————————————————————————————————————————

**CREATE AND DISPLAY LINKED LIST**

#include<iostream>

#include<conio.h>

#include<stdlib.h>

using namespace std;

class Node

{

public:

int data;

Node \*next;

};

class list:public Node

{

Node \*listptr,\*temp;

public:

list()

{

listptr=NULL;

temp=NULL;

}

void create();

void display();

};

int main()

{

list l;

int c;

while(1)

{

cout<<"1.Create\n2.Display\n3.exit\n";

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

cin>>c;

switch(c)

{

case 1:l.create();

break;

case 2:l.display();

break;

case 3:return 0;

}

}

return 0;

}

void list::display()

{

Node \*newnode=listptr;

if(newnode==NULL)

{

cout<<"List is empty";

}

while(newnode!=NULL)

{

cout<<newnode->data;

cout<<"-->";

newnode=newnode->next;

}

cout<<"NULL\n";

}

void list::create()

{

Node \*newnode;

newnode=new Node;

int n;

cout<<"Enter an element:";

cin>>n;

newnode->data=n;

newnode->next=NULL;

if(listptr==NULL)

{

listptr=newnode;

temp=newnode;

temp=listptr;

}

else

{

temp->next=newnode;

temp=temp->next;

}

}

————————————————————————————————————————

**CREATE WITH FOR LOOP AND DISPLAY**

#include<iostream>

#include<conio.h>

#include<stdlib.h>

using namespace std;

class Node

{

public:

int data;

Node \*next;

};

class list:public Node

{

Node \*listptr,\*temp;

public:

list()

{

listptr=NULL;

temp=NULL;

}

void create();

void display();

};

int main()

{

list l;

int c,k,i;

while(1)

{

cout<<"1.Create\n2.Display\n3.exit\n";

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

cin>>c;

switch(c)

{

case 1:cout<<"enter number of nodes:";

cin>>k;

for(i=0;i<k;i++)

{

l.create();

}

break;

case 2:l.display();

break;

case 3:return 0;

}

}

return 0;

}

void list::display()

{

Node \*newnode=listptr;

if(newnode==NULL)

{

cout<<"List is empty";

}

while(newnode!=NULL)

{

cout<<newnode->data;

cout<<"-->";

newnode=newnode->next;

}

cout<<"NULL\n";

}

void list::create()

{

Node \*newnode;

newnode=new Node;

int n;

cout<<"Enter an element:";

cin>>n;

newnode->data=n;

newnode->next=NULL;

if(listptr==NULL)

{

listptr=newnode;

temp=newnode;

temp=listptr;

}

else

{

temp->next=newnode;

temp=temp->next;

}

}

————————————————————————————————————————

**INSERT NODE IN LINKED LIST**

#include<iostream>

#include<conio.h>

#include<stdlib.h>

using namespace std;

class Node

{

public:

int data;

Node \*next;

};

class list:public Node

{

Node \*listptr,\*temp;

public:

list()

{

listptr=NULL;

temp=NULL;

}

void create();

void insert\_start(int n);

void insert\_last(int n);

void insert\_between(int n);

void display();

};

int main()

{

list l;

int c,k,i,n;

while(1)

{

cout<<"1.Create\n2.Insert at start\n3.Insert at last\n4.Insert inbetween\n5.Display\n6.Exit\n";

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

cin>>c;

switch(c)

{

case 1:cout<<"enter number of nodes:";

cin>>k;

for(i=0;i<k;i++)

{

l.create();

}

break;

case 2:cout<<"Enter value to insert at the beginning of list:";

cin>>n;

l.insert\_start(n);

break;

case 3:cout<<"Enter value to insert at the end of list:";

cin>>n;

l.insert\_last(n);

break;

case 4:cout<<"Enter value to insert inbetween list:";

cin>>n;

l.insert\_between(n);

break;

case 5:l.display();

break;

case 6:return 0;

}

}

return 0;

}

void list::create()

