AMAN PRAJAPATI

Linked List Implementation

C++ Program to Implement Singly Linked List

main.cpp

```cpp
#include <iostream>
using namespace std;
struct Node {
    int data;
    struct Node *next;
};
struct Node* head = NULL;
void insert(int new_data) {
    struct Node* new_node = (struct Node*) malloc(sizeof(struct Node));
    new_node->data = new_data;
    new_node->next = head;
    head = new_node;
}
void display() {
    struct Node* ptr;
    ptr = head;
    while (ptr != NULL) {
        cout<< ptr->data <<" ";
        ptr = ptr->next;
    }
}
int main() {
    insert(3);
    insert(1);
    insert(7);
    insert(2);
    insert(9);
    cout<<"The linked list is: ";
    display();
    return 0;
}
```

Console Shell

```
> make -s
> ./main
The linked list is: 9 2 7 1 3 >
```

Implement Doubly Linked List

main.cpp

```cpp
#include <iostream>
using namespace std;
struct Node {
    int data;
    struct Node *prev;
    struct Node *next;
};
struct Node* head = NULL;
void insert(int newdata) {
    struct Node* newnode = (struct Node*) malloc(sizeof(struct Node));
    newnode->data = newdata;
    newnode->prev = NULL;
    newnode->next = head;
    if(head != NULL)
    head->prev = newnode ;
    head = newnode;
}
void display() {
    struct Node* ptr;
    ptr = head;
    while(ptr != NULL) {
        cout<< ptr->data <<" ";
        ptr = ptr->next;
    }
}
int main() {
    insert(3);
    insert(1);
    insert(7);
    insert(2);
    insert(9);
    cout<<"The doubly linked list is: ";
    display();
    return 0;
```

Console Shell

```
make -s
./main
The doubly linked list is: 9 2 7 1 3
```

Circular Linked List

main.cpp

```cpp
#include <iostream>
using namespace std;
struct Node {
  int data;
  struct Node* next;
};
struct Node* addToEmpty(struct Node* last, int data) {
  if (last != NULL) return last;
  struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
  newNode->data = data;
  last = newNode;
 last->next = last;
  return last;
}
struct Node* addFront(struct Node* last, int data) {
  if (last == NULL) return addToEmpty(last, data);
  struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
  newNode->data = data;
  newNode->next = last->next;
  last->next = newNode;
  return last;
}
struct Node* addEnd(struct Node* last, int data) {
  if (last == NULL) return addToEmpty(last, data);
  struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
  newNode->data = data;
  newNode->next = last->next;
  last->next = newNode;
  last = newNode;
  return last;
}
struct Node* addAfter(struct Node* last, int data, int item) {
  if (last == NULL) return NULL;
  struct Node *newNode, *p;
```

Console Shell

```
> make -s
> ./main
2 10 6 8
10 6 2 >
```


```cpp
struct Node* addAfter(struct Node* last, int data, int item) {
  if (last == NULL) return NULL;
  struct Node *newNode, *p;
```


```cpp
  p = last->next;
  do {
  if (p->data == item) {
    newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = p->next;
    p->next = newNode;
    if (p == last) last = newNode;
    return last;
  }
  p = p->next;
  } while (p != last->next);
  cout << "\nThe given node is not present in the list" << endl;
  return last;
}
void deleteNode(Node** last, int key) {
  if (*last == NULL) return;
  if ((*last)->data == key && (*last)->next == *last) {
  free(*last);
  *last = NULL;
  return;
  }
  Node *temp = *last, *d;
  if ((*last)->data == key) {
  while (temp->next != *last) temp = temp->next;
  temp->next = (*last)->next;
  free(*last);
  *last = temp->next;
  }
  while (temp->next != *last && temp->next->data != key) {
  temp = temp->next;
  }
```


```
> make -s
> ./main
2 10 6 8
10 6 2 >
```

main.cpp

```cpp
  while (temp->next != *last && temp->next->data != key) {
  temp = temp->next;
  }
  if (temp->next->data == key) {
  d = temp->next;
  temp->next = d->next;
  free(d);
  }
 }
void traverse(struct Node* last) {
  struct Node* p;
  if (last == NULL) {
  cout << "The list is empty" << endl;
  return;
  }
  p = last->next;
 do {
  cout << p->data << " ";
  p = p->next;
  } while (p != last->next);
 }

int main() {
  struct Node* last = NULL;
  last = addToEmpty(last, 6);
  last = addEnd(last, 8);
  last = addFront(last, 2);
  last = addAfter(last, 10, 2);
  traverse(last);
  deleteNode(&last, 8);
  cout << endl;
  traverse(last);
  return 0;
 }
```

Console Shell

```
> make -s
> ./main
2 10 6 8
10 6 2 >
```

Implement Stack using linked list

main.cpp

```cpp
#include <iostream>
using namespace std;
struct Node {
    int data;
    struct Node *next;
};
struct Node* top = NULL;
void push(int val) {
    struct Node* newnode = (struct Node*) malloc(sizeof(struct Node));
    newnode->data = val;
    newnode->next = top;
    top = newnode;
}
void pop() {
    if(top==NULL)
    cout<<"Stack Underflow"<<endl;
    else {
        cout<<"The popped element is "<< top->data <<endl;
        top = top->next;
    }
}
void display() {
    struct Node* ptr;
    if(top==NULL)
    cout<<"stack is empty";
    else {
        ptr = top;
```

