Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Операционные среды и системное программирование

ОТЧЕТ

к лабораторной работе №5

на тему

**РЕЕСТР И ЖУРНАЛЫ (WINDOWS). ДОСТУП К РЕЕСТРУ WINDOWS. РАБОТА С ЖУРНАЛАМИ WINDOWS. ДРУГИЕ ВСПОМОГАТЕЛЬНЫЕ СРЕДСТВА УПРАВЛЕНИЯ**

Студент О. Л. Дайнович

Преподаватель Н. Ю. Гриценко

Минск 2023

**СОДЕРЖАНИЕ**

[1 Цель работы 3](#_Toc150483813)

[2 Теоретические сведения 4](#_Toc150483814)

[3 Описание функций программы 6](#_Toc150483815)

[Заключение 9](#_Toc150483816)

[Список использованных источников 10](#_Toc150483817)

[Приложение А (обязательное) Листинг кода 11](#_Toc150483818)

# **1 ЦЕЛЬ РАБОТЫ**

Целью выполнения лабораторной работы является создание оконного приложения на Win32 API для создания и управления реестровыми записями Windows, включая создание, изменение и удаление ключей и значений.

# **2 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Работа с реестром Windows на WinAPI позволяет программам получать доступ к системному реестру и выполнять операции чтения, записи и удаления данных из реестра. Реестр Windows является центральным хранилищем конфигурационной информации операционной системы, а также приложений, установленных на компьютере. [1]

Для работы с реестром на WinAPI используются функции из библиотеки advapi32.dll. Вот несколько основных функций, которые могут быть полезны при работе с реестром:

1 RegOpenKeyEx: Используется для открытия существующего ключа реестра или создания нового ключа. Функция возвращает дескриптор открытого ключа, который затем используется в других функциях для обращения к этому ключу.

2 RegCloseKey: Используется для закрытия открытого ключа реестра. Это важно, чтобы освободить ресурсы, связанные с открытым ключом, когда он больше не нужен.

3 RegGetValue: Используется для чтения значения из открытого ключа реестра. Вы можете указать имя ключа и имя значения, и функция вернет данные из реестра. Это может быть полезно, например, для получения настроек приложения из реестра.

4 RegSetValueEx: Используется для записи значения в открытый ключ реестра. Вы указываете имя ключа, имя значения и данные, которые нужно записать. Функция позволяет записать данные различных типов, таких как строки, числа и бинарные данные.

5 RegDeleteKey: Используется для удаления ключа реестра. Вы указываете имя ключа, который нужно удалить. Обратите внимание, что эта функция удаляет только пустые ключи; для удаления ключей с подключами или значениями необходимо использовать другие функции.

Это лишь небольшой набор функций, которые могут быть использованы при работе с реестром Windows на WinAPI. Они позволяют программам читать и изменять данные в реестре, что может быть полезно при настройке и управлении приложениями и операционной системой. Важно быть осторожным при работе с реестром, чтобы не удалить или изменить неправильные данные, что может привести к непредсказуемому поведению системы.

Реестр Windows организован иерархически, подобно файловой системе, с ключами и подключами, которые представляют собой папки, а значения, соответственно, файлы.

Основные корневые ключи в реестре следующие:

1 HKEY\_CLASSES\_ROOT (HKCR): Содержит информацию о связях файлов и регистрации COM-объектов.

2 HKEY\_CURRENT\_USER (HKCU): Содержит настройки и предпочтения текущего пользователя.

3 HKEY\_LOCAL\_MACHINE (HKLM): Содержит системные настройки и конфигурационную информацию.

4 HKEY\_USERS (HKU): Содержит профили пользователей и настройки для всех учетных записей на компьютере.

5 HKEY\_CURRENT\_CONFIG (HKCC): Содержит информацию о текущем профиле оборудования. [2]

# **3 ОПИСАНИЕ ФУНКЦИЙ ПРОГРАММЫ**

В ходе выполнения лабораторной работы было разработано оконное приложение для создания и управления реестровыми записями Windows, включая создание, изменение и удаление ключей и значений (рисунок 1).

