Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Операционные среды и системное программирование

ОТЧЁТ
к лабораторной работе №1
на тему

ОСНОВЫ ПРОГРАММИРОВАНИЯ В WIN 32 API. ОКОННОЕ ПРИЛОЖЕНИЕ WIN 32 С МИНИМАЛЬНОЙ ДОСТАТОЧНОЙ ФУНКЦИОНАЛЬНОСТЬЮ. ОБРАБОТКА ОСНОВНЫХ ОКОННЫХ СООБЩЕНИЙ.

Выполнил студент гр.153504 Сацюк С.В.

Проверил ассистент кафедры информатики Гриценко Н.Ю.

Минск 2023

СОДЕРЖАНИЕ

1 ФОРМУЛИРОВКА ЗАДАЧИ

Целью выполнения лабораторной работы является создание оконного приложения на Win32 API, обладающее минимальным функционалом, позволяющим отработать базовые навыки написания программы на Win32 API, таких как обработка оконных сообщений.

В качестве задачи необходимо построить приложение для чтения и редактирования текстовых документов с возможностью выделения и копирования текста в буфер обмена.

2 ОПИСАНИЕ ФУНКЦИЙ ПРОГРАММЫ

Согласно формулировке задачи, были спроектированы следующие функции программы:

– Открытие текстового файла;
– Создание нового текстового файла;
– Редактирование текстового файла;
– Копирование выделенного текста в буфер обмена;
– Сохранение файла.

1. Открытие файла

Для открытия текстового файла необходимо в меню нажать File-Open и в открывшемся диалоговом окне выбрать нужный файл.

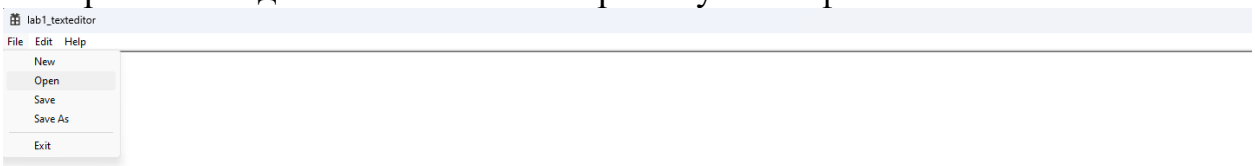

Рисунок 1 – Открытие файла

2. Редактирование текста и копирование выделенного текста в буфер обмена

После открытия файла в окне редактирования появляется текст из него, который можно отредактировать. Для копирования выделенного текста в буфер обмена можно воспользоваться меню: File-Copy.

Рисунок 2 – Копирование выделенного текста в буфер обмена

Рисунок 3 – Результат вставки текста после копирования

3. Сохранение отредактированного текста в файл

Для сохранения отредактированного текста в файл необходимо воспользоваться меню: File-Save. Сохранение произойдет в файл, который был открыт вначале.

