**Name-Vanshikaa**

**CO21**

**102103580**

**Lab 2 :**

Q1 .

#include <iostream>

using namespace std;

struct Linkedlist

{

    int data;

    struct Linkedlist \* next;

};

typedef struct Linkedlist node;

node \* head = NULL, \* temp;

void insertEnd()

{

    temp->next = new node;

    cout << "Enter data: " << endl;

    cin >> temp->next->data;

    temp->next->next = NULL;

    temp = temp->next;

}

void display() {

   struct Linkedlist\* ptr;

   ptr = head;

   while (ptr != NULL) {

      cout<< ptr->data <<" ";

      ptr = ptr->next;

   }

}

void create()

{

    if (head == NULL)

    {

        head  = new node;

        cout << "Enter data: " << endl;

        cin >> head->data;

        head->next = NULL;

        temp = head;

    }

    else

    {

        insertEnd();

    }

}

void insertBeginning()

{

    node \* beg = new node;

    cout << "Enter data: " << endl;

    cin >> beg->data;

    beg->next = head;

    head = beg;

}

void insertMiddle()

{

    int num,ref;

    cout << "Enter data: " << endl;

    cin >> num;

    cout << "Enter the number after which input must be added: " << endl;

    cin >> ref;

    node \* loopPointer = head;

    while( loopPointer->data != ref)

    {

        loopPointer = loopPointer->next;

    }

    node \* temporaryNode = new node;

    temporaryNode->data = num;

    temporaryNode->next = loopPointer->next;

    loopPointer->next = temporaryNode;

}

void deleteEnd()

{

    node \* temporaryPointer = head;

    if (head->next == NULL)

        delete head;

    else{

    while(temporaryPointer->next->next != NULL)

    {

        temporaryPointer = temporaryPointer->next;

    }

        delete temporaryPointer->next;

        temp = temporaryPointer;

        temp->next = NULL;

    }

}

void deleteBeginning()

{

     node \* tempPointer = head;

     head = head->next;

     delete tempPointer;

}

void deleteMiddle()

{

    int num;

    cout << "Enter data to delete: "<< endl;

    cin >> num;

    node \* temporaryPointer = head;

    node \* temporaryPointer2 = NULL;

    while(temporaryPointer->next->data != num)

    {

        temporaryPointer = temporaryPointer->next;

    }

    temporaryPointer2 = temporaryPointer->next;

    temporaryPointer->next = temporaryPointer2->next;

    delete temporaryPointer2;

}

void Search()

{

    int num, count = 1;

    cout << "Enter the data to search for: " << endl;

    cin >> num;

    node \* temporaryPointer = head;

    while(temporaryPointer!= NULL && temporaryPointer->data != num)

    {

        count++;

        temporaryPointer = temporaryPointer->next;

    }

    if (temporaryPointer!= NULL)

        cout << "The required data was found at position " << count << endl;

    else

        cout << "Node not found!!" << endl;

    delete temporaryPointer;

}

int main()

{

    int menu;

    cout << "1. Create node" << endl;

    cout << "2. Display items" << endl;

    cout << "3. Add node at beginning" << endl;

    cout << "4. Add node in the middle" << endl;

    cout << "5. Delete last node" << endl;

    cout << "6. Delete first node" << endl;

    cout << "7. Delete middle node" << endl;

    cout << "8. Search" << endl;

    cout << "9.Exit" << endl;

    do{

    cout << "Enter menu item: " << endl;

    cin >> menu;

    switch(menu)

    {

        case 1:

            create();

            break;

        case 2:

            display();

            break;

        case 3:

            insertBeginning();

            break;

        case 4:

            insertMiddle();

            break;

        case 5:

            deleteEnd();

            break;

        case 6:

            deleteBeginning();

            break;

        case 7:

            deleteMiddle();

            break;

        case 8:

            Search();

            break;

        default:

            break;

    }} while(menu != 9);

    return 0;

}

Output :

1. Create node

2. Display items

3. Add node at beginning

4. Add node in the middle

5. Delete last node

6. Delete first node

7. Delete middle node

8. Search

15.Exit

Enter menu item:

1

Enter data:

2

Enter menu item:

2

2 Enter menu item:

3

Enter data:

20

Enter menu item:

2

20 2 Enter menu item:

4

Enter data:

30

Enter the number after which input must be added:

2

Enter menu item:

2

20 2 30 Enter menu item:

5

Enter menu item:

2

20 2 Enter menu item:

6

Enter menu item:

2

2 Enter menu item:

8

Enter the data to search for:

2

The required data was found at position 1

Enter menu item:

9

Q2 .

