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

ОТЧЕТ

к лабораторной работе №3

на тему

**УПРАВЛЕНИЕ ПАМЯТЬЮ И ВВОДОМ-ВЫВОДОМ, РАСШИРЕННЫЕ ВОЗМОЖНОСТИ ВВОДА-ВЫВОДА WINDOWS**

Студент А. С. Сорокин

Преподаватель Н. Ю. Гриценко

Минск 2023

СОДЕРЖАНИЕ

[1 Цель работы 3](#_Toc146752068)

[2 Теоретические сведения 4](#_Toc146752069)

[3 Результат выполнения 5](#_Toc146752070)

[Заключение 6](#_Toc146752071)

[Список использованных источников 7](#_Toc146752072)

[Приложение А (обязательное) Листинг кода 8](#_Toc146752073)

1. **ЦЕЛЬ РАБОТЫ**

Целью данной лабораторной работы является изучение и практическое применение методов управления памятью и вводом-выводом, а также расширенных функциональных возможностей для ввода-вывода в операционной системе *Windows* через использование *API* подсистемы памяти *Win32*. Основными задачами лабораторной работы являются овладение навыками работы с асинхронными операциями ввода-вывода, осуществление отображения файлов в память, и приобретение практических навыков для разработки эффективных приложений, использующих данные возможности операционной системы *Windows*.

1. **ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Сопоставление файлов — это связь содержимого файла с частью виртуального адресного пространства процесса. Система создает объект сопоставления файлов (также известный как объект раздела) для поддержания этой связи.

Представление файла — это часть виртуального адресного пространства, которую процесс использует для доступа к содержимому файла. Сопоставление файлов позволяет процессу использовать как случайные входные и выходные *(I/O),* так и последовательные операции ввода-вывода.

Обрабатывает чтение из представления файлов и запись в нее с помощью указателей так же, как и с динамически выделенной памятью. Использование сопоставления файлов повышает эффективность, так как файл находится на диске, а представление файла находится в памяти.

Асинхронность означает выполнение задачи или операции, которая не блокирует основной поток выполнения программы. Вместо того, чтобы ждать завершения долгой операции, программа может продолжать выполнение других задач или операций. Асинхронность позволяет увеличить производительность и отзывчивость приложений, так как не блокирует пользовательский интерфейс или другие потоки исполнения во время выполнения задач.

Перекрытие операций ввода-вывода: Функции ReadFile и WriteFile также поддерживают перекрытие операций ввода-вывода с помощью структуры OVERLAPPED. Это позволяет выполнять несколько операций ввода-вывода одновременно и асинхронно.

Управление файловыми указателями: Win32 API предоставляет функции для управления файловыми указателями, такие как SetFilePointer и SetFilePointerEx. Они позволяют устанавливать позицию файлового указателя в файле, читать данные из определенного места или записывать данные в определенное место файла.

Работа с устройствами: Win32 API предоставляет функции для работы с устройствами, такие как CreateFile с указанием имени устройства. Это позволяет выполнять операции ввода-вывода непосредственно с устройствами, например, чтение или запись данных в последовательные порты или сетевые устройства.

1. **РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ**

Основой является приложение способное считывать, изменять и сохранять текст (рисунок 1)*.*

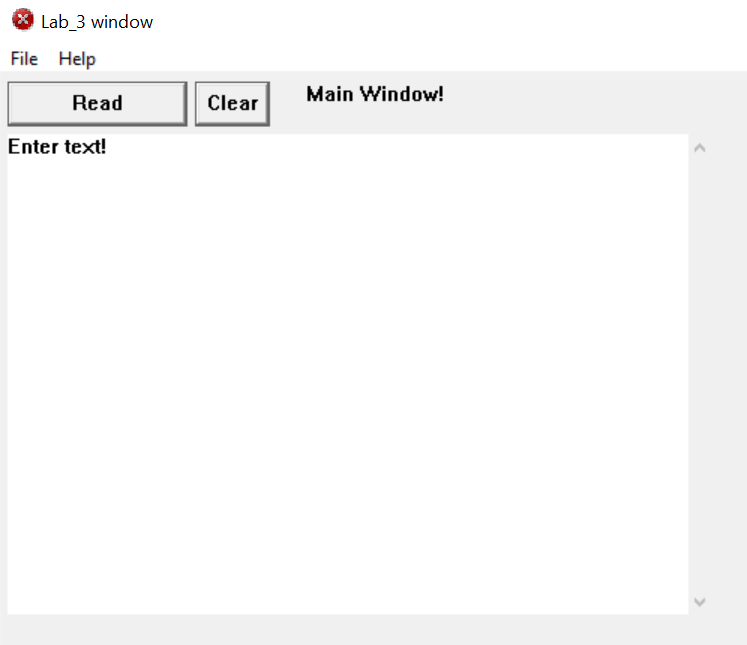


Рисунок 1 – Окно приложения

Была реализована возможность сохранения в файл и последующее его чтение при запуске программы и изменение. (рисунок 2).

