Министерство образования и науки РФ

Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа «Компьютерных технологий и информационных систем»

ОТЧЕТ

по дисциплине ««Теория и технология программирования»

**Лабораторная работа № 10**

**Выполнил:**

Cтудент гр. 5130902/40002 Г.Ю. Рюмин

**Проверил**

Ст. преподаватель А.М. Журавская

Санкт-Петербург

2025 г.

**1. Цель работы.**

Цель задания – изучить методы работы с классом CString.

**2. Задание.**

Написать текстовый редактор с параметрами (индивидуальным заданием), представленными в системной матрице в Таблице 1. В текстовом редакторе должны быть функции открытия файла, проверки расширения файла и его сохранения.

**Вариант:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | 2 | 4 | 5 |

Варианты индивидуальных заданий:

1. Поиск символа (ов).
2. Вставка в начало.
3. Добавление в конец.
4. Отбор (с N до M символа).
5. Копирование строк.
6. Замена всех строчных букв прописными.
7. Замена всех первых строчных букв слов прописными.
8. Поиск в строке.
9. Вставка текста из файла.
10. Удаление символов сначала строки.
11. Удаление символов с конца строки.

## **Код программы**

#define NOMINMAX

#include <iostream>

#include <vector>

#include <string>

#include <fstream>

#include <limits>

#include <stdexcept>

#include <algorithm>

#include <sstream>

#include <locale>

#ifdef \_WIN32

#include <windows.h>

#endif

using namespace std;

long long getValidatedInput(long long minVal = numeric\_limits<long long>::min(), long long maxVal = numeric\_limits<long long>::max()) {

long long value;

string line;

while (true) {

getline(cin, line);

stringstream ss(line);

if ((ss >> value) && (ss >> ws).eof() && value >= minVal && value <= maxVal) {

return value;

}

else {

cerr << "Ошибка ввода. Пожалуйста, введите ТОЛЬКО целое число от " << minVal << " до " << maxVal << ": ";

}

}

}

string getNonEmptyLine() {

string line;

while (true) {

getline(cin, line);

size\_t first = line.find\_first\_not\_of(" \t\n\r\f\v");

if (string::npos == first) {

cerr << "Ошибка ввода. Пожалуйста, введите непустую строку: ";

continue;

}

size\_t last = line.find\_last\_not\_of(" \t\n\r\f\v");

line = line.substr(first, (last - first + 1));

if (!line.empty()) {

return line;

}

}

}

bool checkExtension(const string& filename, const string& expectedExt) {

if (filename.length() >= expectedExt.length()) {

return (0 == filename.compare(filename.length() - expectedExt.length(), expectedExt.length(), expectedExt));

}

else {

return false;

}

}

void displayContent(const vector<string>& lines) {

cout << "--- Содержимое документа ---" << endl;

if (lines.empty()) {

cout << "(Пусто)" << endl;

}

else {

for (size\_t i = 0; i < lines.size(); ++i) {

cout << (i + 1) << ": " << lines[i] << endl;

}

}

cout << "-----------------------------" << endl;

}

void displayMenu() {

cout << endl << "--- Текстовый редактор ---" << endl;

cout << "1. Открыть файл (.txt)" << endl;

cout << "2. Сохранить файл (.txt)" << endl;

cout << "3. Вставить текст в начало строки" << endl;

cout << "4. Выделить подстроку (с N до M символа)" << endl;

cout << "5. Копировать строки" << endl;

cout << "6. Показать содержимое" << endl;

cout << "0. Выход" << endl;

cout << "---------------------------------------" << endl;

}

bool openFile(const string& filename, vector<string>& lines) {

ifstream inFile(filename);

if (!inFile.is\_open()) {

return false;

}

inFile.imbue(locale(""));

lines.clear();

string line;

while (getline(inFile, line)) {

if (!line.empty() && line.back() == '\r') {

line.pop\_back();

}

lines.push\_back(line);

}

inFile.close();

if (inFile.bad()) {

cerr << "Ошибка чтения файла (возможно, диск поврежден или нет прав)." << endl;

return false;

}

return true;

}

bool saveFile(const string& filename, const vector<string>& lines) {

ofstream outFile(filename);

if (!outFile.is\_open()) {

return false;

}

outFile.imbue(locale(""));

for (size\_t i = 0; i < lines.size(); ++i) {

outFile << lines[i] << endl;

}

outFile.close();

return outFile.good();

}

void taskInsertBeginning(vector<string>& lines) {

displayContent(lines);

size\_t lineNum, lineIndex;

string textToInsert;

if (lines.size() != 0) {

cout << "Введите номер строки для вставки (1-" << lines.size() << "): ";

lineNum = getValidatedInput(1, lines.size());

cout << "Введите текст для вставки в начало строки " << lineNum << ": ";

textToInsert = getNonEmptyLine();

lineIndex = lineNum - 1;

