Operations on Linkedlist

# Course title

Data Structure & Algorithm

# Course code

IT-209

**Submitted to** Prof. Azib Mahmood **Submitted by** Hadiqa tul Zahra **Roll Number** 22011556-002

**Semester** Fall 23 **Section** IT-22-A

# Date: 30th October, 2023



**Basic Operations**

# Code:

#include <iostream>

using namespace std;

class node {

public:

    int value;

    node \*next;

    node(int data) {

        value = data;

        next = NULL;

    }

};

void inshead(node\* &head, int val) {

    node \*newNode = new node(val);

    newNode->next = head;

    head = newNode;

    cout << "Inserted " << val << " at the beginning.\n";

}

void display(node \*head) {

    cout << "Linked List: ";

    while (head != NULL) {

        cout << head->value << " -> ";

        head = head->next;

    }

    cout << "NULL\n";

}

void insend(node\* &head, int val) {

    node \*newNode = new node(val);

    if (head == NULL) {

        head = newNode;

    } else {

        node \*temp = head;

        while (temp->next != NULL) {

            temp = temp->next;

       }

        temp->next = newNode;

    }

    cout << "Inserted " << val << " at the end.\n";

}

void insn(node\* &head, int val, int position) {

    if (position == 1) {

        inshead(head, val);

    } else {

        node \*newNode = new node(val);

        node \*temp = head;

        for (int i = 1; i < position - 1 && temp != NULL; ++i) {

            temp = temp->next;

        }

        if (temp == NULL) {

            cout << "Position out of range. Cannot insert.\n";

        } else {

            newNode->next = temp->next;

            temp->next = newNode;

cout << "Inserted " << val << " at position " << position << ".\n";

        }

    }

}

void delbeg(node\* &head) {

    if (head == NULL) {

        cout << "List is empty. Nothing to delete.\n";

    } else {

        node \*temp = head;

        head = head->next;

        delete temp;

        cout << "Deleted from the beginning.\n";

    }

}

void delend(node\* &head) {

    if (head == NULL) {

        cout << "List is empty. Nothing to delete.\n";

    }

else if (head->next == NULL) {

        delete head;

        head = NULL;

        cout << "Deleted from the end.\n";

    }

else {

        node \*temp = head;

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

            temp = temp->next;

        }

        delete temp->next;

        temp->next = NULL;

        cout << "Deleted from the end.\n";

    }

}

void deln(node\* &head, int position) {

    if (position == 1) {

        delbeg(head);

    } else {

        node \*temp = head;

        for (int i = 1; i < position - 1 && temp != NULL; ++i) {

            temp = temp->next;

        }

        if (temp == NULL || temp->next == NULL) {

            cout << "Position out of range. Cannot delete.\n";

        } else {

            node \*toDelete = temp->next;

            temp->next = temp->next->next;

            delete toDelete;

            cout << "Deleted from position " << position << ".\n";}

    }

}

void search(node\* head, int val) {

    int position = 1;

    bool found = false;

    while (head != NULL) {

        if (head->value == val) {

            found = true;

            cout << "Value " << val << " found at position “<<position << ". Enter new value: ";

            int newVal;

            cin >> newVal;

            head->value = newVal;

            cout << "Value updated at position " << position<".\n";

            break;

}

        head = head->next;

        position++;

}

    if (!found) {

        cout << "Value " << val << " not found in the list.\n";

    }

}

void update(node\* head, int position) {

    int count = 1;

    int newVal;

    while (head != NULL) {

        if (count == position) {

            cout << "Enter new value: ";

            cin >> newVal;

            head->value = newVal;

            cout<< "Value updated at position "<< position<<".\n";

            return;

        }

        head = head->next;

        count++;

    }

    cout << "Position " << position << " not found in the list.\n";

}

int main() {

    int ch, val, pos;

    node \*head = NULL;

    do {

        cout << "\nChoose one of the following operations to be performed:";

        cout << "\n1.Insert at beginning\n2.Insert at end\n3.Insert at any position n";

        cout << "\n4.Display/traverse the list\n5.Delete from beginning\n6.Delete from end";

        cout << "\n7.Delete from any position n\n8.Update at any position";

        cout << "\n9.Search and Update at any position\n0.Exit\n";

        cin >> ch;

        switch (ch) {

            case 1:

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

                cin >> val;

                inshead(head, val);

                break;

            case 2:

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

                cin >> val;

                insend(head, val);

                break;

            case 3:

                cout << "Enter value to insert: ";

                cin >> val;

                cout << "Enter position to insert: ";

                cin >> pos;

                insn(head, val, pos);

                break;

            case 4:

                display(head);

                break;

            case 5:

                delbeg(head);

                break;

            case 6:

                delend(head);

                break;

            case 7:

                cout << "Enter position to delete: ";

                cin >> pos;

                deln(head, pos);

                break;

            case 8:

                cout << "Enter position to update: ";

                cin >> pos;

                update(head, pos);

                break;

            case 9:

                cout << "Enter value to search and update: ";

                cin >> val;

                search(head, val);

                break;

            case 0:

                exit(0);

            default:

                cout << "Invalid choice! Please try again.\n";

}

    } while (true);

return 0;

}

# Output:

Choose one of the following operations to be performed:

1.Insert at beginning

2.Insert at end

3.Insert at any position n

4.Display/traverse the list

5.Delete from beginning

6.Delete from end

7.Delete from any position n

8.Update at any position

9.Search and Update at any position

0.Exit

1

Enter value to insert at the beginning: 4

Inserted 4 at the beginning.

