** LAB Task**

**Submitted to:**

Mam Yasmeen Jana

**Submitted by:**

Hassan Raza Bhatti

**Registration No:**

SP22-BCS-096

**Section:**

"B"

**Subject:**

Data Structure and Algorithms

**Date:**

08-Oct-2023

**Comsats University Islamabad,**

**Vehari campus.**

**Activity 1:**

#include <iostream>

using namespace std; // Add this line to include the std namespace

class Node {

public:

int data;

Node\* next;

Node(int value) : data(value), next(nullptr) {}

};

class LinkedList {

private:

Node\* head;

public:

LinkedList() : head(nullptr) {}

void insert(int value) {

Node\* newNode = new Node(value);

if (head == nullptr) {

head = newNode;

} else {

Node\* current = head;

while (current->next != nullptr) {

current = current->next;

}

current->next = newNode;

}

}

void display() {

// Check if the linked list is empty.

if (head == NULL) {

cout << "The linked list is empty." << endl;

return;

}

// Create a temporary pointer to traverse the linked list.

Node\* ptr = head;

cout << "The linked list is:" << endl;

while (ptr != nullptr) {

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

ptr = ptr->next;

}

cout<<endl;

cout<<endl;

ptr = head;

cout << "\*\*\*\*head address:\*\*\*\* " << &head << endl;

cout<<"-------------------------------------"<<endl;

cout << "\*\*\*\*head address:\*\*\*\* " << head << endl;

cout<<"-------------------------------------"<<endl;

cout << "\*\*\*\*ptr address:\*\*\*\* " << &ptr << endl;

cout<<"-------------------------------------"<<endl;

cout << "\*\*\*\*ptr Content:\*\*\*\* " << ptr << endl;

while (ptr != nullptr) {

cout<<"-------------------------------------"<<endl;

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

cout << "ptr:" << ptr << endl;

cout << "ptr->next:" << ptr->next << endl;

ptr = ptr->next;

}

}

~LinkedList() {

Node\* current = head;

while (current != nullptr) {

Node\* next = current->next;

delete current;

current = next;

}

}

};

int main() {

LinkedList list;

list.insert(1);

list.insert(2);

list.insert(20);

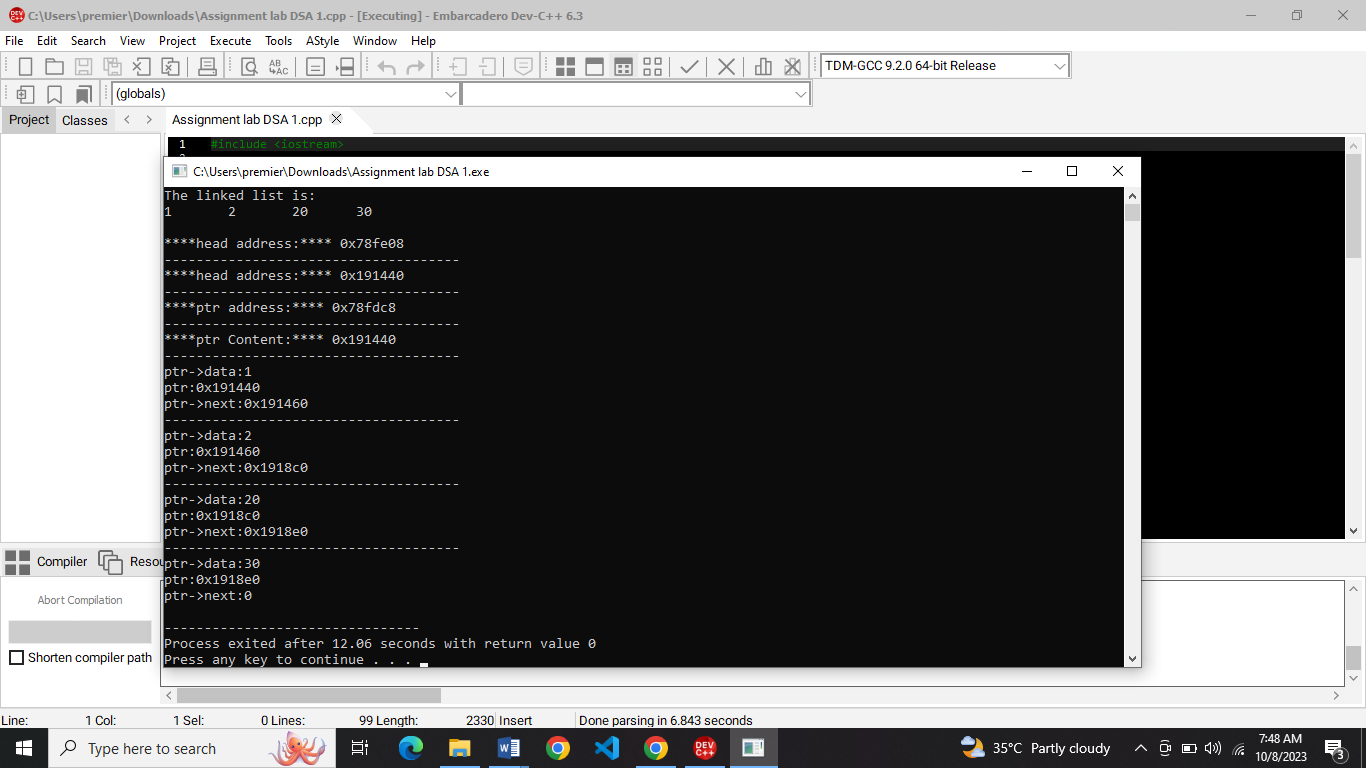
list.insert(30);