}

else {

cout << "Введите текст для вставки в начало строки: ";

textToInsert = getNonEmptyLine();

lineIndex = 0;

lines.push\_back("");

}

try {

lines.at(lineIndex).insert(0, textToInsert);

cout << "Текст успешно вставлен." << endl;

displayContent(lines);

}

catch (const out\_of\_range& oor) {

cerr << "Ошибка: Недопустимый номер строки." << endl;

}

}

void taskSubstring(const vector<string>& lines) {

displayContent(lines);

cout << "Введите номер строки для выделения подстроки (1-" << lines.size() << "): ";

size\_t lineNum = getValidatedInput(1, lines.size());

size\_t lineIndex = lineNum - 1;

try {

const string& targetLine = lines.at(lineIndex);

if (targetLine.empty()) {

cout << "Выбранная строка пуста. Нельзя выделить подстроку." << endl;

return;

}

cout << "Строка " << lineNum << ": \"" << targetLine << "\" (Длина: " << targetLine.length() << ")" << endl;

cout << "Введите начальную позицию N (1-" << targetLine.length() << "): ";

size\_t startPos = getValidatedInput(1, targetLine.length());

cout << "Введите конечную позицию M (" << startPos << "-" << targetLine.length() << "): ";

size\_t endPos = getValidatedInput(startPos, targetLine.length());

size\_t startIndex = startPos - 1;

size\_t length = endPos - startPos + 1;

string sub = targetLine.substr(startIndex, length);

cout << "Выделенная подстрока (символы с " << startPos << " по " << endPos << "): \"" << sub << "\"" << endl;

}

catch (const out\_of\_range& oor) {

cerr << "Ошибка: Недопустимый номер строки или диапазон для подстроки." << endl;

}

}

void taskCopyLines(vector<string>& lines) {

displayContent(lines);

cout << "--- Копирование строк ---" << endl;

if (lines.empty()) {

cout << "Документ пуст, копировать нечего." << endl;

return;

}

cout << "Введите номер НАЧАЛЬНОЙ строки для копирования (1-" << lines.size() << "): ";

size\_t startLineNum = getValidatedInput(1, lines.size());

cout << "Введите номер КОНЕЧНОЙ строки для копирования (" << startLineNum << "-" << lines.size() << "): ";

size\_t endLineNum = getValidatedInput(startLineNum, lines.size());

cout << "Введите номер строки, ПЕРЕД которой нужно вставить скопированные строки (1-" << lines.size() + 1 << "): ";

size\_t insertPosNum = getValidatedInput(1, lines.size() + 1);

size\_t startIndex = startLineNum - 1;

size\_t endIndex = endLineNum - 1;

size\_t insertIndex = insertPosNum - 1;

vector<string> linesToCopy;

linesToCopy.reserve(endIndex - startIndex + 1);

try {

for (size\_t i = startIndex; i <= endIndex; ++i) {

linesToCopy.push\_back(lines.at(i));

}

if (insertIndex > lines.size()) {

cerr << "Ошибка: Недопустимая позиция для вставки." << endl;

return;

}

auto insertIterator = lines.begin() + insertIndex;

lines.insert(insertIterator, linesToCopy.begin(), linesToCopy.end());

cout << "Строки с " << startLineNum << " по " << endLineNum << " скопированы и вставлены перед строкой " << insertPosNum << "." << endl;

displayContent(lines);

}

catch (const out\_of\_range& oor) {

cerr << "Ошибка: Недопустимый номер строки при доступе для копирования." << endl;

}

catch (const bad\_alloc& ba) {

cerr << "Ошибка: Недостаточно памяти для копирования строк." << endl;

}

}

int main() {

#ifdef \_WIN32

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

#endif

setlocale(LC\_ALL, "Russian");

vector<string> documentLines;

string currentFilename = "";

bool fileLoaded = false;

int choice;

do {

displayMenu();

cout << "Выберите опцию: ";

choice = getValidatedInput(0, 6);

switch (choice) {

case 1: {

cout << "Введите имя файла для открытия (например, document.txt): ";

string filename = getNonEmptyLine();

if (!checkExtension(filename, ".txt")) {

cerr << "Ошибка: Поддерживаются только файлы с расширением .txt" << endl;

}

else {

if (openFile(filename, documentLines)) {

currentFilename = filename;

fileLoaded = true;

cout << "Файл '" << filename << "' успешно загружен." << endl;

displayContent(documentLines);

