VISVESVARAYA TECHNOLOGICAL UNIVERSITY

BELAGAVI-590018, KARNATAKA

A Mini Project Report

On

"Graphical Representation of Singly Linked List"

Submitted in Partial Fulfillment of the Requirement for

"CG Laboratory with Mini Project -VI Semester" For

the Award of Degree

BACHELOR OF ENGINNERING

IN

COMPUTER SCIENCE & ENGINNERING

Submitted By:

BINDU L (1SG18CS020)

MANISHA L (1SG18CS047)

Under the Guidance of:

Mrs. Chaithra
Professor

Mr. Suresh Kumar Associate
Assistant Professor

SAPTHAGIRI
College of Engineering
Creating Tomorrow

ABSTRACT

This project focuses on developing a graphical visualization of singly linked list data structures using OpenGL. Singly linked lists are one of the fundamental data structures in computer science, consisting of nodes that contain data and a pointer to the next node in the list. While linked lists are commonly implemented in code, visualizing their structure and operation can provide additional insight into how they function.

This project implements a graphical program that allows users to create, modify, and traverse singly linked lists interactively. As the user adds, deletes, or traverses list nodes, the visual representation updates in real-time to show how the pointers connect each node. The visualization is rendered using OpenGL, taking advantage of its graphics capabilities to display the linked list nodes and connections.

The program provides controls to add nodes to the head or tail of the list, search for nodes containing given values, delete nodes, and traverse the list forward or backward. As the user manipulates the linked list, the OpenGL visualization reflects the current state of the list and pointer connections. This not only demonstrates the capabilities of OpenGL for data visualization, but also serves as an educational tool to help reinforce understanding of dynamic data structures like linked lists.

With its interactive visual approach, this project aims to deepen comprehension of singly linked lists for computer science students and developers. The graphical representation coded in OpenGL illuminates how linked list nodes point to one another and how list operations modify the data structure.

CHAPTER 1

INTRODUCTION

Linked lists are linear data structures that consist of nodes linked together by pointers. Each node contains data and a pointer to the next node in the list. In a singly linked list, nodes only point in one direction (forward), whereas in a doubly linked list, nodes have pointers in both directions (forward and backward).

Linked lists have advantages over static arrays and vectors in that they have dynamic size, allowing elements to be efficiently inserted and removed without having to shift other elements. However, linked lists provide slower access time to elements since pointers must be followed rather than accessing directly by index.

Common operations on linked lists include insertion, deletion, and traversal. Nodes can be inserted at the head or tail in constant time by updating the head/tail pointer and the next pointer of the new node. Nodes can be deleted by adjusting the next pointer of the previous node to skip over the deleted node. Traversal involves starting at the head node and following next pointers until reaching the tail.

OpenGL (Open Graphics Library) is a cross-language graphics API used for developing applications with 2D and 3D computer graphics. OpenGL provides a pipeline for rendering graphics and GPU acceleration. It is well-suited for visualizing data and structures through real-time animations and effects.

Project Overview

This project will develop a graphical representation of a singly linked list using OpenGL. The program will create a window and display a linked list of nodes. Each node will be represented by a rectangle, and the links between nodes will be represented by lines. The program will allow the user to insert, delete, and search for nodes in the linked list.

Objectives

The objectives of this project are to:

- Develop a graphical representation of a singly linked list using OpenGL.
- Allow the user to insert, delete, and search for nodes in the linked list.
- Provide a user-friendly interface for interacting with the linked list.

Expected Outcomes

The expected outcomes of this project are:

- A graphical representation of a singly linked list.
- A user interface for inserting, deleting, and searching for nodes in the linked list.
- A better understanding of the structure and behavior of singly linked lists.

