Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования «Московский технический университет связи и информатики»

Лабораторная №1

«Функции получения системной информации»

Выполнила студентка

группы БВТ1501

Козлова Анна

8 вариант

МОСКВА, 2018

**Цель работы:**

Получение практических навыков по программированию в Win32 API

с использованием аппаратных и системных функций.

**Описание дополнительных функций:**

**ExitWindowsEx**. Завершает работу интерактивного пользователя, завершает работу системы или завершает работу и перезапускает систему. Отправляет сообщение WM\_QUERYENDSESSION всем приложениям, чтобы определить, могут ли они быть завершены.

Синтаксис:

BOOL WINAPI ExitWindowsEx(

\_In\_ UINT  uFlags,

\_In\_ DWORD dwReason

);

**GetKBCodePage.** Считывает загpуженную в данный момент таблицу OEM/ANSI.

Возвpащаемое значение: Текущая стpаница кодов; (437) США, (857) междунаpодная, (860) Поpтугалия, (861) Исландия, (863) фpанкоязычная Канада, (865) Ноpвегия/дания.

Оставлена только для совместимости с 16-битными системами.

**GetSystemDefaultLCID**. Возвращает идентификатор языка системы языка.

Синтаксис:

LCID GetSystemDefaultLCID(void);

После этого, имея этот идентификатор, можно запросить параметры системы, используя GetLocaleInfo.

**GetDoubleClickTime.** Получает Текущее время двойного щелчка мыши. Двойной щелчок-это серия из двух щелчков кнопки мыши, второй происходит в течение указанного времени после первого. Время двойного щелчка - это максимальное количество миллисекунд, которое может произойти между первым и вторым щелчком двойного щелчка. Максимальное время двойного щелчка - 5000 миллисекунд.

Синтаксис:

UINT WINAPI GetDoubleClickTime(void);

**Код:**

#include <windows.h>

#include <iostream>

#include <conio.h>

#define TCHAR wchar\_t

#define cout wcout

#define cin wcin

#define TEXT(p) L##p

using namespace std;

int main() {

setlocale(LC\_ALL, "Russian");

TCHAR buffer[512];

DWORD size = 512;

OSVERSIONINFO osversion;

ACCESSTIMEOUT access;

int sysparam;

if(!GetComputerName(buffer, &size))

cout << TEXT("error of GetComputerName");

cout << TEXT("Имя компьютера: \t\t\t") << buffer << endl;

if (!GetSystemDirectory(buffer, 512))

cout << TEXT("error of GetSystemDirectory");

cout << TEXT("Путь к системному каталогу: \t\t") << buffer << endl;

if(!GetWindowsDirectory(buffer, 512))

cout << TEXT("error of GetWindowsDirectory");

cout << TEXT("Путь к каталогу: \t\t\t") << buffer << endl;

if (!GetTempPath(512, buffer))

cout << TEXT("error of GetTempPath");

cout << TEXT("Путь к каталогу врем файлов: \t\t") << buffer << endl << endl;

memset(&osversion, 0, sizeof(OSVERSIONINFO));

osversion.dwOSVersionInfoSize = sizeof(OSVERSIONINFO);

GetVersionEx((OSVERSIONINFO\*)&osversion);

cout << TEXT("Версия системы: \t\t\t") << osversion.dwMajorVersion << "." << osversion.dwMinorVersion << "." << osversion.dwBuildNumber << endl

<< TEXT("Идентификатор платформы ОС: \t\t") << osversion.dwPlatformId << "

(VER\_PLATFORM\_WIN32\_NT)" << endl << endl;

cout << TEXT("Кол-во клавиш мыши: \t\t\t") << GetSystemMetrics(43) << endl

<< TEXT("Ширина и высота экрана: \t\t") << GetSystemMetrics(0) << " " <<

GetSystemMetrics(1) << endl

<< TEXT("Высота стандартной области заголовка: \t") << GetSystemMetrics(4)

<< endl << endl;

access.cbSize = sizeof(ACCESSTIMEOUT);

SystemParametersInfo(SPI\_GETACCESSTIMEOUT, sizeof(ACCESSTIMEOUT), &access, 0);

cout << TEXT("Временные интервалы: \t\t\t") << access.iTimeOutMSec << endl;