}

else {

cerr << "Ошибка: Не удалось открыть файл '" << filename << "'" << endl;

fileLoaded = false;

}

}

break;

}

case 2: {

if (!fileLoaded && currentFilename.empty()) {

cout << "Нет открытого файла. Введите имя для сохранения (например, new\_doc.txt): ";

currentFilename = getNonEmptyLine();

if (!checkExtension(currentFilename, ".txt")) {

cerr << "Ошибка: Сохранение возможно только в файлы с расширением .txt" << endl;

currentFilename = "";

break;

}

}

else if (currentFilename.empty()) {

cout << "Введите имя файла для сохранения (например, new\_doc.txt): ";

currentFilename = getNonEmptyLine();

if (!checkExtension(currentFilename, ".txt")) {

cerr << "Ошибка: Сохранение возможно только в файлы с расширением .txt" << endl;

currentFilename = "";

break;

}

}

if (!currentFilename.empty()) {

if (saveFile(currentFilename, documentLines)) {

cout << "Файл '" << currentFilename << "' успешно сохранен." << endl;

fileLoaded = true;

}

else {

cerr << "Ошибка: Не удалось сохранить файл '" << currentFilename << "'" << endl;

}

}

break;

}

case 3:

taskInsertBeginning(documentLines);

break;

case 4:

if (documentLines.empty()) {

cerr << "Ошибка: Документ пуст. Сначала откройте файл или добавьте строки." << endl;

}

else {

taskSubstring(documentLines);

}

break;

case 5:

if (documentLines.empty()) {

cerr << "Ошибка: Документ пуст. Сначала откройте файл или добавьте строки." << endl;

}

else {

taskCopyLines(documentLines);

}

break;

case 6:

if (documentLines.empty()) {

cout << "Документ пуст." << endl;

}

else {

displayContent(documentLines);

}

break;

case 0:

cout << "Выход из программы." << endl;

break;

default:

cerr << "Неверный ввод. Пожалуйста, попробуйте снова." << endl;

break;

}

if (choice != 0) {

cout << endl << "Нажмите Enter для продолжения...";

string dummy;

getline(cin, dummy);

}

} while (choice != 0);

return 0;