Choose one of the following operations to be performed:

1.Insert at beginning

2.Insert at end

3.Insert at any position n

4.Display/traverse the list

5.Delete from beginning

6.Delete from end

7.Delete from any position n

8.Update at any position

9.Search and Update at any position

0.Exit

3

Enter value to insert: 3

Enter position to insert: 2

Inserted 3 at position 2.

Choose one of the following operations to be performed:

1.Insert at beginning

2.Insert at end

3.Insert at any position n

4.Display/traverse the list

5.Delete from beginning

6.Delete from end

7.Delete from any position n

8.Update at any position

9.Search and Update at any position

0.Exit

3

Enter value to insert: 5

Enter position to insert: 3

Inserted 5 at position 3.

Choose one of the following operations to be performed:

1.Insert at beginning

2.Insert at end

3.Insert at any position n

4.Display/traverse the list

5.Delete from beginning

6.Delete from end

7.Delete from any position n

8.Update at any position

9.Search and Update at any position

0.Exit

2

Enter value to insert at the end: 8

Inserted 8 at the end.

Choose one of the following operations to be performed:

1.Insert at beginning

2.Insert at end

3.Insert at any position n

4.Display/traverse the list

5.Delete from beginning

6.Delete from end

7.Delete from any position n

8.Update at any position

9.Search and Update at any position

0.Exit

4

Linked List: 4 -> 3 -> 5 -> 8 -> NULL

Choose one of the following operations to be performed:

1.Insert at beginning

2.Insert at end

3.Insert at any position n

4.Display/traverse the list

5.Delete from beginning

6.Delete from end

7.Delete from any position n

8.Update at any position

9.Search and Update at any position

0.Exit

5

Deleted from the beginning.

Choose one of the following operations to be performed:

1.Insert at beginning

2.Insert at end

3.Insert at any position n

4.Display/traverse the list

5.Delete from beginning

6.Delete from end

7.Delete from any position n

8.Update at any position

9.Search and Update at any position

0.Exit

6

Deleted from the end.

Choose one of the following operations to be performed:

1.Insert at beginning

2.Insert at end

3.Insert at any position n

4.Display/traverse the list

5.Delete from beginning

6.Delete from end

7.Delete from any position n

8.Update at any position

9.Search and Update at any position

0.Exit

7

Enter position to delete: 2

Deleted from position 2.

Choose one of the following operations to be performed:

1.Insert at beginning

2.Insert at end

3.Insert at any position n

4.Display/traverse the list

5.Delete from beginning

6.Delete from end

7.Delete from any position n

8.Update at any position

9.Search and Update at any position

0.Exit

8

Enter position to update: 2

Position 2 not found in the list.

Choose one of the following operations to be performed:

1.Insert at beginning

2.Insert at end

3.Insert at any position n

4.Display/traverse the list

5.Delete from beginning

6.Delete from end

7.Delete from any position n

8.Update at any position

9.Search and Update at any position

0.Exit

9

Enter value to search and update: 5

Value 5 not found in the list.

Choose one of the following operations to be performed:

1.Insert at beginning

2.Insert at end

3.Insert at any position n

4.Display/traverse the list

5.Delete from beginning

6.Delete from end

7.Delete from any position n

8.Update at any position

9.Search and Update at any position

0.Exit

9

Enter value to search and update: 3

Value 3 found at position 1. Enter new value: 6

Value updated at position 1.

Choose one of the following operations to be performed:

1.Insert at beginning

2.Insert at end

3.Insert at any position n

4.Display/traverse the list

5.Delete from beginning

6.Delete from end

7.Delete from any position n

8.Update at any position

9.Search and Update at any position

0.Exit

8

Enter position to update: 1

Enter new value: 9

Value updated at position 1.

Choose one of the following operations to be performed:

1.Insert at beginning

2.Insert at end

3.Insert at any position n

4.Display/traverse the list

5.Delete from beginning

6.Delete from end

7.Delete from any position n

8.Update at any position

9.Search and Update at any position

0.Exit

4

Linked List: 9 -> NULL

Choose one of the following operations to be performed:

1.Insert at beginning

2.Insert at end

3.Insert at any position n

4.Display/traverse the list

5.Delete from beginning

6.Delete from end

7.Delete from any position n

8.Update at any position

9.Search and Update at any position

0.Exit

0