Name haris farooq

Class bscs

section 4d

roll number 21f-9064

code

#include <iostream>

#include <string>

#include <stack>

using namespace std;

// Node to represent a line of text in the text editor

struct LineNode {

string line;

LineNode\* next;

LineNode\* prev;

LineNode(const string& text) : line(text), next(nullptr), prev(nullptr) {}

};

// Text editor class

class TextEditor {

private:

LineNode\* head; // Head of the linked list

LineNode\* currentLine; // Current line pointer for cursor position

stack<string> undoStack; // Stack for undo functionality

public:

TextEditor() : head(nullptr), currentLine(nullptr) {}

~TextEditor() {

clear();

}

// Insert a new line at the current cursor position

void insertLine(const string& text) {

LineNode\* newNode = new LineNode(text);

if (!head) {

head = currentLine = newNode;

}

else {

if (currentLine) {

newNode->next = currentLine->next;

newNode->prev = currentLine;

if (currentLine->next) {

currentLine->next->prev = newNode;

}

currentLine->next = newNode;

currentLine = newNode;

}

else {

// If no current line, insert at the end

LineNode\* temp = head;

while (temp->next) {

temp = temp->next;

}

temp->next = newNode;

newNode->prev = temp;

currentLine = newNode;

}

}

undoStack.push("insert");

}

// Delete the current line

void deleteLine() {

if (currentLine) {

LineNode\* temp = currentLine;

if (currentLine->prev) {

currentLine->prev->next = currentLine->next;

}

else {

head = currentLine->next;

}

if (currentLine->next) {

currentLine->next->prev = currentLine->prev;

}

currentLine = currentLine->next;

delete temp;

}

undoStack.push("delete");

}

// Move the cursor to the next line

void moveCursorDown() {

if (currentLine && currentLine->next) {

currentLine = currentLine->next;

}

}

// Move the cursor to the previous line

void moveCursorUp() {

if (currentLine && currentLine->prev) {

currentLine = currentLine->prev;

}

}

// Display the content of the text editor

void display() const {

LineNode\* temp = head;

while (temp) {

if (temp == currentLine) {

cout << "> " << temp->line << endl;

}

else {

cout << " " << temp->line << endl;

}

temp = temp->next;

}

}

// Undo the last operation

void undo() {

if (!undoStack.empty()) {

string lastOperation = undoStack.top();

undoStack.pop();

if (lastOperation == "insert") {

deleteLine();

}

else if (lastOperation == "delete") {

// To undo delete, insert a line at the same position

if (currentLine) {

insertLine("");

}

}

}

}

// Clear all lines in the text editor

void clear() {

LineNode\* temp = head;

while (temp) {

LineNode\* nextNode = temp->next;

delete temp;

temp = nextNode;

}

head = currentLine = nullptr;

while (!undoStack.empty()) {

undoStack.pop();

}

}

};

int main() {

TextEditor textEditor;

system("color 5");

textEditor.insertLine("Hello, this is a simple text editor.");

textEditor.insertLine("You can add multiple lines and move the cursor.");

textEditor.insertLine("Use arrow keys to navigate.");

textEditor.insertLine("Try deleting lines and undoing the changes.");

cout << "Text Editor Content:\n";

textEditor.display();

textEditor.moveCursorDown();

textEditor.moveCursorDown();

textEditor.deleteLine();

cout << "\nText Editor Content after deletion:\n";

textEditor.display();

textEditor.undo();

cout << "\nText Editor Content after undo:\n";

textEditor.display();

textEditor.clear(); // Clear all lines

cout << "\nText Editor Content after clearing:\n";

textEditor.display();

return 0;

}

Result

