Одесский национальный политехнический университет

Кафедра интеллектуальных систем и сетей

Лабораторная работа №5

По предмету «Системное программирование»

На тему «Оконный интерфейс»

Выполнили

Студенты группы АЕ-145

Михайлов О.М.

Зайцева М.О.

Проверил

Олещук О.В.

Одесси 2017

**Цель лабораторной работы**

Реализовать программу, выполняющую следующие действия с реестром:

а) чтение всех названий разделов из "HKEY\_LOCAL\_MACHINE\Software";

б) чтение всех значений из раздела "HKEY\_LOCAL\_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run";

в) чтение произвольно выбранной информации об аппаратуре;

г) определение того, какое приложение ассоциировано с выбранным студентом типом файла (тип файла должен начинаться с той же буквы, что и фамилия студента).

А так же:

1. Программа запускается 10 раз, после чего сообщает о необходимости приобрести полнофункциональную версию.

**Ход выполнения работы:**

// Lab3.cpp: определяет точку входа для приложения.

//

#include "stdafx.h"

#include "Lab3.h"

#include <ctime>

#define MAX\_LOADSTRING 100

#define ENGLISH\_LANGUAGE 0

#define RUSSIAN\_LANGUAGE 1

#define MAX\_VALUE\_NAME 16383

// Глобальные переменные:

HINSTANCE hInst; // текущий экземпляр

WCHAR szTitle[MAX\_LOADSTRING]; // Текст строки заголовка

WCHAR szWindowClass[MAX\_LOADSTRING]; // имя класса главного окна

int iVscrollPos; // величина прокрутки

int StringCounter = 0; // кол-во строк

TCHAR \*szDescibeText1; // Тексты для локализации

TCHAR \*szDescibeText2;

TCHAR \*szDescibeText3;

TCHAR \*szDescibeText4;

DWORD iLanguage;

\_TCHAR szTestString[] = \_T("0");

static const TCHAR g\_szRegSubKey[] = TEXT("Software\\SPLab"); // путь к настройкам приложения

static const TCHAR g\_szRegSubKey2[] = TEXT("Software\\Microsoft\\Windows\\CurrentVersion\\Run"); // путь по заданию

static const TCHAR g\_szRegSubKey3[] = TEXT("Software"); // путь по заданию

static const TCHAR g\_szRegSubKey4[] = TEXT("HARDWARE\\DESCRIPTION\\System"); // путь с информацией о системе

static const TCHAR g\_szRegSubKey5[] = TEXT(".cpp");

// путь к ключу с ассоциациями

HKEY hKey; // ключ настроек приложения

ATOM MyRegisterClass(HINSTANCE hInstance);

BOOL InitInstance(HINSTANCE, int);

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

int APIENTRY wWinMain(\_In\_ HINSTANCE hInstance,

\_In\_opt\_ HINSTANCE hPrevInstance,

\_In\_ LPWSTR lpCmdLine,

\_In\_ int nCmdShow)

{

UNREFERENCED\_PARAMETER(hPrevInstance);

UNREFERENCED\_PARAMETER(lpCmdLine);

// TODO: разместите код здесь.

// Инициализация глобальных строк

LoadStringW(hInstance, IDS\_APP\_TITLE, szTitle, MAX\_LOADSTRING);

LoadStringW(hInstance, IDC\_LAB3, szWindowClass, MAX\_LOADSTRING);

MyRegisterClass(hInstance);

// Выполнить инициализацию приложения:

if (!InitInstance(hInstance, nCmdShow))

{

return FALSE;

}

HACCEL hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_LAB3));

MSG msg;

// Цикл основного сообщения:

while (GetMessage(&msg, nullptr, 0, 0))

{

if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

return (int)msg.wParam;

}

//

// ФУНКЦИЯ: MyRegisterClass()

//

// НАЗНАЧЕНИЕ: регистрирует класс окна.

