SINGLY LINKED LIST

#include<iostream>

using namespace std;

class Node{

    public:

    int data;

    Node\* next;

    Node(int data){

        this->data=data;

        this->next=NULL;

    }

};

class LinkedList{

    public:

    void addFirst(Node\* &head,int data){

        Node\* newNode = new Node(data);

        if(head==NULL){

            head=newNode;

        }

        else{

            newNode->next=head;

            head=newNode;

        }

    }

    void addLast(Node\* &head, int data){

        Node\* newNode=new Node(data);

        Node\* currNode=head;

        if(head==NULL){

            head=newNode;

        }

        else{

            while(currNode->next!=NULL){

                currNode=currNode->next;

            }

            currNode->next=newNode;

        }

    }

    int deleteFirst(Node\* &head){

        Node\* currNode=head;

        if(head==NULL){

           cout << "No nodes to display" << endl;

        }

        head=head->next;

        free(currNode);

    }

    void deleteEnd(Node\* &head){

        Node\* currNode=head;

        Node\* ptr;

        if(head==NULL){

            cout << "No nodes to display" << endl;

        }

        while(currNode->next!=NULL){

            ptr=currNode;

            currNode=currNode->next;

        }

        ptr->next=NULL;

        free(currNode);

    }

    void addatSpecific(Node\* &head,int location,int data){

        Node\* newNode=new Node(data);

        Node\* currNode=head;

        for(int i=0;i<location;i++){

            if(currNode->next!=NULL){

                currNode=currNode->next;

            }

        }

        newNode->next=currNode->next;

        currNode->next=newNode;

    }

    void deletetSpecific(Node\* &head,int location){

        Node\* currNode=head;

        Node\* previous;

        for(int i=0;i<=location;i++){

            previous=currNode;

            currNode=currNode->next;

        }

        previous->next=currNode->next;

        free(currNode);

    }

    void printList(Node\* &head){

        Node\* currNode=head;

        while(currNode!=NULL){

            cout << currNode->data << " -> ";

            currNode=currNode->next;

        }

        cout << "NULL";

    }

};

int main(){

    LinkedList ll;

    Node\* head=NULL;

    int ch;

    int position;

    int element;

    do{

        cout << "\n1.AddFirst 2.AddLast 3.printList 4.DeleteFirst 5.DeleteLast 6.Insert at Specific position 7. Delete at Specific position 8.Exit" << endl;

        cout << "Enter the choice: " << endl;

        cin >> ch;

        switch(ch){

            case 1: cout << "Enter the element: " << endl;

                    cin >> element;

                    ll.addFirst(head,element);

                    break;

            case 2: cout << "Enter the element: " << endl;

                    cin >> element;

                    ll.addLast(head,element);

                    break;

            case 3: ll.printList(head);

                    break;

            case 4: ll.deleteFirst(head);

                    break;

            case 5: ll.deleteEnd(head);

                    break;

            case 6: cout << "Enter the position: ";

                    cin >> position;

                    cout << "Enter the element:";

                    cin >> element;

                    ll.addatSpecific(head,position,element);

                    break;

            case 7: cout << "Enter the position: ";

                    cin >> position;

                    ll.deletetSpecific(head,position);

            case 8: exit(0);

        }

    }while(1);

}

DOUBLY LINKED LIST

#include<iostream>

using namespace std;

class Node{

    public:

    int data;

    Node\* prev;

    Node\* next;

    Node(int data){

        this->data=data;

        this->next=NULL;

        this->prev=NULL;

    }

};

class DoublyLinkedList{

    public:

    void addFirst(Node\* &head,int data){

        Node\* newNode= new Node(data);

        newNode->next=head;

        if(head!=NULL){

            head->prev=newNode;

        }

        head=newNode;

    }

    void addLast(Node\* &head,int data){

        Node\* newNode=new Node(data);

        Node\* currNode=head;

        if(head==NULL){

            addFirst(head,data);

        }

        while(currNode->next!=NULL){

            currNode=currNode->next;

        }

        currNode->next=newNode;

        newNode->prev=currNode;

    }

    void printList(Node\* head){

        Node\* currNode=head;

        while(currNode!=NULL){

            cout << currNode->data << "->";

            currNode=currNode->next;

        }

        cout << "NULL";

    }

};

int main(){

    DoublyLinkedList dll;

    Node\* head=NULL;

    int ch;

    int element;

    //do{

    for(;;){

        cout << "\n 1.AddFirst 2.AddLast 2.printList" << endl;

        cout << "Enter the choice: " << endl;

        cin >> ch;

        switch(ch){

            case 1: cout << "Enter the element: " << endl;

                    cin >> element;

                    dll.addFirst(head,element);

                    break;

            case 2: cout << "Enter the element: " << endl;

                    cin >> element;

                    dll.addLast(head,element);

                    break;

            case 3: dll.printList(head);

                    break;

        }

    //}while(1);

    }

}

CIRCULAR LINKED LIST

#include<iostream>

using namespace std;

class Node{

    public:

    int data;