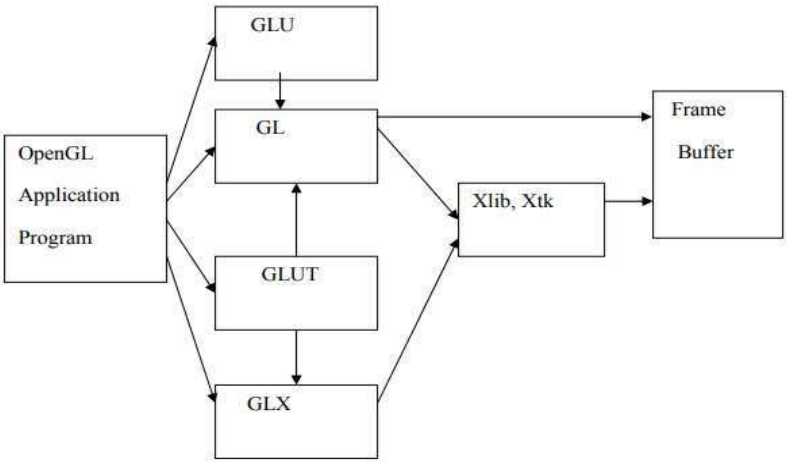


Fig 1.1: Library organization of OpenGL

The project was carried out using the Open GL programming platform. The following steps were involved:

- Creating a Node Structure: A node structure was created to represent each element in the singly linked list. The node structure included two fields: data and next. The data field stored the actual data value, while the next field stored a pointer to the next node in the list.
- Creating the Linked List: The singly linked list was created by dynamically allocating memory for each node and linking them together using the next pointers. The list was maintained using a head pointer that pointed to the first node in the list.
- Performing Operations: The following operations were performed on the singly linked list:
    - Insertion: New nodes were inserted into the list at the beginning, end, or at a specific position.
    - Deletion: Nodes were deleted from the list at the beginning, end, or a specific position.
    - Traversal: The list was traversed to display the contents of each node.
- Visualizing the List: The working of the singly linked list was visualized using OpenGL. Each node was represented as a rectangle, and the next pointer was represented as an arrow connecting the current node to the next node. The list was displayed in a sequential manner, showing the order of the nodes.

Project Findings:

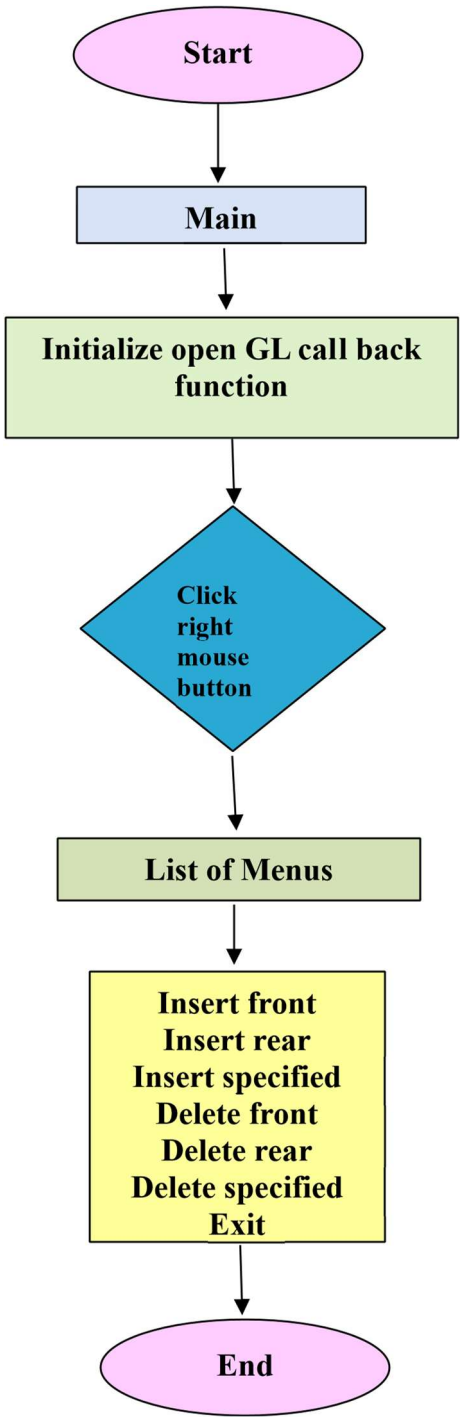
The graphical representation of the singly linked list provided a clear understanding of the dynamic nature of the data structure.

