**Doubly Link-list**

#include<iostream>

#include<stdlib.h>

#include<new>

using namespace std;

class node{

public:

    int data;

    node \*next;

    node \*prev;

};

 int getdata(){

     int value;

     cout<<"enter the value : ";

     cin>>value;

     return value;

 }

 void insert\_atstart(node \*\*head){

    int value = getdata();   //gets the value

    node \*new\_node=new node(); //allocates the memory to new node

    if(\*head == NULL){

        new\_node->next = NULL;

        new\_node->prev = NULL;

        new\_node->data = value;

        (\*head)=new\_node;

    }

    else{

         new\_node->data = value;

        new\_node->next = (\*head);

        new\_node->prev = NULL;

        (\*head)->prev = new\_node;

        (\*head)=new\_node;

    }

}

void insert\_atend(node \*\*head){

   int value = getdata();

   node \*new\_node = new node();         //allocate memory to new node

   node \*last = \*head;                  // stores the address reference of head

   new\_node->data = value;

   new\_node->next = NULL;

   if(\*head == NULL){

    new\_node->prev = NULL;

    \*head = new\_node;

     return;

   }

   while(last->next != NULL)            //traverse to last node

        last = last->next;

   last->next = new\_node;     // change the next of last node to recently created node

   new\_node->prev = last;      //set last to prev of new node

   return;

}

void insert\_afterval(node \*\*head){

            int value = getdata();

            int uservalue;

            cout<<"enter the aftervalue :";

            cin>>uservalue;

            node \*new\_node = new node();

            node \*curr = NULL;

            node \*temp = NULL;

            curr = \*head;

            while(curr){

                if(curr->data == uservalue){

                    break;

                }

                curr = curr->next;

            }

            new\_node->data = value;

            temp = curr->next;

            curr->next = new\_node;

            new\_node->prev = curr;

            new\_node->next = temp;

}

void delete\_atstart(node \*\*head)

{

    node \*temp;

    if((\*head) == NULL)

    {

        cout<<"UNDERFLOW";

    }

    else if((\*head)->next == NULL)

    {

        (\*head) = NULL;

        free(\*head);

        cout<<"\n Node Deleted \n";

    }

    else

    {

        temp = \*head;

        \*head = (\*head) -> next;

        (\*head) -> prev = NULL;

        free(temp);

        cout<<" \n  Node Deleted\n";

    }

}

void delete\_atend(node \*\*head)

{

    node \*temp = \*head;

    if((\*head) == NULL)

    {

        cout<<"UNDERFLOW";

    }

    else if(temp->next == NULL)

    {

        (\*head) = NULL;

        temp = temp->next;

        cout<<"\n Node Deleted \n";

    }

    else

    {

        while(temp->next != NULL)

        {

            temp = temp -> next;

        }

        temp -> prev -> next = NULL;

        temp = temp->next;

        cout<<"\nNode Deleted\n";

    }

}

void delete\_value(node \*\*head)

{

    node \*temp;

    int value;

    cout<<"Enter the value to be deleted : ";

    cin>>value;

    temp = \*head;

    if(temp->data == value && temp->next == NULL){

        \*head = NULL;

        free(temp);

        cout<<"list is empty";

    }

    else if(temp->data == value && temp->next != NULL){

        temp->next->prev = NULL;

        temp = temp->next;

    }

    else{

        while(temp->data != value && temp->next != NULL)

            temp = temp->next;

        if(temp == NULL){

            cout<<"value is not found";

        }

        else if(temp->next == NULL){

            temp->prev->next = NULL;

            temp = temp->next;

        }

        else{

            temp->prev->next = temp->next;

            temp->next->prev = temp->prev;

            temp = temp->next;

        }

    }

}

void display(node \*head)

{

    int count\_no=0;

    while(head != NULL){

        cout<<head->data<<" ";

        head=head->next;

        count\_no++;

    }

    cout<<" \n no of nodes in the linked list are : " << count\_no;

}

 int main()

{

    node \*head = NULL;

    int choice;

    cout<<" 1 for insert at beginning \n 2 for insert at end \n 3 for insert after the given value \n 4 for delete from beginning";

    cout<<"\n 5 delete from end \n 6 delete the given value\n 7 display \n";

    cout<<"enter your choice : ";

    cin>>choice;

    while(choice!=0){

      if(choice == 1){

        insert\_atstart(&head);

      }

      else if(choice == 2){

        insert\_atend(&head);

      }

      else if(choice == 3){

        insert\_afterval(&head);

      }

      else if(choice == 4){

        delete\_atstart(&head);

      }

      else if(choice == 5){

        delete\_atend(&head);

      }

      else if(choice == 6){

        delete\_value(&head);

      }

      else if(choice == 7){

        display(head);

      }

      else{

        cout<<"incorrect choice";

        cout<<"enter your choice : ";

        cin>>choice;

      }

      cout<<"\n enter your choice : ";

      cin>>choice;

    }

    return 0;