Console Shell

```
> make -s
> ./main
1) Push in stack
2) Pop from stack
3) Display stack
4) Exit
Enter choice:
1
Enter value to be pushed:
4
Enter choice:
1
Enter value to be pushed:
3
Enter choice:
1
Enter value to be pushed:
6
Enter choice:
2
The popped element is 6
Enter choice:
3
Stack elements are: 3 4
Enter choice:
```

main.cpp

```cpp
    if(top==NULL)
    cout<<"stack is empty";
    else {
        ptr = top;
        cout<<"Stack elements are: ";
        while (ptr != NULL) {
            cout<< ptr->data <<" ";
            ptr = ptr->next;
        }
    }
    cout<<endl;
}
int main() {
    int ch, val;
    cout<<"1) Push in stack"<<endl;
    cout<<"2) Pop from stack"<<endl;
    cout<<"3) Display stack"<<endl;
    cout<<"4) Exit"<<endl;
    do {
        cout<<"Enter choice: "<<endl;
        cin>>ch;
        switch(ch) {
            case 1: {
                cout<<"Enter value to be pushed:"<<endl;
                cin>>val;
                push(val);
                break;
            }
```

Console Shell

```
> make -s
> ./main
1) Push in stack
2) Pop from stack
3) Display stack
4) Exit
Enter choice:
1
Enter value to be pushed:
4
Enter choice:
1
Enter value to be pushed:
3
Enter choice:
1
Enter value to be pushed:
6
Enter choice:
2
The popped element is 6
Enter choice:
3
Stack elements are: 3 4
Enter choice:
```

main.cpp

```cpp
            cout<<"Enter value to be pushed:"<<endl;
            cin>>val;
            push(val);
            break;
        }
        case 2: {
            pop();
            break;
        }
        case 3: {
            display();
            break;
        }
        case 4: {
            cout<<"Exit"<<endl;
            break;
        }
        default: {
            cout<<"Invalid Choice"<<endl;
        }
    }
}while(ch!=4);
return 0;
}
```

Console Shell

```
> make -s
> ./main
1) Push in stack
2) Pop from stack
3) Display stack
4) Exit
Enter choice:
1
Enter value to be pushed:
4
Enter choice:
1
Enter value to be pushed:
3
Enter choice:
1
Enter value to be pushed:
6
Enter choice:
2
The popped element is 6
Enter choice:
3
Stack elements are: 3 4
Enter choice:
```

Implement Queue using Linked List

main.cpp

```cpp
#include <iostream>
using namespace std;
struct node {
    int data;
    struct node *next;
};
struct node* front = NULL;
struct node* rear = NULL;
struct node* temp;
void Insert() {
    int val;
    cout<<"Insert the element in queue : "<<endl;
    cin>>val;
    if (rear == NULL) {
        rear = (struct node *)malloc(sizeof(struct node));
        rear->next = NULL;
        rear->data = val;
        front = rear;
    } else {
        temp=(struct node *)malloc(sizeof(struct node));
        rear->next = temp;
        temp->data = val;
        temp->next = NULL;
        rear = temp;
    }
}
void Delete() {
    temp = front;
```

Console Shell

```
> make -s
> ./main
1) Insert element to queue
2) Delete element from queue
3) Display all the elements of queue
4) Exit
Enter your choice :
1
Insert the element in queue :
4
Enter your choice :
1
Insert the element in queue :
5
Enter your choice :
1
Insert the element in queue :
6
Enter your choice :
2
Element deleted from queue is : 4
Enter your choice :
3
Queue elements are: 5 6
Enter your choice :
```

main.cpp

```cpp
    if (front == NULL) {
        cout<<"Underflow"<<endl;
        return;
    }
    else
    if (temp->next != NULL) {
        temp = temp->next;
        cout<<"Element deleted from queue is : "<<front->data<<endl;
        free(front);
        front = temp;
    } else {
        cout<<"Element deleted from queue is : "<<front->data<<endl;
        free(front);
        front = NULL;
        rear = NULL;
    }
}
void Display() {
    temp = front;
    if ((front == NULL) && (rear == NULL)) {
        cout<<"Queue is empty"<<endl;
        return;
    }
    cout<<"Queue elements are: ";
    while (temp != NULL) {
        cout<<temp->data<<" ";
        temp = temp->next;
    }
    cout<<endl;
```

Console Shell

```
> make -s
> ./main
1) Insert element to queue
2) Delete element from queue
3) Display all the elements of queue
4) Exit
Enter your choice :
1
Insert the element in queue :
4
Enter your choice :
1
Insert the element in queue :
5
Enter your choice :
1
Insert the element in queue :
6
Enter your choice :
2
Element deleted from queue is : 4
Enter your choice :
3
Queue elements are: 5 6
Enter your choice :
```

main.cpp

```cpp
}
int main() {
    int ch;
    cout<<"1) Insert element to queue"<<endl;
    cout<<"2) Delete element from queue"<<endl;
    cout<<"3) Display all the elements of queue"<<endl;
    cout<<"4) Exit"<<endl;
    do {
        cout<<"Enter your choice : "<<endl;
        cin>>ch;
        switch (ch) {
            case 1: Insert();
            break;
            case 2: Delete();
            break;
            case 3: Display();
            break;
            case 4: cout<<"Exit"<<endl;
            break;
            default: cout<<"Invalid choice"<<endl;
        }
    } while(ch!=4);
    return 0;
}
```

Console Shell

```
make -s
./main
1) Insert element to queue
2) Delete element from queue
3) Display all the elements of queue
4) Exit
Enter your choice :
1
Insert the element in queue :
4
Enter your choice :
1
Insert the element in queue :
5
Enter your choice :
1
Insert the element in queue :
6
Enter your choice :
2
Element deleted from queue is : 4
Enter your choice :
3
Queue elements are: 5 6
Enter your choice :
```