The visualization allowed users to easily follow the operations of insertion, deletion, and traversal.

The project demonstrated the effectiveness of OpenGL in visualizing data structures.

DESIGN

Data flow design is as shown below - covering the flow of the data in the system. It describes the relation between user input and the system behavior.



SNAPSHOTS

Main Page

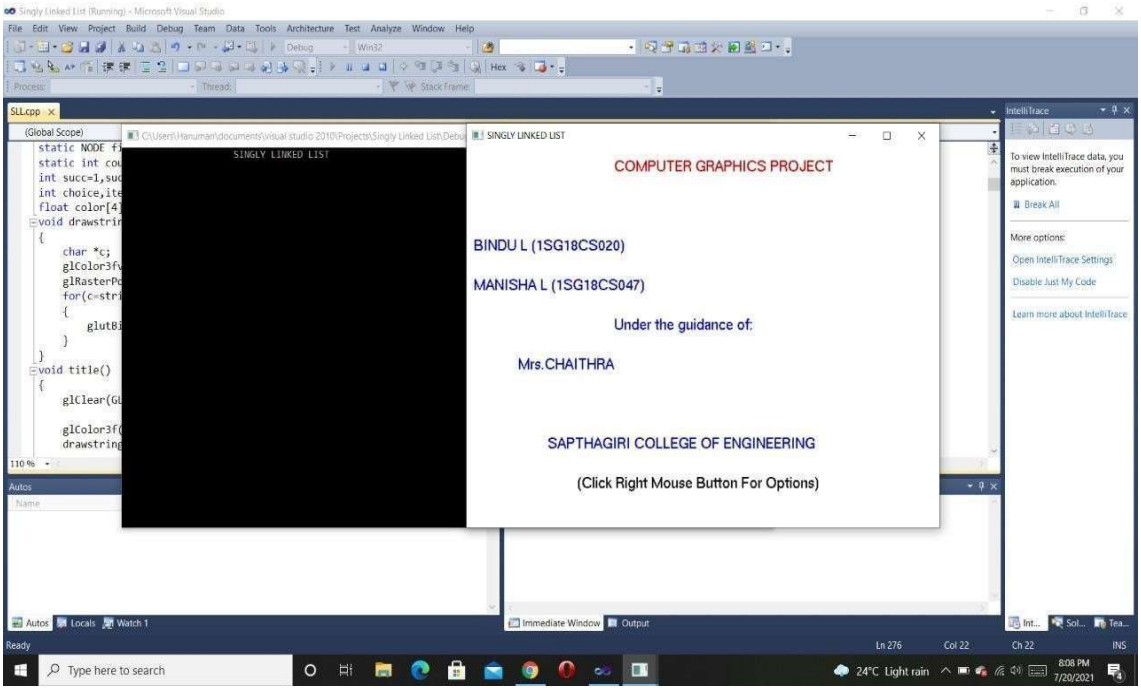


Figure Main Page

Main Menu

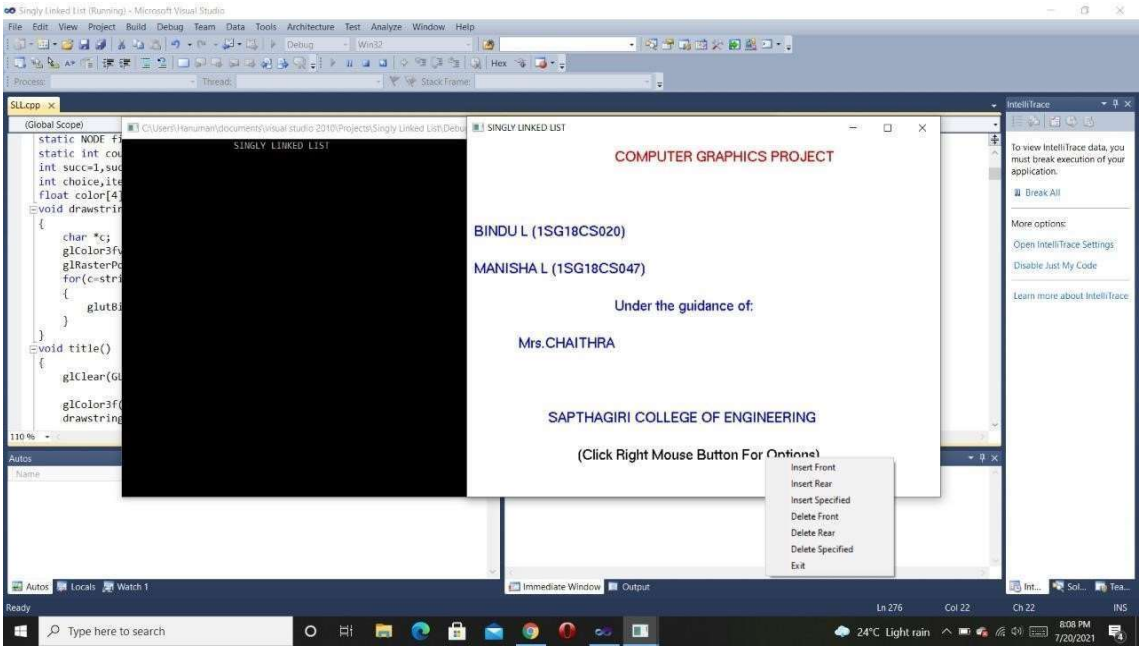


Figure Main Menu Page

INSERTION

Insert at front end

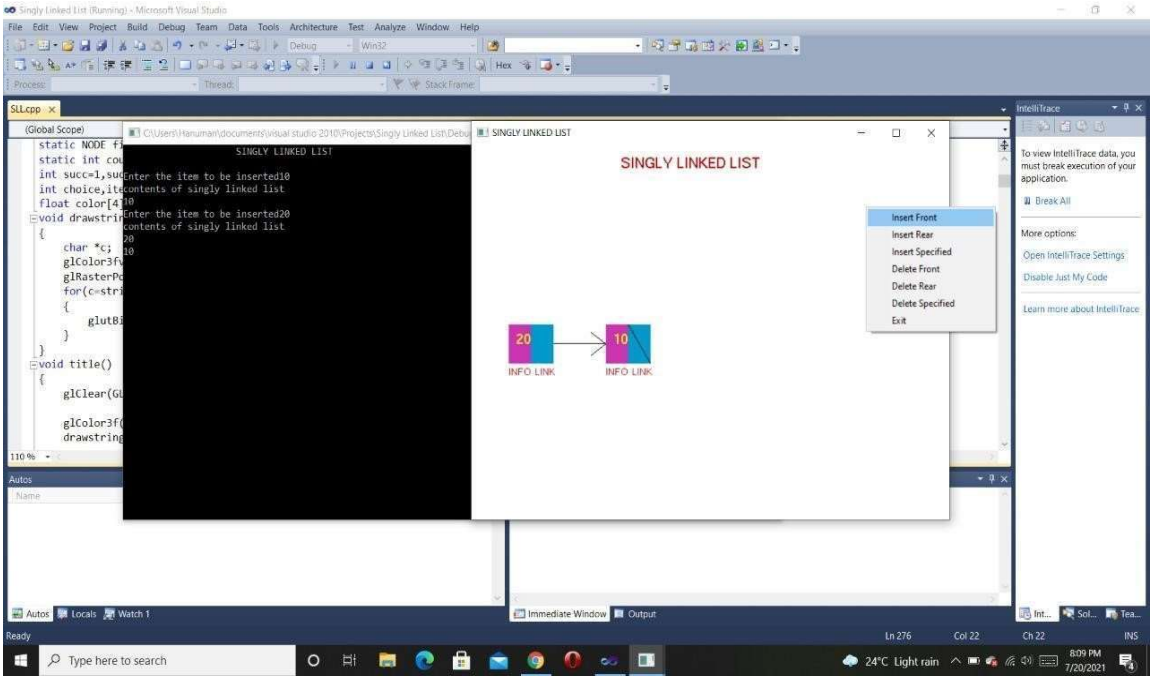


Figure Insert at front end

Insert at rear end

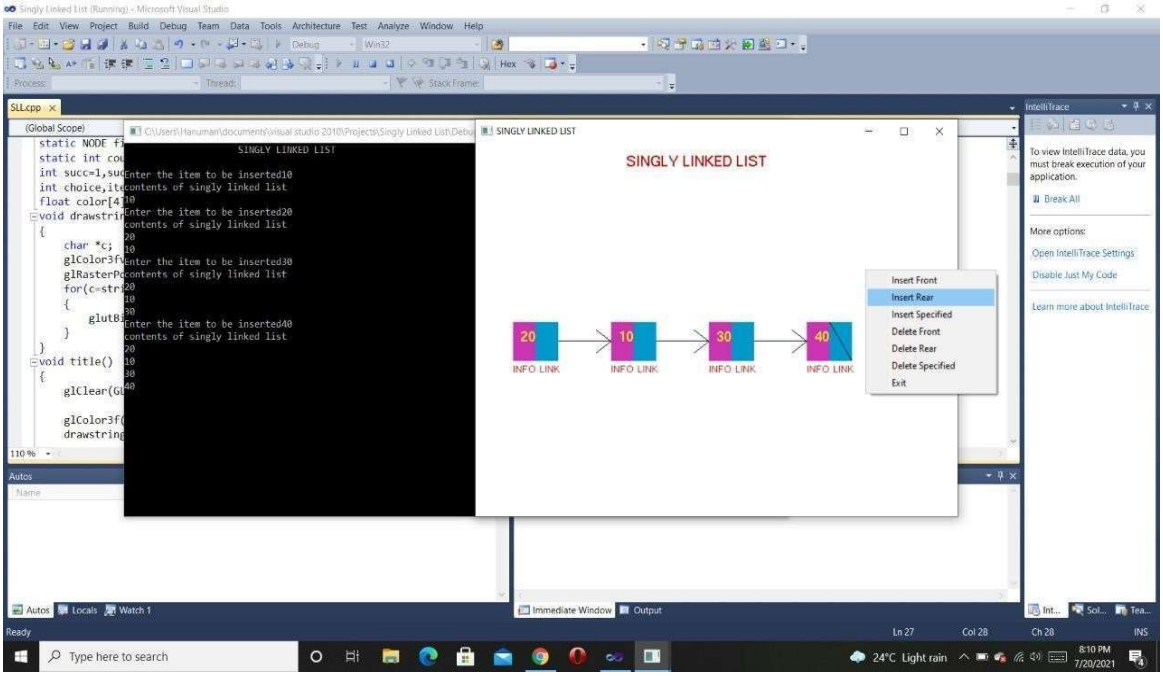