}

**Пример работы программы**

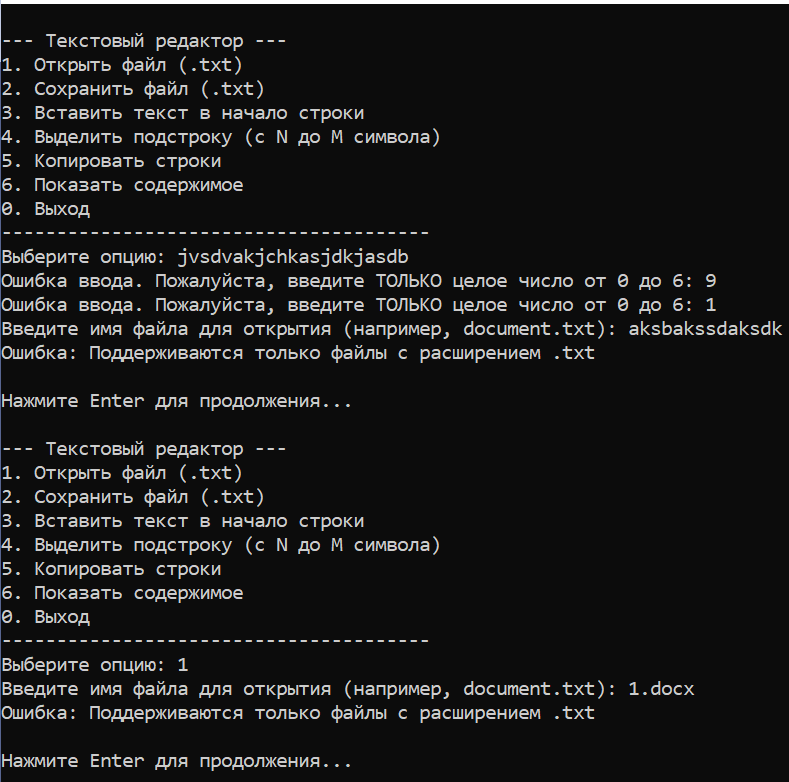


Рисунок 1 – Пример работы программы с некорректным вводом

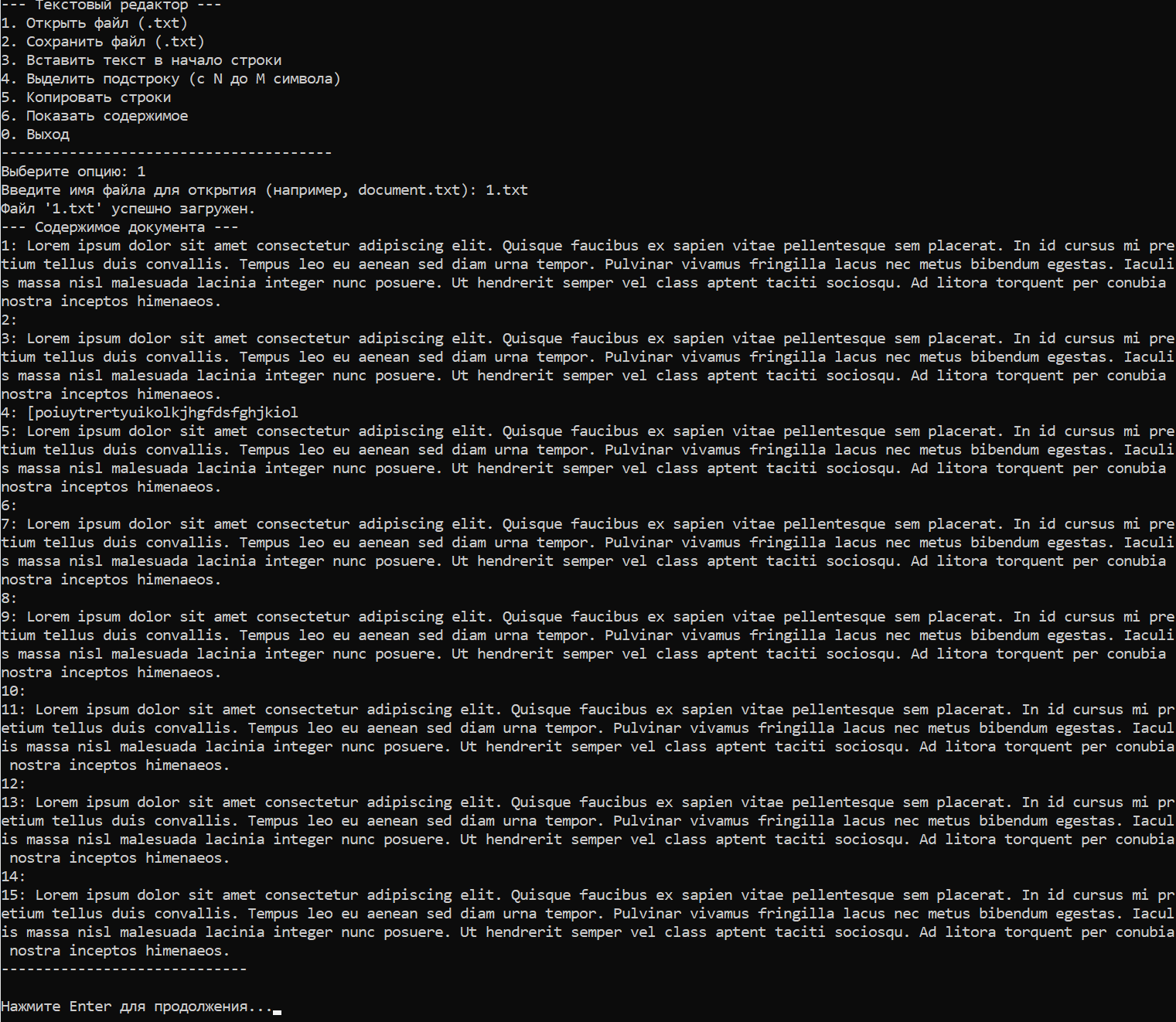