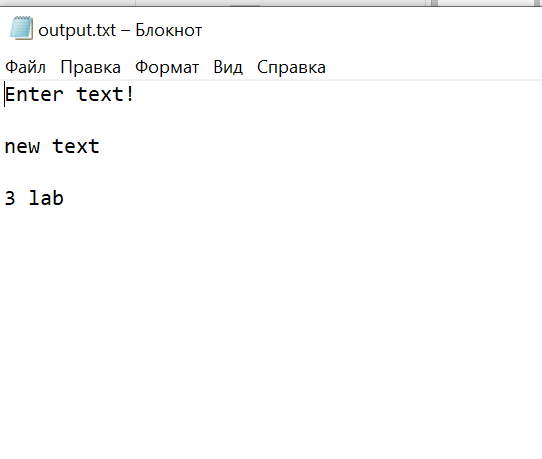


Рисунок 2– Файл *output.txt*

# ЗАКЛЮЧЕНИЕ

В результате лабораторной работы были изучены основные принципы работы с вводом-выводом: управление памятью, контроль асинхронных операций, отображение файлов в память. Для приложения была добавлена возможность сохранения и загрузки в файл, вывода информации о количестве символов.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Основы программирования для Win32 API [Электронный ресурс]. – Режим доступа: <https://dims.karelia.ru/win32/>.
2. Сопоставление файлов [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/windows/win32/memory/file-mapping.

# ПРИЛОЖЕНИЕ А (обязательное) Листинг кода

Листинг 1 – Файл *Lab\_3.cpp*

#ifndef UNICODE

#define UNICODE

#endif

#include <windows.h>

#include "SoftwareDefinitions.h"

#include <string>

/\*

#pragma comment(lib, "ws2\_32.lib")

#pragma comment(lib, "iphlpapi.lib")

#include <WinSock2.h>

#include <WS2tcpip.h>

//#include <iphlpapi.h>

\*/

int WINAPI WinMain(HINSTANCE hInst, HINSTANCE hPrevInst, LPSTR args, int ncmdshow) {

WNDCLASS SoftwareMainClass = NewWindowClass((HBRUSH)COLOR\_WINDOW, LoadCursor(NULL, IDC\_HAND), hInst, LoadIcon(NULL, IDI\_ERROR),

L"MainWndClass", SoftwareMainProcedure);

if (!RegisterClassW(&SoftwareMainClass)) { return -1; }

CreateWindow(L"MainWndClass", L"Lab\_3 window", WS\_OVERLAPPEDWINDOW | WS\_VISIBLE, 100, 100, 750, 500, NULL, NULL, NULL, NULL);

MSG msg = { };

while (GetMessage(&msg, NULL, NULL, NULL)) {

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return 0;

}

WNDCLASS NewWindowClass(HBRUSH BGColor, HCURSOR Cursor, HINSTANCE hInst, HICON Icon, LPCWSTR Name, WNDPROC Procedure) {

WNDCLASS NWC = { 0 };

NWC.hIcon = Icon;

NWC.hCursor = Cursor;

NWC.hInstance = hInst;

NWC.lpszClassName = Name;

NWC.hbrBackground = BGColor;

NWC.lpfnWndProc = Procedure;

return NWC;

}

LRESULT CALLBACK SoftwareMainProcedure(HWND hWnd, UINT msg, WPARAM wp, LPARAM lp) {

switch (msg)

{

case WM\_COMMAND:

switch (wp) {

case OnMenuClicked1:

MessageBoxA(hWnd, "menu 1 was klicked", "Menu worked", MB\_OK);

break;

case OnClear:

SetWindowTextA(hEditControl, "");

break;

case OnRead:

readChars = GetWindowTextA(hEditControl, Buffer, TextBufferSize);

//SetWindowTextA(hStaticControl, Buffer);

SetWindowTextA(hStaticControl, ("Symbols read: " + std::to\_string(readChars)).c\_str());

break;

case OnSave:

SaveData("D:\\OSiSP\\libs\\output.txt");

break;

case OnLoad:

LoadData("D:\\OSiSP\\libs\\output.txt");

break;

case OnExitSoftware:

PostQuitMessage(0);

break;

