**FINAL ASSINGMENT**

|  |  |
| --- | --- |
| **TOPIC** | **PAGE NO:** |
| **DOUBLE LINK LIST** | **1** |
| **CIRCULAR LINK LIST** | **21** |
| **BINARY SEARCH TREE** | **11** |

**Doubly Linked List**

**Write a program to delete the first node in a doubly linked list**

#include <iostream>

using namespace std;

**OUTPUT:**

Original List: 1 2 3

List after deletion: 2 3

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

Process exited after 0.2311 seconds with return value 0

struct Node {

int data;

Node\* prev;

Node\* next;

};

void deleteFirstNode(Node\*\* head) {

if (\*head) {

Node\* temp = \*head;

\*head = (\*head)->next;

if (\*head) (\*head)->prev = NULL;

delete temp;

}

}

void printList(Node\* head) {

while (head) {

cout << head->data << " ";

head = head->next;

}

cout << endl;

}

int main() {

Node\* head = new Node{1, NULL, new Node{2, NULL, new Node{3, NULL, NULL}}};

head->next->prev = head;

head->next->next->prev = head->next;

cout << "Original List: ";

printList(head);

deleteFirstNode(&head);

cout << "List after deletion: ";

printList(head);

return 0;

}

**How can you delete the last node in a doubly linked list? Write the code.**

#include <iostream>

**OUTPUT:**  
Original List: 1 2 3

List after deletion: 1 2

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

Process exited after 0.1862 seconds with return value 0

using namespace std;

struct Node {

int data;

Node\* prev;

Node\* next;

};

void deleteLastNode(Node\*\* head) {

if (\*head == NULL) return;

if ((\*head)->next == NULL) {

delete \*head;

\*head = NULL;

} else {

Node\* temp = \*head;

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

temp = temp->next;

}

delete temp->next;

temp->next = NULL;

}

}

void printList(Node\* head) {

while (head != NULL) {

cout << head->data << " ";

head = head->next;

}

cout << endl;

}

int main() {

Node\* head = new Node{1, NULL, new Node{2, NULL, new Node{3, NULL, NULL}}};

head->next->prev = head;

head->next->next->prev = head->next;

cout << "Original List: ";

printList(head);

deleteLastNode(&head);

cout << "List after deletion: ";

printList(head);

return 0;

}

**Write code to delete a node by its value in a doubly linked list**

#include <iostream>

using namespace std;

**OUTPUT:**

Original List: 1 2 3

List after deletion: 1 3

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

Process exited after 0.2046 seconds with return value 0

struct Node {

int data;

Node\* prev;

Node\* next;

};

void deleteNode(Node\*\* head, int value) {

if (\*head == NULL) return;

if ((\*head)->data == value) {

Node\* temp = \*head;

\*head = (\*head)->next;

if (\*head) (\*head)->prev = NULL;

delete temp;

return;

}

Node\* temp = \*head;

while (temp->next) {

if (temp->next->data == value) {

Node\* nodeToDelete = temp->next;

temp->next = nodeToDelete->next;

if (nodeToDelete->next) nodeToDelete->next->prev = temp;

delete nodeToDelete;

return;

}

temp = temp->next;

void printList(Node\* head) {

while (head) {

cout << head->data << " ";

head = head->next;

}

cout << endl;

}

int main() {

Node\* head = new Node{1, NULL, new Node{2, NULL, new Node{3, NULL, NULL}}};

head->next->prev = head;

head->next->next->prev = head->next;

cout << "Original List: ";

printList(head);

deleteNode(&head, 2);

cout << "List after deletion: ";

printList(head);

return 0;

}

**How would you delete a node at a specific position in a doubly linked list? Show it in code.**

#include <iostream>

**OUTPUT:**

Original List: 1 2 3 4

List after deletion: 1 2 4

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

Process exited after 0.2311 seconds with return value 0

using namespace std;

struct Node {

int data;

Node\* prev;

Node\* next;

};

void deleteAtPosition(Node\*\* head, int position) {

if (\*head == NULL || position < 0) return;

if (position == 0) {

Node\* temp = \*head;

\*head = (\*head)->next;

if (\*head) (\*head)->prev = NULL;

delete temp;

return;

}

Node\* temp = \*head;

for (int i = 0; i < position - 1 && temp->next; i++) {

temp = temp->next;

}

if (temp->next) {

Node\* nodeToDelete = temp->next;

temp->next = nodeToDelete->next;

if (nodeToDelete->next) nodeToDelete->next->prev = temp;

delete nodeToDelete;