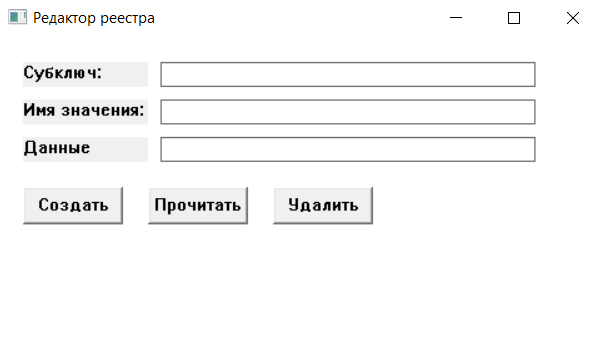


Рисунок 1 – Графический интерфейс приложения

В случае успешного а – добавления записи в реестр на экран выводится соответствующее сообщение (рисунок 2).

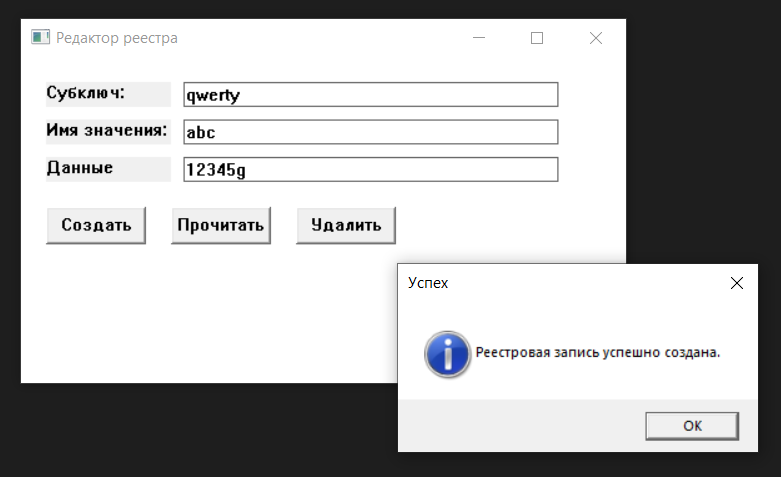


Рисунок 2 — Успешное добавление записи в реестр

После выполнения команды соответствующая запись появляется в реестре (рисунок 3).

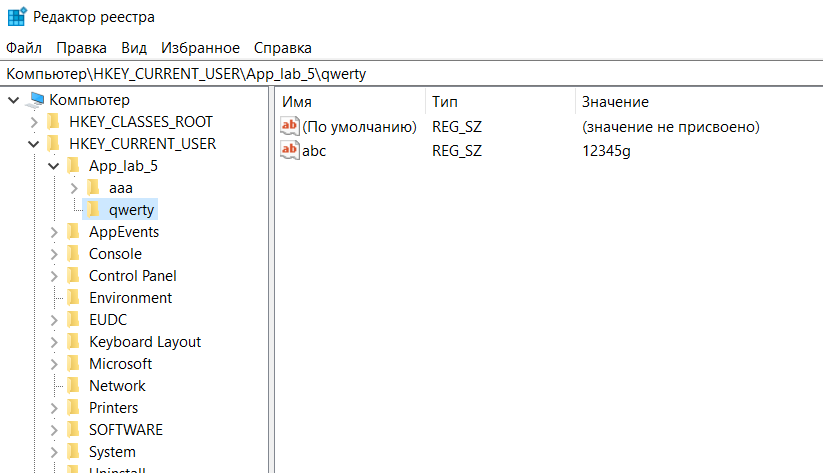


Рисунок 3 — Запись, созданная пользователем

Помимо возможности создания записей их также можно считывать. Для этого требуется ввести значение субключа и имя значения, которое пользователь хочет прочитать (рисунок 4).