#include <iostream>

using namespace std;

class Node {

public:

    int data;

    Node\* next;

};

Node\* insert(Node\* head, int new\_data)

{

    Node\* new\_node = new Node();

    new\_node->data = new\_data;

    new\_node->next = head;

    head = new\_node;

    return head;

}

Node\* deletes(Node\* head, int x)

{

    if (!head)

        return head;

    while (head && head->data == x)

        head = head->next;

    Node \*curr = head, \*prev = nullptr;

    while (curr) {

        if (curr->data == x)

            prev->next = curr->next;

        else

            prev = curr;

        curr = curr->next;

    }

    return head;

}

void display(Node\* node)

{

    while (node) {

        cout << node->data << " ";

        node = node->next;

    }

}

int main()

{

    Node\* head = NULL;

    int n;

    cout<<"Enter number of elements : "<<endl;

    cin>>n;

    for(int i=0;i<n;i++)

    {

        int temp;

        cin>>temp;

        head=insert(head,temp);

    }

    int key;

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

    cin>>key;

    cout << "Linked List:\n ";

    display(head);

    head = deletes(head, key);

    if (!head)

        cout << "No element present\n"

            << endl;

    else {

        cout << "\nAfter Deletion: \n";

        display(head);

    }

}

Output :

Enter number of elements :

7

1

2

1

2

1

3

1

Enter the element to be deleted: 1

Linked List:

3 1 2 1 2 1

**Lab 3 :**

Q1 . Circular Linked List

// Cicular Linked List

#include<iostream>

using namespace std;

struct Linkedlist

{

    int data;

    struct Linkedlist \* next;

};

typedef struct Linkedlist node;

node \* head = NULL, \* temp;

void insertEnd()

{

    node \* temp = new node;

    cout << "Enter data: " << endl;

    cin >> temp->data;

    temp->next = head->next;

    head->next = temp;

    temp = head;

}

void create()

{

    if (head == NULL)

    {

        head  = new node;

        cout << "Enter data: " << endl;

        cin >> head->data;

        head->next = head;

    }

    else

    {

        insertEnd();

    }

}

void display()

{

    node \* disp = head;

    do {

            cout << disp->data << endl;

            disp = disp->next;

        } while(disp != head);

}

void insertBeginning()

{

    node \* beg = new node;

    cout << "Enter data: " << endl;

    cin >> beg->data;

    beg->next = head;

    node \* temp = head;

    while(temp->next != head)

    {

        temp = temp->next;

    }

    temp->next = beg;

    head = beg;

}

void insertMiddle()

{

    int num,ref;

    cout << "Enter data: " << endl;

    cin >> num;

    cout << "Enter the number after which input must be added: " << endl;

    cin >> ref;

    node \* loopPointer = head;

    while( loopPointer->data != ref)

    {

        loopPointer = loopPointer->next;

    }

    node \* temporaryNode = new node;

    temporaryNode->data = num;

    temporaryNode->next = loopPointer->next;

    loopPointer->next = temporaryNode;

}

void deleteEnd()

{

    node \* temp = head;

    if (head->next == head)

        {

            delete head;

            head = NULL;

        }

    else{

        node \* ptr = head->next;

        while(ptr->next!=head)

        {

            ptr = ptr->next;

        }

        ptr->next = head->next;

        delete head;

        head = ptr;

    }

}

void deleteBeginning()

{

     node \* tempPointer = head->next;

     head->next = head->next->next;

     delete tempPointer;

}

void deleteMiddle()

{

    int num;

    cout << "Enter data to delete: "<< endl;

    cin >> num;

    node \* temporaryPointer = head->next;

    node \* temporaryPointer2 = NULL;

    while(temporaryPointer->next->data != num)