//

ATOM MyRegisterClass(HINSTANCE hInstance)

{

WNDCLASSEXW wcex;

wcex.cbSize = sizeof(WNDCLASSEX);

wcex.style = CS\_HREDRAW | CS\_VREDRAW;

wcex.lpfnWndProc = WndProc;

wcex.cbClsExtra = 0;

wcex.cbWndExtra = 0;

wcex.hInstance = hInstance;

wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_LAB3));

wcex.hCursor = LoadCursor(nullptr, IDC\_ARROW);

wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW + 1);

wcex.lpszClassName = szWindowClass;

wcex.lpszMenuName = NULL;

wcex.hIconSm = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI\_SMALL));

return RegisterClassExW(&wcex);

}

//

// ФУНКЦИЯ: InitInstance(HINSTANCE, int)

//

// НАЗНАЧЕНИЕ: сохраняет обработку экземпляра и создает главное окно.

//

// КОММЕНТАРИИ:

//

// В данной функции дескриптор экземпляра сохраняется в глобальной переменной, а также

// создается и выводится на экран главное окно программы.

//

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)

{

hInst = hInstance; // Сохранить дескриптор экземпляра в глобальной переменной

HWND hWnd = CreateWindowW(szWindowClass, szTitle, WS\_OVERLAPPEDWINDOW | WS\_VSCROLL, CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0, nullptr, nullptr, hInstance, nullptr);

if (!hWnd)

{

return FALSE;

}

ShowWindow(hWnd, nCmdShow);

UpdateWindow(hWnd);

return TRUE;

}

// ФУНКЦИЯ: WndProc(HWND, UINT, WPARAM, LPARAM)

//

// НАЗНАЧЕНИЕ: обрабатывает сообщения в главном окне.

// WM\_COMMAND — обработать меню приложения

// WM\_PAINT — отрисовать главное окно

// WM\_DESTROY — отправить сообщение о выходе и вернуться

//

//

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)

{

switch (message)

{

case WM\_CREATE:

{

DWORD disposition;

//создаёт указанный ключ реестра

RegCreateKeyEx(HKEY\_CURRENT\_USER, g\_szRegSubKey, 0, NULL, REG\_OPTION\_NON\_VOLATILE,

KEY\_ALL\_ACCESS, NULL, &hKey, &disposition);

//записываем в реестр

RegSetValueEx(hKey, \_T("Count of run"), 0, REG\_DWORD, (BYTE\*)szTestString, sizeof(szTestString));

DWORD dwBufLen = MAX\_PATH;

RegGetValue(hKey, g\_szRegSubKey, \_T("Test String"), RRF\_RT\_REG\_SZ, NULL, (BYTE\*)szTestString - 1, &dwBufLen);

RegCloseKey(hKey);

switch (iLanguage) {

case ENGLISH\_LANGUAGE: {

szDescibeText1 = \_T("HKEY\_LOCAL\_MACHINE\\Software");

szDescibeText2 = \_T("HKEY\_LOCAL\_MACHINE\\Software\\Microsoft\\Windows\\CurrentVersion\\Run");

szDescibeText3 = \_T("Some random data about architecture");

szDescibeText4 = \_T("Apps, associated with filetype \".cpp\"");

break;

}

}

break;

}

case WM\_PAINT:

{

StringCounter = 0;

int StringHight = 20;

TCHAR text[100];

DWORD lenght = sizeof(text);

TCHAR achValue[MAX\_VALUE\_NAME];

achValue[0] = '\0';

DWORD cchValue = MAX\_VALUE\_NAME;

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hWnd, &ps);

LONG l = RegOpenKeyEx(HKEY\_LOCAL\_MACHINE, g\_szRegSubKey3, 0, KEY\_ENUMERATE\_SUB\_KEYS | KEY\_READ | KEY\_QUERY\_VALUE, &hKey);

TextOut(hdc, 0, (StringCounter++ - iVscrollPos)\*StringHight, szDescibeText1, \_tcslen(szDescibeText1));

StringCounter++;

for (int nIndex = 0; l != ERROR\_NO\_MORE\_ITEMS; nIndex++) {

//перечисляет подключи указанного открытого ключа реестра

l = RegEnumKey(hKey, nIndex, text, lenght);

TextOut(hdc, 0, (StringCounter++ - iVscrollPos)\*StringHight, text, \_tcslen(text));