Рисунок 2 – Пример работы программы с корректным вводом: открытие файла

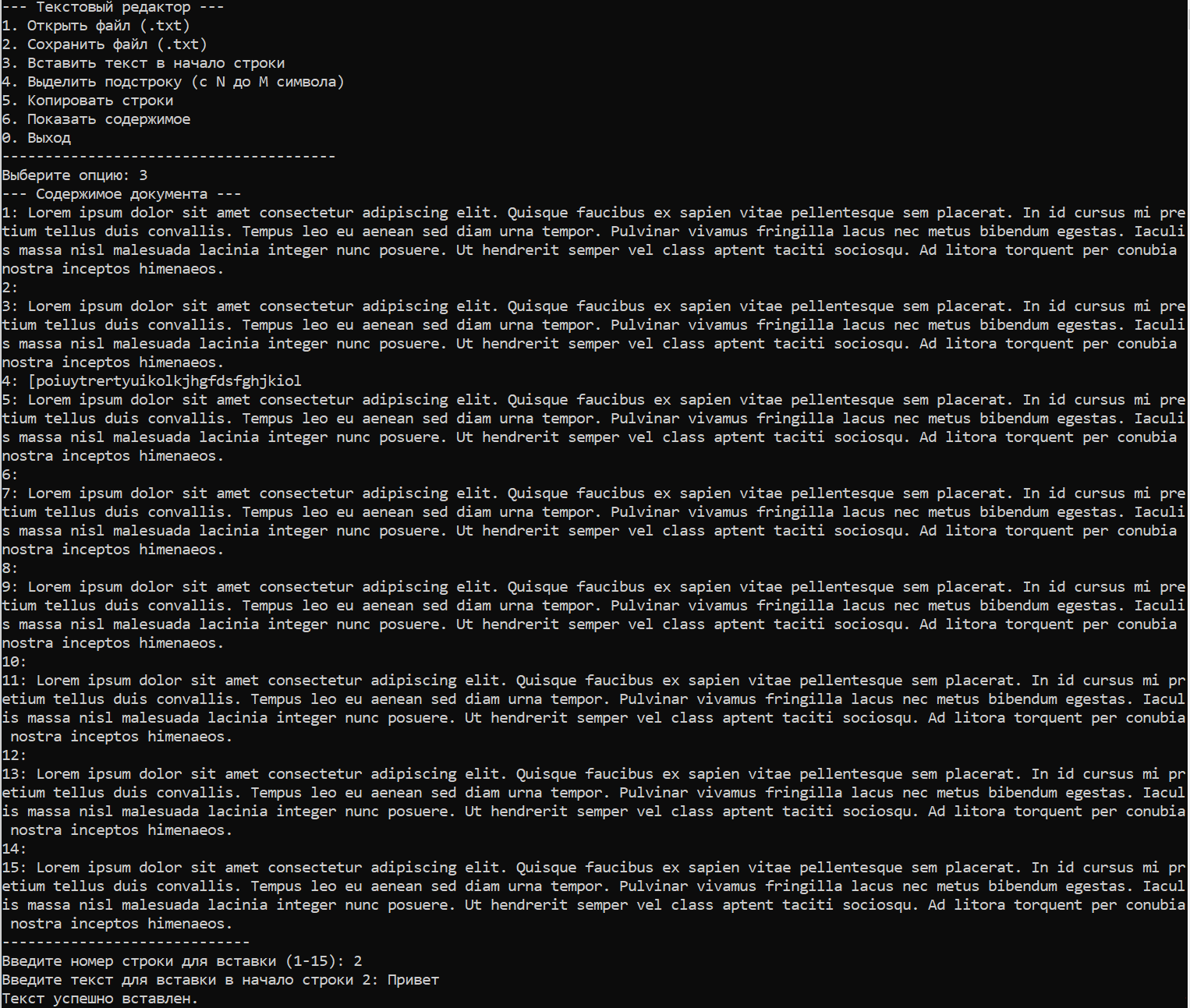


Рисунок 3 – Пример работы программы: вставка текста в начало строки (часть 1)