}

}

void printList(Node\* head) {

while (head) {

cout << head->data << " ";

head = head->next;

}

cout << endl;

}

int main() {

Node\* head = new Node{1, NULL, new Node{2, NULL, new Node{3, NULL, new Node{4, NULL, NULL}}}};

head->next->prev = head;

head->next->next->prev = head->next;

head->next->next->next->prev = head->next->next;

cout << "Original List: ";

printList(head);

deleteAtPosition(&head, 2);

cout << "List after deletion: ";

printList(head);

return 0;

**After deleting a node, how will you write the forward and reverse traversal functions?**

#include <iostream>

**OUTPUT:**

Original List: 1 2 3 4

List after deletion: 1 2 4

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

Process exited after 0.2154 seconds with return value 0

using namespace std;

struct Node {

int data;

Node\* prev;

Node\* next;

};

// Forward Traversal

void forwardTraversal(Node\* head) {

while (head != NULL) {

cout << head->data << " ";

head = head->next;

}

cout << endl;

}

// Reverse Traversal

void reverseTraversal(Node\* head) {

if (head == NULL) return;

Node\* temp = head;

while (temp->next != NULL) {

temp = temp->next;

}

while (temp != NULL) {

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

temp = temp->prev;

}

cout << endl;

}

int main() {

// Create a doubly linked list

Node\* head = new Node{1, NULL, new Node{2, NULL, new Node{3, NULL, new Node{4, NULL, NULL}}}};

head->next->prev = head;

head->next->next->prev = head->next;

head->next->next->next->prev = head->next->next;

cout << "Forward Traversal: ";

forwardTraversal(head);

cout << "Reverse Traversal: ";

reverseTraversal(head);

return 0;

}

**Circular Linked List**

**• Write a program to delete the first node in a circular linked list.**

#include <iostream>

using namespace std;

OUTPUT:

Original List: 1 3

List after deletion: 3

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

Process exited after 0.1993 seconds with return value 0

struct Node {

int data;

Node\* next;

};

void deleteFirstNode(Node\*\* head) {

if (\*head == NULL) return;

if ((\*head)->next == \*head) {

delete \*head;

\*head = NULL;

} else {

Node\* temp = \*head;

while (temp->next != \*head) {

temp = temp->next;

}

Node\* firstNode = \*head;

\*head = (\*head)->next;

temp->next = \*head;

delete firstNode;

}

void printList(Node\* head) {

if (head == NULL) return;

Node\* temp = head;

do {

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

temp = temp->next;

} while (temp != head);

cout << endl;

}

int main() {

Node\* head = new Node{1, NULL};

head->next = new Node{2, NULL};

head->next->next = new Node{3, head};

head->next = head->next->next;

cout << "Original List: ";

printList(head);

deleteFirstNode(&head);

cout << "List after deletion: ";

printList(head);

return 0;

}

**How can you delete the last node in a circular linked list? Write the code**

#include <iostream>

**OUTPUT:**

Original List: 1 2 3

List after deletion: 1 2

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

Process exited after 0.2182 seconds with return value 0

using namespace std;

struct Node {

int data;

Node\* next;

};

void deleteLastNode(Node\*\* head) {

if (\*head == NULL) return;

if ((\*head)->next == \*head) {

delete \*head;

\*head = NULL;

} else {

Node\* temp = \*head;

while (temp->next->next != \*head) {

temp = temp->next;

}

delete temp->next;

temp->next = \*head;

}

}

void printList(Node\* head) {

Node\* temp = head;

do {

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

temp = temp->next;

} while (temp != head);

cout << endl;

}

int main() {

Node\* head = new Node{1, NULL};

head->next = new Node{2, NULL};

head->next->next = new Node{3, head};

cout << "Original List: ";

printList(head);

deleteLastNode(&head);

cout << "List after deletion: ";

printList(head);

return 0;

}

**Write a function to delete a node by its value in a circular linked list.**

#include <iostream>

using namespace std;

**OUTPUT:**

Original List: 1 2 3

List after deletion: 1 2

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

Process exited after 0.2043 seconds with return value 0

struct Node {

int data;

Node\* next;

};

void deleteNode(Node\*\* head, int value) {

if (\*head == NULL) return;

if ((\*head)->data == value) {

if ((\*head)->next == \*head) {

delete \*head;

\*head = NULL;

} else {

Node\* temp = \*head;

while (temp->next != \*head) {

temp = temp->next;

}

temp->next = (\*head)->next;

delete \*head;

\*head = temp->next;

}

return;

