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. ОТОБРАЖЕНИЕ ФАЙЛОВ В ПАМЯТЬ**

Выполнил:

студент гр. 153503

Татаринов В.В.

Проверил:

Гриценко Н.Ю.

Минск 2023

**СОДЕРЖАНИЕ**

[1 Цель работы 3](#_Toc146836467)

[2 Теоретические сведения 4](#_Toc146836468)

[3 Полученные результаты 6](#_Toc146836469)

[Выводы 7](#_Toc146836470)

[Список использованных источников 8](#_Toc146836471)

[Приложение А (обязательное) листинг кода 9](#_Toc146836472)

# **1 ЦЕЛЬ РАБОТЫ**

Изучить возможности операций ввода и вывода данных в *Win32 API*, организовать асинхронные операции ввода и вывода. Освоить и применить методы управления памятью и расширенные функциональные возможности *Windows* для ввода и вывода. Создать приложение *Win32 API*, реализующее операции асинхронного ввода и вывода данных, используя отображение файлов в память.

**2 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

*Win* *API* представляет собой основной набор интерфейсов прикладного программирования (*API*) *Microsoft*, доступных в операционных системах *Microsoft Windows*. Название *Windows API* в совокупности относится к нескольким различным реализациям платформы, которые часто называются собственными, одним из которых является *Win32 API*. Почти все программы *Windows* взаимодействуют с *Windows API*.

*Win32 API* был изначально разработан для 32-битных версий *Windows*, но в некоторых случаях его можно использовать и в более новых версиях *Windows*. Он представляет собой набор функций и библиотек для разработки приложений под операционные системы *Windows*. Этот *API* ориентирован на 32-битные системы (хотя существует и 64-битная версия, известная как *Win64* *API*), и он предоставляет доступ к основным службам и функциям *Windows*, таким как создание окон, управление ресурсами, работа с файлами и многими другими.

В *Win32* *API* операции ввода и вывода (*I/O*) играют ключевую роль в разработке приложений, так как они позволяют программам взаимодействовать с файлами, устройствами и сетевыми ресурсами. Операции *I/O* предоставляют широкий набор функций и механизмов для эффективной и гибкой работы с данными.

*Win32 API* предоставляет низкоуровневый доступ к функциональности *Windows*. Это позволяет создавать более гибкие и настраиваемые приложения, но также требует более тщательной работы с деталями.

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. Это также позволяет процессу эффективно работать с большим файлом данных, например с базой данных, без необходимости сопоставлять весь файл с памятью. Несколько процессов также могут использовать сопоставленные с памятью файлы для совместного использования данных.

В ходе выполнения данной лабораторной работы использовались следующие инструменты и концепции:

* *CreateFile* в *Win32* *API* используется для создания или открытия файла, устройства или каталога и возврата дескриптора файла, который может использоваться для выполнения операций с этим файлом. *CreateFile* является одной из основных функций для работы с файлами в *Win32* *API*. В качестве параметров передается строка, представляющая путь к файлу или устройству,

желаемый доступ к файлу (чтение, запись, выполнение и т. д.), режим совместного доступа к файлу другими процессами, действие, которое следует предпринять, если файл уже существует или должен быть создан.

* *SetFilePointer* предназначенна для управления указателем позиции в файле. Эта функция используется для перемещения указателя позиции в файле, чтобы установить текущую позицию, с которой будут считываться или записываться данные.
* *SetEndOfFile* в *Win32* *API* используется для установки конца файла на заданную позицию. Это позволяет изменять размер файла путем усечения или расширения файла. Обычно она используется для изменения размера файла на определенное количество байтов.
* *CreateFileMapping* используется для создания объекта файлового отображения в памяти. В качестве параметром дескриптор файла, который будет связан с файловым отображением, размер файла (в байтах), который будет отображен.

# **3 ПОЛУЧЕННЫЕ РЕЗУЛЬТАТЫ**

В результате выполнения лабораторной работы было создано оконное приложение, позволяющее вводить данные и асинхронно записывать их в файл, используя отображение файла в память. В качестве входных данных пользователем вводится любая строка. При нажатии на кнопку записи, происходит запись данных в файл, при нажатии на кнопку чтения, происходит чтение из файла.