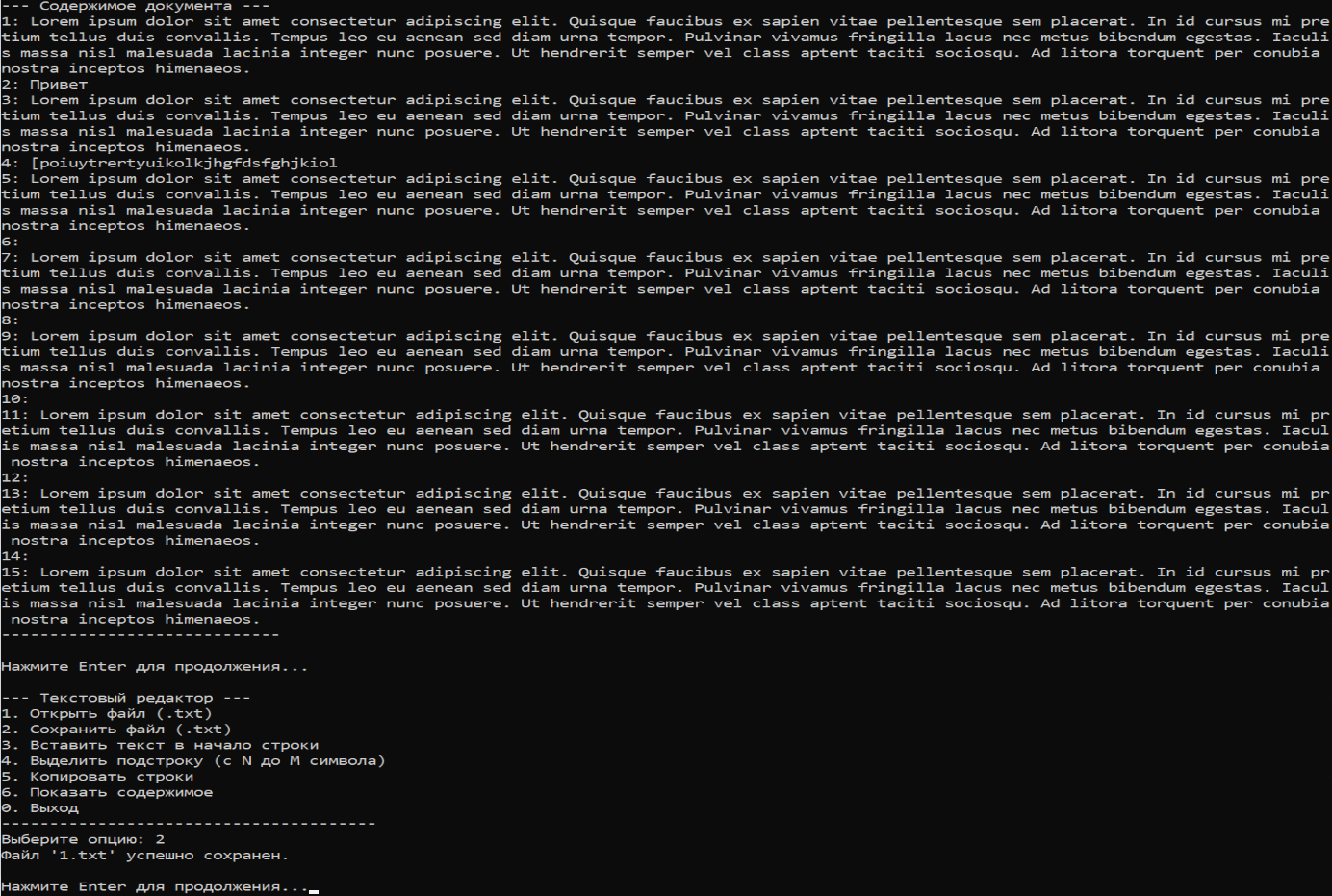


Рисунок 4 – Пример работы программы: вставка в начало строки (часть 2)

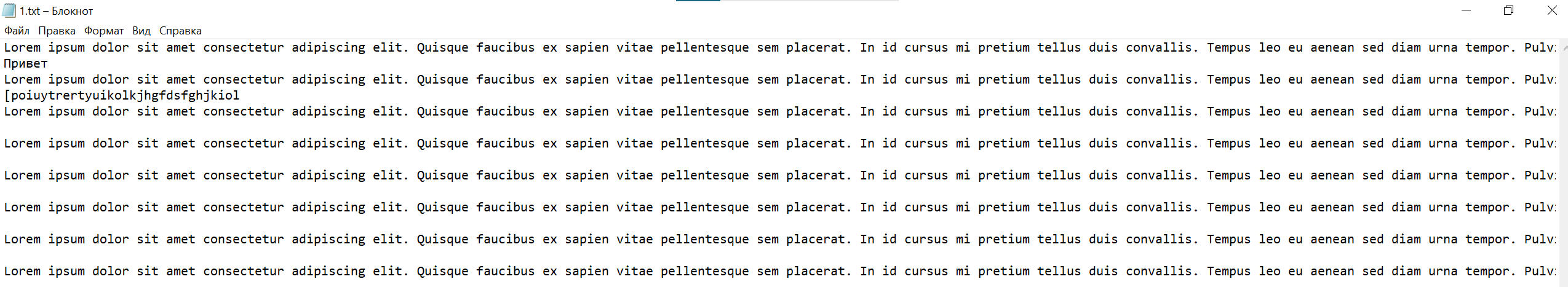


Рисунок 5 – Пример работы программы: вставка в начало строки (часть 3)

