/\*Write a C++ program to create a single linked list and display them

note

Creation

Create a list with required number of nodes by accessing data elements,

i.e Read the data & create the linked list with required number of nodes until zero is pressed

Display

1. If no nodes are present then print "List is empty".

2. Otherwise display all elements in list.

sample

input & output:

1. Create

2. Display

3. Exit

Enter your choice:1

Enter data to create nodes & press '0' to exit: 10

20

30

40

50

0

1. Create

2. Display

3. Exit

Enter your choice:2

10 20 30 40 50

1. Create

2. Display

3. Exit

Enter your choice:3

Exiting

\*/

#include<iostream>

#include<cstdlib>

using namespace std;

class Node {

public:

int data;

Node\* next;

public:

Node() {

data = 0;

next = NULL;

}

Node(int x) {

data = x;

next = NULL;

}

};

class LinkedList

{

public:

Node \*head;

int count;

LinkedList()

{

head = NULL;

count=0;

}

void create(int num);

void display();

};

void LinkedList ::create(int num)

{

Node \*temp,\*nn;

if(head==NULL)

{

head=new Node(num);

count++;

}

else

{

for(temp=head;temp->next!=NULL;temp=temp->next);

nn=new Node(num);

temp->next=nn;

count++;

}

}

void LinkedList::display()

{

Node \*temp;

if(head==NULL)

{

cout<<"List is empty"<<endl;

}

else

{

for(temp=head;temp!=NULL;temp=temp->next)

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

cout<<endl;

}

}

int main()

{

LinkedList l;

Node \*temp;

int num,pos,ch=1;

while(ch!=3)

{

cout<<"\n1. Create"<<endl;

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

cout<<"3. Exit"<<endl;

cout<<"Enter your choice:";

cin>>ch;

switch(ch)

{

case 1:

cout<<"Enter data to create nodes & press '0' to exit: ";

do{

cin>>num;

if(num==0)

break;

l.create(num);

}while(1);

break;

case 2:

l.display();

break;

case 3:

cout<<"Exiting\n";

break;

default:

cout<<"Wrong Choice\n";

}

}

}

/\*Write a C++ program to create a single linked list and perform search operation

note:

Creation

Create a list with required number of nodes by accessing data elements,

i.e Read the data & create the linked list with required number of nodes until zero is pressed

For Searching

1. Read data, then if list is empty then print "List is empty"

2. Read data, if element is present then print "Element found 50"

3. Read data, if element is not present then print "Element not found 50"

Display

1. If no nodes are present then print "List is empty".

2. Otherwise display all elements in list.

sample input & output

1. Create

2. Search an element in list

3. Display

4. Exit

Enter your choice:2

Enter num: 15

List is empty

1. Create

2. Search an element in list

3. Display

4. Exit

Enter your choice:3

List is empty

1. Create

2. Search an element in list

3. Display

4. Exit

Enter your choice:1

Enter data to create nodes & press '0' to exit: 10

20

30

0

1. Create

2. Search an element in list

3. Display

4. Exit

Enter your choice:3

10 20 30

1. Create

2. Search an element in list

3. Display

4. Exit

Enter your choice:2

Enter num: 20

Element found 20

1. Create

2. Search an element in list

3. Display

4. Exit

Enter your choice:2

Enter num: 40

Element not found 40

1. Create

2. Search an element in list

3. Display

4. Exit

Enter your choice:4

Exiting

\*/

#include<iostream>

#include<cstdlib>

using namespace std;

class Node {

public:

int data;

Node\* next;

public:

Node() {

data = 0;

next = NULL;

}

Node(int x) {

data = x;

next = NULL;

}

};

class LinkedList {

public:

Node \*head;

int count;

LinkedList() {

head = NULL;

count=0;

}

void create(int num);

void display();

void search(int k);

};

void LinkedList ::create(int num)

{

Node \*temp,\*nn;

if(head==NULL)

{

head=new Node(num);

count++;

