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

РЕЕСТРЫ И ЖУРНАЛЫ (WINDOWS)

Студент Н. С. Шмидт
Преподаватель Н. Ю. Гриценко

Минск 2023

СОДЕРЖАНИЕ

1 ЦЕЛЬ РАБОТЫ

Целью данной лабораторной работы является изучение основных аспектов работы с реестром и журналами Windows, а также ознакомление с другими вспомогательными средствами управления, предоставляемыми операционной системой, с целью приобретения навыков администрирования. Данные методы будут продемонстрированы на оконном приложении, позволяющим создавать и управлять реестровыми записями Windows, включая создание, изменение и удаление ключей и значений.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

1 Реестр Windows: Реестр Windows (Windows Registry) - это централизованное хранилище, где хранятся настройки и конфигурационные данные операционной системы и установленных приложений. Реестр организован в виде древовидной структуры, где корневыми узлами являются пятеро главных ключей: HKEY_CLASSES_ROOT, HKEY_CURRENT_USER, HKEY_LOCAL_MACHINE, HKEY_USERS и HKEY_CURRENT_CONFIG.

2 HKEY: HKEY (Handle to the Key) - это основной тип данных для работы с ключами в реестре. Каждый ключ имеет уникальное имя внутри своего родительского ключа и может содержать значения, связанные с этим ключом.

3 Значения реестра: В реестре хранятся не только ключи, но и их ассоциированные значения. Значения могут быть различных типов, таких как строковые, числовые, бинарные и другие. Они используются для хранения настроек и параметров.

Взаимодействие с реестром через WinAPI

Для взаимодействия с реестром Windows используется WinAPI (Application Programming Interface).

Основные функции для работы с реестром включают:

1 RegOpenKeyEx: Для открытия существующего ключа в реестре.

2 RegCreateKeyEx: Для создания нового ключа в реестре.

3 RegQueryValueEx: Для получения значения, ассоциированного с ключом.

4 RegSetValueEx: Для установки значения для ключа.

5 RegDeleteKey: Для удаления ключа.

6 RegEnumKey/RegEnumValue: Для перечисления подключей и значений ключа соответственно.

2 РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ

Основой является приложение, в котором можно создавать ключи, изменять их значение, удалять эти ключи а также просматривать существует ли ключ и просматривать его значение (рисунок 1).

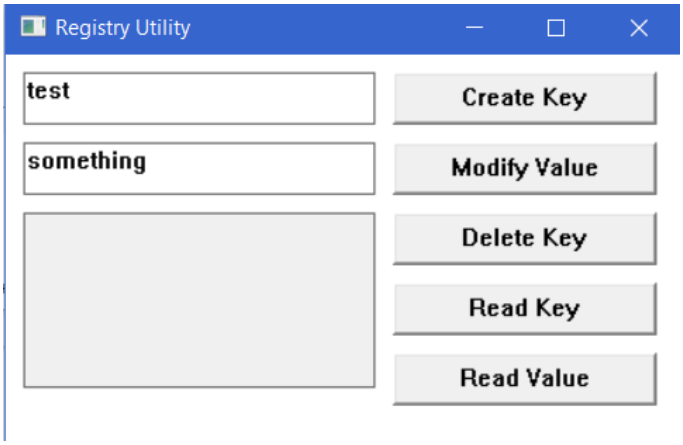

Рисунок 1 – Окно приложения

В ходе выполнения задания было создано приложение которое позволяет работать с реестром Windows(создавать, редактироват, удалять и просматривать). По нажатию на кнопку Create Key ключ введенный в первой строке будет создан, после этого следует задать ему значение по нажатию кнопки Modify Value(эта кнопка также может редактировать значение ключа), это можно увидеть в самом реестре Windows(рисунок 2).