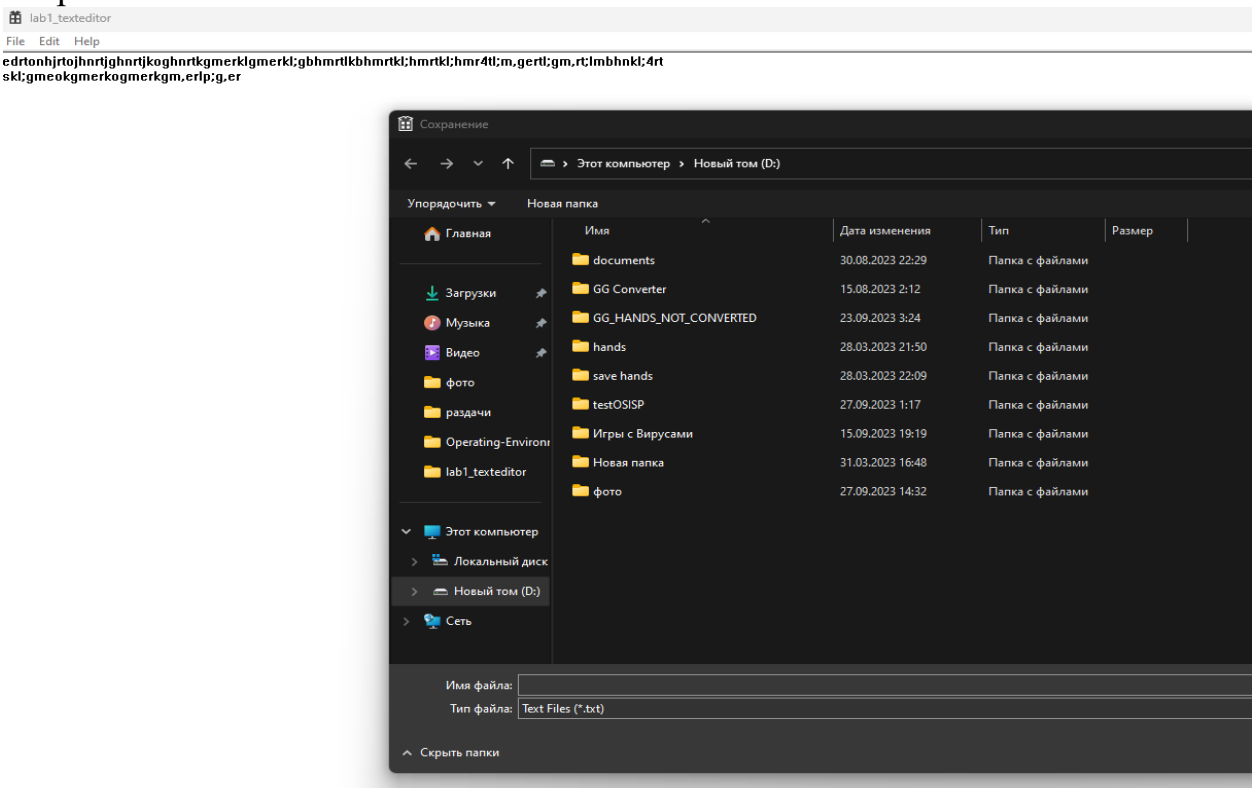

Рисунок 4 – Сохранение файла

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Build desktop Windows apps using the Win32 API [Электронный ресурс]. – Электронные данные. – Режим доступа: https://learn.microsoft.com/en-us/windows/win32/

[2] RegisterHotKey function (winuser.h) - Win32 apps – Электронные данные. – Режим доступа: https://learn.microsoft.com/enus/windows/win32/api/winuser/nf-winuser-registerhotkey/

[3] CreateWindowExA function (winuser.h) - Win32 apps – Электронные данные. – Режим доступа: https://learn.microsoft.com/enus/windows/win32/api/winuser/nf-winuser-createwindowexa