{

Node \*newnode=new Node;

int n;

cout<<"Enter an element:";

cin>>n;

newnode->data=n;

newnode->next=NULL;

if(listptr==NULL)

{

listptr=newnode;

temp=newnode;

temp=listptr;

}

else

{

temp->next=newnode;

temp=temp->next;

}

}

void list::insert\_start(int n)

{

Node \*newnode=new Node;

newnode->data=n;

newnode->next=listptr;

listptr=newnode;

}

void list::insert\_last(int n)

{

Node \*newnode=new Node;

Node \*temp;

temp=listptr;

newnode->data=n;

while(temp->next!=NULL)

{

temp=temp->next;

}

temp->next=newnode;

}

void list::insert\_between(int n)

{

Node \*newnode=new Node;

newnode->data=n;

newnode->next=temp->next;

temp->next=newnode;

}

void list::display()

{

Node \*newnode=listptr;

if(newnode==NULL)

{

cout<<"List is empty";

}

while(newnode!=NULL)

{

cout<<newnode->data;

cout<<"-->";

newnode=newnode->next;

}

cout<<"NULL\n";

}

————————————————————————————————————————

**DELETE NODE IN LINKED LIST**

#include<iostream>

#include<conio.h>

#include<stdlib.h>

using namespace std;

class Node

{

public:

int data;

Node \*next;

};

class list:public Node

{

Node \*listptr,\*temp;

public:

list()

{

listptr=NULL;

temp=NULL;

}

void create();

void insert\_start(int n);

void insert\_last(int n);

void insert\_between(int n);

void delete\_start();

void delete\_last();

void delete\_between();

void display();

};

int main()

{

list l;

int c,k,i,n;

while(1)

{

cout<<"1.Create\n2.Insert at first\n3.Insert at last\n4.Insert inbetween\n5.Delete at start\n6.Delete at last\n7.Delete inbetween\n8.Display\n9.Exit\n";

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

cin>>c;

switch(c)

{

case 1:cout<<"enter number of nodes:";

cin>>k;

for(i=0;i<k;i++)

{

l.create();

}

break;

case 2:cout<<"Enter value to insert at the beginning of list:";

cin>>n;

l.insert\_start(n);

break;

case 3:cout<<"Enter value to insert at the end of list:";

cin>>n;

l.insert\_last(n);

break;

case 4:cout<<"Enter value to insert inbetween list:";

cin>>n;

l.insert\_between(n);

break;

case 5:l.delete\_start();

break;

case 6:l.delete\_last();

break;

case 7:l.delete\_between();

break;

case 8:l.display();

break;

case 9:return 0;

}

}

return 0;

}

void list::create()

{

Node \*newnode=new Node;

int n;

cout<<"Enter an element:";

cin>>n;

newnode->data=n;

newnode->next=NULL;

if(listptr==NULL)

{

listptr=newnode;

temp=newnode;

temp=listptr;

}

else

{

temp->next=newnode;

temp=temp->next;

}

}

void list::insert\_start(int n)

{

Node \*newnode=new Node;

newnode->data=n;

newnode->next=listptr;

listptr=newnode;

}

void list::insert\_last(int n)

{

Node \*newnode=new Node;

Node \*temp;

temp=listptr;

newnode->data=n;

while(temp->next!=NULL)

{

temp=temp->next;

}

temp->next=newnode;

}

void list::insert\_between(int n)

{

Node \*newnode=new Node;

newnode->data=n;

newnode->next=temp->next;

temp->next=newnode;

}

void list::delete\_start()

{

Node \*temp;

temp=listptr;

listptr=listptr->next;

free(temp);

}

void list::delete\_last()

{

Node \*q,\*temp;

temp=listptr;

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

{

temp=temp->next;

}

q=temp->next;

free(q);

temp->next=NULL;

}

void list::delete\_between()

{

Node \*q,\*temp;

int n,c=1;

cout<<"Enter node to delete:";

cin>>n;

temp=listptr;

while(c!=n)