}

else

{

for(temp=head;temp->next!=NULL;temp=temp->next);

nn=new Node(num);

temp->next=nn;

count++;

}

}

void LinkedList::display()

{

if(head==NULL)

{

cout<<"List is empty"<<endl;

}

else

{

Node \*ptr=head;

while(ptr!=NULL)

{

cout<<ptr->data<<" ";

ptr=ptr->next;

}

cout<<endl;

}

}

void LinkedList :: search(int x)

{

Node\* ptr = head;

if(ptr==NULL)

cout<<"List is empty"<<endl;

else

{

while(ptr->data != x && ptr->next!=NULL )

{

ptr = ptr->next;

}

if(ptr->data==x)

cout<<"Element found "<<x<<endl;

else

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

}

}

int main()

{

LinkedList l;

Node \*temp;

int num,pos,ch=1;

while(ch!=4)

{

cout<<"\n1. Create"<<endl;

cout<<"2. Search an element in list"<<endl;

cout<<"3. Display"<<endl;

cout<<"4. Exit"<<endl;

cout<<"Enter your choice:";

cin>>ch;

switch(ch)

{

case 1:

cout<<"Enter data to create nodes & press '0' to exit: ";

do{

cin>>num;

if(num==0)

break;

l.create(num);

}while(1);

break;

case 2:

cout<<"Enter num: ";

cin>>num;

l.search(num);

break;

case 3:

l.display();

break;

case 4:

cout<<"Exiting\n";

break;

default:

cout<<"Wrong Choice\n";

}

}

}

/\*Write a c++ program to implement insertion operation into single linked list & display them

note:

1. Insert at beginning position

2. insert at ending position

3. For insertion read position and data

a. if position is 1 & when list is empty - insert element at position 1

and print statement "Creating the list with head node"

b. or if position value is within number of elements in list then write code to insert element after required position in the list

c. if position value exceeds number of elements in list or position<=0 then print "Position not found"

4. Display

If no nodes are present then print "List is empty".

Otherwise display all elements in list.

sample

input & output:

1. Insert at beginning position

2. Insert at ending position

3. Insert after required position

4. Display

5. Exit

Enter your choice:3

Enter pos: 0

Enter data: 20

Position not found

1. Insert at beginning position

2. Insert at ending position

3. Insert after required position

4. Display

5. Exit

Enter your choice:3

Enter pos: 1

Enter data: 10

Creating the list with head Node

1. Insert at beginning position

2. Insert at ending position

3. Insert after required position

4. Display

5. Exit

Enter your choice:1

Enter data: 11

1. Insert at beginning position

2. Insert at ending position

3. Insert after required position

4. Display

5. Exit

Enter your choice:1

Enter data: 22

1. Insert at beginning position

2. Insert at ending position

3. Insert after required position

4. Display

5. Exit

Enter your choice:2

Enter data: 33

1. Insert at beginning position

2. Insert at ending position

3. Insert after required position

4. Display

5. Exit

Enter your choice:3

Enter pos: 4

Enter data: 44

1. Insert at beginning position

2. Insert at ending position

3. Insert after required position

4. Display

5. Exit

Enter your choice:4

22 11 10 33 44

1. Insert at beginning position

2. Insert at ending position

3. Insert after required position

4. Display

5. Exit

Enter your choice:5

Exiting

\*/

#include<iostream>

#include<cstdlib>

using namespace std;

class Node {

public:

int data;

Node\* next;

public:

Node() {

data = 0;

next = NULL;

}

Node(int x) {

data = x;

next = NULL;

}

};

class LinkedList

{

public:

Node \*head;

int count;

LinkedList()

{

head = NULL;

count=0;

}

void insertBeg(int num);

void insertEnd(int num);

void insertAfterPos(int num,int pos);

void display();

};

void LinkedList::insertBeg(int num)

{

Node \*nn,\*temp;

nn=new Node(num);

if(head==NULL)