list.display();

return 0;

}

OUTPUT:



Activity 2: Menu

#include <iostream>

using namespace std;

class Node

{

private:

int data;

Node \*next;

public:

Node \*head;

Node()

{

head == NULL;

}

// inserion at the Begining\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void insert\_at\_begin(int n)

{

if (head == NULL)

{

head = new Node();

head->data = n;

head->next = NULL;

}

else

{

Node \*p;

p = new Node();

p->data = n;

// p->next=NULL;

p->next = head;

head = p;

}

disp();

}

// inserion at the END\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void insert\_at\_end(int n)

{

if (head == NULL)

{

head = new Node();

head->data = n;

head->next = NULL;

}

else

{

Node \*p, \*temp;

temp = head;

while (temp->next != NULL)

{

temp = temp->next;

}

p = new Node();

p->next = NULL;

p->data = n;

temp->next = p;

}

disp();

}

// inserion at after Specific Location\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void insert\_at\_afterval(int loc, int n)

{

if (head == NULL)

{

head = new Node();

head->data = n;

head->next = NULL;

}

else

{

Node \*temp;

temp = head;

while (temp->data != loc)

{

temp = temp->next;

}

Node \*p;

p = new Node();

p->data = n;

p->next = temp->next;

temp->next = p;

}

cout << "\nInserted Successfully After location: " << loc;

disp();

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*deletion\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//deletion at begining

void delete\_at\_begin()

{

if (head == NULL)

{

cout << "Linked list not Exist";

}

else

{

Node \*temp;

temp = head;

head = temp->next;

delete temp;

temp = NULL;

}

disp();

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//deletion at END

void delete\_at\_end(){

if(head==NULL){

cout<<"No Nodes Exist"<<endl;

}

else{

Node \*ptr;

Node \*ptr1;

ptr = head;

while(ptr->next!=NULL){

ptr1 = ptr;

ptr = ptr->next;

}

ptr1->next = NULL;

delete(ptr);

}

disp();

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//deletion at specific value

void delete\_at\_value(int x){

Node \*temp, \*temp1;

temp = head;

if (temp->data == x)

{

head = temp->next;

delete temp;

return;

}

temp1 = temp;

while (temp != NULL)

{

if (temp->data == x)

{

temp1->next = temp->next;

delete temp;

return;