{

temp=temp->next;

c++;

}

q=temp->next;

temp->next=q->next;

free(q);

}

void list::display()

{

Node \*newnode=listptr;

if(newnode==NULL)

{

cout<<"List is empty";

}

while(newnode!=NULL)

{

cout<<newnode->data;

cout<<"-->";

newnode=newnode->next;

}

cout<<"NULL\n";

}

————————————————————————————————————————

**REVERSE OF LINKED LIST**

#include<iostream>

#include<conio.h>

#include<stdlib.h>

using namespace std;

class Node

{

public:

int data;

Node \*next;

};

class list:public Node

{

Node \*listptr,\*temp;

public:

list()

{

listptr=NULL;

temp=NULL;

}

void create();

void reverse();

void display();

};

int main()

{

list l;

int c,k,i;

while(1)

{

cout<<"1.Create\n2.Reverse\n3.Display\n4.exit\n";

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

cin>>c;

switch(c)

{

case 1:cout<<"enter number of nodes:";

cin>>k;

for(i=0;i<k;i++)

{

l.create();

}

break;

case 2:l.reverse();

break;

case 3:l.display();

break;

case 4:return 0;

}

}

return 0;

}

void list::display()

{

Node \*newnode;

if(listptr==NULL)

{

cout<<"List is empty";

}

else

{

newnode=listptr;

while(newnode!=NULL)

{

cout<<newnode->data;

cout<<"-->";

newnode=newnode->next;

}

cout<<"NULL\n";

}

}

void list::create()

{

Node \*newnode;

newnode=new Node;

int n;

cout<<"Enter an element:";

cin>>n;

newnode->data=n;

newnode->next=NULL;

if(listptr==NULL)

{

listptr=newnode;

temp=newnode;

temp=listptr;

}

else

{

temp->next=newnode;

temp=temp->next;

}

}

void list::reverse()

{

Node \*back,\*curr,\*forw;

forw=listptr;

curr=NULL;

while(forw!=NULL)

{

back=curr;

curr=forw;

forw=forw->next;

curr->next=back;

}

listptr=curr;

}

-------------------------------------------------------------------------------------------------------------------------------

**CREATE + DISPLAY + INSERT + DELETE + REVERSE + LENGTH**

#include<iostream>

#include<conio.h>

#include<stdlib.h>

using namespace std;

class Node

{

public:

int data;

Node \*next;

};

class list:public Node

{

Node \*listptr,\*temp;

public:

list()

{

listptr=NULL;

temp=NULL;

}

void create();

void insert\_start(int n);

void insert\_last(int n);

void insert\_between(int n);

void delete\_start();

void delete\_last();

void delete\_between();

void reverse();

void length();

void display();

};

int main()

{

list l;

int c,k,i,n;

while(1)

{

cout<<"1.Create\n2.Insert at first\n3.Insert at last\n4.Insert inbetween\n5.Delete at start\n6.Delete at last\n7.Delete inbetween\n8.Reverse\n9.Length of linked list\n10.Display\n11.Exit\n";

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

cin>>c;

switch(c)

{

case 1:cout<<"enter number of nodes:";

cin>>k;

for(i=0;i<k;i++)

{

l.create();

}

break;

case 2:cout<<"Enter value to insert at the beginning of list:";

cin>>n;

l.insert\_start(n);

break;

case 3:cout<<"Enter value to insert at the end of list:";

cin>>n;

l.insert\_last(n);

break;

case 4:cout<<"Enter value to insert inbetween list:";

cin>>n;

l.insert\_between(n);

break;

case 5:l.delete\_start();

break;

case 6:l.delete\_last();

break;

case 7:l.delete\_between();

break;

case 8:l.reverse();

break;

case 9:l.length();

break;

case 10:l.display();

break;

case 11:return 0;

}

}

return 0;

}

void list::create()

{

Node \*newnode=new Node;

int n;

cout<<"Enter an element:";

cin>>n;

newnode->data=n;

newnode->next=NULL;

if(listptr==NULL)

