**Bahria University, Lahore Campus**

Department of Computer Science

Lab Journal 04

**(Spring 2024)**

|  |  |  |
| --- | --- | --- |
| Course: | **Compiler Construction – Lab** | Date: 9-3-2024\_\_\_ |
| Course Code: | CSL 323 | Max Marks: 10 |
| Faculty’s Name: | Mr. M Mudassar |  |

Name: AFFAN AHMAD\_\_\_ Enroll No: \_03-134221-003\_\_ Class: \_BS(CS)5-A\_\_\_

Objective(s):

Upon completion of this lab session, learners will be able to:

* The objective of this exercise is to get you to write, compile, and run singly link list program in C++ from the scratch by defining a header file for the insertion, searching, and deletion of nodes in the list.

Lab Tasks:

Your lab report is expected to contain the following for each exercise:

* C++ Source Code (any file)
* Screenshot of your output (optional)

## Task 1:

Write a C++ program that implement the concept of singly link list in such a way that it ask user to add a new node in the list either in the beginning, end, or after any certain key node. Same is required for searching and deleting a node. All the functions related to insertion, searching, and deletion of node must be defined in a header file and that header file would be added in the main program which asks user either s/he wants to insert, search, or delete a node, or exit the program. Insertion and deletion would performed according to users interest either at start, end, or at aby specific location.

**PROGRAM:**

**LinkedList.h Header File:**

#ifndef LINKEDLIST\_H

#define LINKEDLIST\_H

struct Node {

int data;

Node\* next;

};

class LinkedList {

private:

Node\* head;

public:

LinkedList();

~LinkedList();

void insertAtBeginning(int value);

void insertAtEnd(int value);

void insertAfterKey(int key, int value);

void search(int value);

void deleteNode(int value);

void displayList();

};

#endif // LINKEDLIST\_H

**LinkedList.cpp Implementation File:**

#include "LinkedList.h"

#include <iostream>

using namespace std;

LinkedList::LinkedList() : head(nullptr) {}

LinkedList::~LinkedList() {

Node\* current = head;

Node\* next;

while (current != nullptr) {

next = current->next;

delete current;

current = next;

}

}

void LinkedList::insertAtBeginning(int value) {

Node\* newNode = new Node{value, head};

head = newNode;

}

void LinkedList::insertAtEnd(int value) {

Node\* newNode = new Node{value, nullptr};

if (head == nullptr) {

head = newNode;

return;

}

Node\* current = head;

while (current->next != nullptr) {

current = current->next;

}

current->next = newNode;

}

void LinkedList::insertAfterKey(int key, int value) {

Node\* current = head;

while (current != nullptr && current->data != key) {

current = current->next;

}

if (current != nullptr) {

Node\* newNode = new Node{value, current->next};

current->next = newNode;

} else {

std::cout << "Key not found in the list." << std::endl;

}

}

void LinkedList::search(int value) {

Node\* current = head;

int position = 1;

while (current != nullptr && current->data != value) {

current = current->next;

position++;

}

if (current != nullptr) {

std::cout << "Value " << value << " found at position " << position << std::endl;

} else {

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

}

}

void LinkedList::deleteNode(int value) {

Node\* current = head;

Node\* prev = nullptr;

while (current != nullptr && current->data != value) {

prev = current;

current = current->next;

}

if (current != nullptr) {

if (prev == nullptr) {

head = current->next;

} else {

prev->next = current->next;

}

delete current;

cout << "Node with value " << value << " deleted." << std::endl;

} else {

cout << "Value " << value << " not found in the list. Deletion failed." << std::endl;

}

}

void LinkedList::displayList() {

Node\* current = head;

while (current != nullptr) {

cout << current->data << " ";

current = current->next;

}

cout << endl;

}

**main.cpp Main Program:**

#include <iostream>

using namespace std;

#include "LinkedList.h"

int main() {

LinkedList list;

while (true) {

cout << "\nChoose an operation:\n"

<< "1. Insert a node\n"

<< "2. Search a node\n"

<< "3. Delete a node\n"

<< "4. Display the list\n"

<< "5. Exit\n";

int choice;

cin >> choice;

switch (choice) {

case 1: {

cout << "Choose insertion type:\n"

<< "1. At the beginning\n"

<< "2. At the end\n"

<< "3. After a specific key\n";

int insertType;

cin >> insertType;

int value, key;

switch (insertType) {

case 1:

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

cin >> value;

list.insertAtBeginning(value);

break;

case 2:

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

cin >> value;

list.insertAtEnd(value);

break;

case 3:

cout << "Enter the key after which to insert: ";

cin >> key;

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

cin >> value;

list.insertAfterKey(key, value);

break;

default:

std::cout << "Invalid choice.\n";

break;

}

break;

}

case 2: {

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

int value;

cin >> value;

list.search(value);

break;

}

case 3: {

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

int value;

cin >> value;

list.deleteNode(value);

break;

}

case 4:

cout << "Current list: ";

list.displayList();

break;

case 5:

cout << "Exiting the program.\n";

return 0;

default:

cout << "Invalid choice.\n";

break;

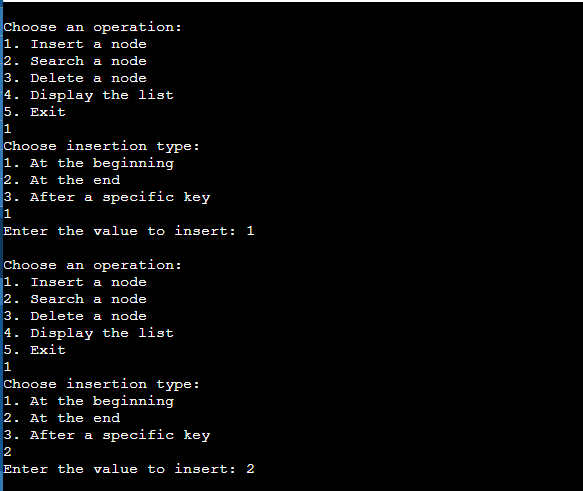
}

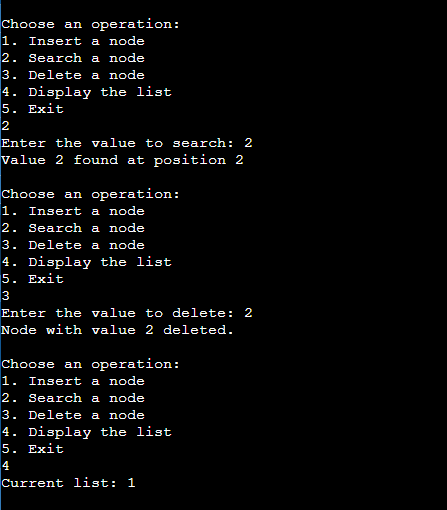
}

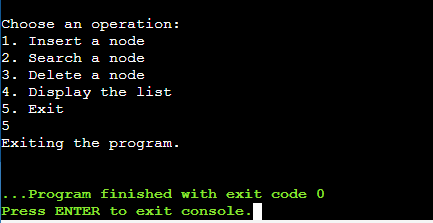
return 0;

}

**OUTPUT:**







**Lab Grading Sheet :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Max Marks** | **Obtained Marks** | **Comments(*if any*)** |
| 1. | 10 |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
| **Total** | **10** |  | **Signature** |

**Note: Attempt all tasks and get them checked by your Lab Instructor.**