{

head=nn;

count++;

}

else

{

nn->next=head;

head=nn;

count++;

}

}

void LinkedList::insertEnd(int num)

{

Node \*nn,\*temp;

nn=new Node(num);

if(head==NULL)

{

head=nn;

count++;

}

else

{

for(temp=head;temp->next!=NULL;temp=temp->next);

temp->next=nn;

count++;

}

}

void LinkedList :: insertAfterPos(int num,int pos)

{

Node \*nn,\*temp;

int i;

if(pos==1 && head==NULL)

{

head=new Node(num);

count=1 ;

cout<<"Creating the list with head Node"<<endl;

return;

}

else if(pos>count||pos<=0)

{

cout<<"Position not found"<<endl;

return;

}

else

{

for(i=1,temp=head;i<pos;temp=temp->next,i=i+1);

nn=new Node(num);

nn->next=temp->next;

temp->next=nn;

count++;

}

}

void LinkedList::display()

{

Node \*temp;

if(head==NULL)

{

cout<<"List is empty"<<endl;

}

else

{

for(temp=head;temp!=NULL;temp=temp->next)

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

cout<<endl;

}

}

int main()

{

LinkedList l;

Node \*temp;

int num,pos,ch=1;

while(ch!=5)

{

cout<<"1. Insert at beginning position"<<endl;

cout<<"2. Insert at ending position"<<endl;

cout<<"3. Insert after required position"<<endl;

cout<<"4. Display"<<endl;

cout<<"5. Exit"<<endl;

cout<<"Enter your choice:";

cin>>ch;

switch(ch)

{

case 1:cout<<"Enter data: ";

cin>>num;

l.insertBeg(num);

break;

case 2:cout<<"Enter data: ";

cin>>num;

l.insertEnd(num);

break;

case 3:

cout<<"Enter pos: ";

cin>>pos;

cout<<"Enter data: ";

cin>>num;

l.insertAfterPos(num,pos);

break;

case 4:

l.display();

break;

case 5:

cout<<"Exiting\n";

break;

default:

cout<<"Wrong Choice\n";

}

}

}

/\*Write a C++ program to create a single linked list and perform all deletion operations on it

note:

1. Creation

Create a list with required number of nodes by accessing data elements,

i.e Read the data & create the linked list with required number of nodes until zero

is pressed

2.Delete beginning node

if list is empty then print "List is empty"

or delete the beginning node & print eg: "10 is deleted"

3.Delete Ending node

if list is empty then print "List is empty"

or delete the ending node & print eg: "10 is deleted"

4.Deleting particular node

Read data, then if list is empty then print "List is empty"

Read data, if element is present then delete it & print eg: "10 is deleted"

Read data, if element is not present then print "10 not found"

5. Display

If no nodes are present then print "List is empty".

Otherwise display all elements in list.

sample

input & output

1. Create

2. Delete at beginning position

3. Delete at ending position

4. Delete a particular node

5. Display

6. Exit

Enter your choice:1

Enter data to create nodes & press '0' to exit: 10

20

30

40

50

0

1. Create

2. Delete at beginning position

3. Delete at ending position

4. Delete a particular node

5. Display

6. Exit

Enter your choice:2

10 is deleted

1. Create

2. Delete at beginning position

3. Delete at ending position

4. Delete a particular node

5. Display

6. Exit

Enter your choice:3

50 is deleted

1. Create

2. Delete at beginning position

3. Delete at ending position

4. Delete a particular node

5. Display

6. Exit

Enter your choice:5

20 30 40

1. Create

2. Delete at beginning position

3. Delete at ending position

4. Delete a particular node

5. Display

6. Exit

Enter your choice:4

Enter data: 30

30 is deleted

1. Create

2. Delete at beginning position

3. Delete at ending position

4. Delete a particular node

5. Display

6. Exit

Enter your choice:4

Enter data: 40

40 is deleted

1. Create

2. Delete at beginning position

3. Delete at ending position

4. Delete a particular node

5. Display

6. Exit

Enter your choice:2

20 is deleted

1. Create

2. Delete at beginning position

3. Delete at ending position

4. Delete a particular node

5. Display

6. Exit

Enter your choice:5

List is empty

1. Create

2. Delete at beginning position

3. Delete at ending position

4. Delete a particular node

5. Display

6. Exit

Enter your choice:6

Exiting

\*/

#include<iostream>

#include<cstdlib>

using namespace std;

class Node {

public:

int data;

