БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий

и робототехники

Кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»

О Т Ч Ё Т

о лабораторной работе № 4

Дисциплина

«Системное программирование»

Тема

«Работа с файлами»

Выполнил: студент гр. 10702217 Храмков Д. С.

Проверил: Разорёнов Н. А.

Минск 2019

***Лабораторная работа № 4***

**РАБОТА С ФАЙЛАМИ**

**Цель работы:** изучить основы работы с двоичным и текстовым файлами на базе WIN32 API.

**Изучаемые вопросы**

1. Создание текстового файла. Структура текстового файла.
2. Запись/чтение текстового файла.
3. Создание двоичного файла. Структура двоичного файла.
4. Запись/чтение двоичного файла.
5. Атрибуты файла. Чтение/установка атрибутов файла.
6. Преобразование информации при записи и чтении в/из файла.
7. Внутреннее представление информации разного типа в оперативной и дисковой памяти (файлах).
8. Дампы памяти.

***Постановка задачи***

Разработать WIN32 Application с диалогами, которое должно обеспечить:

1. Запись в оба типа файлов данных следующих типов (значение данных задаются через диалоговое окно):
   * BYTE, UINT;
   * INT32/INT64;
   * WORD, DWORD;
   * LONG, BOOL;
   * Float; Double;
   * Char;
   * Строки (в файле должно быть три и более строки)
2. Чтение данных из двоичного файла и их отображение в диалоговом окне.
3. Чтение данных из текстового файла и их отображение в диалоговом окне.
4. Отображение дампов значений данных файлов.
5. Для сохранения файла и его загрузки использовать стандартные диалоги посредством вызова функций GetOpenFileName() и GetSaveFileName().

В результате выполнения работ студент должен уметь анализировать кодировку данных в символьном/текстовом и двоичном форматах на основе дампов памяти (файлов). Дамп памяти – отображение байтов памяти (оперативной или дисковой) в шестнадцатеричной системе.

1. **Создание текстового файла. Структура текстового файла.**

**Текстовый файл создаётся функцией** CreateFile(L"files\\text\\textFile.txt", GENERIC\_READ | GENERIC\_WRITE, 0, NULL, CREATE\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL). Первый параметр отвечает за имя файла, второй – за режимы доступа к нему, третий определяет, будет ли файл использоваться кем-то ещё во время выполнения программы, четвёртый отвечает за параметры безопасности, пятый параметр говорит создавать файл постоянно (на случай если по заданному пути файла нет), шестой отвечает за атрибуты и флаги файла, а седьмой задаёт шаблон, по которому файл создаётся.

Текстовый файл является последовательностью символов, зачастую сгруппированных в строки. Данные в таких файлах хранятся в виде цепочки ASCII-кодов и могут обрабатываться любым текстовым редакотром. Каждая строка завершается маркером конца строки. На практике такой маркер представляет собой последовательность из двух символов: \r (маркер возврата каретки) и \n (маркер перевода строки). Конец файла обозначается нуль-символом: \0.

HANDLE hFile = CreateFile(name, GENERIC\_READ, 0, NULL, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, NULL);

1. **Запись/чтение текстового файла.**

**Сначала выбираем файл, из которого будет осуществляться чтение функцией** GetOpenFileName(&file)**.**

**Для записи в файл используется функция WriteFile(hFile, text, sizeof(text), 0, NULL), которая при помощи обработчика hFile записывает данные из буфера text в выбранный при помощи функции GetSaveFileName(&file) файл.**

if (!GetSaveFileName(&file)) {

MessageBox(hDlg, L"Something went wrong", L"Error", MB\_OK | MB\_ICONERROR);

return 0;

}

HWND edit = GetDlgItem(hDlg, IDC\_TXTEDIT);

GetWindowTextA(edit, text, MaxLength);

hFile = CreateFile(name, GENERIC\_WRITE, 0, NULL, CREATE\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL);

WriteFile(hFile, text, sizeof(text), 0, NULL);

**Для чтения из текстового файла используется функция** ReadFile(hFile, text, MaxLength, &nCharRead, NULL), которая с помощью обработчика hFile передаёт в буфер text содержимое файла, выбранного с помощью функции GetOpenFileName(&file).

if (!GetOpenFileName(&file)) {

MessageBox(hDlg, L"Something went wrong", L"Error", MB\_OK | MB\_ICONERROR);

return 0;

}

HWND edit = GetDlgItem(hDlg, IDC\_TXTEDIT);

hFile = CreateFile(name, GENERIC\_READ, 0, NULL, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, NULL);

ReadFile(hFile, text, MaxLength, &nCharRead, NULL);

SetDlgItemTextA(hDlg, IDC\_TXTEDIT, text);

1. **Создание двоичного файла. Структура двоичного файла.**

**Двоичные файлы представляют из себя последовательность битов. По сути, двоичный файл является аналогом оперативной памяти.**

hFile = CreateFile(L"files\\binary\\binFile.bin", GENERIC\_READ | GENERIC\_WRITE, 0, NULL, CREATE\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL);

1. **Запись/чтение двоичного файла.**

**Чтение двоичного файла:**

if (!GetOpenFileName(&file)) {

MessageBox(hDlg, L"Something went wrong", L"Error", MB\_OK | MB\_ICONERROR);

return 0;

}

edit = GetDlgItem(hDlg, IDC\_BINEDIT);

hFile = CreateFile(name, GENERIC\_READ, 0, NULL, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, NULL);

BYTE byte;

UINT uint;

INT32 int32;

