

Stack Text Editor

Submitted To Usman Ghani

Submitted By Khadija Javed

Roll No 1227(3M)

Project Data structure & Algorithm

Department Computer Science

***Project Report: Text Editor Application***

**1. Description**

The Text Editor application is a simple C++ program designed to perform basic text editing operations. Users can insert and delete characters, navigate through the text, and use undo and redo functionality to manage their edits. The application uses efficient data structures to track changes and ensure smooth user interactions.

### 2. Introduction

#### Purpose

In modern text editing applications, basic functionalities such as text insertion, deletion, and navigation are essential. Additionally, the ability to undo and redo changes is critical for user convenience and error correction. This application is designed to demonstrate these fundamental text editing features using C++.

**Data Structure Used**

* Vector: The `vector` data structure is used to store the text. It provides dynamic array capabilities, allowing efficient insertion and deletion of characters at any position.
* Stack: Two stacks (`undoStack` and `redoStack`) are used to manage the undo and redo operations. Each stack stores the state of the text, enabling easy rollback and reapplication of changes.

Suitability

* Vector: Chosen for its dynamic resizing ability and efficient element access, insertion, and deletion.
* Stack: Ideal for managing the Last-In-First-Out (LIFO) nature of undo and redo operations, ensuring quick access to the most recent text state.

3. Data Structure Functions

The following table lists the functions available in the data structures and their usage in the project:

|  |
| --- |
|  |
| | Function | Data Structure | Description | |
| |--------------|----------------|----------------------------------------------------| |
| | `insert` | `vector` | Inserts a character at the current cursor position | |
| | `erase` | `vector` | Deletes a character at the current cursor position | |
| | `push` | `stack` | Saves the current state of the text | |
| | `pop` | `stack` | Reverts to the previous state | |
| | `top` | `stack` | Accesses the current state without removing it | |
| | `empty` | `stack` | Checks if the stack is empty | |

4. Functional Requirements

**Data Store**

Text Storage: The application stores text in a `vector<char>`.

**Basic Editing**

* Text Entry: Users can enter text using the `insert` method.
* Cursor Navigation: Users can navigate through the text using methods like `next()`, `previous()`, `start()`, `end()`, and `curr\_position(int)`.

Undo/Redo Functionality

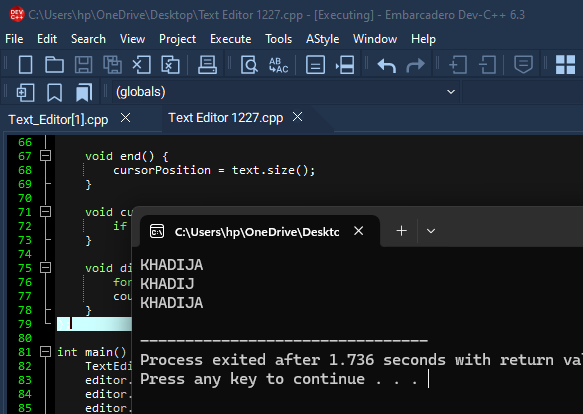
* Undo Operations: The application uses an `undoStack` to track all editing operations and allows users to undo the most recent edit.
* Redo Operations: The `redoStack` tracks undone operations, allowing users to redo an undone edit.

Stack Management: Both undo and redo stacks have a clear limit to the number of operations they can support, ensuring efficient memory usage.

*5. Test Cases*

**Test Case 1: Basic Insertion and Display**

* Description: Insert characters into the text and display the result.
* Input: 'K', 'H', 'A', 'D', 'I', 'J', 'A'
* Expected Output: "KHADIJA"
* Actual Output: "KHADIJA"



cpp

editor.insert('K');

editor.insert('H');

editor.insert('A');

editor.insert('D');

editor.insert('I');

editor.insert('J');

editor.insert('A');

editor.display();

```

Test Case 2: Undo Operation

Description: Undo the last insertion and display the result.

Expected Output: "KHADIJ"

Actual Output: "KHADIJ"

**cpp**

editor.undo();

editor.display();

```

Test Case 3: Redo Operation

* Description: Redo the last undone insertion and display the result.
* Expected Output: "KHADIJA"
* Actual Output: "KHADIJA"

**cpp**

editor.redo();

editor.display();

https://github.com/khadijaJaved1/texteditor

6. Conclusion

This Text Editor application demonstrates the core functionalities required for basic text manipulation. By leveraging efficient data structures like `vector` and `stack`, the application ensures smooth performance for insertions, deletions, and undo/redo operations. The clear and concise implementation provides a solid foundation for more advanced text editing features in future developments.