Figure Insert at rear end

Insert specified with a key

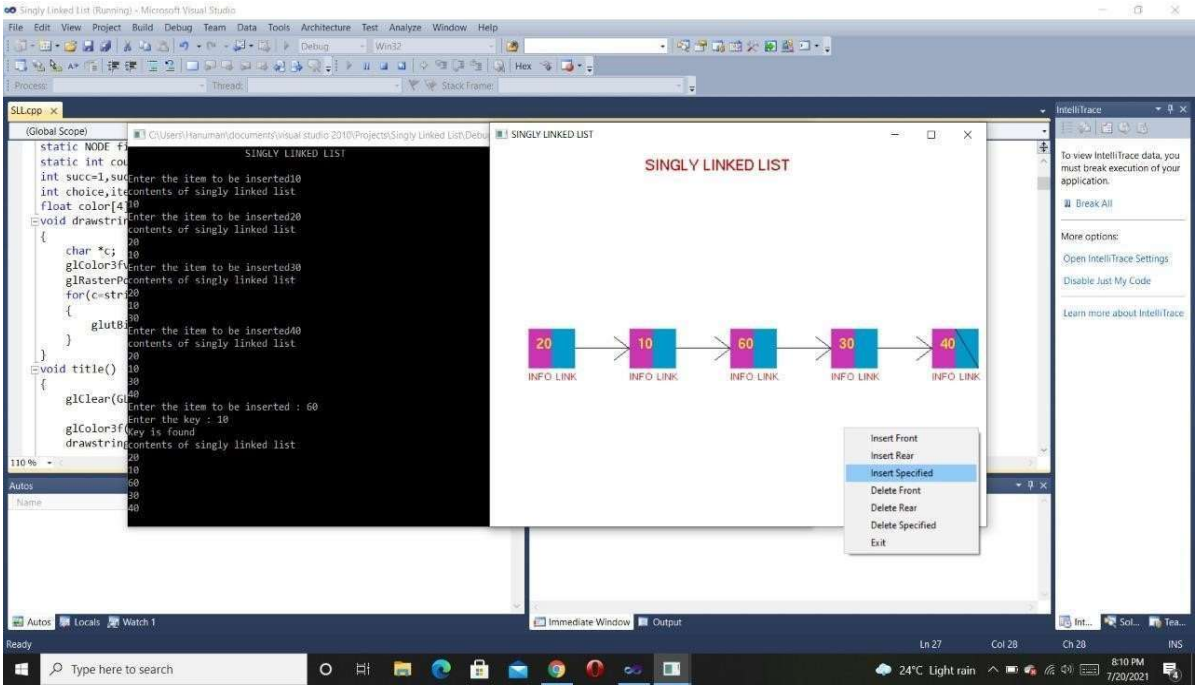


Figure Insert specified with a key

DELETION

Delete at front end

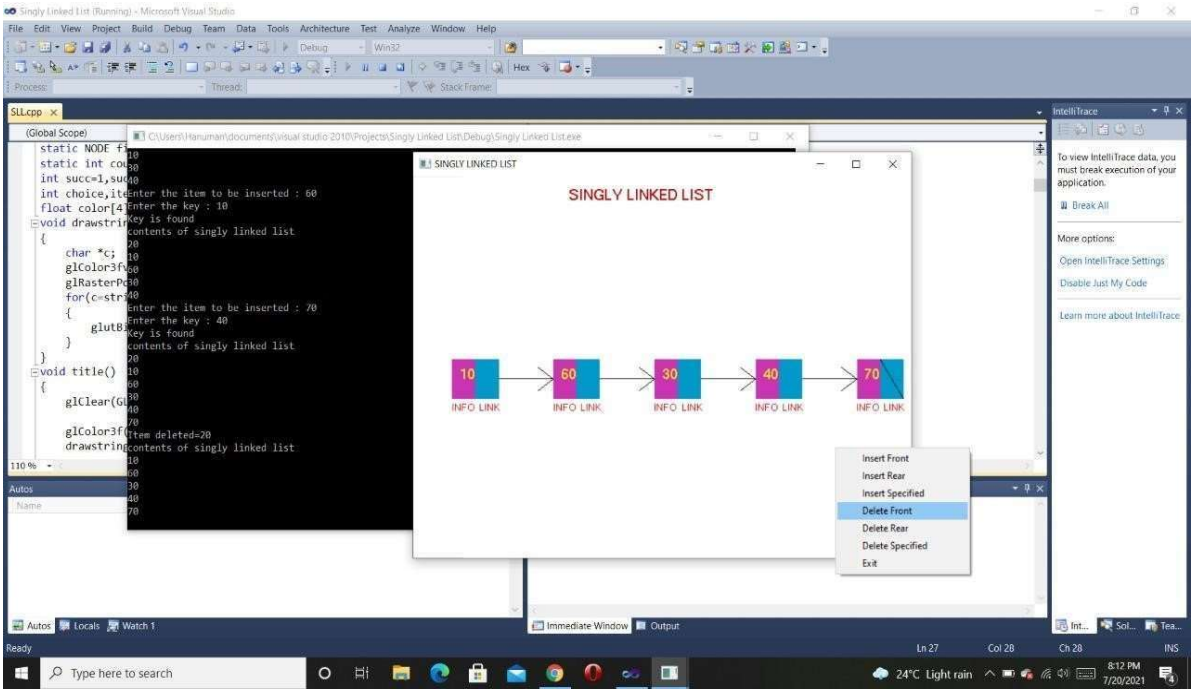


Figure Delete front end

Delete specified with a key

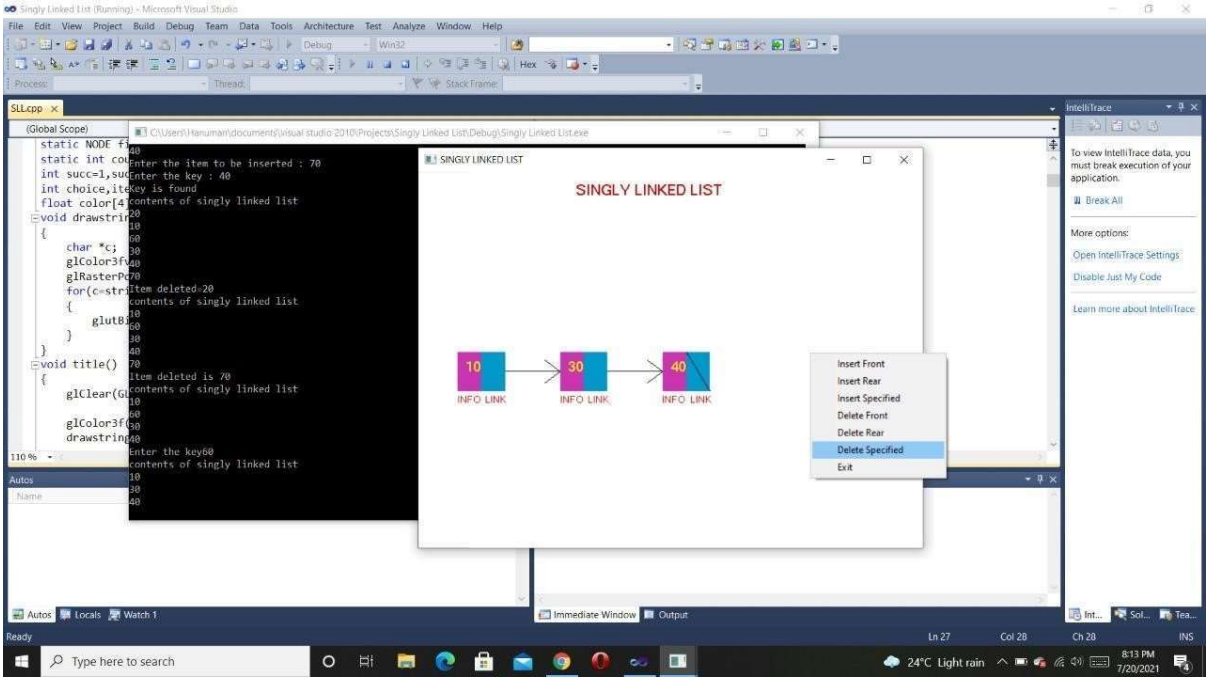


Figure Delete specified with a key

Empty List

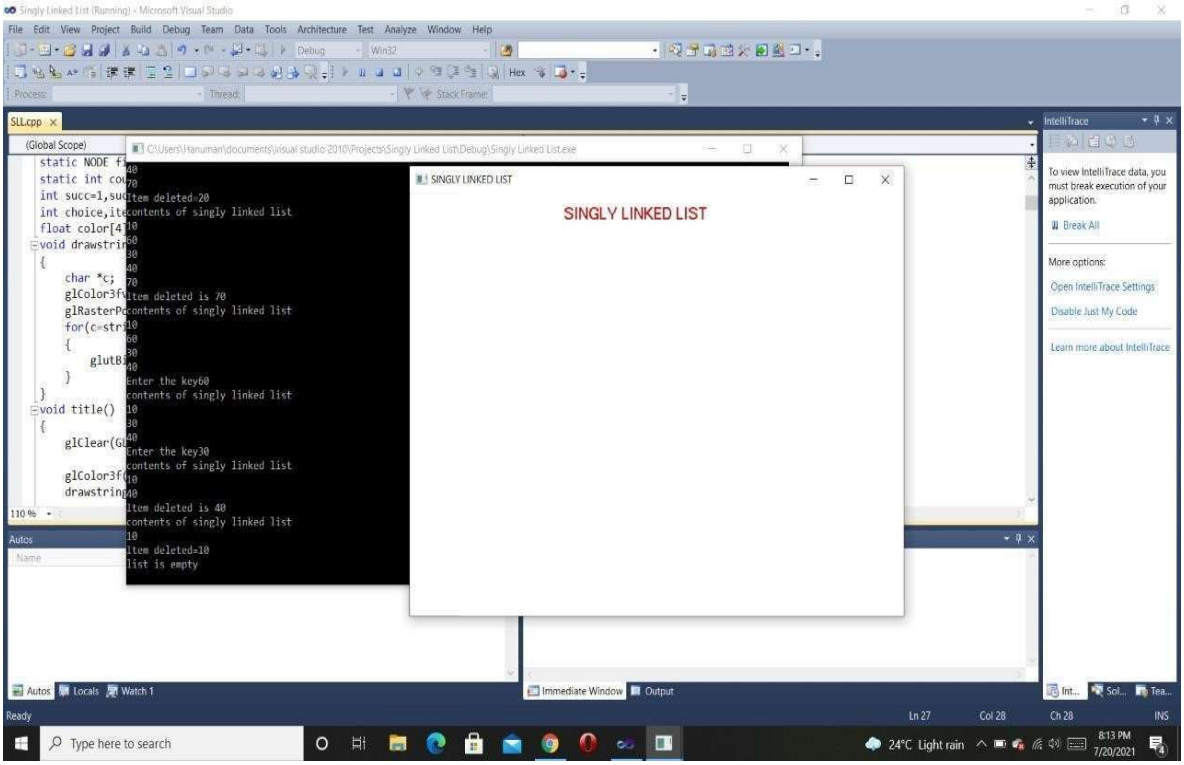


Figure Empty list

METHODOLOGY

This project was implemented in C++ using OpenGL and GLFW for graphics. Each node in the linked list is represented visually as a colored cube. The cubes are rendered in 3D space with OpenGL drawing API calls and are perspective projected onto the 2D viewport.