    {

        temporaryPointer = temporaryPointer->next;

    }

    temporaryPointer2 = temporaryPointer->next;

    temporaryPointer->next = temporaryPointer2->next;

    delete temporaryPointer2;

}

void Search()

{

    int num, count = 1;

    cout << "Enter the data to search for: " << endl;

    cin >> num;

    node \* temporaryPointer = head->next;

    if (head->data == num)

    {

        cout << "The required element was found at the end of the list" << endl;

    }

    else{

    while(temporaryPointer!= head && temporaryPointer->data != num)

    {

        count++;

        temporaryPointer = temporaryPointer->next;

    }

    if (temporaryPointer!= head)

        cout << "The required data was found at position " << count << endl;

    else

        cout << "Node not found!!" << endl;

    }

}

int main()

{

    int menu;

    cout << "1. Create node" << endl;

    cout << "2. Display items" << endl;

    cout << "3. Add node at beginning" << endl;

    cout << "4. Add node in the middle" << endl;

    cout << "5. Delete last node" << endl;

    cout << "6. Delete first node" << endl;

    cout << "7. Delete middle node" << endl;

    cout << "8. Search" << endl;

    cout << "9.Exit" << endl;

    do{

    cout << "Enter menu item: " << endl;

    cin >> menu;

    switch(menu)

    {

        case 1:

            create();

            break;

        case 2:

            display();

            break;

        case 3:

            insertBeginning();

            break;

        case 4:

            insertMiddle();

            break;

        case 5:

            deleteEnd();

            break;

        case 6:

            deleteBeginning();

            break;

        case 7:

            deleteMiddle();

            break;

        case 8:

            Search();

            break;

        default:

            break;

    }} while(menu != 9);

    return 0;

}

Output :

1. Create node

2. Display items

3. Add node at beginning

4. Add node in the middle

5. Delete last node

6. Delete first node

7. Delete middle node

8. Search

9.Exit

Enter menu item:

1

Enter data:

10

Enter menu item:

2

10

Enter menu item:

3

Enter data:

20

Enter menu item:

2

20

10

Enter menu item:

3

Enter data:

30

Enter menu item:

2

30

20

10

Enter menu item:

4

Enter data:

40

Enter the number after which input must be added:

30

Enter menu item:

2

30

40

20

10

Enter menu item:

5

Enter menu item:

2

30

40

20

Enter menu item:

Enter the data to search for:

40

The required data was found at position 2

Double Linked list

#include <stdio.h>

#include <stdlib.h>

struct node {

    int info;

    struct node \*prev, \*next;

};

struct node\* start = NULL;

void traverse()

{

    if (start == NULL) {

        printf("\nList is empty\n");

        return;

    }

    struct node\* temp;

    temp = start;

    while (temp != NULL) {

        printf("Data = %d\n", temp->info);

        temp = temp->next;

    }

}

void insertAtFront()

{

    int data;

    struct node\* temp;

    temp = (struct node\*)malloc(sizeof(struct node));

    printf("\nEnter number to be inserted: ");

    scanf("%d", &data);

    temp->info = data;

    temp->prev = NULL;

    temp->next = start;

    start = temp;

}

void insertAtEnd()

{

    int data;

    struct node \*temp, \*trav;

    temp = (struct node\*)malloc(sizeof(struct node));

    temp->prev = NULL;

    temp->next = NULL;

    printf("\nEnter number to be inserted: ");

    scanf("%d", &data);

    temp->info = data;

    temp->next = NULL;

    trav = start;

    if (start == NULL) {

        start = temp;

    }

    else {

        while (trav->next != NULL)

            trav = trav->next;

        temp->prev = trav;

        trav->next = temp;

    }

}

void insertAtPosition()

{

    int data, pos, i = 1;

    struct node \*temp, \*newnode;

    newnode = malloc(sizeof(struct node));

    newnode->next = NULL;

    newnode->prev = NULL;

    printf("\nEnter position : ");

    scanf("%d", &pos);

    if (start == NULL) {

        start = newnode;

        newnode->prev = NULL;

        newnode->next = NULL;

    }

    else if (pos == 1) {

      insertAtFront();