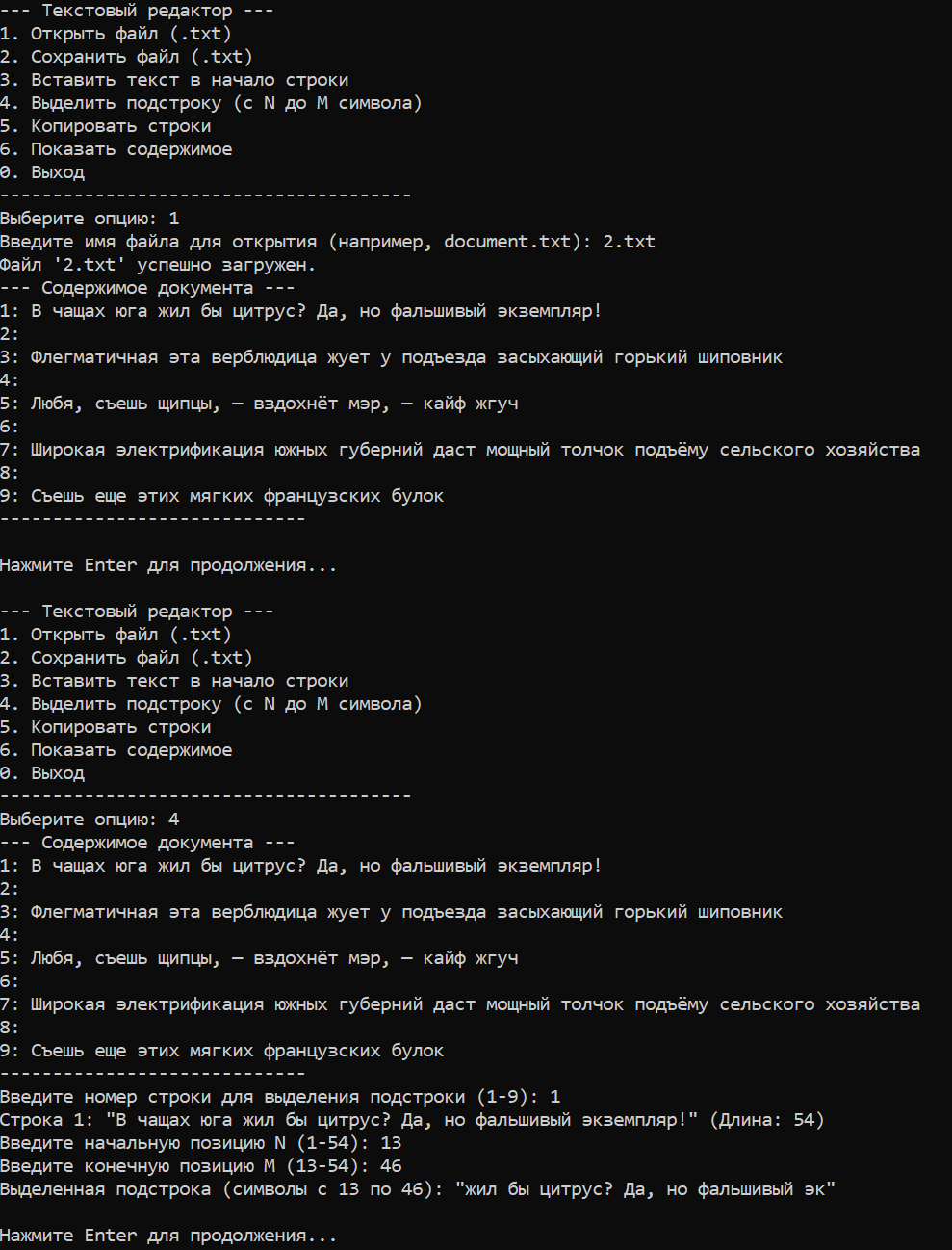


Рисунок 6 – Пример работы программы: выделение подстроки в файле

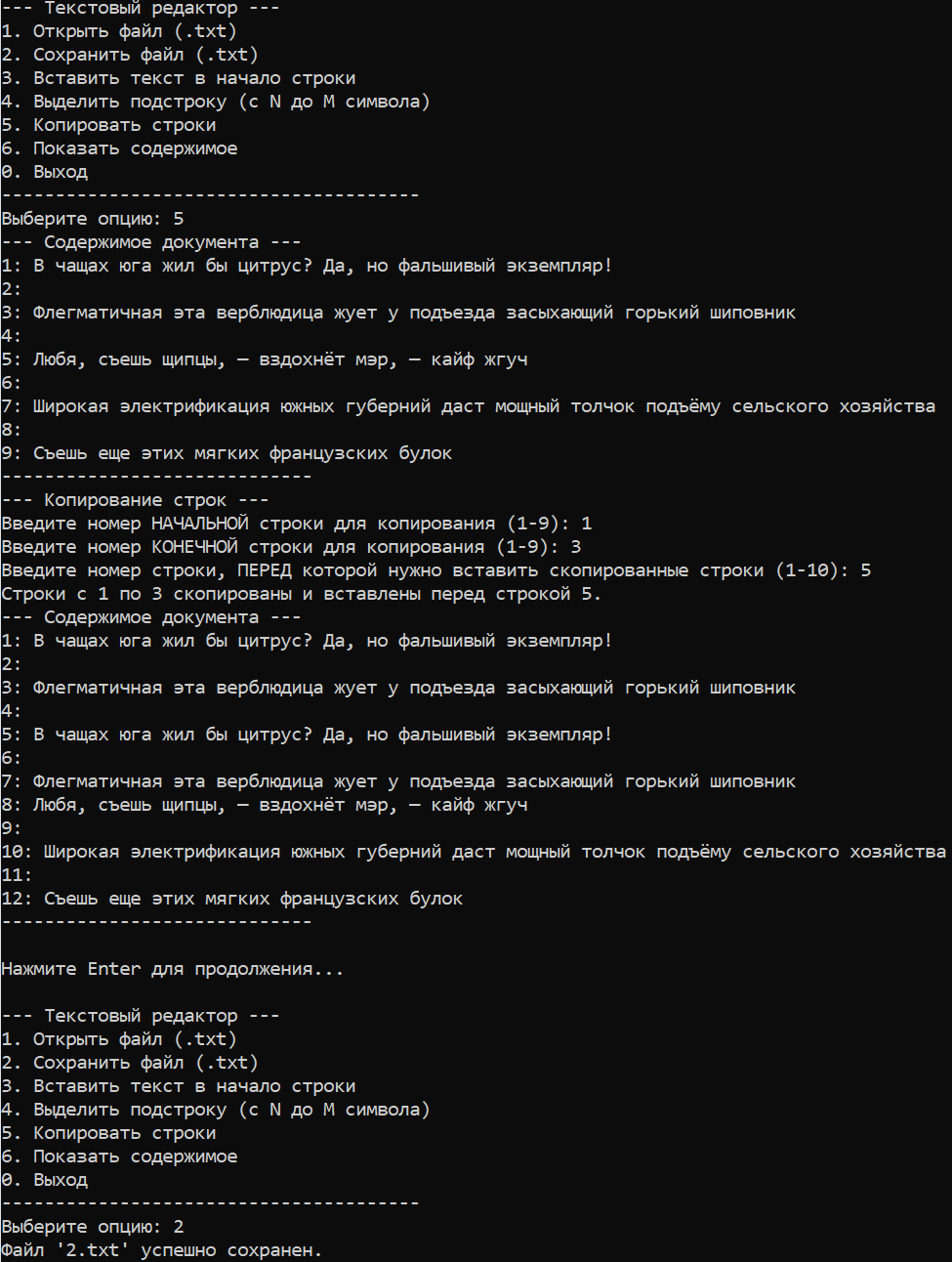


Рисунок 7 – Пример работы программы: копирование строк (часть 1)

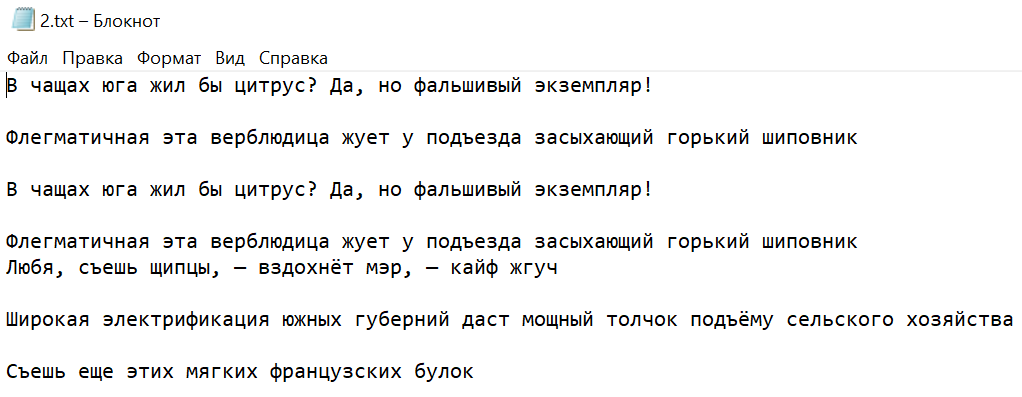


Рисунок 7 – Пример работы программы: копирование строк (часть 2)

## **Вывод**

В ходе выполнения данной лабораторной работы были изучены и применены на практике основные принципы работы с текстовыми данными в C++ с использованием стандартных библиотек, включая класс std::string и контейнер std::vector. Был разработан консольный текстовый редактор, реализующий базовые функции открытия, сохранения текстовых файлов в кодировке ANSI, а также специфические операции манипуляции строками согласно варианту задания, такие как вставка текста, выделение подстроки и копирование строк. Особое внимание было уделено обработке пользовательского ввода для предотвращения ошибок и обеспечению корректной работы с кириллицей в консоли Windows и текстовых файлах, что позволило закрепить навыки создания интерактивных консольных приложений и работы с файловой системой.