ПРИЛОЖЕНИЕ А

Листинг кода

Файл TextEditor.cpp

```
#include <windows.h>
#include "resource.h"

HINSTANCE hInst; // Дескриптор текущего экземпляра приложения
HWND hMainWnd;   // Дескриптор главного окна
HWND hEdit;      // Дескриптор элемента управления EDIT (текстовое поле)
HMENU hMenu;     // Дескриптор меню приложения

static WCHAR szFileName[MAX_PATH] = L""; // Объявляем глобально имя файла

// Прототипы функций
LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

int APIENTRY WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine,
int nCmdShow)
{
    UNREFERENCED_PARAMETER(hPrevInstance);
    UNREFERENCED_PARAMETER(lpCmdLine);

    WNDCLASSEX wcex = { sizeof(WNDCLASSEX), CS_HREDRAW | CS_VREDRAW, WndProc, 0, 0,
GetModuleHandle(nullptr), nullptr, nullptr, nullptr, nullptr, L"TextEditor", nullptr
};
    RegisterClassEx(&wcex);

    hInst = hInstance;
    RECT rc = { 0, 0, 800, 600 };
    AdjustWindowRect(&rc, WS_OVERLAPPEDWINDOW, FALSE);
    hMainWnd = CreateWindow(L"TextEditor", L"Text Editor", WS_OVERLAPPEDWINDOW,
CW_USEDEFAULT, CW_USEDEFAULT,
        rc.right - rc.left, rc.bottom - rc.top, nullptr, nullptr, hInstance,
nullptr);

    if (!hMainWnd)
    {
        return FALSE;
    }

    ShowWindow(hMainWnd, nCmdShow);
    UpdateWindow(hMainWnd);

    // Создание меню
    hMenu = CreateMenu();
    HMENU hFileMenu = CreateMenu();
    HMENU hEditMenu = CreateMenu();

    // Добавляем пункты меню для меню "File"
    AppendMenu(hMenu, MF_POPUP, (UINT_PTR)hFileMenu, L"File");
    AppendMenu(hFileMenu, MF_STRING, IDM_FILE_NEW, L"New");
    AppendMenu(hFileMenu, MF_STRING, IDM_FILE_OPEN, L"Open");
    AppendMenu(hFileMenu, MF_STRING, IDM_FILE_SAVE, L"Save");
    AppendMenu(hFileMenu, MF_STRING, IDM_FILE_SAVEAS, L"Save As");

    // Добавляем пункты меню для меню "Edit"
    AppendMenu(hMenu, MF_POPUP, (UINT_PTR)hEditMenu, L"Edit");
    AppendMenu(hEditMenu, MF_STRING, IDM_EDIT_COPY, L"Copy");
    AppendMenu(hEditMenu, MF_STRING, IDM_EDIT_PASTE, L"Paste");
    AppendMenu(hEditMenu, MF_STRING, IDM_EDIT_CUT, L"Cut");
    AppendMenu(hEditMenu, MF_STRING, IDM_EDIT_SELECTALL, L"Select All");

    // Добавляем меню "Help" в главное меню
```

```
    AppendMenu(hMenu, MF_STRING, IDM_HELP_ABOUT, L"Help");

    // Устанавливаем меню в главное окно
    SetMenu(hMainWnd, hMenu);

    // Основной цикл сообщений
    MSG msg;
    while (GetMessage(&msg, nullptr, 0, 0))
    {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }

    return static_cast<int>(msg.wParam);
}

// Обработчик сообщений окна
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{

    switch (message)
    {

    case WM_CREATE:
    {
        hEdit = CreateWindowEx(
            WS_EX_CLIENTEDGE,
            L"EDIT",
            NULL,
            WS_CHILD | WS_VISIBLE | WS_VSCROLL | WS_HSCROLL | ES_MULTILINE |
ES_AUTOVSCROLL | ES_AUTOHSCROLL,
            0, 0, 800, 600,
            hWnd,
            (HMENU)IDC_TEXT_EDIT,
            hInst,
            NULL);

        if (hEdit == NULL)
        {
            MessageBox(NULL, L"Cannot create edit control.", L"Error", MB_OK |
MB_ICONERROR);
            return -1;
        }

        // Устанавливаем меню в главное окно
        SetMenu(hWnd, hMenu);
    }
    break;

    case WM_COMMAND:
    {
        int wmId = LOWORD(wParam);
        int wmEvent = HIWORD(wParam);

        if (wmEvent == 0) // Проверяем, что это сообщение от меню
        {
            // Обработка элементов меню
            switch (wmId)
            {
            case IDM_EXIT:
                DestroyWindow(hWnd);
                break;
            case IDM_FILE_NEW:
                // Обработка пункта меню "New"
```

```
                SendMessage(hEdit, WM_SETTEXT, 0, (LPARAM)L""); // Очищаем текстовое
поле
                break;
            case IDM_FILE_OPEN:
            {
                OPENFILENAME ofn;

                ZeroMemory(&ofn, sizeof(ofn));
                ofn.lStructSize = sizeof(ofn);
                ofn.hwndOwner = hMainWnd;
                ofn.lpstrFilter = L"Text Files (*.txt)\0*.txt\0All Files
(*.*)\0*.*\0";
                ofn.lpstrFile = szFileName;
                ofn.nMaxFile = MAX_PATH;
                ofn.Flags = OFN_FILEMUSTEXIST;

                if (GetOpenFileName(&ofn))
                {
                    // Открываем выбранный файл для чтения
                    HANDLE hFile = CreateFile(szFileName, GENERIC_READ, 0, NULL,
OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
                    if (hFile != INVALID_HANDLE_VALUE)
                    {
                        DWORD dwFileSize = GetFileSize(hFile, NULL);
                        if (dwFileSize != INVALID_FILE_SIZE)
                        {
                            // Выделение памяти для чтения файла
                            wchar_t* pBuffer = new wchar_t[dwFileSize /
sizeof(wchar_t) + 1];
                            if (pBuffer)
                            {
                                DWORD dwBytesRead;
                                if (ReadFile(hFile, pBuffer, dwFileSize,
&dwBytesRead, NULL))
                                {
                                    pBuffer[dwBytesRead / sizeof(wchar_t)] = L'\0';