    }

    else {

      printf("\nEnter number to be inserted: ");

    scanf("%d", &data);

    newnode->info = data;

    temp = start;

        while (i < pos - 1) {

            temp = temp->next;

            i++;

        }

        newnode->next = temp->next;

        newnode->prev = temp;

        temp->next = newnode;

        temp->next->prev = newnode;

    }

}

void deleteFirst()

{

    struct node\* temp;

    if (start == NULL)

        printf("\nList is empty\n");

    else {

        temp = start;

        start = start->next;

        if (start != NULL)

            start->prev = NULL;

        free(temp);

    }

}

void deleteEnd()

{

    struct node\* temp;

    if (start == NULL)

        printf("\nList is empty\n");

    temp = start;

    while (temp->next != NULL)

        temp = temp->next;

    if (start->next == NULL)

        start = NULL;

    else {

        temp->prev->next = NULL;

        free(temp);

    }

}

void deletePosition()

{

    int pos, i = 1;

    struct node \*temp, \*position;

    temp = start;

    if (start == NULL)

        printf("\nList is empty\n");

    else {

        printf("\nEnter position : ");

        scanf("%d", &pos);

        if (pos == 1) {

            deleteFirst();

            if (start != NULL) {

                start->prev = NULL;

            }

            free(position);

            return;

        }

        while (i < pos - 1) {

            temp = temp->next;

            i++;

        }

        position = temp->next;

        if (position->next != NULL)

            position->next->prev = temp;

        temp->next = position->next;

        free(position);

    }

}

int main()

{

    int choice;

    while (1) {

        printf("\n\t1  To see list\n");

        printf("\t2  For insertion at"

               " starting\n");

        printf("\t3  For insertion at"

               " end\n");

        printf("\t4  For insertion at "

               "any position\n");

        printf("\t5  For deletion of "

               "first element\n");

        printf("\t6  For deletion of "

               "last element\n");

        printf("\t7  For deletion of "

               "element at any position\n");

        printf("\t8 To exit\n");

        printf("\nEnter Choice :\n");

        scanf("%d", &choice);

        switch (choice) {

        case 1:

            traverse();

            break;

        case 2:

            insertAtFront();

            break;

        case 3:

            insertAtEnd();

            break;

        case 4:

            insertAtPosition();

            break;

        case 5:

            deleteFirst();

            break;

        case 6:

            deleteEnd();

            break;

        case 7:

            deletePosition();

            break;

        case 8:

            exit(1);

            break;

        default:

            printf("Incorrect Choice. Try Again \n");

            continue;

        }

    }

    return 0;

}

Output :

1 To see list

2 For insertion at starting

3 For insertion at end

4 For insertion at any position

5 For deletion of first element

6 For deletion of last element

7 For deletion of element at any position

8 To exit

Enter Choice :

2

Enter number to be inserted: 1

1 To see list

2 For insertion at starting

3 For insertion at end

4 For insertion at any position

5 For deletion of first element

6 For deletion of last element

7 For deletion of element at any position

8 To exit

Enter Choice :

3

Enter number to be inserted: 3

1 To see list

2 For insertion at starting

3 For insertion at end

4 For insertion at any position

5 For deletion of first element

6 For deletion of last element

7 For deletion of element at any position

8 To exit

Enter Choice :

8

Q2 .

#include <iostream>

using namespace std;

struct cll{

    int data;

    struct cll \* next;

};

typedef struct cll node;

node \* last = NULL;

void create()

{

    if (last == NULL)

    {

        last = new node;

        last->next = last;

        cin >> last->data;

    }

    else

    {

        node \* temp = new node;

        temp->next = last->next;

        last->next = temp;

        last = temp;

        cin >> last->data;

    }

}

void tempDisplay()

{

    node \* temp = last->next;

    cout<<"The output is: \n";

    do

    {

        cout << temp->data << endl;

        if (temp == last)

            cout << last->next->data;

        temp = temp->next;

    } while(temp != last->next);

}

int main()

{

    int n;

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

    cin >> n;

    for (int i = 0; i < n; i++)

    {

        create();

    }

    tempDisplay();

    return 0;

}

Output :

Enter the size:

5

20

100

40

80

60

The output is:

20

100

40

80

60

20