Node\* next;

public:

Node() {

data = 0;

next = NULL;

}

Node(int x) {

data = x;

next = NULL;

}

};

class LinkedList {

public:

Node \*head;

int count;

LinkedList() {

head = NULL;

count=0;

}

void create(int num);

void deleteBeg();

void deleteEnd();

void deleteNode(int x);

void display();

};

void LinkedList ::create(int num)

{

Node \*temp,\*nn;

if(head==NULL)

{

head=new Node(num);

count++;

}

else

{

for(temp=head;temp->next!=NULL;temp=temp->next);

nn=new Node(num);

temp->next=nn;

count++;

}

}

void LinkedList::display()

{

Node \*temp;

if(head==NULL)

{

cout<<"List is empty"<<endl;

}

else

{

for(temp=head;temp!=NULL;temp=temp->next)

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

cout<<endl;

}

}

void LinkedList::deleteBeg()

{

if(head==NULL)

{

cout<<"List is empty"<<endl;

return;

}

if(head->next==NULL)

{

cout<<head->data<<" is deleted"<<endl;

head=NULL;

count--;

return;

}

cout<<head->data<<" is deleted"<<endl;

head=head->next;

count--;

}

void LinkedList::deleteEnd()

{

Node \*temp,\*ptr;

if(head==NULL)

{

cout<<"List is empty"<<endl;

return;

}

if(head->next==NULL)

{

cout<<head->data<<" is deleted"<<endl;

head=NULL;

count--;

return;

}

for(temp=head;temp->next->next!=NULL;temp=temp->next);

ptr=temp->next;

temp->next=NULL;

cout<<ptr->data<<" is deleted"<<endl;

delete ptr;

count--;

}

void LinkedList :: deleteNode(int x)

{

Node \*temp,\*ptr;

if(head==NULL)

{

cout<<"List is empty"<<endl;

return;

}

if(head->data==x)

{

temp=head;

head= head->next;

cout<<temp->data<<" is deleted"<<endl;

delete temp;

count--;//use this when u combine all operations together

return;

}

for(temp=head;temp->next!=NULL && temp->next->data!=x;temp=temp->next);

if(temp->next==NULL)

{

cout<<x<<" not found"<<endl;

return;

}

ptr=temp->next;

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

cout<<ptr->data<<" is deleted"<<endl;

delete ptr;

count--;

}

int main()

{

LinkedList l;

Node \*temp;

int num,pos,ch=1;

while(ch!=6)

{

cout<<"\n1. Create"<<endl;

cout<<"2. Delete at beginning position"<<endl;

cout<<"3. Delete at ending position"<<endl;

cout<<"4. Delete a particular node"<<endl;

cout<<"5. Display"<<endl;

cout<<"6. Exit"<<endl;

cout<<"Enter your choice:";

cin>>ch;

switch(ch)

{

case 1:

cout<<"Enter data to create nodes & press '0' to exit: ";

do{

cin>>num;

if(num==0)

break;

l.create(num);

}while(1);

break;

case 2:l.deleteBeg();

break;

case 3:l.deleteEnd();

break;

case 4:

cout<<"Enter data: ";

cin>>num;

l.deleteNode(num);

break;

case 5:

l.display();

break;

case 6:

cout<<"Exiting\n";

break;

default:

cout<<"Wrong Choice\n";

}

}

}

/\*Write a C++ program to perform insertion of node after required position,

deletion and searching of particular node in single linked list.