Интерфейс программы представляет собой 2 поля для ввода и кнопки чтения и записи (рисунок 1).

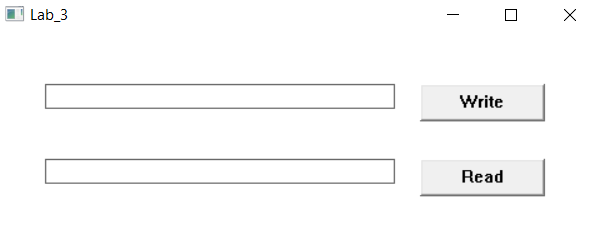


Рисунок 1 – Интерфейс приложения

Результат записи и чтения из файла (рисунок 2).

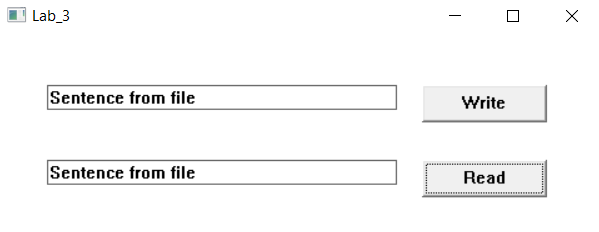


Рисунок 2 – Результат работы программы

# **ВЫВОДЫ**

В результате выполнения лабораторной работы были изучены возможности операций ввода и вывода данных в *Win32* *API*. Было создано приложение *Win32* *API*, реализующее операции асинхронного ввода и вывода данных, используя отображение файлов в память.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Чтение из файлов и запись в файлы [Электронный ресурс]. – Режим доступа:  [https://learn.microsoft.com/ru-ru/windows/win32/fileio/reading-from-and-writing-to-files](https://learn.microsoft.com/ru-ru/windows/win32/api%20–%20Дата%20доступа%2023.09.2023).
2. Сопоставление файлов [Электронный ресурс]. – Режим доступа:  [https://learn.microsoft.com/ru-ru/windows/win32/memory/file-mapping](https://learn.microsoft.com/ru-ru/windows/win32/api%20–%20Дата%20доступа%2023.09.2023).

# **ПРИЛОЖЕНИЕ А (обязательное) Листинг кода**

Листинг 1 **–** Файл *Lab\_3\_WINAPI32.cpp*:

#include <windows.h>

#include <cstdlib>

#define ID\_FILE\_WRITE 100

#define ID\_FILE\_READ 101

HMENU hMenu;

HWND hResult, hSource;

wchar\_t data[100];

int numBytesToWrite;

wchar\_t res[100];

HANDLE writeCompleteEvent;

void AddControls(HWND hWnd);

void AddMenus(HWND hWnd);

void WriteToFileAsync(HWND hWnd);

void ReadFromFileAsync(HWND hWnd);

DWORD WINAPI WriteAsync(LPVOID filePointer);

DWORD WINAPI ReadAsync(LPVOID filePointer);

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

int WINAPI WinMain(HINSTANCE hInst, HINSTANCE hPrevInst, LPSTR arts, int ncmdshow)

{

WNDCLASSW wc = { 0 };

wc.hbrBackground = CreateSolidBrush(RGB(255, 255, 255));

wc.hCursor = LoadCursor(NULL, IDC\_ARROW);

wc.hInstance = hInst;

wc.lpszClassName = L"Class";

wc.lpfnWndProc = WindowProcedure;

if (!RegisterClassW(&wc))

return -1;

CreateWindowW(L"Class", L"Lab\_3", WS\_OVERLAPPEDWINDOW | WS\_VISIBLE, 20, 20, 500, 200, NULL, NULL, NULL, NULL);

MSG msg = { 0 };

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

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return 0;

}

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

{

switch (msg)

{

case WM\_COMMAND:

switch (wp)

{

case ID\_FILE\_WRITE:

{

WriteToFileAsync(hWnd);

break;

}

case ID\_FILE\_READ:

{

ReadFromFileAsync(hWnd);

break;

}

}

break;

case WM\_CREATE:

AddMenus(hWnd);

AddControls(hWnd);

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

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

}

}

void AddMenus(HWND hWnd)

{

hMenu = CreateMenu();