}

break;

case WM\_CREATE:

MainWndAddMenus(hWnd);

MainWndAddWidgets(hWnd);

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

return DefWindowProc(hWnd, msg, wp, lp);

}

}

void MainWndAddMenus(HWND hwnd) {

HMENU RootMenu = CreateMenu();

HMENU SubMenu = CreateMenu();

AppendMenu(SubMenu, MF\_STRING, OnClear, L"Clear");

AppendMenu(SubMenu, MF\_STRING, OnSave, L"Save");

AppendMenu(SubMenu, MF\_STRING, OnLoad, L"Load");

AppendMenu(SubMenu, MF\_SEPARATOR, NULL, NULL);

AppendMenu(SubMenu, MF\_STRING, OnExitSoftware, L"Exit");

AppendMenu(RootMenu, MF\_POPUP, (UINT\_PTR)SubMenu, L"File");

AppendMenu(RootMenu, MF\_STRING, (UINT\_PTR)SubMenu, L"Help");

SetMenu(hwnd, RootMenu);

}

void MainWndAddWidgets(HWND hWnd) {

hStaticControl = CreateWindowA("static", "Main Window!", WS\_VISIBLE | WS\_CHILD | ES\_CENTER, 200, 5, 100, 30, hWnd, NULL, NULL, NULL);

hEditControl = CreateWindowA("edit", "Enter text!", WS\_VISIBLE | WS\_CHILD | ES\_MULTILINE | WS\_VSCROLL, 5, 40, 470, 320, hWnd, NULL, NULL, NULL);

hErrorControl = CreateWindowA("static", "", WS\_VISIBLE | WS\_CHILD | ES\_CENTER, 500, 50, 100, 30, hWnd, NULL, NULL, NULL);

CreateWindowA("button", "Clear", WS\_VISIBLE | WS\_CHILD | ES\_CENTER, 130, 5, 50, 30, hWnd, (HMENU)OnClear, NULL, NULL);

CreateWindowA("button", "Read", WS\_VISIBLE | WS\_CHILD | ES\_CENTER, 5, 5, 120, 30, hWnd, (HMENU)OnRead, NULL, NULL);

}

void SaveData(LPCSTR path) {

HANDLE SourceFile = CreateFileA(

path,

GENERIC\_WRITE,

0,

NULL,

CREATE\_ALWAYS,

FILE\_ATTRIBUTE\_NORMAL,

NULL);

if (SourceFile == INVALID\_HANDLE\_VALUE)

{

SetWindowTextA(hErrorControl, ("Error with open file"));

return;

}

DWORD saveLenth = GetWindowTextLength(hEditControl) + 1;

char\* data = new char[saveLenth];

saveLenth = GetWindowTextA(hEditControl, data, saveLenth);

DWORD bytesIterated;

WriteFile(SourceFile, data, saveLenth, &bytesIterated, NULL);

CloseHandle(SourceFile);

delete[] data;

SetWindowTextA(hErrorControl, ("File save"));

}

void LoadData(LPCSTR path) {

HANDLE hFile = CreateFileA(

path,

GENERIC\_READ,

FILE\_SHARE\_READ,

NULL,

OPEN\_EXISTING,

FILE\_ATTRIBUTE\_NORMAL,

NULL);

DWORD bytesIterated;

ReadFile(hFile, Buffer, TextBufferSize, &bytesIterated, NULL);

SetWindowTextA(hEditControl, Buffer);

CloseHandle(hFile);

SetWindowTextA(hErrorControl, ("File open"));

}}

Листинг 2 – Файл *SoftwareDefinitions.h*

#pragma once

#define OnMenuClicked1 1

#define OnMenuClicked2 2

#define OnMenuClicked3 3

#define OnExitSoftware 4

#define OnClear 5

#define OnRead 6

#define OnSave 7

#define OnLoad 8

#define TextBufferSize 1024

char Buffer[TextBufferSize];

HWND hEditControl;

HWND hStaticControl;

HWND hErrorControl;

int readChars;

const char\* filePath = "D:\\OSiSP\\libs\\output.txt";

LRESULT CALLBACK SoftwareMainProcedure(HWND hWnd, UINT msg, WPARAM wp, LPARAM lp);

WNDCLASS NewWindowClass(HBRUSH BGColor, HCURSOR Cursor, HINSTANCE hInst, HICON Icon, LPCWSTR Name, WNDPROC Procedure);

void MainWndAddMenus(HWND hwnd);

void MainWndAddWidgets(HWND hwnd);

void SaveData(LPCSTR path);

void LoadData(LPCSTR path);