note:

For insertion read position and data

1. if position is 1 & when list is empty - insert element at position 1

and print statement "Creating the list with head node"

2. or if position value is within number of elements in list then write code to insert element after required position in the list

3. or if position value exceeds number of elements in list then print "Position not found"

For Deletion

1. Read data, then if list is empty then print "List is empty"

2. Read data, if element is present then delete it & print eg: "10 is deleted"

3. Read data, if element is not present then print "10 not found"

For Searching

1. Read data, then if list is empty then print "List is empty"

2. Read data, if element is present then print "Element found 50"

3. Read data, if element is not present then print "Element not found 50"

Using display, display all functions in list

\*/

#include<iostream>

#include<cstdlib>

using namespace std;

class node {

public:

int data;

node\* next;

public:

node() {

data = 0;

next = NULL;

}

node(int x) {

data = x;

next = NULL;

}

};

class LinkedList {

public:

node \*head;

int count;

LinkedList() {

head = NULL;

count=0;

}

void addAfterPos(int num,int pos);

void search(int k);

void deleteNode(int x);

void display();

};

void LinkedList :: addAfterPos(int num,int pos)

{

int tmp;

node \*n = new node(num);

node \*temp=head;

if(pos==1 && temp==NULL)

{

head=n;

count++;

cout<<"Creating the list with head node"<<endl;

}

else if(pos<=count && temp!=NULL)

{

tmp=1;

while(tmp!=pos)

{

temp=temp->next;

tmp++;

}

if(tmp==count)

temp->next=n;

else

{

n->next=temp->next;

temp->next=n;

}

count++;

}

else

cout<<"Position not found"<<endl;

}

void LinkedList :: deleteNode(int x)

{

node \*ptr = head,\*temp;

int flag=0;

if(ptr==NULL)

cout<<"List is empty"<<endl;

else if(ptr->data==x)

{

head= head->next;

cout<<x<<" is deleted"<<endl;

flag=1;

delete ptr;

}

else

{

while(ptr!=NULL)

{

if(ptr->data!=x)

{

temp=ptr;

ptr=ptr->next;

}

else

{

temp->next=ptr->next;

cout<<x<<" is deleted"<<endl;

flag=1;

break;

}

}

if(flag==0)

cout<<x<<" not found"<<endl;

}

}

void LinkedList::display()

{

if(head==NULL)

{

cout<<"List is empty"<<endl;

}

else

{

node \*ptr=head;

while(ptr!=NULL)

{

cout<<ptr->data<<" ";

ptr=ptr->next;

}

cout<<endl;

}

}

void LinkedList :: search(int x)

{

node\* ptr = head;

if(ptr==NULL)

cout<<"List is empty"<<endl;

else

{

while(ptr->data != x && ptr->next!=NULL )

{

ptr = ptr->next;

}

if(ptr->data==x)

cout<<"Element found "<<x<<endl;

else

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

}

}

int main()

{

LinkedList l;

node \*temp;

int num,pos,ch=1;

while(ch!=5)

{

cout<<"\n1. Insert After Node"<<endl;

cout<<"2. Delete a node"<<endl;

cout<<"3. Search an element in list"<<endl;

cout<<"4. Display"<<endl;

cout<<"5. Exit"<<endl;

cout<<"Enter your choice:";

cin>>ch;

switch(ch)

{

case 1:

cout<<"Enter pos: ";

cin>>pos;

cout<<"Enter data: ";

cin>>num;

l.addAfterPos(num,pos);

break;

case 2:

cout<<"Enter data: ";

cin>>num;

l.deleteNode(num);

break;

case 3:

cout<<"Enter num: ";

cin>>num;

l.search(num);

break;

case 4:

l.display();

break;

case 5:

cout<<"Exiting\n";

break;

default:

cout<<"Wrong Choice\n";

}

}

}

/\*Write a c++ program creation, insertion(at start position & ending position) and search operations

using single linked list

note:

1. create a class node,declare required variables to store element & address

use node(), to initialize default values of node

use node(int x), to create new node

2. write the logic for the functions inorder to perform required operations specified in LinkedList class

sample

input & output

1. Insert node at front.

2. Insert node at rear.

3. Display

4. Exit

Enter your choice: 1

Enter data: 20

1. Insert node at front.

2. Insert node at rear.

1. Insert node at front.

2. Insert node at rear.

3. Display

4. Exit

Enter your choice: 1

Enter data: 10

1. Insert node at front.

2. Insert node at rear.

1. Insert node at front.

2. Insert node at rear.

3. Display

4. Exit

Enter your choice: 2

Enter data: 30

1. Insert node at front.

2. Insert node at rear.

1. Insert node at front.

2. Insert node at rear.

3. Display

4. Exit

Enter your choice: 2

Enter data: 40

1. Insert node at front.

2. Insert node at rear.

1. Insert node at front.

2. Insert node at rear.

3. Display

4. Exit

Enter your choice: 4

10 20 30 40

1. Insert node at front.

2. Insert node at rear.

1. Insert node at front.

2. Insert node at rear.

3. Display

4. Exit

Enter your choice: 5

exiting

\*/

#include<iostream>

#include<cstdlib>

using namespace std;

class node {

public:

int data;

node\* next;

public:

node() {

data = 0;

next = NULL;

}

node(int x) {

data = x;

next = NULL;

}

};

class LinkedList {

public:

node\* head;

LinkedList() {

head = NULL;

}

void addFront(int);

void addRear(int);

//void addAfter(int num,int old);

// node\* search(int k);

void display();

};

void LinkedList::addFront(int num)

{

node \*n=new node(num);

if(head==NULL)

{

head=n;

}

else

{

node \*ptr=head;

n->next=ptr;

head=n;

}

}

void LinkedList::addRear(int num)

{

node \*n=new node(num);

if(head==NULL)

{

head=n;

}

else

{

node \*ptr=head;

while(ptr->next!=NULL)

{

ptr=ptr->next;

}

ptr->next=n;

}

}

/\*void LinkedList :: addAfter(int num,int old) {

node \*n;

n = new node(num);

node \*temp1=search(old);

node \*temp2=head;

if(temp1->next!=NULL)

{

temp2=temp1->next;

temp1->next=n;

n->next=temp2;

}

else {

temp1->next=n;

}

}

node\* LinkedList :: search(int x) {

node\* ptr = head;

while(ptr->data != x || ptr->next == NULL ) {

ptr = ptr->next;

}

return ptr;

}\*/

void LinkedList::display()

{

node \*temp=head;

if(temp==NULL)

cout<<"empty list"<<endl;

else

{

while(temp!=NULL)

{

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

temp=temp->next;

}

}

cout<<endl;

}

int main()

{

LinkedList l;

node \*temp;

int num,nd,ch=1,ele;

while(ch!=4)

{

cout<<"1. Insert node at front."<<endl;

cout<<"2. Insert node at rear."<<endl;

//cout<<"3. Insert After Node"<<endl;

//cout<<"3. Search"<<endl;

cout<<"3. Display"<<endl;

cout<<"4. Exit"<<endl;

cout<<"Enter your choice: ";

cin>>ch;

switch(ch)

{

case 1:

cout<<"Enter data: ";

cin>>num;

l.addFront(num);

break;

case 2:

cout<<"Enter data: ";

cin>>num;

l.addRear(num);

break;

/\*case 3:

cout<<"Enter node and data: ";

cin>>nd>>num;

l.addAfter(num,nd);

break;

case 3:

cout<<"enter number: "<<endl;

cin>>num;

temp=l.search(num);

if(num==temp->data)

cout<<num<<" element found"<<endl;

break;\*/

case 3:

l.display();

break;

case 4:

cout<<"exiting\n";

break;

default:

cout<<"Wrong Choice\n";

}

}

}