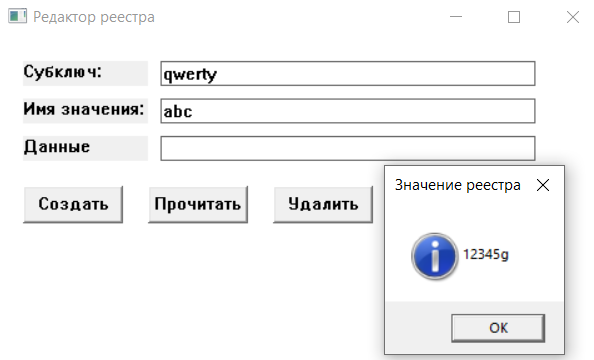


Рисунок 4 — Успешное чтение записи

Возможность удаления записей также присутствует. Для этого пользователю требуется ввести субключ положения записи (рисунок 5).

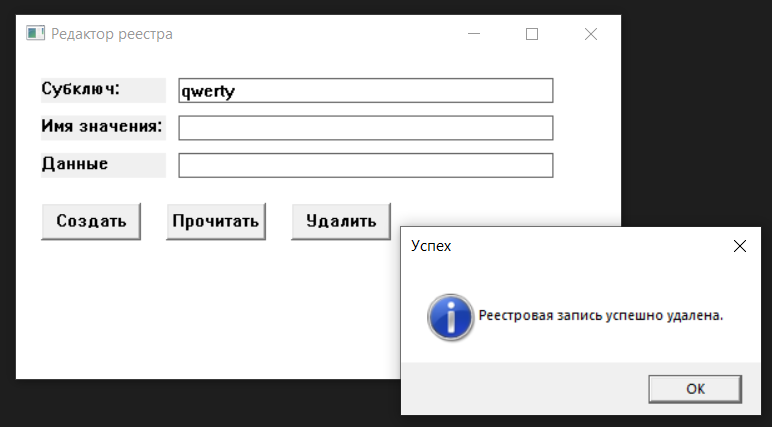


Рисунок 5 — Успешное удаление записи

Также присутствует обработка ошибок, к примеру таких как попытка удаления несуществующей записи (рисунок 6).

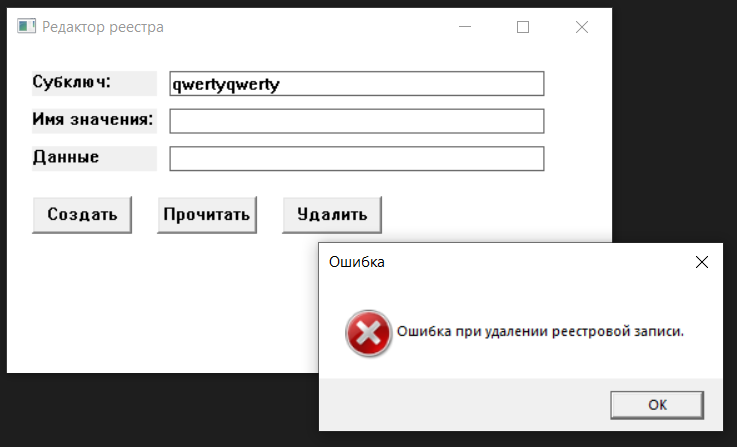


Рисунок 5 — Сообщение об ошибке

# **ЗАКЛЮЧЕНИЕ**

В результате лабораторной работы были изучены принципы работы с реестром Windows в Win32 API. Было создано оконное приложение, позволяющее создавать, считывать и удалять записи в реестре Windows.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] Синхронизация процессов Win32 API [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ ru-ru/windows/win32/sync/ interprocess-synchronization.

[2] Сведения о реестре Windows для опытных пользователей [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/ troubleshoot/windows-server/performance/windows-registry-advanced-users.

