**Лабораторная работа №2**

**Задание для выполнения**

Разработать программу, реализующую следующие задачи:

1. Создать текстовый файл (можно с использованием notepad).

2. Создасть объект File на базе созданного в предыдущем пункте файла, используя АРI-функцию CreateFile. Вывести значение дескриптора объекта File.

3. Используя дескриптор объекта File-mapping, а также API-функцию MapViewOfFile отобразить части файла в память. Данная функция назначает область виртуальной памяти, выделяемой этому файлу. Базовый адрес выделенной области памяти является дескриптором представления этой области в виде отображения файла.

4. Используя базовый адрес и функцию CopyMemory прочитайте информацию из отображаемого файла. Согласно индивидуальному варианту задания измените текстовый файл и запишите информацию в этот же файл: Изменить регистр содержимого файла

5. Закрыть все дескрипторы.

**Решение:**

import ctypes

import ctypes.wintypes as wintypes

import os

# === Шаг 1. Создание текстового файла

filename = "example.txt"

with open(filename, "w", encoding="utf-8") as f:

    f.write("It is lab 2 of OS")  # Исходный текст

# === Шаг 2. Открытие файла через CreateFile

kernel32 = ctypes.windll.kernel32

GENERIC\_READ = 0x80000000

GENERIC\_WRITE = 0x40000000

OPEN\_EXISTING = 3

FILE\_SHARE\_READ = 1

FILE\_SHARE\_WRITE = 2

hFile = kernel32.CreateFileW(

    filename,

    GENERIC\_READ | GENERIC\_WRITE,

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

    None,

    OPEN\_EXISTING,

    0,

    None

)

if hFile == -1:

    raise ctypes.WinError()

print(f"Дескриптор файла: {hFile}")

# === Шаг 3. Создание объекта отображения файла

PAGE\_READWRITE = 0x04

hMap = kernel32.CreateFileMappingW(

    hFile,

    None,

    PAGE\_READWRITE,

    0,

    0,

    None

)

if not hMap:

    raise ctypes.WinError()

# === Шаг 4. Отображение файла в память

FILE\_MAP\_ALL\_ACCESS = 0xF001F

lpBaseAddress = kernel32.MapViewOfFile(

    hMap,

    FILE\_MAP\_ALL\_ACCESS,

    0,

    0,

    0

)

if not lpBaseAddress:

    raise ctypes.WinError()

# === Чтение содержимого

length = os.path.getsize(filename)

buffer = (ctypes.c\_char \* length).from\_address(lpBaseAddress)

original\_text = buffer.raw.decode('utf-8')

print(f"Исходное содержимое файла: {original\_text}")

# === Изменение регистра и запись обратно

modified\_text = original\_text.swapcase()

modified\_bytes = modified\_text.encode('utf-8')

# Перезаписываем память

ctypes.memmove(lpBaseAddress, modified\_bytes, len(modified\_bytes))

print(f"Изменённое содержимое файла: {modified\_text}")

# === Шаг 5. Закрытие всех дескрипторов

kernel32.UnmapViewOfFile(lpBaseAddress)

kernel32.CloseHandle(hMap)

kernel32.CloseHandle(hFile)

**Результат:**