Node\* temp = \*head;

while (temp->next != \*head) {

if (temp->next->data == value) {

Node\* nodeToDelete = temp->next;

temp->next = nodeToDelete->next;

delete nodeToDelete;

return;

}

temp = temp->next;

}

void printList(Node\* head) {

Node\* temp = head;

do {

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

temp = temp->next;

} while (temp != head);

cout << endl;

}

int main() {

Node\* head = new Node{1, NULL};

head->next = new Node{2, NULL};

head->next->next = new Node{3, head};

cout << "Original List: ";

printList(head);

deleteNode(&head, 2);

cout << "List after deletion: ";

printList(head);

return 0;

}

**How will you delete a node at a specific position in a circular linked list? Write code for it.**

#include <iostream>

**OUTPUT:**

Original List: 1 2 3

List after deletion: 1 3

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

Process exited after 0.3457 seconds with return value 0

using namespace std;

struct Node {

int data;

Node\* next;

};

void deleteAtPosition(Node\*\* head, int position) {

if (\*head == NULL) return;

if (position == 0) {

if ((\*head)->next == \*head) {

delete \*head;

\*head = NULL;

} else {

Node\* temp = \*head;

while (temp->next != \*head) {

temp = temp->next;

}

temp->next = (\*head)->next;

delete \*head;

\*head = temp->next;

}

return;

}

Node\* temp = \*head;

for (int i = 0; i < position - 1 && temp->next != \*head; i++) {

temp = temp->next;

}

if (temp->next != \*head) {

Node\* nodeToDelete = temp->next;

temp->next = nodeToDelete->next;

delete nodeToDelete;

}

void printList(Node\* head) {

Node\* temp = head;

do {

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

temp = temp->next;

} while (temp != head);

cout << endl;

}

int main() {

Node\* head = new Node{1, NULL};

head->next = new Node{2, NULL};

head->next->next = new Node{3, head};

cout << "Original List: ";

printList(head);

deleteAtPosition(&head, 1);

cout << "List after deletion: ";

printList(head);

return 0;

}

**Write a program to show forward traversal after deleting a node in a circular linked list.**

#include <iostream>

using namespace std;

**OUTPUT:**

Original List: 1 2 3

List after deletion: 1 3

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

Process exited after 0.2279 seconds with return value 0

struct Node {

int data;

Node\* next;

};

void deleteNode(Node\*\* head, int value) {

if (\*head == NULL) return;

if ((\*head)->data == value) {

Node\* temp = \*head;

while (temp->next != \*head) {

temp = temp->next;

}

temp->next = (\*head)->next;

delete \*head;

\*head = temp->next;

return;

}

Node\* temp = \*head;

while (temp->next != \*head) {

if (temp->next->data == value) {

Node\* nodeToDelete = temp->next;

temp->next = nodeToDelete->next;

delete nodeToDelete;

return;

}

temp = temp->next;

void forwardTraversal(Node\* head) {

if (head == NULL) return;

Node\* temp = head;

do {

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

temp = temp->next;

} while (temp != head);

cout << endl;

}

int main() {

Node\* head = new Node{1, NULL};

head->next = new Node{2, NULL};

head->next->next = new Node{3, head};

cout << "Original List: ";

forwardTraversal(head);

deleteNode(&head, 2);

cout << "List after deletion: ";

forwardTraversal(head);

return 0;

}

**Binary Search Tree**

**• Write a program to count all the nodes in a binary search tree.**

#include <iostream>

**OUTPUT:**

Number of nodes: 5

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

Process exited after 0.196 seconds with return value 0

using namespace std;

struct Node {

int data;

Node\* left;

Node\* right;

};

int countNodes(Node\* root) {

if (root == NULL) return 0;

return 1 + countNodes(root->left) + countNodes(root->right);

}

int main() {

Node\* root = new Node{8};

root->left = new Node{3};

root->right = new Node{10};

root->left->left = new Node{1};

root->left->right = new Node{6};

cout << "Number of nodes: " << countNodes(root) << endl;

return 0;

}

**How can you search for a specific value in a binary search tree? Write the code.**

#include <iostream>

using namespace std;

**OUTPUT:**

Value 6 found in the tree.