}

temp1 = temp;

temp = temp->next;

}

}

// Display\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void disp()

{

Node \*temp;

temp = head;

if (temp == NULL)

{

cout << " \nNo data is in the list..\n\n\n" << endl;

return;

}

else

{

cout << " \nThe items present in the list are :";

while (temp != NULL)

{

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

temp = temp->next;

}

cout <<"\n\n\n"<<endl;

}

}

};

//\*\*\*\*\*--------------DOUBLY------------\*\*\*\*\*

class Node2

{

private:

int data;

Node2 \*next;

Node2 \*prv;

public:

Node2 \*head;

Node2(){

head=NULL;

}

//INsertion\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void insert\_beg(int n){

if(head==NULL){

head=new Node2();

head->data=n;

head->next=NULL;

head->prv=NULL;

}

else{

Node2 \*p;

p=new Node2();

p->data=n;

p->next= head;

p->prv= NULL;

head=p;

}

disp();

}

void insert\_end(int n){

if(head==NULL){

head=new Node2();

head->data =n;

head->next=NULL;

head->prv=NULL;

}

else{

Node2 \*p;

p=new Node2();

p->data=n;

p->next = NULL;

Node2 \*temp;

temp = head;

while(temp->next!=NULL){

temp = temp->next;

}

p->prv = temp;

temp->next = p;

}

disp();

}

void insert\_at\_val(int val, int n){

if(head==NULL)

{

head=new Node2();

head->data=n;

head->next=NULL;

head->prv=NULL;

}

else

{

Node2 \*temp;

Node2 \*p;

p=new Node2();

p->data=n;

temp=head;

while(temp->data!= val)

{

temp = temp->next;

}

p->next= temp->next;

temp->next = p;

p->prv = temp;

}

disp();

}

//DEletion\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void del\_beg(){

if(head==NULL){

cout<<"No Nodes Exist"<<endl;

}

else{

Node2 \*ptr;

ptr = head;

head = head->next;

head->prv = ptr;

delete(ptr);

ptr = NULL;

}

disp();

}

void del\_end(){

if(head==NULL){

cout<<"No Nodes Exist"<<endl;

}

else{

Node2 \*temp;

Node2 \*temp1;

temp = head;

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

temp1 = temp;

temp = temp->next;

}

temp->next = NULL;

temp->prv = temp1;

delete(temp);

}

disp();

}

void del\_at\_value(int val) {

if (head == NULL) {

cout << "List is empty. Cannot delete." << endl;

return;

}

Node2\* ptr = head;

while (ptr != NULL && ptr->data != val) {

ptr = ptr->next;

}

if (ptr == NULL) {

cout << "Value not found in the list." << endl;

return;

}

if (ptr->prv != NULL) {

ptr->prv->next = ptr->next;

} else {

head = ptr->next;

}

if (ptr->next != NULL) {

ptr->next->prv = ptr->prv;

}

delete ptr;

disp();

}

// Display\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void disp()

{

Node2 \*temp;

temp = head;

if (temp == NULL)

{

cout << " \nNo data is in the list..\n\n\n" << endl;

return;

}

else

{

cout << " \nThe items present in the list are :";

while (temp != NULL)

{

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

temp = temp->next;

}

cout <<"\n\n\n"<<endl;

}

}

};

class Node3 {

private:

int data;

Node3\* next;

public:

Node3\* head;

Node3() {

head = NULL;

}

void insert\_endd(int n) {

if (head == NULL) {

head = new Node3();

head->data = n;

head->next = head; // Circular: Point to itself

} else {

Node3\* p, \* ptr;

ptr = head;

while (ptr->next != head) {

ptr = ptr->next;

}

p = new Node3();

p->data = n;

p->next = head;

ptr->next = p;

}

dispp();

}

void insert\_begg(int n) {

if (head == NULL) {

head = new Node3();

head->data = n;

head->next = head; // Circular: Point to itself

} else {

Node3\* p, \* ptr;

ptr = head;

while (ptr->next != head) {

ptr = ptr->next;

