end

3. Insert at mid

4. Delete at beginning

5. Delete at end

6. Delete by value

7. Search

8. Display

9. Exit

Enter your choice : 2

Enter data to insert at end : 7

1. Insert at beginning

2. Insert at end

3. Insert at mid

4. Delete at beginning

5. Delete at end

6. Delete by value

7. Search

8. Display

9. Exit

Enter your choice : 3

Enter data to insert : 9

Enter position to insert at : 2

1. Insert at beginning

2. Insert at end

3. Insert at mid

4. Delete at beginning

5. Delete at end

6. Delete by value

7. Search

8. Display

9. Exit

Enter your choice : 8

Linked list : 5->9->7->(Head)

1. Insert at beginning

2. Insert at end

3. Insert at mid

4. Delete at beginning

5. Delete at end

6. Delete by value

7. Search

8. Display

9. Exit

Enter your choice : 4

1. Insert at beginning

2. Insert at end

3. Insert at mid

4. Delete at beginning

5. Delete at end

6. Delete by value

7. Search

8. Display

9. Exit

Enter your choice : 8

Linked list : 9->7->(Head)

1. Insert at beginning

2. Insert at end

3. Insert at mid

4. Delete at beginning

5. Delete at end

6. Delete by value

7. Search

8. Display

9. Exit

Enter your choice : 7

Enter value to search : 7

7 found at position 2.

1. Insert at beginning

2. Insert at end

3. Insert at mid

4. Delete at beginning

5. Delete at end

6. Delete by value

7. Search

8. Display

9. Exit

Enter your choice : 6

Enter value to delete: 7

1. Insert at beginning

2. Insert at end

3. Insert at mid

4. Delete at beginning

5. Delete at end

6. Delete by value

7. Search

8. Display

9. Exit

Enter your choice : 8

Linked list : 9->(Head)

1. Insert at beginning

2. Insert at end

3. Insert at mid

4. Delete at beginning

5. Delete at end

6. Delete by value

7. Search

8. Display

9. Exit

Enter your choice : 9