МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М. Є. Жуковського

«Харківський авіаційний інститут»

Факультет радіоелектроніки, комп'ютерних систем та інфокомунікацій

Кафедра комп'ютерних систем, мереж і кібербезпеки

# Лабораторна робота № 1

з «Системне програмування»

(назва дисципліни)

на тему: «Кодування файдів ANCI та UNICODE»

Виконав: студент 4 курсу групи № 535-ст2 напряму підготовки (спеціальності) 123 - Комп’ютерна інженерія

(шифр і назва напряму підготовки (спеціальності))

Саенко О.О.

(прізвище й ініціали студента)

Прийняв: ст. викладач \_\_\_\_\_\_

Годунов Олександр Сергійович

(посада, науковий ступінь, прізвище й ініціали)

Національна шкала: \_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_

Оцінка: ECTS \_\_\_\_\_

**Мета роботи:**

1. Вивчити системні виклики Win32 API, які дозволяють отримувати інформацію про помилку.
2. Вивчити функції для збору інформації про систему.

**Постановка завдання:**

1. Написати програму, яка генерує помилку під час виконання системного виклику та виводить системний опис цієї помилки. Програма має мати два режими роботи, які визначаються ключем, переданим через командний рядок (-e - виведення помилки, -s - виведення інформації про систему).
2. Написати програму, яка дозволяє перекодувати текст з кодування ASCII в Unicode і навпаки. В якості вхідних даних використовується текстовий файл. Програма при запуску отримує параметр командного рядка, який визначає початкове кодування файлу (-a - файл у кодуванні ANSI, -u - файл у кодуванні Unicode).

Виконаний код:

#include "pch.h"

#define BUFFER\_SIZE 200

void ErrorExit(LPTSTR lpszFunction)

{

// Retrieve the system error message for the last-error code

LPVOID lpMsgBuf;

LPVOID lpDisplayBuf;

DWORD dw = GetLastError();

FormatMessage(

FORMAT\_MESSAGE\_ALLOCATE\_BUFFER |

FORMAT\_MESSAGE\_FROM\_SYSTEM |

FORMAT\_MESSAGE\_IGNORE\_INSERTS,

NULL,

dw,

MAKELANGID(LANG\_NEUTRAL, SUBLANG\_DEFAULT),

(LPTSTR)&lpMsgBuf,

0, NULL);

// Display the error message and exit the process

lpDisplayBuf = (LPVOID)LocalAlloc(LMEM\_ZEROINIT,

(lstrlen((LPCTSTR)lpMsgBuf) + lstrlen((LPCTSTR)lpszFunction) + 40) \* sizeof(TCHAR));

StringCchPrintf((LPTSTR)lpDisplayBuf,

LocalSize(lpDisplayBuf) / sizeof(TCHAR),

TEXT("%s failed with error %d: %s"),

lpszFunction, dw, lpMsgBuf);

MessageBox(NULL, (LPCTSTR)lpDisplayBuf, TEXT("Error"), MB\_OK);

LocalFree(lpMsgBuf);

LocalFree(lpDisplayBuf);

ExitProcess(dw);

}

void asciiToUnicode() {

HANDLE firstFile = CreateFile(L"inFileAnci.txt",

FILE\_ALL\_ACCESS,

FILE\_SHARE\_READ,

NULL,

OPEN\_EXISTING,

FILE\_ATTRIBUTE\_NORMAL,

NULL);

HANDLE secondFile = CreateFile(L"outFileUni.txt",

FILE\_ALL\_ACCESS,

FILE\_SHARE\_READ,

NULL,

OPEN\_EXISTING,

FILE\_ATTRIBUTE\_NORMAL,

NULL);

LPSTR lpBuffer = new CHAR[BUFFER\_SIZE];

LPWSTR lpwBuffer = new WCHAR[BUFFER\_SIZE];

DWORD dwBytesRead = 0;

ReadFile(firstFile,

lpBuffer,

BUFFER\_SIZE - 1,

&dwBytesRead,

NULL

);

MultiByteToWideChar(CP\_ACP,

0,

lpBuffer,

dwBytesRead,

lpwBuffer,

dwBytesRead

);

WriteFile(secondFile,

lpwBuffer,

dwBytesRead\*2,

&dwBytesRead,

NULL);

TCHAR functionName[] = TEXT("asciiToUnicode");

if(GetLastError() != 0) ErrorExit(functionName);

}

void unicodeToAscii() {

HANDLE firstFile = CreateFile(L"inFileUni.txt",

FILE\_ALL\_ACCESS,

FILE\_SHARE\_READ,

NULL,

OPEN\_EXISTING,

FILE\_ATTRIBUTE\_NORMAL,

NULL);