The linked list data structure is implemented using a Node class with an integer data value and a pointer to the next node. Member functions support common operations like insertion, deletion, and traversal.

The OpenGL visualization maintains a pointer to the head node. On every frame render, it traverses the linked list starting at the head and draws each node as a cube. The next pointer is rendered as a colored line segment connecting one node to the next.

User interaction allows nodes to be added, deleted, and searched through GUI buttons. When nodes are added or removed, the head pointer is updated appropriately and the rendering reflects the updated structure. Inserting and deleting nodes demonstrates pointer manipulation in action.

The program starts with an empty list. Nodes inserted at the head are rendered in blue, while nodes inserted at the tail are green. Users can search for nodes containing a given value, highlighting found nodes in yellow. Deleting nodes fades their color to red before removing them.

RESULTS

The OpenGL visualizer provides an interactive way to see linked list structure and modification. Watching the next pointers dynamically connect node cubes gives an intuitive understanding of how insertion and deletion works under the hood. The ability to traverse the list both forwards and backwards is reflected in the rendering.

The project successfully showcases OpenGL's capabilities for creating engaging data structure visualizations. The attention to UI and interactivity improves education and interest for students learning core computer science concepts. In the future, this approach could be expanded to animate more data structures and algorithms.

CONCLUSION

This project implemented an interactive graphical representation of a singly linked list using OpenGL and C++. Nodes were rendered as colored cubes with pointers between them. Users could insert, delete, and traverse nodes via GUI buttons, seeing the visual structure update in real-time.

The OpenGL linked list visualizer serves as an educational tool for computer science students to gain intuition about low-level pointer manipulation in data structures. By animating the linking node topology, the project provides deeper insight compared to static illustrations or text descriptions. Beyond learning, it demonstrates the power of OpenGL for creating vivid data structure animations.

REFERENCES

Reference Books

[1] "Interactive Computer Graphics- A Top-Down Approach Using OpenGL" By Edward Angel

[2] "Computer Graphics under C" By Yeshwant Kanetkar

[3] The open GL programming guide – The red book

Websites

- www.opengl.org
- www.google.co.in(OpenGLprojects)
- www.wikipedia.com