SetMenu(hWnd, hMenu);

}

void AddControls(HWND hWnd)

{

hSource = CreateWindowW(L"Edit", L"", WS\_VISIBLE | WS\_CHILD | WS\_BORDER, 40, 40, 280, 20, hWnd, NULL, NULL, NULL);

CreateWindowW(L"Button", L"Write", WS\_VISIBLE | WS\_CHILD, 340, 40, 100, 30, hWnd, (HMENU)ID\_FILE\_WRITE, NULL, NULL);

hResult = CreateWindowW(L"Edit", L"", WS\_VISIBLE | WS\_CHILD | WS\_BORDER, 40, 100, 280, 20, hWnd, NULL, NULL, NULL);

CreateWindowW(L"Button", L"Read", WS\_VISIBLE | WS\_CHILD, 340, 100, 100, 30, hWnd, (HMENU)ID\_FILE\_READ, NULL, NULL);

}

void WriteToFileAsync(HWND hWnd)

{

LPCWSTR filename = L"mapped\_file.txt";

HANDLE fileHandle = CreateFile(

filename,

GENERIC\_READ | GENERIC\_WRITE,

FILE\_SHARE\_READ | FILE\_SHARE\_WRITE,

NULL,

CREATE\_ALWAYS,

FILE\_ATTRIBUTE\_NORMAL | FILE\_FLAG\_OVERLAPPED,

NULL

);

GetWindowText(hSource, data, 100);

numBytesToWrite = (wcslen(data) + 1) \* sizeof(wchar\_t);

DWORD fileSize = numBytesToWrite;

SetFilePointer(fileHandle, fileSize, NULL, FILE\_BEGIN);

SetEndOfFile(fileHandle);

HANDLE fileMapping = CreateFileMapping(

fileHandle,

NULL,

PAGE\_READWRITE,

0,

fileSize,

NULL

);

LPVOID filePointer = MapViewOfFile(fileMapping, FILE\_MAP\_WRITE, 0, 0, 0);

HANDLE writeThread = CreateThread(NULL, 0, WriteAsync, filePointer, 0, NULL);

WaitForSingleObject(writeThread, INFINITE);

CloseHandle(writeThread);

UnmapViewOfFile(filePointer);

CloseHandle(fileMapping);

CloseHandle(fileHandle);

}

void ReadFromFileAsync(HWND hWnd)

{

LPCWSTR filename = L"mapped\_file.txt";

HANDLE fileHandle = CreateFile(

filename,

GENERIC\_READ | GENERIC\_WRITE,

FILE\_SHARE\_READ | FILE\_SHARE\_WRITE,

NULL,

OPEN\_EXISTING,

FILE\_ATTRIBUTE\_NORMAL | FILE\_FLAG\_OVERLAPPED,

NULL

);

DWORD fileSize = numBytesToWrite;

SetFilePointer(fileHandle, fileSize, NULL, FILE\_BEGIN);

SetEndOfFile(fileHandle);

HANDLE fileMapping = CreateFileMapping(

fileHandle,

NULL,

PAGE\_READWRITE,

0,

fileSize,

NULL

);

LPVOID filePointer = MapViewOfFile(fileMapping, FILE\_MAP\_WRITE, 0, 0, 0);

HANDLE readThread = CreateThread(NULL, 0, ReadAsync, filePointer, 0, NULL);

WaitForSingleObject(readThread, INFINITE);

CloseHandle(readThread);

SetWindowText(hResult, res);

UnmapViewOfFile(filePointer);

CloseHandle(fileMapping);

CloseHandle(fileHandle);

}

DWORD WINAPI WriteAsync(LPVOID filePointer)

{

CopyMemory(filePointer, data, numBytesToWrite);

SetEvent(writeCompleteEvent);

return 0;

}

DWORD WINAPI ReadAsync(LPVOID filePointer)

{

WaitForSingleObject(writeCompleteEvent, INFINITE);

wchar\_t\* Buffer = new wchar\_t[numBytesToWrite / sizeof(wchar\_t)];

CopyMemory(Buffer, filePointer, numBytesToWrite);

wcscpy\_s(res, Buffer);

wcscat\_s(res, L"\0");

return 0;

}