    Node\* next;

    Node(int val){

        data=val;

        next=NULL;

    }

};

class CircularLinkedList{

    public:

    void insertatHead(Node\* &head,int val){

        Node\* newNode=new Node(val);

        if(head==NULL){

            newNode->next=newNode;

            head=newNode;

            return;

        }

        Node\* currNode=head;

        while(currNode!=head){

            currNode=currNode->next;

        }

        currNode->next=newNode;

        newNode->next=head;

        head=newNode;

    }

    void insertatTail(Node\* &head,int val){

        Node\* newNode=new Node(val);

        if(head==NULL){

            insertatHead(head,val);

            return;

        }

        Node\* currNode=head;

        while(currNode->next!=head){

            currNode=currNode->next;

        }

        currNode->next=newNode;

        newNode->next=head;

    }

    void printList(Node\* head){

        Node\* currNode=head;

        do{

            cout << currNode->data << "->";

            currNode=currNode->next;

        }while(currNode!=head);

        cout << head->data;

    }

};

int main(){

    Node\* head=NULL;

    CircularLinkedList cll;

    int ch;

    int element;

    do{

        cout << "\nENTER THE CHOICE: " << endl;

        cout << "1. InsertatHead 2. InsertatTail 3.PrintList" << endl;

        cin >> ch;

        switch(ch){

            case 1: cout << "Enter the element: ";

                    cin >> element;

                    cll.insertatHead(head,element);

                    break;

            case 2: cout << "Enter the element: ";

                    cin >> element;

                    cll.insertatTail(head,element);

                    break;

            case 3: cll.printList(head);

                    break;

        }

    }while(1);

}

STACK USING ARRAYS

#include<iostream>

using namespace std;

int stack[5];

int n=5;

int top=-1;

void push(int element){

    if(top==n-1){

        cout << "Overflow";

    }

    else{

        top++;

        stack[top]=element;

    }

}

void pop(){

    if(top==-1){

        cout << "Underflow";

    }

    else{

        top--;

    }

}

void display(){

    //for(int i=top;i>=0;i--){

        cout << stack[top];

    //}

}

void Displapelementspresent(){

    if(top>=0){

        for(int i=top;i>=0;i--){

            cout << stack[i] << " ";

        }

    }

}

int main(){

    int val;

    int ch;

    do{

        cout << endl << "1.Push 2.Display 3.Pop 4.Display all elements present";

        cout << endl << "Enter the choice :";

        cin >> ch;

        switch(ch){

            case 1: cout << "Enter the value: ";

                    cin >> val;

                    push(val);

                    break;

            case 2: display();

                    break;

            case 3: pop();

                    break;

            case 4: Displapelementspresent();

                    break;

        }

    }while(1);

}

STACK USING LINKED LIST

#include<iostream>

using namespace std;

class Node{

    public:

    int data;

    Node\* next;

    Node(int val){

        data=val;

        next=NULL;

    }

};

Node\* top;

class Stack{

    public:

    void push(int val){           // push function is same as insertion at head in linked list

        Node\* newNode=new Node(val);

        if(top==NULL){

            top=newNode;

        }

        else{

            newNode->next=top;

            top=newNode;

        }

    }

    void pop(){

        Node\* temp=top;

        if(top==NULL){

            cout << "No Nodes to delete";

        }

        else{

            int data=top->data;

            cout << "Deleted data is: " << data;

            top=top->next;

            free(temp);

        }

    }

    void peek(){

        if(top==NULL){

            cout << "No nodes to display";

        }

        else{

            cout << "Top most element: " << top->data << endl;

        }

    }

    void display(){

        Node\* temp=top;

        if(top==NULL){

            cout << "Stack Underflow";

        }

        else{

            while(temp!=NULL){

                cout << temp->data << "->";

                temp=temp->next;

            }

            cout << "NULL";

        }

    }

};

int main(){

    Stack s;

    int ch;

    int element;

    do{

        cout << "\n1.Push 2.Pop 3.Peek 4.Display" << endl;

        cout << "Enter the choice: " << endl;

        cin >> ch;

        switch(ch){

            case 1: cout << "Enter the element: " << endl;

                    cin >> element;

                    s.push(element);

                    break;

            case 2: s.pop();

                    break;

            case 3: s.peek();

                    break;

            case 4: s.display();

                    break;

        }

    }while(1);

}

QUEUE USING ARRAYS

#include<iostream>

#define n 5

using namespace std;

int queue[n];

int front=-1;

int rear=-1;

void enqueue(int element){

    if(rear==n-1){

        cout << "Queue is full" << endl;

    }

    else{

        if(front==-1){

            front=0;

        }

        rear++;

        queue[rear]=element;

    }

}

void dequeue(){

    if(front==-1 && rear==-1){

        cout << "Queue is Empty";

    }

    else{

        int deleteddata=queue[front];

        cout << "Deleted element is: " << deleteddata << endl;

        if(front == rear){

            front = -1;

            rear = -1;

        }

        else{

            front++;