// Добавляем нулевой символ в конец, чтобы сделать строку
                                    SetWindowTextW(hEdit, pBuffer); // Устанавливаем
содержимое файла в текстовое поле
                                }

                                delete[] pBuffer;
                            }
                        }

                        CloseHandle(hFile);
                    }
                    // Обновляем имя файла
                    wcscpy_s(szFileName, MAX_PATH, ofn.lpstrFile);

                    // Устанавливаем название файла в заголовок окна
                    SetWindowTextW(hMainWnd, szFileName);
                }
            }
            break;
            case IDM_FILE_SAVE:
                // Обработка пункта меню "Save"
                if (szFileName[0] == L'\0') // Если у нас нет имени файла (файл не
был открыт или сохранен ранее)
                {
                    // Если нет имени файла, вызываем функцию "Save As"
                    SendMessage(hMainWnd, WM_COMMAND, IDM_FILE_SAVEAS, 0);
                }
                else // Если у нас уже есть имя файла
                {
```

```
                    // Открываем выбранный файл для записи
                    HANDLE hFile = CreateFile(szFileName, GENERIC_WRITE, 0, NULL,
CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
                    if (hFile != INVALID_HANDLE_VALUE)
                    {
                        int textLength = GetWindowTextLength(hEdit);
                        wchar_t* buffer = new wchar_t[textLength + 1];
                        GetWindowText(hEdit, buffer, textLength + 1);

                        DWORD dwBytesWritten;
                        WriteFile(hFile, buffer, textLength * sizeof(wchar_t),
&dwBytesWritten, NULL);

                        delete[] buffer;
                        CloseHandle(hFile);
                    }
                }
                break;

            case IDM_FILE_SAVEAS:
            {
                OPENFILENAME ofn;

                ZeroMemory(&ofn, sizeof(ofn));
                ofn.lStructSize = sizeof(ofn);
                ofn.hwndOwner = hMainWnd;
                ofn.lpstrFilter = L"Text Files (*.txt)\0*.txt\0All Files
(*.*)\0*.*\0";
                ofn.lpstrFile = szFileName; // Используем глобальное имя файла
                ofn.nMaxFile = MAX_PATH;
                ofn.Flags = OFN_OVERWRITEPROMPT;

                if (GetSaveFileName(&ofn))
                {
                    int textLength = GetWindowTextLength(hEdit);
                    wchar_t* buffer = new wchar_t[textLength + 1];
                    GetWindowText(hEdit, buffer, textLength + 1);

                    // Добавляем расширение .txt, если его нет
                    if (wcsstr(szFileName, L".") == NULL)
                    {
                        wcscat_s(szFileName, MAX_PATH, L".txt");
                    }

                    // Открываем выбранный файл для записи
                    HANDLE hFile = CreateFile(szFileName, GENERIC_WRITE, 0, NULL,
CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
                    if (hFile != INVALID_HANDLE_VALUE)
                    {
                        DWORD dwBytesWritten;
                        WriteFile(hFile, buffer, textLength * sizeof(wchar_t),
&dwBytesWritten, NULL);
                        CloseHandle(hFile);
                    }

                    delete[] buffer;
                }
            }
            break;

            case IDM_EDIT_COPY:
                SendMessage(hEdit, WM_COPY, 0, 0);
                break;

            case IDM_EDIT_PASTE:
```

```
                SendMessage(hEdit, WM_PASTE, 0, 0);
                break;
            case IDM_EDIT_CUT:
                SendMessage(hEdit, WM_CUT, 0, 0);
                break;
            case IDM_EDIT_SELECTALL:
                SendMessage(hEdit, EM_SETSEL, 0, -1);
                break;


            case IDM_HELP_ABOUT:
                // Обработка пункта меню "About"
                MessageBox(hWnd, L"Text Editor v1.0\n\n© by Stas", L"About",
MB_ICONINFORMATION | MB_OK);
                break;
            default:
                return DefWindowProc(hWnd, message, wParam, lParam);
            }

        }
        else if (wmId == IDC_TEXT_EDIT && wmEvent == EN_UPDATE)
        {
            // Обработка изменений в текстовом поле
            int textLength = GetWindowTextLength(hEdit);
            wchar_t* buffer = new wchar_t[textLength + 1];
            GetWindowText(hEdit, buffer, textLength + 1);
            // Теперь переменная 'buffer' содержит текст из текстового поля
            delete[] buffer;
        }
        else
        {
            return DefWindowProc(hWnd, message, wParam, lParam);
        }
    }
    break;



    case WM_PAINT:
    {
        PAINTSTRUCT ps;
        HDC hdc = BeginPaint(hWnd, &ps);  //код отрисовки окна


        EndPaint(hWnd, &ps);
    }
    break;
    case WM_SIZE:
        // TODO: Обработка изменения размера окна
        break;
    case WM_DESTROY:
        PostQuitMessage(0); //завершению основного цикла сообщений и закрытию
приложения
        break;
    default:
        return DefWindowProc(hWnd, message, wParam, lParam);
    }
    return 0;
}
```