HANDLE secondFile = CreateFile(L"outFileAnci.txt",

FILE\_ALL\_ACCESS,

FILE\_SHARE\_READ,

NULL,

OPEN\_EXISTING,

FILE\_ATTRIBUTE\_NORMAL,

NULL);

LPSTR lpBuffer = new CHAR[BUFFER\_SIZE];

LPWSTR lpwBuffer = new WCHAR[BUFFER\_SIZE];

TCHAR functionName[] = TEXT("unicodeToAscii");

DWORD dwBytesRead = 0;

ReadFile(firstFile,

lpwBuffer,

BUFFER\_SIZE - 1,

&dwBytesRead,

NULL

);

INT size = WideCharToMultiByte(CP\_ACP,

0,

lpwBuffer,

dwBytesRead / sizeof(WCHAR),

NULL,

0,

NULL,

NULL

);

WideCharToMultiByte(CP\_ACP,

0,

lpwBuffer,

dwBytesRead/sizeof(WCHAR),

lpBuffer,

size,

NULL,

NULL

);

WriteFile(secondFile,

lpBuffer,

size,

&dwBytesRead,

NULL);

if (GetLastError() != 0) ErrorExit(functionName);

}

void systemInfo() {

SYSTEM\_INFO siSysInfo;

// Copy the hardware information to the SYSTEM\_INFO structure.

GetSystemInfo(&siSysInfo);

// Display the contents of the SYSTEM\_INFO structure.

printf("Hardware information: \n");

printf(" OEM ID: %u\n", siSysInfo.dwOemId);

printf(" Number of processors: %u\n",

siSysInfo.dwNumberOfProcessors);

printf(" Page size: %u\n", siSysInfo.dwPageSize);

printf(" Processor type: %u\n", siSysInfo.dwProcessorType);

printf(" Minimum application address: %lx\n",

siSysInfo.lpMinimumApplicationAddress);

printf(" Maximum application address: %lx\n",

siSysInfo.lpMaximumApplicationAddress);

printf(" Active processor mask: %u\n",

siSysInfo.dwActiveProcessorMask);

}

int main(int argc, char\* argv[])

{

if (argc == 2 && \_stricmp(argv[1], "-e") == 0)

{

LocalAlloc(LHND, 90000000000);

// Вызываем функцию ErrorExit для вывода описания ошибки 8

TCHAR functionName[] = TEXT("createError");

ErrorExit(functionName); // Замените "YourFunctionNameHere" на имя вашей функции

}

else if (argc == 2 && \_stricmp(argv[1], "-s") == 0) {

systemInfo();

}

else if (argc == 2 && \_stricmp(argv[1], "-a") == 0) {

asciiToUnicode();

}

else if (argc == 2 && \_stricmp(argv[1], "-u") == 0) {

unicodeToAscii();

}

return 0;

}

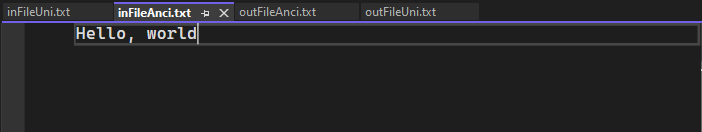


Рисунок 1 – Ввідні дані у кодуванні ANCI

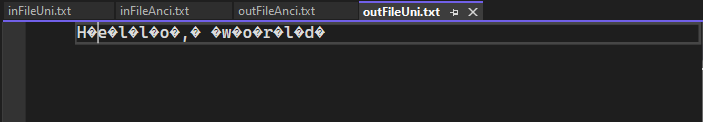


Рисунок 2 - Вихідні дані у кодуванні Unicode

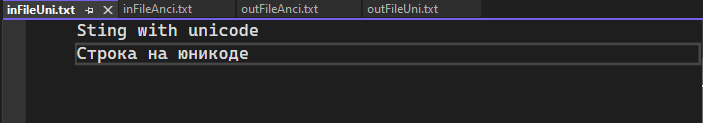


Рисунок 3 - Ввідні дані у кодуванні Unicode

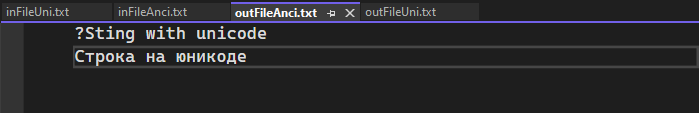


Рисунок 4 - Вихідні дані у кодуванні ANCI

Таблиця тестових даних для перекодування тексту:

|  |  |  |
| --- | --- | --- |
| **Назва тесту** | **Вхідні дані** | **Вихідні дані** |
| Test 1 | ASCII-текст у файлі input.txt | Unicode-текст у файлі output.txt |
| Test 2 | Unicode-текст у файлі input.txt | ASCII-текст у файлі output.txt |

**Висновок:**

Під час виконання лабораторної роботи були розроблені дві програми для опанування системних викликів Win32 API та функцій для збору інформації про систему.