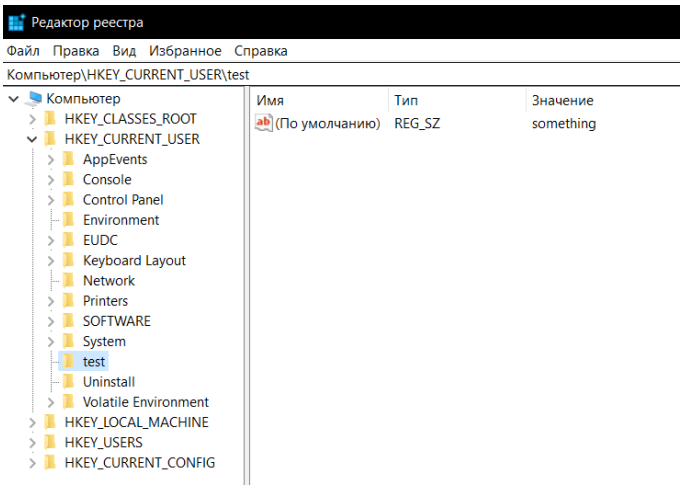

Рисунок 2 – Окно редактора реестра после создания ключа и добавления ему значения

Также это приложение позволяет просматривать существует ли ключ с введённым названием(рисунок 3)

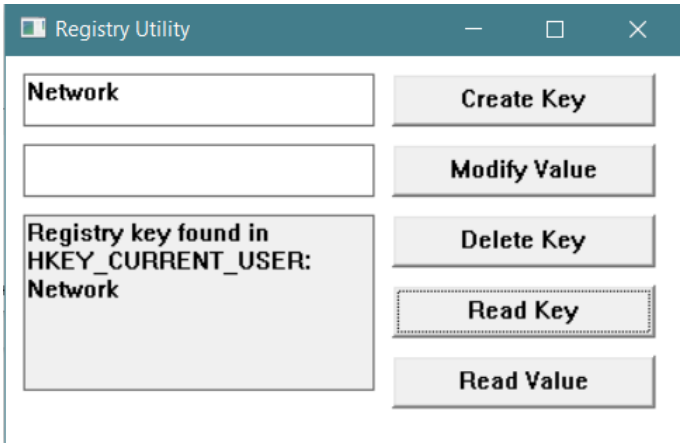

Рисунок 3 – Окно приложения после нажатия на кнопку Read Key

Если такой ключ существует, то по нажатию на кнопку Read Value будет выведено его значение(рисунок 4)

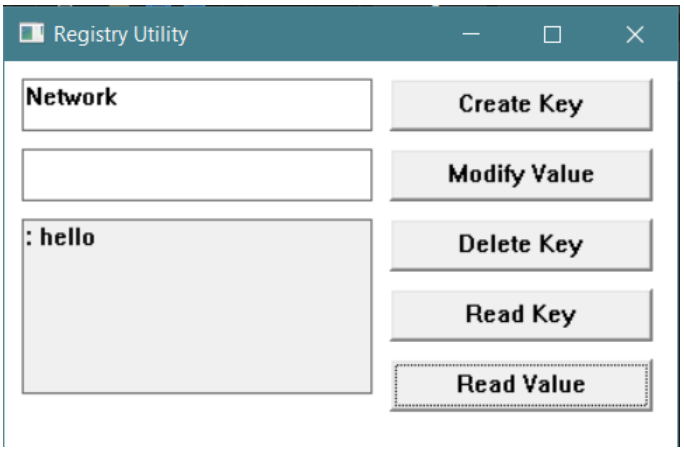

Рисунок 4 – Окно приложения после нажатия на кнопку Read Value

ЗАКЛЮЧЕНИЕ

В результате лабораторной работы были изучены основные аспекты работы с реестром и журналами Windows, а также другие вспомогательные средства управления, предоставляемыми операционной системой, с целью приобретения навыков администрирования. Было создано оконное приложение, позволяющее создавать и управлять реестровыми записями Windows, включая создание, изменение и удаление ключей и значений.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Основы программирования для Win32 API [Электронный ресурс]. – Режим доступа: https://dims.karelia.ru/win32/.

[2] Щупак Ю. Win32 API. Разработка приложений для Windows. – СПб: Питер, 2008. – 592 с.: ип.