}

p = new Node3();

p->data = n;

p->next = head;

ptr->next = p;

head=p;

}

dispp();

}

void insert\_at\_valuee(int pos, int n) {

if (head == NULL) {

cout << "List is empty. Cannot insert." << endl;

return;

}

Node3\* ptr;

ptr = head;

while (ptr->data != pos) {

ptr = ptr->next;

}

Node3\* p;

p = new Node3();

p->data = n;

p->next = ptr->next;

ptr->next = p;

dispp();

}

void dispp() {

if (head == NULL) {

cout << "No data is in the list." << endl;

return;

}

Node3\* ptr = head;

do {

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

ptr = ptr->next;

} while (ptr != head);

cout<<endl;

}

void del\_begg() {

if (head == NULL) {

cout << "List is empty. Cannot delete." << endl;

return;

}

Node3\* temp = head;

Node3\* ptr = head;

while (ptr->next != head) {

ptr = ptr->next;

}

head = head->next;

ptr->next = head;

delete temp;

dispp();

}

void del\_endd() {

if (head == NULL) {

cout << "List is empty. Cannot delete." << endl;

return;

}

if (head->next == head) {

delete head;

head = NULL;

return;

}

Node3\* ptr = head;

Node3\* prev = nullptr;

while (ptr->next != head) {

prev = ptr;

ptr = ptr->next;

}

prev->next = head;

delete ptr;

dispp();

}

void del\_at\_valuee(int val) {

if (head == NULL) {

cout << "List is empty. Cannot delete." << endl;

return;

}

if (head->data == val) {

Node3\* temp = head;

Node3\* ptr = head;

while (ptr->next != head) {

ptr = ptr->next;

}

head = head->next;

ptr->next = head;

delete temp;

return;

}

Node3\* ptr = head;

Node3\* prev = nullptr;

do {

prev = ptr;

ptr = ptr->next;

} while (ptr != head && ptr->data != val);

if (ptr != head) {

prev->next = ptr->next;

delete ptr;

} else {

cout << "Value not found in the list." << endl;

}

dispp();

}

};

int main()

{

Node obj;

Node2 obj2;

Node3 obj3;

int n, v, id, mn;

do

{

cout << "Select any One Linked List" << endl;

cout << "1: SINGLY" << endl;

cout << "2: DOUBLY" << endl;

cout << "3: CIRCULAR" << endl;

cin >> mn;

switch (mn){

case 1:

min:

cout << "Select any One Operation You want to Perform.." << endl;

cout << "1: INSERTION" << endl;

cout << "2: DELETION" << endl;

cin >> id;

switch (id)

{

case 1:

cout << "1: To add Node at Begining" << endl;

cout << "2: To add Node at End" << endl;

cout << "3: To add Node at Specific Location" << endl;

cout << "4: to Back" << endl;

cout << "5: to exit" << endl;

cin >> n;

switch (n)

{

case 1:

cout << "\nEnter the value to insert: ";

cin >> v;

obj.insert\_at\_begin(v);

break;

case 2:

cout << "\nEnter the value to insert: ";

cin >> v;

obj.insert\_at\_end(v);

break;

case 3:

int o, loc;

cout << "Enter location value: ";

cin >> loc;

cout << "Enter the value to insert: ";

cin >> v;

obj.insert\_at\_afterval(loc, v);

break;

case 4:

goto min;

case 5:

exit(1);

default:

cout << "Choose valid Option" << endl;

break;

}

break;

system("pause");

case 2:

cout << "1: To Delete Node from Begining" << endl;

cout << "2: To Delete Node from End" << endl;

cout << "3: To Delete Specific Node" << endl;

cout << "4: to Back" << endl;

cout << "5: to exit" << endl;

cin >> n;

switch (n)

{

case 1:

cout<<"Node deleted from Begining....";

obj.delete\_at\_begin();

break;

case 2:

cout<<"Node deleted from END....";

obj.delete\_at\_end();

break;

