**Лабораторная работа. РЕЕСТР WINDOWS**

Цель. Изучить способ создания раздела реестра, записи и чтения в раздел реестра.

Краткая теория. Реестр – *это централизованное хранилище системных настроек*. Управляется подсистемой **Configuration Manager** ядра операционной системы.

Реестр играет ключевую роль в конфигурации систем Windows и в управлении ими. Он является хранилищем как для общесистемных настроек, так и для настроек для каждого пользователя. Хотя многие считают реестр некими статичными данными, хранящимися на жестком диске, реестр также является окном в различные находящиеся в памяти структуры, обслуживаемые исполняющей системой и ядром Windows.

**Шесть корневых разделов**

**HKEY\_CURRENT\_USER** Хранит данные, связанные с текущим пользователем, вошедшим в систему

**HKEY\_USERS** Хранит информацию обо всех учетных записях, имеющихся на машине

**HKEY\_CLASSES\_ROOT** Хранит файловые связи и информацию о регистрации объектов, относящихся к модели компонентных объектов — Component Object Model (COM)

**HKEY\_LOCAL\_MACHINE** Хранит информацию, связанную с системой

**HKEY\_PERFORMANCE\_DATA** Хранит информацию о производительности

**HKEY\_CURRENT\_CONFIG** Хранит определенную информацию о текущем профиле

Типы данных в реестре:

**REG\_BINARY** – двоичное число (по форме представления – 16ичное)

**REG\_DWORD** – 4-ех байтовое целое число

**REG\_EXPAND\_SZ** – строковое значение переменной длины

**REG\_MULTI\_SZ** – список строковых значений

**REG\_SZ** – строка фиксированной длины

**REG\_RESOURCE\_LIST** – список устройств в 16-ичной форме представления

**REG\_RESOURCE\_REQUIREMENTS\_LIST** – список устройств, обсл. драйвером

**REG\_NONE –** данные без определенного типа (часто записываются в реестр прикладными программами)

**REG\_LINK**  – строка в формате UNICODE

**REG\_QWORD** – 64-битовое целое число

**Чтение и запись информации из реестра**

#include "stdafx.h"

#include <windows.h>

#include <iostream>

using namespace std;

#define WIN\_32\_LEAN\_AND\_MEAN

void writeToRegistry(void)

{

DWORD lRv;

HKEY hKey;

//Check if the registry exists

lRv = RegOpenKeyEx(

HKEY\_CURRENT\_USER,

L"Software\\OV\_German",

0,

KEY\_WRITE,

&hKey

);

if (lRv != ERROR\_SUCCESS)

{

DWORD dwDisposition;

// Create a key if it did not exist

lRv = RegCreateKeyEx(

HKEY\_CURRENT\_USER,

L"Software\\OV\_German", //"Use Multi-Byte Character Set" by using L

0,

NULL,

REG\_OPTION\_NON\_VOLATILE,

KEY\_ALL\_ACCESS,

NULL,

&hKey,

&dwDisposition

);

DWORD dwValue = 14;

RegSetValueEx(

hKey,

L"Read more books",

0,

REG\_DWORD,

reinterpret\_cast<BYTE \*>(&dwValue),

sizeof(dwValue)

);

RegCloseKey(hKey);

}

}

//Чтение из реестра

void readValueFromRegistry(void)

{

HKEY hKey;

//Check if the registry exists

DWORD lRv = RegOpenKeyEx(

HKEY\_CURRENT\_USER,

L"Software\\OV\_GERMAN",

0,

KEY\_READ,

&hKey

);

if (lRv == ERROR\_SUCCESS)

{

DWORD BufferSize = sizeof(DWORD);

DWORD dwRet;

DWORD cbData;

DWORD cbVal = 0;

dwRet = RegQueryValueEx(

hKey,

L"Read more books",

NULL,

NULL,

(LPBYTE)&cbVal,

&cbData

);

if( dwRet == ERROR\_SUCCESS )

cout<<"\nValue of OV\_German is " << cbVal << endl;

else cout<<"\nRegQueryValueEx failed " << dwRet << endl;

}

else

{

cout<<"RegOpenKeyEx failed " << lRv << endl;

}

}

int main()

{

writeToRegistry();

readValueFromRegistry();

getchar();

return 0;

}

+++++++++++++++++++++++++++++++++++++++++++++++++++++++

Основные методы:

int CreateRegistryKey(CString ConstantKeyPath);

int ReadRegistryKeyAttributes(CString ConstantKeyPath);

int UpdateRegistryKeyAttribute(CString ConstantKeyPath, CString Attribute, DWORD value);

int DeleteRegistryKey(CString ConstantKeyPath);

Удаление узла из реестра

int DeleteRegistryKey(CString ConstantKeyPath)

{

int lReturn = RegDeleteKey(HKEY\_CURRENT\_USER, ConstantKeyPath);

return lReturn;

}

Создание узла реестра:

………….

int iResult;

CString KeyPath = \_T("Software\\RCD\_Technologies\\Rajib\_Test");

iResult =  CreateRegistryKey(KeyPath);

if (iResult<0)

{

cout << "CreateRegistryKey Operation Failed" << endl;

return -1;

}

cout << "<-- CreateRegistryKey Operation Successfull -->" << endl;

Открытие узла:

HKEY MyRegistryKey;

if (RegOpenKeyEx(HKEY\_CURRENT\_USER, ConstantKeyPath, 0, KEY\_ALL\_ACCESS, &MyRegistryKey) != ERROR\_SUCCESS)

{

cout << "KeyOpen Failed" << endl;

return -1;

}

Можно записать несколько значений:

lRv = RegCreateKeyEx(

HKEY\_CURRENT\_USER,

L"Software\\OV\_German2", //"Use Multi-Byte Character Set" by using L

0,

NULL,

REG\_OPTION\_NON\_VOLATILE,

KEY\_ALL\_ACCESS,

NULL,

&hKey,

&dwDisposition

);

DWORD dwValue = 14;

TCHAR sr[]=TEXT("be happy");

RegSetValueEx(

hKey,

L"Read more books",

0,

REG\_DWORD,

reinterpret\_cast<BYTE \*>(&dwValue),

sizeof(dwValue)

);

RegSetValueEx(

hKey,

L"LG",

0,

REG\_SZ,

reinterpret\_cast<BYTE \*>(&sr), // строка не должна содержать пробелов!

sizeof(dwValue)

);

RegCloseKey(hKey);

ЗАДАНИЕ.

1. Создать собственный раздел в реестре
2. Записать в него несколько значений.
3. Продемонстрировать чтение значения из созданного раздела.
4. Записать в раздел реестра адрес расположения программы NotePad.exe и написать код для ее запуска.
5. Удалить раздел из реестра.
6. Провести поиск раздела в реестре (самостоятельно установить, какая для этого используется команда).