[3] Microsoft Learn[Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/windows/win32/fileio/i-o-concepts – Дата доступа 08.10.2023

ПРИЛОЖЕНИЕ А
(обязательное)
Листинг кода

Листинг 1 – Файл *main.cpp*

```
#include <windows.h>
#include <tchar.h>

HINSTANCE hInst;
HWND hMainWindow;
HWND hKeyEdit;
HWND hValueEdit;
HWND hOutputEdit;

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);
void CreateRegistryKey();
void ModifyRegistryValue();
void DeleteRegistryKey();
void ReadRegistryKey();
void ReadRegistryValue();

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR
lpCmdLine, int nCmdShow)
{
    hInst = hInstance;
    WNDCLASSEX 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, IDI_APPLICATION);
    wcex.hCursor = LoadCursor(NULL, IDC_ARROW);
    wcex.hbrBackground = (HBRUSH)(COLOR_WINDOW + 1);
    wcex.lpszMenuName = NULL;
    wcex.lpszClassName = _T("RegistryUtility");
    wcex.hIconSm = LoadIcon(wcex.hInstance, IDI_APPLICATION);
    if (!RegisterClassEx(&wcex))
    {
        MessageBox(NULL, _T("Call to RegisterClassEx failed!"), _T("Error"),
NULL);
        return 1;
    }

    hMainWindow = CreateWindow(_T("RegistryUtility"), _T("Registry Utility"),
WS_OVERLAPPEDWINDOW, CW_USEDEFAULT, CW_USEDEFAULT, 400, 260, NULL, NULL,
hInstance, NULL);

    if (!hMainWindow)
    {
```

```
        MessageBox(NULL, _T("Call to CreateWindow failed!"), _T("Error"),
NULL);
        return 1;
    }

    ShowWindow(hMainWindow, nCmdShow);
    UpdateWindow(hMainWindow);
    MSG msg;

    while (GetMessage(&msg, NULL, 0, 0))
    {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }

    return (int)msg.wParam;
}

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    switch (message)
    {
    case WM_CREATE:
        hKeyEdit = CreateWindowEx(0, _T("EDIT"), _T(""), WS_CHILD | WS_VISIBLE
| WS_BORDER, 10, 10, 200, 30, hWnd, (HMENU)100, hInst, NULL);
        hValueEdit = CreateWindowEx(0, _T("EDIT"), _T(""), WS_CHILD |
WS_VISIBLE | WS_BORDER, 10, 50, 200, 30, hWnd, (HMENU)101, hInst, NULL);
        hOutputEdit = CreateWindowEx(0, _T("EDIT"), _T(""), WS_CHILD |
WS_VISIBLE | WS_BORDER | ES_MULTILINE | ES_READONLY, 10, 90, 200, 100, hWnd,
(HMENU)102, hInst, NULL);
        CreateWindow(_T("BUTTON"), _T("Create Key"), WS_CHILD | WS_VISIBLE |
BS_PUSHBUTTON, 220, 10, 150, 30, hWnd, (HMENU)200, hInst, NULL);
        CreateWindow(_T("BUTTON"), _T("Modify Value"), WS_CHILD | WS_VISIBLE |
BS_PUSHBUTTON, 220, 50, 150, 30, hWnd, (HMENU)201, hInst, NULL);
        CreateWindow(_T("BUTTON"), _T("Delete Key"), WS_CHILD | WS_VISIBLE |
BS_PUSHBUTTON, 220, 90, 150, 30, hWnd, (HMENU)202, hInst, NULL);
        CreateWindow(_T("BUTTON"), _T("Read Key"), WS_CHILD | WS_VISIBLE |
BS_PUSHBUTTON, 220, 130, 150, 30, hWnd, (HMENU)203, hInst, NULL);
        CreateWindow(_T("BUTTON"), _T("Read Value"), WS_CHILD | WS_VISIBLE |
BS_PUSHBUTTON, 220, 170, 150, 30, hWnd, (HMENU)204, hInst, NULL);
        break;

    case WM_COMMAND:
        if (HIWORD(wParam) == BN_CLICKED)
        {
            switch (LOWORD(wParam))
            {
            case 200:
                CreateRegistryKey();
                break;
            case 201:
                ModifyRegistryValue();
                break;
```

```
            case 202:
                DeleteRegistryKey();
                break;
            case 203:
                ReadRegistryKey();
                break;
            case 204:
                ReadRegistryValue();
                break;
            }
        }
        break;

    case WM_DESTROY:
        PostQuitMessage(0);
        break;

    default:
        return DefWindowProc(hWnd, message, wParam, lParam);
