

**Department of Computer Science**

**Data Structures and Algorithms- Lab 4**

Class: BSCS-3D

Name: Faisal Tayyab

Enrollment Number: 01-134232-057

Date: 11-10-2024

Github Link: https://github.com/FaisalTayyab/Lab-Journal-4.git

**Exercise 1**

Implement the (class) Linked List with Dummy head node to create a list of integers. You need to provide the implementation of the member functions as described in the following.

**class HList**

**{**

**private:**

**Node \* head;**

**public:**

**HList();**

|  |  |
| --- | --- |
| **// Checks if the list is empty or not**  **bool emptyList();**  **// Inserts a new node with value ‘newV’ after node with value ‘oldV’in the list**  **void insert\_after(int oldV, int newV);**  **// Inserts a new node at the start of the list**  **void insert\_begin(int value);**  **// Inserts a new node at the end of the list**  **void insert\_end(int value);**  **// Deletes a node of value ‘val’ from the list**  **void delete\_Node(int val);**  **// Deletes a node from the beginning of the list**  **void delete\_begin();**  **// Deletes a node from the end of the list**  **void delete\_end();**  **// Displays the values stored in the list**  **void traverse();**  **};**  #include<iostream>  usingnamespace std;  classNode {  public:  int data;  Node\* next;  };  classHList {  public:  Node\* head;  public:  HList()  {  head = newNode;  head->data = 0;  head->next = NULL;  }  bool is\_empty()  {  if (head->next == NULL)  {  returntrue;  }  else  {  returnfalse;  }  }  void Insert\_Begin(intValue)  {  Node\* temp = newNode;  temp->data = Value;  temp->next = NULL;  if (is\_empty())  {  head->next = temp;  }  else  {  temp->next = head->next;  head->next = temp;  }  }  void Insert\_End(intValue)  {  Node\* temp = newNode;  temp->data = Value;  temp->next = NULL;  if (is\_empty())  {  head->next = temp;  }  else  {  Node\* t = head;  while (t->next != NULL)  {  t = t->next;  }  t->next = temp;  }  }  void Insert\_After(intOldV, intNewV)  {  if (is\_empty())  {  cout <<"List is empty!"<< endl;  }  Node\* p = head->next;  while (p != NULL&& p->data != OldV)  {  p = p->next;  }  if (p == NULL)  {  cout <<"Node not found!"<< endl;  }  Node\* temp = newNode;  temp->data = NewV;  temp->next = p->next;  p->next = temp;  }  void Delete\_Node(intValue)  {  if (is\_empty())  {  cout <<"List is empty!"<< endl;  }  Node\* p = head->next;  Node\* q = head;  while (p != NULL&& p->data != Value)  {  q = p;  p = p->next;  }  if (p == NULL)  {  cout <<"Node not found!"<< endl;  }  q->next = p->next;  delete p;  }  void Delete\_Begin()  {  if (is\_empty())  {  cout <<"List is empty!"<< endl;  return;  }  Node\* temp = head->next;  head->next = temp->next;  delete temp;  }  void Delete\_End()  {  if (is\_empty())  {  cout <<"List is empty!"<< endl;  return;  }  Node\* p = head->next;  Node\* q = head;  while (p->next != NULL)  {  q = p;  p = p->next;  }  q->next = NULL;  delete p;  }  void traverse()  {  Node\* current = head->next;  cout <<"List is: "<< endl;  while (current != NULL)  {  cout << current->data <<" ";  current = current->next;  }  cout << endl;  }  };  int main()  {  HList li;  li.Insert\_Begin(10);  li.Insert\_Begin(20);  li.Insert\_Begin(30);  li.Insert\_End(40);  li.Insert\_End(50);  li.Insert\_After(30, 60);  li.traverse();  li.Delete\_Begin();  li.traverse();  li.Delete\_End();  li.traverse();  li.Delete\_Node(20);  li.traverse();  return 0;  }    **Exercise 2**   |  | | --- | | Implement the (class) Circular Linked List to create a list of integers. You need to provide the implementation of the member functions as described in the following.  **class CList**  **{**  **private:**  **Node \*head;**  **public:**  **CList();**  **// Checks if the list is empty or not**  **bool emptyList();**  **// Inserts a new node with value ‘value’ at position ‘pos’ in the list (restrict user from entering pos=1)**  **void insert\_at(int pos, int value);**  **// Inserts a new node at the start of the list**  **void insert\_begin(int value);**  **// Inserts a new node at the end of the list**  **void insert\_end(int value);**  **// Deletes a node from position ‘pos’ of the list (restrict user from entering first node)**  **void deleteNode(int pos);**  **// Deletes a node from the beginning of the list**  **void delete\_begin();**  **// Deletes a node from the end of the list**  **void delete\_end();**  **// Displays the values stored in the list**  **void traverse();**  **};** | |

#include<iostream>

usingnamespace std;

structNode {

int data;

Node\* next;

};

classCList {

private:

Node\* head;

public:

CList() {

head = NULL;

}

void Insert\_Begin(intValue) {

Node\* temp = newNode;

temp->data = Value;

if (head == NULL) {

head = temp;

temp->next = head;

}

else {

Node\* p = head;

while (p->next != head) {

p = p->next;

}

p->next = temp;

temp->next = head;

head = temp;

}

}

void Insert\_End(intValue) {

Node\* temp = newNode;

temp->data = Value;

if (head == NULL) {

head = temp;

temp->next = head;

}

else {

Node\* p = head;

while (p->next != head) {

p = p->next;

}

p->next = temp;

temp->next = head;

}

}

void Insert\_After(intpos, intvalue) {

if (head == NULL) {

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

return;

}

Node\* temp = head;

do {

if (temp->data == pos) {

Node\* newNode = newNode;

newNode->data = value;

newNode->next = temp->next;

temp->next = newNode;

return;