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

Process exited after 0.1943 seconds with return value 0

struct Node {

int data;

Node\* left;

Node\* right;

};

Node\* search(Node\* root, int value) {

if (root == NULL || root->data == value) return root;

if (value < root->data) return search(root->left, value);

return search(root->right, value);

}

int main() {

Node\* root = new Node{8};

root->left = new Node{3};

root->right = new Node{10};

root->left->left = new Node{1};

root->left->right = new Node{6};

int value = 6;

Node\* result = search(root, value);

if (result != NULL) {

cout << "Value " << value << " found in the tree." << endl;

} else {

cout << "Value " << value << " not found in the tree." << endl;

}

return 0;

**Write code to traverse a binary search tree in in-order, pre-order, and postorder.**

#include <iostream>

OUTPUT:

In-order traversal: 1 3 6 8 10

Pre-order traversal: 8 3 1 6 10

Post-order traversal: 1 6 3 10 8

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

Process exited after 0.2508 seconds with return value 0

using namespace std;

struct Node {

int data;

Node\* left;

Node\* right;

};

// In-order traversal

void inOrder(Node\* root) {

if (root) {

inOrder(root->left);

cout << root->data << " ";

inOrder(root->right);

}

}

// Pre-order traversal

void preOrder(Node\* root) {

if (root) {

cout << root->data << " ";

preOrder(root->left);

preOrder(root->right);

}

// Post-order traversal

void postOrder(Node\* root) {

if (root) {

postOrder(root->left);

postOrder(root->right);

cout << root->data << " ";

}

int main() {

Node\* root = new Node{8};

root->left = new Node{3};

root->right = new Node{10};

root->left->left = new Node{1};

root->left->right = new Node{6};

cout << "In-order traversal: ";

inOrder(root);

cout << endl;

cout << "Pre-order traversal: ";

preOrder(root);

cout << endl;

cout << "Post-order traversal: ";

postOrder(root);

cout << endl;

return 0;

}

**How will you write reverse in-order traversal for a binary search tree? Show it in code.**

#include <iostream>

**OUTPUT:**

Reverse in-order traversal: 10 8 6 3 1

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

Process exited after 0.2034 seconds with return value 0

using namespace std;

struct Node {

int data;

Node\* left;

Node\* right;

};

void reverseInOrder(Node\* root) {

if (root) {

reverseInOrder(root->right);

cout << root->data << " ";

reverseInOrder(root->left);

}

int main() {

Node\* root = new Node{8};

root->left = new Node{3};

root->right = new Node{10};

root->left->left = new Node{1};

root->left->right = new Node{6};

cout << "Reverse in-order traversal: ";

reverseInOrder(root);

cout << endl;

return 0;

}

**Write a program to check if there are duplicate values in a binary search tree.**

#include <iostream>

**OUTPUT:**

Duplicate value found: 8

Duplicate value found: 3

Duplicate value found: 1

Duplicate value found: 3

Duplicate value found: 10

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

Process exited after 0.2005 seconds with return value 0

using namespace std;

struct Node {

int data;

Node\* left;

Node\* right;

};

bool checkDuplicates(Node\* root, int value) {

if (root == NULL) return false;

if (root->data == value) return true;

return checkDuplicates(root->left, value) || checkDuplicates(root->right, value);

}

void traverse(Node\* root) {

if (root == NULL) return;

if (checkDuplicates(root, root->data)) {

cout << "Duplicate value found: " << root->data << endl;

}

traverse(root->left);

traverse(root->right);

int main() {

Node\* root = new Node{8};

root->left = new Node{3};

root->right = new Node{10};

root->left->left = new Node{1};

root->left->right = new Node{3}; // duplicate value

traverse(root);

**How can you delete a node from a binary search tree? Write code for deleting a leaf, a node with one child, and a node with two children.**

#include <iostream>

**OUTPUT:**

Original tree: 1 3 6 8 10

Tree after deletion: 1 3 8 10

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

Process exited after 0.1953 seconds with return value 0

using namespace std;

struct Node {

int data;

Node\* left;

Node\* right;

};

Node\* deleteNode(Node\* root, int value) {

if (root == NULL) return root;

if (value < root->data) root->left = deleteNode(root->left, value);

else if (value > root->data) root->right = deleteNode(root->right, value);

else {

if (root->left == NULL) return root->right;

else if (root->right == NULL) return root->left;

Node\* temp = root->right;

while (temp->left) temp = temp->left;

root->data = temp->data;

root->right = deleteNode(root->right, temp->data);

}

return root;

}

void printInorder(Node\* root) {

if (root) {

printInorder(root->left);

cout << root->data << " ";

printInorder(root->right);

}

}

int main() {

Node\* root = new Node{8};

root->left = new Node{3};

root->right = new Node{10};

root->left->left = new Node{1};

root->left->right = new Node{6};

cout << "Original tree: ";

printInorder(root);

cout << endl;

root = deleteNode(root, 6);

cout << "Tree after deletion: ";

printInorder(root);

cout << endl;

return 0;

}