        }

    }

}

void search(int var){

    for(int i=front;i<=rear;i++){

        if(queue[i]==var){

            cout << "Element is found and is at " << i << " position" << endl;

        }

        else{

            cout << "Element is not found" << endl;

        }

    }

}

void display(){

    cout <<  "Elements in the queue are: " << endl;

    for(int i=front;i<=rear;i++){

        cout << queue[i] << " ";

    }

}

int main(){

    int val;

    int ch;

    do{

        cout << endl << "1.Enqueue 2.Display 3.Dequeue 4.Search 5.Exit";

        cout << endl << "Enter the choice :";

        cin >> ch;

        switch(ch){

            case 1: cout << "Enter the value: ";

                    cin >> val;

                    enqueue(val);

                    break;

            case 2: display();

                    break;

            case 3: dequeue();

                    break;

            case 4: cout << "Enter the value: ";

                    cin >> val;

                    search(val);

                    break;

            case 5: exit(0);

                    break;

        }

    }while(1);

}

QUEUE USING LINKED LIST

#include<iostream>

using namespace std;

class Node{

    public:

    int data;

    Node\* next;

    Node(int val){

        this->data=val;

        this->next=NULL;

    }

};

class Queue{

    public:

    Node\* front;

    Node\* back;

    Queue(){

        front=NULL;

        back=NULL;

    }

    void enqueue(int value){

        Node\* newNode=new Node(value);

        if(front==NULL){

            front=newNode;

            back=newNode;

            return;

        }

        else{

            back->next=newNode;

            back=newNode;

        }

    }

    void dequeue(){

        if(front==NULL){

            cout << "No nodes to delete" << endl;

        }

        else{

            Node\* temp=front;

            int data=front->data;

            cout << "Deleted element: " << data << endl;

            front=front->next;

            free(temp);

        }

    }

    void display(){

        Node\* temp=front;

        if(front==NULL){

            cout << "No nodes to display" << endl;

        }

        while(temp!=NULL){

            cout << temp->data << "->";

            temp=temp->next;

        }

        cout << "NULL" << endl;

    }

};

int main(){

    Queue q;

    int ch;

    int element;

    do{

        cout << "1.Enqueue 2.Dequeue 3.Display 4.Exit" << endl;

        cout << "Enter the choice: " << endl;

        cin >> ch;

        switch (ch)

        {

            case 1: cout << "Enter the element: " << endl;

                    cin >> element;

                    q.enqueue(element);

                    break;

            case 2: q.dequeue();

                    break;

            case 3: q.display();

                    break;

            case 4: exit(0);

        }

    }while(1);

}

STANDARD TEMPLATE LIBRARIES

STACK

//Stack is based on LIFO

#include<iostream>

#include<stack>

using namespace std;

int main(){

    stack<int> s;

    int ch;

    int element;

    do{

        cout << "\n1.Push 2.Pop 3.Top 4.Size" << endl;

        cout << "Enter the choice: ";

        cin >> ch;

        switch (ch)

        {

        case 1: cout << "Enter the elements into the stack: " << endl;

                cin >> element;

                s.push(element);

                break;

        case 2: s.pop();

                break;

        case 3: cout << endl << s.top();

                break;

        case 4: cout << endl << s.size();

        default:

            break;

        }

    }while(1);

}

LIST

#include<iostream>

#include<list>

using namespace std;

void showlist(list <int> g)

{

    list <int> :: iterator it;

    for(it = g.begin(); it != g.end(); it++)

        cout <<\*it << " ";

    cout <<'\n';

}

int main(){

    list<int> l;

    int ch;

    int element;

    do{

        cout << "1.Push Front 2.Push Back 3.Pop Front 4.Pop Back 5.Reverse 6.Sort 7.Display" << endl;

        cout << endl << "Enter the choice: ";

        cin >> ch;

        switch(ch){

            case 1: cout << "Enter the elements into the list: ";

                    cin >> element;

                    l.push\_front(element);

                    break;

            case 2: cout << "Enter the element to the list: ";

                    cin >> element;

                    l.push\_back(element);

                    break;

            case 3: l.pop\_front();

                    break;

            case 4: l.pop\_back();

                    break;

            case 5: l.reverse();

                    break;

            case 6: l.sort();

                    break;

            case 7: showlist(l);

        }

    }while(1);

}

QUEUE

//Queue is based on FIFO

#include<iostream>

#include<queue>

using namespace std;

void showq(queue<int> gq)

{

    queue<int> g = gq;

    while (!g.empty()) {

        cout << g.front() << " ";

        g.pop();

    }

    cout <<'\n';

}

int main(){

    queue<int> q;

    int ch;

    int element;

    do{

        cout << "\n1.Push 2.Pop 3.Display" << endl;

        cout << "Enter the choice: ";

        cin >> ch;

        switch (ch)

        {

        case 1: cout << "Enter the elements into the stack: " << endl;

                cin >> element;

                q.push(element);

                break;

        case 2: q.pop();

                break;

        case 3: showq(q);

                break;

        }

    }while(1);

}