}

RegCloseKey(hKey);

l = RegOpenKeyEx(HKEY\_LOCAL\_MACHINE, g\_szRegSubKey2, 0, KEY\_ENUMERATE\_SUB\_KEYS | KEY\_READ | KEY\_QUERY\_VALUE, &hKey);

StringCounter++;

TextOut(hdc, 0, (StringCounter++ - iVscrollPos)\*StringHight, szDescibeText2, \_tcslen(szDescibeText2));

StringCounter++;

for (int nIndex = 0; l != ERROR\_NO\_MORE\_ITEMS; nIndex++) {

//перечисляет значения в указанном открытом ключе реестра

l = RegEnumValue(hKey, nIndex, achValue, &cchValue, NULL, NULL, NULL, NULL);

TextOut(hdc, 0, ((StringCounter++) - iVscrollPos)\*StringHight, achValue, \_tcslen(achValue));

}

RegCloseKey(hKey);

l = RegOpenKeyEx(HKEY\_LOCAL\_MACHINE, g\_szRegSubKey4, 0, KEY\_ENUMERATE\_SUB\_KEYS | KEY\_READ | KEY\_QUERY\_VALUE, &hKey);

StringCounter++;

TextOut(hdc, 0, (StringCounter++ - iVscrollPos)\*StringHight, szDescibeText3, \_tcslen(szDescibeText3));

StringCounter++;

//чтение версии биоса

DWORD cData = 255;

TCHAR SystemBiosVersion[255] = { '\0' };

//возвращает тип и данние указанного значения по имени, ассоциирующимся с открытым ключом реестра

RegQueryValueEx(hKey, \_T("SystemBiosVersion"), NULL, NULL, (LPBYTE)SystemBiosVersion, &cData);

for (TCHAR\* p = SystemBiosVersion; \*p; p += \_tcslen(p) + 1)

{

TextOut(hdc, 0, ((StringCounter++) - iVscrollPos)\*StringHight, p, \_tcslen(p));

}

RegCloseKey(hKey);

l = RegOpenKeyEx(HKEY\_CLASSES\_ROOT, g\_szRegSubKey5, 0, KEY\_ENUMERATE\_SUB\_KEYS | KEY\_READ | KEY\_QUERY\_VALUE, &hKey);

//вывод имени файла формата .cpp

DWORD cData2 = 255;

TCHAR AssosiatedApp[255] = { '\0' };

RegQueryValueEx(hKey, NULL, NULL, NULL, (PBYTE)&AssosiatedApp, &cData2);

StringCounter++;

TextOut(hdc, 0, (StringCounter++ - iVscrollPos)\*StringHight, szDescibeText4, \_tcslen(szDescibeText4));

StringCounter++;

TextOut(hdc, 0, (StringCounter++ - iVscrollPos)\*StringHight, AssosiatedApp, \_tcslen(SystemBiosVersion));

RegCloseKey(hKey);

EndPaint(hWnd, &ps);

}

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

case WM\_MOUSEWHEEL:

{

iVscrollPos -= (short)HIWORD(wParam) / WHEEL\_DELTA;

iVscrollPos = max(0, min(iVscrollPos, StringCounter));

InvalidateRect(hWnd, NULL, TRUE);

break;

}

case WM\_VSCROLL:

switch (LOWORD(wParam)) {

case SB\_LINEUP: iVscrollPos--; break;

case SB\_LINEDOWN: iVscrollPos++; break;

case SB\_THUMBPOSITION: iVscrollPos = HIWORD(wParam); break;

} iVscrollPos = max(0, min(iVscrollPos, StringCounter));

if (iVscrollPos != GetScrollPos(hWnd, SB\_VERT)) {

SetScrollPos(hWnd, SB\_VERT, iVscrollPos, TRUE);

InvalidateRect(hWnd, NULL, TRUE);

}

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

return 0;

}

Вывод

В данной лабораторной работе мы разработали принципы работы с реестром путём использования функций: Создать ключ - RegCreateKeyEx(). Открыть существующий ключ: RegOpenKeyEx(). Закрыть ключ - RegCloseKey(). Чтение данных - RegGetValue(). Запись данных - RegSetValueEx(). Удаление данных - RegDeleteValue().