{

listptr=newnode;

temp=newnode;

temp=listptr;

}

else

{

temp->next=newnode;

temp=temp->next;

}

}

void list::insert\_start(int n)

{

Node \*newnode=new Node;

newnode->data=n;

newnode->next=listptr;

listptr=newnode;

}

void list::insert\_last(int n)

{

Node \*newnode=new Node;

Node \*temp;

temp=listptr;

newnode->data=n;

while(temp->next!=NULL)

{

temp=temp->next;

}

temp->next=newnode;

}

void list::insert\_between(int n)

{

Node \*newnode=new Node;

newnode->data=n;

newnode->next=temp->next;

temp->next=newnode;

}

void list::delete\_start()

{

Node \*temp;

temp=listptr;

listptr=listptr->next;

free(temp);

}

void list::delete\_last()

{

Node \*q,\*temp;

temp=listptr;

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

{

temp=temp->next;

}

q=temp->next;

free(q);

temp->next=NULL;

}

void list::delete\_between()

{

Node \*q,\*temp;

int n,c=1;

cout<<"Enter node to delete:";

cin>>n;

temp=listptr;

while(c!=n)

{

temp=temp->next;

c++;

}

q=temp->next;

temp->next=q->next;

free(q);

}

void list::reverse()

{

Node \*back,\*curr,\*forw;

forw=listptr;

curr=NULL;

while(forw!=NULL)

{

back=curr;

curr=forw;

forw=forw->next;

curr->next=back;

}

listptr=curr;

}

void list::length()

{

temp=listptr;

int count=0;

while(temp)

{

temp=temp->next;

count++;

}

cout<<"length="<<count<<endl;

}

void list::display()

{

Node \*newnode=listptr;

if(newnode==NULL)

{

cout<<"List is empty";

}

while(newnode!=NULL)

{

cout<<newnode->data;

cout<<"-->";

newnode=newnode->next;

}

cout<<"NULL\n";

}

————————————————————————————————————————

**CONCATENATE TWO LINKED LISTS**

#include<iostream>

#include<conio.h>

#include<stdlib.h>

using namespace std;

class Node

{

public:

int data;

Node \*next;

};

class list:public Node

{

Node \*t1,\*t2,\*l1,\*l2;

public:

list()

{

t1=NULL;

t2=NULL;

l1=NULL;

l2=NULL;

}

void createA();

void createB();

void displayA();

void displayB();

void display();

void concatenate();

};

int main()

{

list a;

int c,k,i;

while(1)

{

cout<<"1.Create linked list A\n2.Display linked list A\n3.Create linked list B\n4.Display linked list B\n5.Concatenate\n6.Display\n7.Exit\n";

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

cin>>c;

switch(c)

{

case 1:cout<<"enter number of nodes:";

cin>>k;

for(i=0;i<k;i++)

{

a.createA();

}

break;

case 2:a.displayA();

break;

case 3:cout<<"enter number of nodes:";

cin>>k;

for(i=0;i<k;i++)

{

a.createB();

}

break;

case 4:a.displayB();

break;

case 5:a.concatenate();

break;

case 6:a.display();

break;

case 7:return 0;

}

}

return 0;

}

void list::displayA()

{

Node \*newnode=l1;

if(newnode==NULL)

{

cout<<"List is empty";

}

while(newnode!=NULL)

{

cout<<newnode->data;

cout<<"-->";

newnode=newnode->next;

}

cout<<"NULL\n";

}

void list::createA()

{

Node \*newnode;

newnode=new Node;

int n;

cout<<"Enter an element:";

cin>>n;

newnode->data=n;

newnode->next=NULL;

if(l1==NULL)

{

l1=newnode;

t1=newnode;

t1=l1;

}

else

{

t1->next=newnode;

t1=t1->next;

}

}

void list::displayB()

{

Node \*newnode=l2;

if(newnode==NULL)

{

cout<<"List is empty";

