Neh Samir Joshi

19BEC1112

21/01/2022

CSE2003

Data Structures and Algorithms

[LAB]

LAB – 2

Stacks and Queues using Linked Lists

Aim: To learn how to implement stacks, queues and circular queues and its applications using Linked Lists.

Software Required: Code editor (e.g. VS Code, Dev C++), GCC/G++ compiler

Task 1: To create a stack using Linked list and perform basic operations

Code:

```
#include <iostream>
using namespace std;

struct Node
{
    int value;
    Node *next;
};
int top = -1;

void Display(struct Node *head)
{
    cout << "-----------" << endl;
    while (head != NULL)
    {
        cout << head->value << endl;
        head = head->next;
```

```
    }
    cout << "-----------" << endl;
    return;
}
void Push(struct Node **head, int data)
{
    struct Node *temp = new Node;
    struct Node *n = *head;
    temp->value = data;
    temp->next = NULL;
    if (top == -1)
    {
        (*head) = temp;
        top += 1;
        return;
    }
    while (n->next != NULL)
    {
        n = n->next;
    }
    top += 1;
    n->next = temp;
    return;
}

void Pop(struct Node **head)
{
    struct Node *n = *head;
    if (top == -1)
    {
        cout << "Stack is empty!" << endl;
        return;
    }
    if (top == 0)
    {
        *head = NULL;
        top = -1;
        return;
    }
    while (n->next->next != NULL)
    {
        n = n->next;
    }
    struct Node *del = n->next;
    free(del);
```

```
    n->next = NULL;
    top = top - 1;
    return;
}

int main()
{

    struct Node *head = new Node;
    Push(&head, 20);
    Push(&head, 30);
    Display(head);
    cout << "Now popping" << endl;
    Pop(&head);
    Display(head);
    return 0;
}
```

Output:

```
PS C:\Users\OS\Desktop\6th Semester\CSE2003 Data Structures and Algorithms\Code\LAB2> cd "c:\Users\OS\Desktop\6th Semester\CSE200
3 Data Structures and Algorithms\Code\LAB2\" ; if ($?) { g++ stack_using_ll.cpp -o stack_using_ll } ; if ($?) { .\stack_using_ll
}
----------
20
30
----------
Now popping
----------
20
----------
```

Task 2: Implement queue using Linked Lists and perform enqueue() and dequeue() operations

Code:

```
#include <iostream>
using namespace std;

struct Node
{
    int value;
    Node *next;
};
int rear = -1;

void Display(struct Node *head)
{
    cout << "-----------" << endl;
    while (head != NULL)
    {
        cout << head->value << endl;
        head = head->next;
    }
    cout << "-----------" << endl;
    return;
}

void Enqueue(struct Node **head, int data)
{
    struct Node *temp = new Node;
    temp->value = data;
    if (rear == -1)
    {
        temp->next = NULL;
        *head = temp;
        rear += 1;
        return;
    }
    temp->next = *head;
    *head = temp;
    rear++;
    return;
}
void Dequeue(struct Node **head)
```

```
{
    struct Node *n = *head;
    if (rear == -1)
    {
        cout << "Stack is empty!" << endl;
        return;
    }
    if (rear == 0)
    {
        *head = NULL;
        rear = -1;
        return;
    }
    while (n->next->next != NULL)
    {
        n = n->next;
    }
    n->next = NULL;
    rear = rear - 1;
    return;
}

int main()
{
    struct Node *head = new Node;
    Enqueue(&head, 10);
    Display(head);
    Enqueue(&head, 20);
    Display(head);
    Dequeue(&head);
    Display(head);
    return 0;
}
```

Output:

```
PS C:\Users\OS\Desktop\6th Semester\CSE2003 Data Structures and Algorithms\Code\LAB2> cd "c:\Users\OS\Desktop\6th Semester\CSE200
3 Data Structures and Algorithms\Code\LAB2\" ; if ($?) { g++ queue_using_ll.cpp -o queue_using_ll } ; if ($?) { .\queue_using_ll
}
----------
10
----------
----------
20
10
----------
----------
20
----------
PS C:\Users\OS\Desktop\6th Semester\CSE2003 Data Structures and Algorithms\Code\LAB2>
```