SystemParametersInfo(SPI\_GETFLATMENU, 0, &sysparam, 0);

cout << TEXT("Flat меню: \t\t\t\t") << sysparam << endl;

SystemParametersInfo(SPI\_GETBEEP, 0, &sysparam, 0);

cout << TEXT("Признак разрешения звуковых сигналов: \t") << sysparam << endl <<

endl;

int lpaElements[] = { COLOR\_MENUTEXT, COLOR\_MENU };

COLORREF color[] = { GetSysColor(COLOR\_MENUTEXT), GetSysColor(COLOR\_MENU) };

COLORREF newColor[] = { RGB(255, 0, 0), RGB(0, 255, 0) };

SetSysColors(2, lpaElements, newColor);

cout << TEXT("Изменён цвет рамки активного окна и фон меню.") << endl;

SYSTEMTIME time;

GetLocalTime(&time);

cout << endl << TEXT("Местное время: \t\t\t\t")

<< time.wDay << "." << time.wMonth << "." << time.wYear << endl;

cout << TEXT("Время работы текущего сеанса: \t\t") << GetTickCount() << endl <<

endl;

cout << TEXT("Текущая кодовая страница: \t\t") << GetKBCodePage() << endl;

cout << TEXT("Локальный ID системы: \t\t\t") << GetSystemDefaultLCID() << endl;

//только в 16битной системе

//int caretTime = GetCaretBlinkTime();

//cout << TEXT("Частота мерцания каретки изменена на 100 c ") << caretTime <<

endl;

//SetCaretBlinkTime(100);

int clickTime = GetDoubleClickTime();

SetDoubleClickTime(10000);

cout << TEXT("Время двойного щелчка изменено на 1000 с ") << clickTime << endl;

//выключение виндовс

HANDLE hToken;

TOKEN\_PRIVILEGES tkp;

//получить токен

OpenProcessToken(GetCurrentProcess(),

TOKEN\_ADJUST\_PRIVILEGES | TOKEN\_QUERY, &hToken);

//получить LUID

LookupPrivilegeValue(NULL, SE\_SHUTDOWN\_NAME,

&tkp.Privileges[0].Luid);

tkp.PrivilegeCount = 1; //привелегия

tkp.Privileges[0].Attributes = SE\_PRIVILEGE\_ENABLED;

//привелегии для выключения

AdjustTokenPrivileges(hToken, FALSE, &tkp, 0,

(PTOKEN\_PRIVILEGES)NULL, 0);

//выключить

ExitWindowsEx(EWX\_SHUTDOWN, 0);

system("pause");

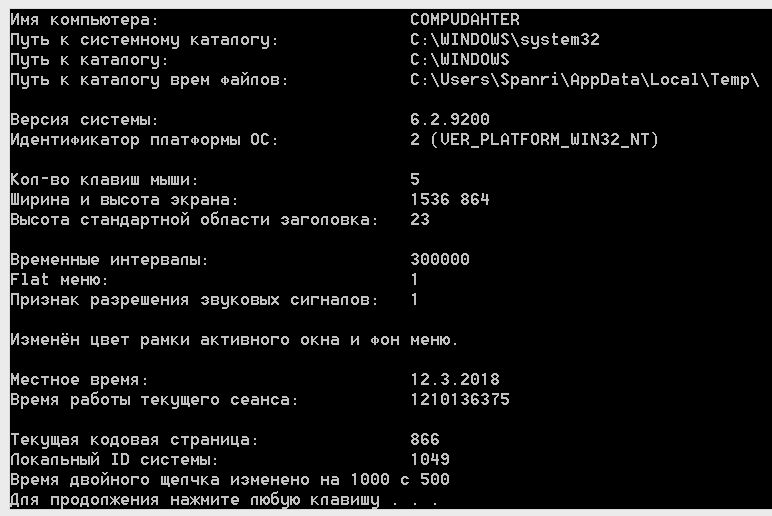
SetSysColors(2, lpaElements, color);

SetDoubleClickTime(500);

return 0;

}

**Пример работы:**



**Вывод:**

В результате проделанной работы научились получать системную информацию.