}

while(newnode!=NULL)

{

cout<<newnode->data;

cout<<"-->";

newnode=newnode->next;

}

cout<<"NULL\n";

}

void list::createB()

{

Node \*newnode;

newnode=new Node;

int n;

cout<<"Enter an element:";

cin>>n;

newnode->data=n;

newnode->next=NULL;

if(l2==NULL)

{

l2=newnode;

t2=newnode;

t2=l2;

}

else

{

t2->next=newnode;

t2=t2->next;

}

}

void list::display()

{

Node \*newnode=l1;

if(newnode==NULL)

{

cout<<"List is empty";

}

while(newnode!=NULL)

{

cout<<newnode->data;

cout<<"-->";

newnode=newnode->next;

}

cout<<"NULL\n";

}

void list::concatenate()

{

Node \*head1,\*head2,\*p;

head1=l1;

head2=l2;

p=head1;

while(p->next!=NULL)

{

p=p->next;

}

p->next=head2;

}

-------------------------------------------------------------------------------------------------------------------------------

**CONCATENATE TWO LINKED LISTS**

#include<iostream>

#include<conio.h>

#include<stdlib.h>

using namespace std;

class Node

{

public:

int data;

Node \*next;

};

class list:public Node

{

Node \*t1,\*t2,\*l1,\*l2;

public:

list()

{

t1=NULL;

t2=NULL;

l1=NULL;

l2=NULL;

}

void createA();

void createB();

void displayA();

void displayB();

void display();

void concatenate();

};

int main()

{

list a;

int c,k,i;

while(1)

{

cout<<"1.Create linked list A\n2.Display linked list A\n3.Create linked list B\n4.Display linked list B\n5.Concatenate\n6.Display\n7.Exit\n";

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

cin>>c;

switch(c)

{

case 1:cout<<"enter number of nodes:";

cin>>k;

for(i=0;i<k;i++)

{

a.createA();

}

break;

case 2:a.displayA();

break;

case 3:cout<<"enter number of nodes:";

cin>>k;

for(i=0;i<k;i++)

{

a.createB();

}

break;

case 4:a.displayB();

break;

case 5:a.concatenate();

break;

case 6:a.display();

break;

case 7:return 0;

}

}

return 0;

}

void list::displayA()

{

Node \*newnode=l1;

if(newnode==NULL)

{

cout<<"List is empty";

}

while(newnode!=NULL)

{

cout<<newnode->data;

cout<<"-->";

newnode=newnode->next;

}

cout<<"NULL\n";

}

void list::createA()

{

Node \*newnode;

newnode=new Node;

int n;

cout<<"Enter an element:";

cin>>n;

newnode->data=n;

newnode->next=NULL;

if(l1==NULL)

{

l1=newnode;

t1=newnode;

t1=l1;

}

else

{

t1->next=newnode;

t1=t1->next;

}

}

void list::displayB()

{

Node \*newnode=l2;

if(newnode==NULL)

{

cout<<"List is empty";

}

while(newnode!=NULL)

{

cout<<newnode->data;

cout<<"-->";

newnode=newnode->next;

}

cout<<"NULL\n";

}

void list::createB()

{

Node \*newnode;

newnode=new Node;

int n;

cout<<"Enter an element:";

cin>>n;

newnode->data=n;

newnode->next=NULL;

if(l2==NULL)

{

l2=newnode;

t2=newnode;

t2=l2;

}

else

{

t2->next=newnode;

t2=t2->next;

}

}

void list::display()

{

Node \*newnode=l1;

if(newnode==NULL)

{

cout<<"List is empty";

}

while(newnode!=NULL)

{

cout<<newnode->data;

cout<<"-->";

newnode=newnode->next;

}

cout<<"NULL\n";

}

void list::concatenate()

{

Node \*head1,\*head2,\*p;

head1=l1;

head2=l2;

p=head1;

while(p->next!=NULL)

{

p=p->next;

}

p->next=head2;

}