Task 3: Implement a circular queue using Linked Lists

Code:

```
#include <iostream>
using namespace std;

struct Node {
    int data;
    Node *next;
};
void Display(struct Node *head)
{
    if (head == NULL)
    {
        cout << "Circular queue is empty!" << endl;
        return;
    }
    struct Node *initial = head;
    do
    {
        cout << head->data << endl;
        head = head->next;
    } while (head != initial);
}

int GetLength(struct Node *head)
{
    int length = 0;
    if (head == NULL)
    {
        return length;
```

```
    }
    else if (head != NULL && head->next == NULL)
        return 1;
    else
    {
        struct Node *initial = head;
        head = head->next;
        length = 1;
        while (head != initial)
        {
            length++;
            head = head->next;
        }
        return length;
    }
}

struct Node *Enqueue(struct Node *head, int data){
    struct Node *temp = new Node;
    temp->data = data;
    if (GetLength(head) == 0)
    {
        head = temp;
        head->next = head;
    }
    else
    {
        struct Node *initial = head;
        while (head->next != initial)
        {
            head = head->next;
        }
        temp->next = initial;
        head->next = temp;
        head = initial;
    }
    return head;
}

struct Node *Dequeue(struct Node *head){
    int l = GetLength(head);

    if (l==0) {
        cout << "Circular Queue is empty!" << endl;
```

```
    }
    else if(l==1){
        head=NULL;
    }
    else {
        struct Node* newHead = head->next;
        struct Node* initial = head;
        while (head->next != initial) head = head->next;
        head->next = newHead;
        head = newHead;
    }
    return head;
}

int main(){
    struct Node *head = new Node;
    head = NULL;
    head = Enqueue(head, 10);
    head = Enqueue(head, 20);
    head = Enqueue(head, 30);
    head = Enqueue(head, 40);
    head = Dequeue(head);
    head = Dequeue(head);
    Display(head);
}
```

Output:

```
PS C:\Users\OS\Desktop\6th Semester\CSE2003 Data Structures and Algorithms\Code\LAB2> cd "c:\Users\OS\Desktop\6th Semester\CSE200
3 Data Structures and Algorithms\Code\LAB2\" ; if ($?) { g++ circular_queue.cpp -o circular_queue } ; if ($?) { .\circular_queue
}
30
40
PS C:\Users\OS\Desktop\6th Semester\CSE2003 Data Structures and Algorithms\Code\LAB2>
```

Task 4: Implement the solution of tower of Hanoi for “n” number of disks

Code:

```cpp
#include <iostream>
using namespace std;

void solveProblem(int n, int from, int to, int intermidate){
    if (n==0){
        return;
    }
    solveProblem(n-1, from, intermidate, to);
    cout << "Moving disk " << n << " from rod " << from << " to rod " << to
<<endl;
    solveProblem(n-1, intermidate, to, from);
}

int main(){
    solveProblem(2, 1, 3, 2);
    return 0;
}
```

Output:

```
PS C:\Users\OS\Desktop\6th Semester\CSE2003 Data Structures and Algorithms\Code\LAB2> cd "c:\Users\OS\Desktop\6th Semester\CSE200
3 Data Structures and Algorithms\Code\LAB2\" ; if ($?) { g++ tower_of_hanoi.cpp -o tower_of_hanoi } ; if ($?) { .\tower_of_hanoi
}
Moving disk 1 from rod 1 to rod 2
Moving disk 2 from rod 1 to rod 3
Moving disk 1 from rod 2 to rod 3
PS C:\Users\OS\Desktop\6th Semester\CSE2003 Data Structures and Algorithms\Code\LAB2>
```

[For n=2]

Task 5: Implement balancing parenthesis to check if the given string of parentheses are appropriately balanced.

Code:

This has been implemented using Stack. For the stack code, refer to Task 1.

For all the other functions, this is the code:

```
char GetTop(struct Node *head)
{
    char top;
    while (head->next != NULL)
    {
        head = head->next;
    }
    top = head->data;
    return top;
}
bool CheckBalanced(string s, struct Node *head)
{
    for (int i = 0; i < s.length(); i++)
    {
        if (s[i] == '{' || s[i] == '[' || s[i] == '(')
        {
            head = Push(head, s[i]);
            continue;
        }
        if (GetLength(head) == 0)
            return false;

        if (s[i] == ')')
        {
            char top = GetTop(head);
            head = Pop(head);
            if (top == '[' || top == '{')
            {
                break;
                return false;
            } else continue;
        }
        else if (s[i] == ']')
        {
            char top = GetTop(head);
            head = Pop(head);
            if (top == '(' || top == '{')
            {
```

```
                break;
                return false;
            } else continue;
        }
        else if (s[i] == '}')
        {
            char top = GetTop(head);
            head = Pop(head);
            if (top == '[' || top == '('){
                break;
                return false;
            } else continue;
        }
    }
    if (GetLength(head) == 0)
        return true;
    else
        return false;
}

int main()
{
    struct Node *head = new Node;
    head = NULL;
    if (CheckBalanced("{{{{()}}}}", head))
        cout << "Balanced!" << endl;
    else
        cout << "Unbalanced!" << endl;
}
```

Output:

```
PS C:\Users\OS\Desktop\6th Semester\CSE2003 Data Structures and Algorithms\Code\LAB2> cd "c:\Users\OS\Desktop\6th Semester\CSE200
3 Data Structures and Algorithms\Code\LAB2\" ; if ($?) { g++ balancing_parantheses.cpp -o balancing_parantheses } ; if ($?) { .\b
alancing_parantheses }
Balanced!
PS C:\Users\OS\Desktop\6th Semester\CSE2003 Data Structures and Algorithms\Code\LAB2>
```

Conclusion

Thus we have successfully implemented stacks and queues using Linked Lists and seen some of their basic applications.

Neh Samir Joshi

19BEC1112