INT64 int64;

ReadFile(hFile, &byte, sizeof(byte), 0, NULL);

ReadFile(hFile, &uint, sizeof(uint), 0, NULL);

ReadFile(hFile, &int32, sizeof(int32), 0, NULL);

ReadFile(hFile, &int64, sizeof(int64), 0, NULL);

**Запись в двоичный файл:**

if (!GetSaveFileName(&file)) {

MessageBox(hDlg, L"Something went wrong", L"Error", MB\_OK | MB\_ICONERROR);

return 0;

}

hFile = CreateFile(name, GENERIC\_WRITE, 0, NULL, CREATE\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL);

BYTE byte = 123;

UINT uint = 2893;

INT32 int32 = -673649;

INT64 int64 = 134;

WriteFile(hFile, &byte, sizeof(byte), 0, NULL);

WriteFile(hFile, &uint, sizeof(uint), 0, NULL);

WriteFile(hFile, &int32, sizeof(int32), 0, NULL);

WriteFile(hFile, &int64, sizeof(int64), 0, NULL);

1. **Атрибуты файла. Чтение/установка атрибутов файла.**

**Для получения атрибутов файла используется функция GetFileAttributes(name), для того, чтобы задать атрибуты – SetFileAttributes(name, FILE\_ATTRIBUTES\_HIDDEN).**

if (!GetOpenFileName(&file)) {

GetFileAttributes(name);

}

break;

if (!GetOpenFileName(&file)) {

SetFileAttributes(name, FILE\_ATTRIBUTE\_NORMAL);

}

break;

1. **Преобразование информации при записи и чтении в/из файла.**

**Для считывания информации из файла используется функция** ReadFile(), а для записи – WriteFile(). При записи/чтении в/из файла каждый тип данных записывается и считывается отдельно, выделяя в качестве размера для записи файла размер записываемого типа данных.

ReadFile(hFile, text, MaxLength, &nCharRead, NULL);

WriteFile(hFile, &byte, sizeof(byte), 0, NULL);

1. **Внутреннее представление информации разного типа в оперативной и дисковой памяти (файлах).**

**В оперативной памяти информация хранится и обрабатывается в шестнадцатеричном виде, завершается нуль-символом. Представление информации в файлах зависит от типа файлов. К примеру, в текстовом файле информация представлена в виде строк, завершающихся символами возврата каретки и маркером перевода строки.**

**Каждый тип данных имеет свой размер, который надо учитывать при работе с файлами.**

WriteFile(hFile, &c, sizeof(c), 0, NULL);

1. **Дампы памяти.**

**Дамп памяти – содержимое оперативной памяти процесса, ядра или всей операционной системы.**

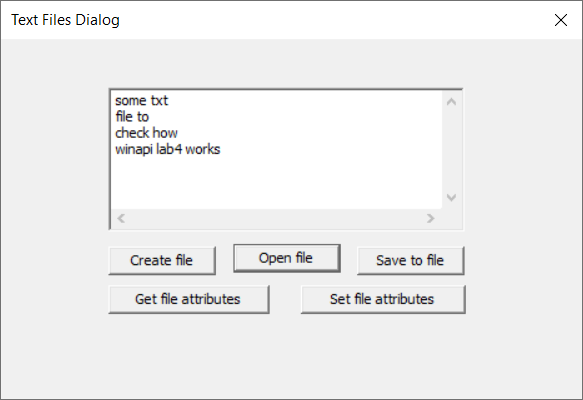
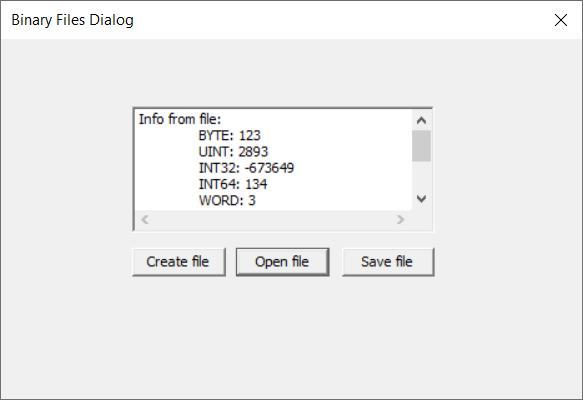
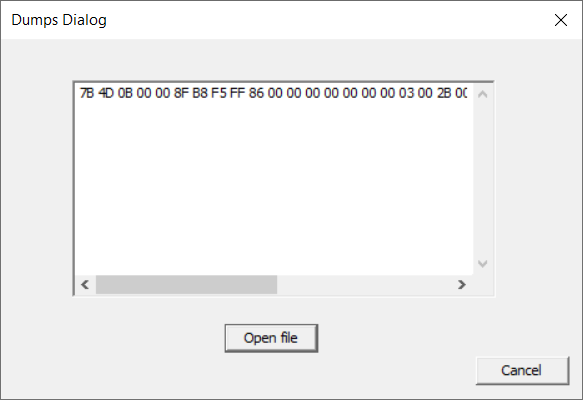
CHAR \*q = new CHAR[size];

for (int i = 0, k = 0; i < size; i++) {

k += sprintf\_s(q + k, size, "%02x ", (unsigned char)text[i]);

}

**Результаты работы программы:**

**Выводы:**

1. **Научились записывать и выводить информацию в/из файл(а);**
2. **Изучили способ отображения дампов значений файлов;**
3. **Научились устанавливать указатели в файле для чтения/записи;**
4. **Изучили структуру файлов.**