}

case 3:

cout << "Enter the node value to Delete: ";

cin >> v;

obj.delete\_at\_value(v);

obj.disp();

break;

default:

cout << "Choose valid Option" << endl;

break;

}

break;

//end of case singly

//-----------------------------------------------

case 2:

tg:

cout << "Select any One Operation You want to Perform.." << endl;

cout << "1: INSERTION" << endl;

cout << "2: DELETION" << endl;

cin >> id;

switch (id){

//insertion in doubly

case 1:

cout << "1: To add Node at Begining" << endl;

cout << "2: To add Node at End" << endl;

cout << "3: To add Node at Specific Location" << endl;

cout << "4: to Back" << endl;

cout << "5: to exit" << endl;

cin >> n;

switch (n)

{

case 1:

cout << "\nEnter the value to insert: ";

cin >> v;

obj2.insert\_beg(v);

break;

case 2:

cout << "\nEnter the value to insert: ";

cin >> v;

obj2.insert\_end(v);

break;

case 3:

int o, loc;

cout << "Enter location value: ";

cin >> loc;

cout << "Enter the value to insert: ";

cin >> v;

obj2.insert\_at\_val(loc, v);

case 4:

goto tg;

case 5:

exit(1);

default:

cout << "Choose valid Option" << endl;

break;

}

break;

//deletion in doubly

case 2:

cout << "1: To Delete Node from Begining" << endl;

cout << "2: To Delete Node from End" << endl;

cout << "3: To Delete Specific Node" << endl;

cout << "4: to Back" << endl;

cout << "5: to exit" << endl;

cin >> n;

switch (n)

{

case 1:

cout<<"Node deleted from Begining....";

obj2.del\_beg();

break;

case 2:

cout<<"Node deleted from END....";

obj2.del\_end();

break;

case 3:

cout << "Enter the node value to Delete: ";

cin >> v;

obj2.del\_at\_value(v);

break;

default:

cout << "Choose valid Option" << endl;

break;

}

}

break;

//end of case doubly

//-----------------------------------

case 3:

gb:

cout << "Select any One Operation You want to Perform.." << endl;

cout << "1: INSERTION" << endl;

cout << "2: DELETION" << endl;

cin >> id;

switch (id){

//insertion in Circular

case 1:

cout << "1: To add Node at Begining" << endl;

cout << "2: To add Node at End" << endl;

cout << "3: To add Node at Specific Location" << endl;

cout << "4: to Back" << endl;

cout << "5: to exit" << endl;

cin >> n;

switch (n)

{

case 1:

cout << "\nEnter the value to insert: ";

cin >> v;

obj3.insert\_begg(v);

break;

case 2:

cout << "\nEnter the value to insert: ";

cin >> v;

obj3.insert\_endd(v);

break;

case 3:

int o, loc;

cout << "Enter location value: ";

cin >> loc;

cout << "Enter the value to insert: ";

cin >> v;

obj3.insert\_at\_valuee(loc, v);

case 4:

goto gb;

case 5:

exit(1);

default:

cout << "Choose valid Option" << endl;

break;

}

break;

//deletion in Circular

case 2:

cout << "1: To Delete Node from Begining" << endl;

cout << "2: To Delete Node from End" << endl;

cout << "3: To Delete Specific Node" << endl;

cout << "4: to Back" << endl;

cout << "5: to exit" << endl;

cin >> n;

switch (n)

{

case 1:

cout<<"Node deleted from Begining....";

obj3.del\_begg();

break;

case 2:

cout<<"Node deleted from END....";

obj3.del\_endd();

break;

case 3:

cout << "Enter the node value to Delete: ";

cin >> v;

obj3.del\_at\_valuee(v);

break;

default:

cout << "Choose valid Option" << endl;

break;

}

}

break;

//end of case circular

default:

cout << "Choose valid Option" << endl;

break;

//end of singly/doubly switch

}

} while (n != 4);

system("pause");

return 0;

}

OUTPUT :