}

temp = temp->next;

} while (temp != head);

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

}

void Delete\_Node(intpos)

{

Node\* temp = head, \* t = NULL;

if (head == NULL)

{

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

}

else {

if (head->data == pos)

{

do {

t = temp;

temp = temp->next;

} while (temp != head);

head = head->next;

delete temp;

t->next = head;

}

else {

do {

t = temp;

temp = temp->next;

} while (temp->data != pos&& temp != head);

if (temp != head) {

t->next = temp->next;

delete temp;

}

else {

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

}

}

}

}

void Delete\_Begin() {

if (head == NULL) {

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

return;

}

Node\* temp = head;

if (temp->next == head) {

delete head;

head = NULL;

}

else {

Node\* last = head;

while (last->next != head) {

last = last->next;

}

head = head->next;

last->next = head;

delete temp;

}

}

void Delete\_End() {

if (head == NULL) {

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

return;

}

Node\* temp = head;

if (temp->next == head) {

delete head;

head = NULL;

}

else {

Node\* prev = NULL;

while (temp->next != head) {

prev = temp;

temp = temp->next;

}

prev->next = head;

delete temp;

}

}

void traverse() {

if (head == NULL) {

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

return;

}

Node\* temp = head;

cout <<"List is: "<< endl;

do {

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

temp = temp->next;

} while (temp != head);

cout << endl;

}

};

int main() {

CList li;

li.Insert\_Begin(10);

li.Insert\_Begin(20);

li.Insert\_Begin(30);

li.Insert\_Begin(40);

li.Insert\_End(50);

li.Insert\_End(60);

li.Insert\_After(50, 70);

li.traverse();

li.Delete\_Begin();

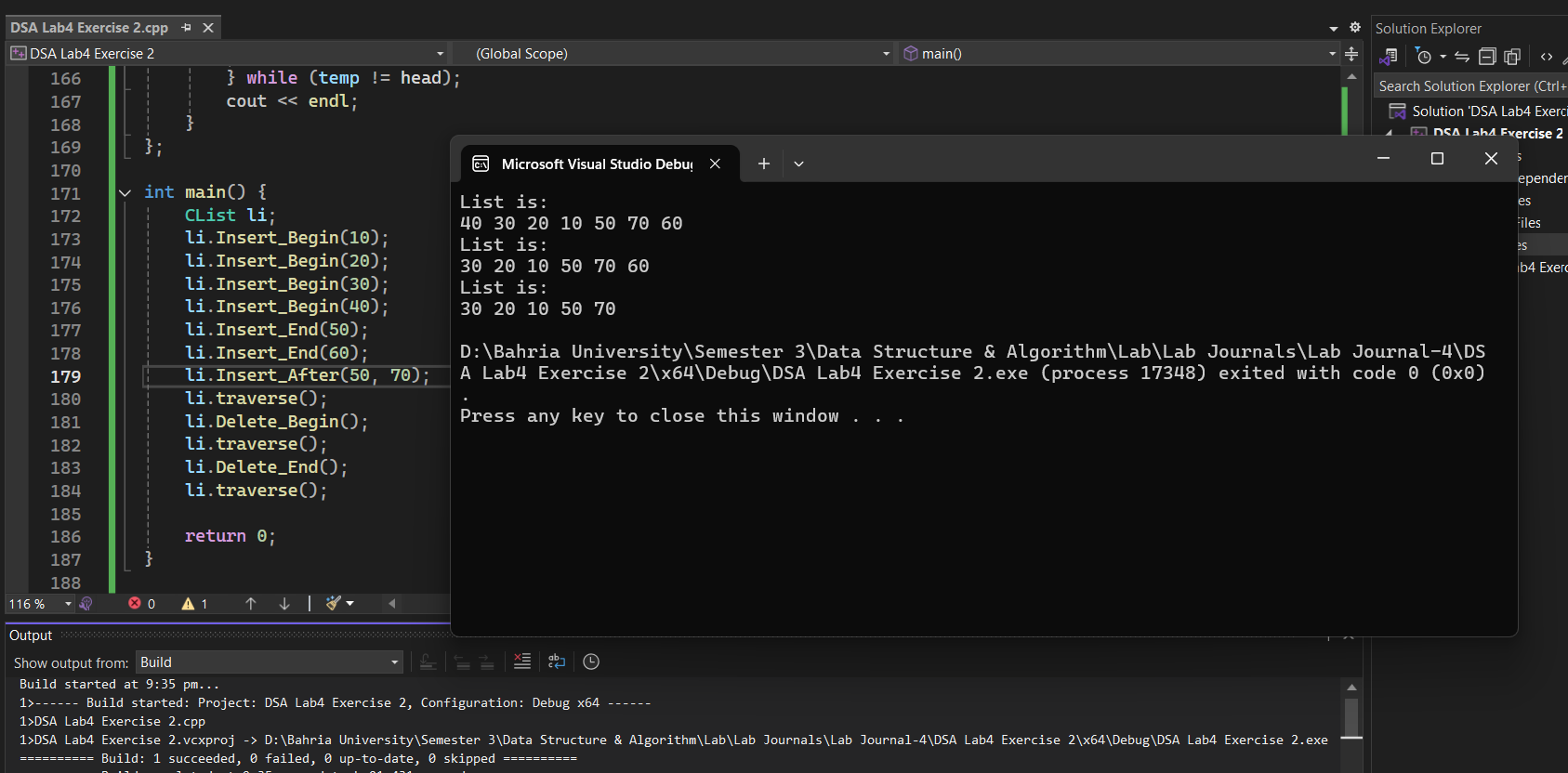
li.traverse();

li.Delete\_End();

li.traverse();

return 0;

}

****