# **ПРИЛОЖЕНИЕ А**

# **(обязательное)**

# **Листинг кода**

Листинг 1 – Файл App.cpp

#include <Windows.h>

#include <iostream>

enum {

EDIT\_SUBKEY = 1001,

EDIT\_VALUENAME,

EDIT\_VALUEDATA,

BUTTON\_CREATE,

BUTTON\_READ,

BUTTON\_DELETE

};

LRESULT CALLBACK WindowProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam);

HWND CreateAppWindow(HINSTANCE hInstance);

void CreateRegistryKey(HWND hwnd);

void ReadRegistryKey(HWND hwnd);

void DeleteRegistryKey(HWND hwnd);

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)

{

HWND hwnd = CreateAppWindow(hInstance);

if (hwnd == NULL)

return -1;

ShowWindow(hwnd, nCmdShow);

MSG msg;

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

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return static\_cast<int>(msg.wParam);

}

LRESULT CALLBACK WindowProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam)

{

switch (uMsg)

{

case WM\_COMMAND:

{

switch (LOWORD(wParam))

{

case BUTTON\_CREATE:

CreateRegistryKey(hwnd);

break;

case BUTTON\_READ:

ReadRegistryKey(hwnd);

break;

case BUTTON\_DELETE:

DeleteRegistryKey(hwnd);

break;

}

}

break;

case WM\_DESTROY:

PostQuitMessage(0);

return 0;

}

return DefWindowProc(hwnd, uMsg, wParam, lParam);

}

HWND CreateAppWindow(HINSTANCE hInstance)

{

WNDCLASS wc = { 0 };

wc.lpfnWndProc = WindowProc;

wc.hInstance = hInstance;

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

wc.lpszClassName = L"RegistryAppClass";

if (!RegisterClass(&wc))

{

MessageBox(NULL, L"Ошибка при регистрации класса окна.", L"Ошибка", MB\_OK | MB\_ICONERROR);

return NULL;

}

HWND hwnd = CreateWindowEx(

0,

L"RegistryAppClass",

L"Редактор реестра",

WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, CW\_USEDEFAULT, 500, 300,

NULL, NULL, hInstance, NULL);

if (hwnd == NULL)

{

MessageBox(NULL, L"Ошибка при создании окна.", L"Ошибка", MB\_OK | MB\_ICONERROR);

return NULL;

}

CreateWindow(

L"STATIC", L"Субключ:",

WS\_VISIBLE | WS\_CHILD,

20, 20, 100, 20,

hwnd, NULL, hInstance, NULL);

CreateWindow(

L"EDIT", L"",

WS\_VISIBLE | WS\_CHILD | WS\_BORDER,

130, 20, 300, 20,

hwnd, reinterpret\_cast<HMENU>(EDIT\_SUBKEY), hInstance, NULL);

CreateWindow(

L"STATIC", L"Имя значения:",

WS\_VISIBLE | WS\_CHILD,

20, 50, 100, 20,

hwnd, NULL, hInstance, NULL);

CreateWindow(

L"EDIT", L"",

WS\_VISIBLE | WS\_CHILD | WS\_BORDER,

130, 50, 300, 20,

hwnd, reinterpret\_cast<HMENU>(EDIT\_VALUENAME), hInstance, NULL);

CreateWindow(

L"STATIC", L"Данные значения:",

WS\_VISIBLE | WS\_CHILD,

20, 80, 100, 20,

hwnd, NULL, hInstance, NULL);

CreateWindow(

L"EDIT", L"",

WS\_VISIBLE | WS\_CHILD | WS\_BORDER,

130, 80, 300, 20,

hwnd, reinterpret\_cast<HMENU>(EDIT\_VALUEDATA), hInstance, NULL);

CreateWindow(

L"BUTTON", L"Создать",

WS\_VISIBLE | WS\_CHILD,

20, 120, 80, 30,

hwnd, reinterpret\_cast<HMENU>(BUTTON\_CREATE), hInstance, NULL);

CreateWindow(

L"BUTTON", L"Прочитать",

WS\_VISIBLE | WS\_CHILD,

120, 120, 80, 30,

hwnd, reinterpret\_cast<HMENU>(BUTTON\_READ), hInstance, NULL);