}

**Output :**

**1 for insert at beginning**

**2 for insert at end**

**3 for insert after the given value**

**4 for delete from beginning**

**5 delete from end**

**6 delete the given value**

**7 display**

**enter your choice : 1**

**enter the value : 5**

**enter your choice : 2**

**enter the value : 4**

**enter your choice : 7**

**5 4**

**no of nodes in the linked list are : 2**

**enter your choice : 1**

**enter the value : 3**

**enter your choice : 7**

**3 5 4**

**no of nodes in the linked list are : 3**

**enter your choice : 6**

**Enter the value to be deleted : 5**

**enter your choice : 7**

**3 4**

**no of nodes in the linked list are : 2**

**enter your choice : 4**

**Node Deleted**

**enter your choice : 7**

**4**

**no of nodes in the linked list are : 1**

**enter your choice : 5**

**Node Deleted**

**enter your choice : 7**

**no of nodes in the linked list are : 0**

**enter your choice :**

**Doubly Circular Linklist**

#include<iostream>

#include<stdlib.h>

#include<new>

using namespace std;

class node{

public:

    int data;

    node \*next;

    node \*prev;

};

int getdata(){

     int value;

     cout<<"enter the value : ";

     cin>>value;

     return value;

 }

void insert\_atstart(node \*\*head){

       int value = getdata();

        node \*new\_node = new node();

        new\_node->data = value;

        if(\*head == NULL){

        new\_node->next = new\_node;

        new\_node->prev = new\_node;

        (\*head)=new\_node;

        }

        else{

       node \*last = (\*head)->prev;

       new\_node->data = value;

       new\_node->next = (\*head);

       new\_node->prev = last;

       last->next = (\*head)->prev = new\_node;

       (\*head) = new\_node;

        }

}

void insert\_atend(node \*\*head)

{

    int value = getdata();

    node \*new\_node = new node();

    new\_node->data = value;

    if(\*head == NULL){

        new\_node->next = new\_node;

        new\_node->prev = new\_node;

        (\*head)=new\_node;

    }

    else{

         node \*last = (\*head)->prev;

         new\_node->next = (\*head);

         (\*head)->prev = new\_node;

         new\_node->prev = last;

         last->next = new\_node;

    }

}

void insert\_afterval(node \*\*head)

{

            int value = getdata();

            int uservalue;

            cout<<"enter the aftervalue :";

            cin>>uservalue;

            node \*new\_node = new node();

            new\_node->data = value;

            node \*temp = (\*head);

            while (temp->data != uservalue)

                 temp = temp->next;

            node \*next\_node = temp->next;

            temp->next = new\_node;

            new\_node->prev = temp;

            new\_node->next = next\_node;

            next\_node->prev = new\_node;

}

void delete\_atstart(node \*\*head)

{

    node \*temp;

    if((\*head) == NULL)

    {

        cout<<"UNDERFLOW";

    }

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

    {

        (\*head) = NULL;

        free(\*head);

        cout<<"\n Node Deleted \n";

    }

    else

    {

        temp = \*head;

        while(temp->next != (\*head))

           temp = temp->next;

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

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

        free(head);

        (\*head) = temp -> next;

        cout<<"\nNode Deleted\n";

    }

}

void delete\_atend(node \*\*head)

{

    node \*temp = \*head;

    if((\*head) == NULL)

    {

        cout<<"UNDERFLOW";

    }

    else if(temp->next == (\*head))

    {

        (\*head) = NULL;

        temp = temp->next;

        cout<<"\n Node Deleted \n";

    }

    else

    {

        while(temp->next != (\*head))

        {

            temp = temp -> next;

        }

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

        (\*head)->prev = temp->prev ;

         free(temp);

        cout<<"\nNode Deleted\n";

    }

}

void delete\_value(node \*\*head)

{

    node \*temp;

    int value;

    cout<<"Enter the value to be deleted : ";

    cin>>value;

    temp = \*head;

    if(temp->data == value && temp->next == NULL){

        \*head = NULL;

        free(temp);

        cout<<"list is empty";

    }

    else if(temp->data == value && temp->next != NULL){

        temp->next->prev = temp->next;

        temp->next->next = temp->next;

        (\*head) = temp->next;

        free(temp);

    }

    else{

        while(temp->data != value && temp->next != (\*head))

            temp = temp->next;

        if(temp == NULL){

            cout<<"value is not found";

        }

        else if(temp->next == NULL){

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

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

            free(temp);

        }

        else{

            temp->prev->next = temp->next;

            temp->next->prev = temp->prev;

            temp = temp->next;

        }

    }

}

void display(node\* head)

{

     node \*temp = head;

    while (temp->next != head)

    {

        cout<< temp->data<<" ";

        temp = temp->next;

    }

    cout<<temp->data;

}

int main()

{

    node \*head = NULL;

    int choice;

    cout<<" 1 insert at beginning \n 2 insert at end \n 3 insert after the given value \n 4 delete from beginning";

    cout<<"\n 5 delete from end \n 6 delete the given value \n 7 display \n";

    cout<<"enter your choice : ";

    cin>>choice;

    while(choice!=0){

      if(choice == 1){

        insert\_atstart(&head);