CreateWindow(

L"BUTTON", L"Удалить",

WS\_VISIBLE | WS\_CHILD,

220, 120, 80, 30,

hwnd, reinterpret\_cast<HMENU>(BUTTON\_DELETE), hInstance, NULL);

return hwnd;

}

void CreateRegistryKey(HWND hwnd)

{

WCHAR subKey[256];

WCHAR valueName[256];

WCHAR valueData[256];

GetDlgItemText(hwnd, EDIT\_SUBKEY, subKey, 256);

GetDlgItemText(hwnd, EDIT\_VALUENAME, valueName, 256);

GetDlgItemText(hwnd, EDIT\_VALUEDATA, valueData, 256);

std::wstring fullKey = L"App\_lab\_5\\";

fullKey += subKey;

HKEY hKey = HKEY\_CURRENT\_USER;

HKEY hRegistryKey;

LONG result = RegCreateKeyEx(hKey, fullKey.c\_str(), 0, NULL, REG\_OPTION\_NON\_VOLATILE, KEY\_WRITE, NULL, &hRegistryKey, NULL);

if (result == ERROR\_SUCCESS)

{

result = RegSetValueEx(hRegistryKey, valueName, 0, REG\_SZ, reinterpret\_cast<const BYTE\*>(valueData), (wcslen(valueData) + 1) \* sizeof(wchar\_t));

if (result == ERROR\_SUCCESS)

MessageBox(hwnd, L"Реестровая запись успешно создана.", L"Успех", MB\_OK | MB\_ICONINFORMATION);

else

MessageBox(hwnd, L"Ошибка при установке значения реестра.", L"Ошибка", MB\_OK | MB\_ICONERROR);

RegCloseKey(hRegistryKey);

}

else

{

MessageBox(hwnd, L"Ошибка при создании реестровой записи.", L"Ошибка", MB\_OK | MB\_ICONERROR);

}

}

void ReadRegistryKey(HWND hwnd)

{

WCHAR subKey[256];

WCHAR valueName[256];

GetDlgItemText(hwnd, EDIT\_SUBKEY, subKey, 256);

GetDlgItemText(hwnd, EDIT\_VALUENAME, valueName, 256);

HKEY hKey = HKEY\_CURRENT\_USER;

HKEY hRegistryKey;

std::wstring fullKey = L"App\_lab\_5\\";

fullKey += subKey;

LONG result = RegOpenKeyEx(hKey, fullKey.c\_str(), 0, KEY\_READ, &hRegistryKey);

if (result == ERROR\_SUCCESS)

{

WCHAR valueData[256];

DWORD bufferSize = sizeof(valueData);

result = RegQueryValueEx(hRegistryKey, valueName, NULL, NULL, reinterpret\_cast<LPBYTE>(valueData), &bufferSize);

if (result == ERROR\_SUCCESS)

{

MessageBox(hwnd, valueData, L"Значение реестра", MB\_OK | MB\_ICONINFORMATION);

}

else

{

MessageBox(hwnd, L"Ошибка при чтении значения реестра.", L"Ошибка", MB\_OK | MB\_ICONERROR);

}

RegCloseKey(hRegistryKey);

}

else

{

MessageBox(hwnd, L"Ошибка при открытии реестровой записи.", L"Ошибка", MB\_OK | MB\_ICONERROR);

}

}

void DeleteRegistryKey(HWND hwnd)

{

WCHAR subKey[256];

GetDlgItemText(hwnd, EDIT\_SUBKEY, subKey, 256);

HKEY hKey = HKEY\_CURRENT\_USER;

std::wstring fullKey = L"App\_lab\_5\\";

fullKey += subKey;

LONG result = RegDeleteKey(hKey, fullKey.c\_str());

if (result == ERROR\_SUCCESS)

{

MessageBox(hwnd, L"Реестровая запись успешно удалена.", L"Успех", MB\_OK | MB\_ICONINFORMATION);

}

else

{

MessageBox(hwnd, L"Ошибка при удалении реестровой записи.", L"Ошибка", MB\_OK | MB\_ICONERROR);

}

}