      }

      else if(choice == 2){

        insert\_atend(&head);

      }

      else if(choice == 3){

        insert\_afterval(&head);

      }

      else if(choice == 4){

        delete\_atstart(&head);

      }

      else if(choice == 5){

        delete\_atend(&head);

      }

      else if(choice == 6){

        delete\_value(&head);

      }

      else if(choice == 7){

        display(head);

      }

      else{

        cout<<"incorrect choice";

        cout<<"enter your choice : ";

        cin>>choice;

      }

      cout<<"\n enter your choice : ";

      cin>>choice;

    }

    return 0;

}

Output :

**1 for insert at beginning**

**2 for insert at end**

**3 for insert after the given value**

**4 for delete from beginning**

**5 delete from end**

**6 delete the given value**

**7 display**

**enter your choice : 1**

**enter the value : 3**

**enter your choice : 2**

**enter the value : 9**

**enter your choice : 7**

**3 9**

**no of nodes in the linked list are : 2**

**enter your choice : 1**

**enter the value : 5**

**enter your choice : 7**

**5 3 9**

**no of nodes in the linked list are : 3**

**enter your choice : 6**

**Enter the value to be deleted : 5**

**enter your choice : 7**

**3 9**

**no of nodes in the linked list are : 2**

**enter your choice : 4**

**Node Deleted**

**enter your choice : 7**

**9**

**no of nodes in the linked list are : 1**

**enter your choice : 5**

**Node Deleted**

**enter your choice : 7**

**no of nodes in the linked list are : 0**

**enter your choice :**

**Perform Bubble, selection, insertion sort**

#include<iostream>

using namespace std;

class Sorting{

    public:

    int list[10], i;

    void getData() {

        i = 0;

        while(i < 10){

            cout << "Enter The " << i << " index element : ";

            cin >> list[i];

            i++;

        }

    }

    void print() {

        i = 0;

        while(i < 10){

            cout << "The Element At index" << i << " : " << list[i] << endl;

            i++;

        }

    }

    void bubblesort() {

         for(i = 0; i < 10 - 1; i++) {

             for(int j = 0; j < 10 - i - 1; j++) {

                 if(list[j] > list[j+1]) {

                     list[j] += list[j+1];

                     list[j+1] = list[j] - list[j+1];

                     list[j] -= list[j+1];

                 }

             }

         }

         print();

    }

    void selectionsort() {

        int lowest\_index;

         for(i = 0; i < 10 - 1; i++) {

            lowest\_index = i;

            for(int j = i + 1; j < 10; j++) {

                if(list[j] < list[lowest\_index]) {

                    lowest\_index = j;

                }

            }

            list[i] += list[lowest\_index];

            list[lowest\_index] = list[i] - list[lowest\_index];

            list[i] -= list[lowest\_index];

         }

         print();

    }

    void insertionsort() {

        int found\_low;

         for(i = 1; i < 10 - 1; i++) {

             if( list[i-1] > list[i] ){

                 int j = i - 1;

                 found\_low = list[i];

                 while(j >= 0 && list[j] > list[i]){

                     list[j + 1] = list[j];

                     j--;

                 }

                 list[j + 1] = found\_low;

             }

         }

         print();

    }

};

int main(){

    Sorting s;

    s.getData();

    int choice = 0;

    while(1){

        cout << "1. Perform Bubble Sort." << endl << "2. Perform Selection Sort." << endl << "3. Perform Insertion Sort." << endl << "4. Re-insert Data into Array."<< endl << "5. Exit." << endl;

        cin >> choice;

        switch(choice){

            case 1:

                s.bubblesort();

                break;

            case 2:

                s.selectionsort();

                break;

            case 3:

                s.insertionsort();

                break;

            case 4:

                s.getData();

                break;

            case 5:

                exit(0);

            default:

                cout << "Invalid Choice" << endl;

        }

    }

}

**Ouput :**

PS E:\MCA\MCA SEM 3\DS> .\bubblesort.exe

Enter The 0 index element : 45

Enter The 1 index element : 12

Enter The 2 index element : 34

Enter The 3 index element : 87

Enter The 4 index element : 3

Enter The 5 index element : 6

Enter The 6 index element : 9

Enter The 7 index element : 10

Enter The 8 index element : 17

Enter The 9 index element : 23

1. Perform Bubble Sort.

2. Perform Selection Sort.

3. Perform Insertion Sort.

4. Re-insert Data into Array.

5. Exit.

1

The Element At index0 : 3

The Element At index1 : 6

The Element At index2 : 9

The Element At index3 : 10

The Element At index4 : 12

The Element At index5 : 17

The Element At index6 : 23

The Element At index7 : 34

The Element At index8 : 45

The Element At index9 : 87

1. Perform Bubble Sort.

2. Perform Selection Sort.

3. Perform Insertion Sort.

4. Re-insert Data into Array.

5. Exit.

5