        break;
    }
    return 0;
}

void CreateRegistryKey()
{
    TCHAR keyName[256];
    GetWindowText(hKeyEdit, keyName, 256);
    HKEY hKey;

    if (RegCreateKey(HKEY_CURRENT_USER, keyName, &hKey) == ERROR_SUCCESS)
    {
        RegCloseKey(hKey);
        MessageBox(hMainWindow, _T("Registry key created successfully."),
_T("Success"), MB_OK | MB_ICONINFORMATION);
    }
    else
    {
        MessageBox(hMainWindow, _T("Failed to create registry key."),
_T("Error"), MB_OK | MB_ICONERROR);
    }
}

void ModifyRegistryValue()
{
    TCHAR keyName[256];
    TCHAR value[256];
    GetWindowText(hKeyEdit, keyName, 256);
    GetWindowText(hValueEdit, value, 256);

    HKEY hKey;
    if (RegOpenKey(HKEY_CURRENT_USER, keyName, &hKey) == ERROR_SUCCESS)
    {
```

```
        if (RegSetValueEx(hKey, NULL, 0, REG_SZ, (BYTE*)value, (_tcslen(value)
+ 1) * sizeof(TCHAR)) == ERROR_SUCCESS)
        {
            RegCloseKey(hKey);
            MessageBox(hMainWindow,      _T("Registry      value      modified
successfully."), _T("Success"), MB_OK | MB_ICONINFORMATION);
        }
        else
        {
            MessageBox(hMainWindow,  _T("Failed  to  modify  registry  value."),
_T("Error"), MB_OK | MB_ICONERROR);
        }
    }
    else
    {
        MessageBox(hMainWindow,  _T("Registry  key  not  found."),  _T("Error"),
MB_OK | MB_ICONERROR);
    }
}

void DeleteRegistryKey()
{
    TCHAR keyName[256];
    GetWindowText(hKeyEdit, keyName, 256);

    if (RegDeleteKey(HKEY_CURRENT_USER, keyName) == ERROR_SUCCESS)
    {
        MessageBox(hMainWindow,  _T("Registry  key  deleted  successfully."),
_T("Success"), MB_OK | MB_ICONINFORMATION);
    }
    else
    {
        MessageBox(hMainWindow,  _T("Failed  to  delete  registry  key."),
_T("Error"), MB_OK | MB_ICONERROR);
    }
}

void ReadRegistryKey()
{
    TCHAR keyName[256];
    GetWindowText(hKeyEdit, keyName, 256);

    HKEY hKey;
    if (RegOpenKey(HKEY_CURRENT_USER, keyName, &hKey) == ERROR_SUCCESS)
    {
        TCHAR outputText[1024];
        wsprintf(outputText, _T("Registry key found in HKEY_CURRENT_USER: %s"),
keyName);
        SetWindowText(hOutputEdit, outputText);
        RegCloseKey(hKey);
    }
    else
    {
```

```
        TCHAR outputText[1024];
        wsprintf(outputText, _T("Registry key not found: %s"), keyName);
        SetWindowText(hOutputEdit, outputText);
    }
}

void ReadRegistryValue()
{
    TCHAR keyName[256];
    GetWindowText(hKeyEdit, keyName, 256);
    HKEY hKey;

    if (RegOpenKeyEx(HKEY_CURRENT_USER, keyName, 0, KEY_READ, &hKey) ==
ERROR_SUCCESS)
    {
        TCHAR outputText[1024];
        outputText[0] = _T('\0');
        DWORD index = 0;
        TCHAR valueName[256];
        DWORD valueNameSize = sizeof(valueName);

        while (RegEnumValue(hKey, index, valueName, &valueNameSize, NULL, NULL,
NULL, NULL) == ERROR_SUCCESS)
        {
            DWORD valueType;
            DWORD valueSize;
            if (RegQueryValueEx(hKey, valueName, NULL, &valueType, NULL,
&valueSize) == ERROR_SUCCESS)
            {
                LPBYTE valueData = (LPBYTE)malloc(valueSize);

                if (RegQueryValueEx(hKey, valueName, NULL, &valueType,
valueData, &valueSize) == ERROR_SUCCESS)
                {
                    _tcscat_s(outputText, 1024, valueName);
                    _tcscat_s(outputText, 1024, _T(": "));

                    if (valueType == REG_SZ)
                    {
                        _tcscat_s(outputText, 1024, (LPTSTR)valueData);
                    }
                    else if (valueType == REG_DWORD)
                    {
                        DWORD value = *(DWORD*)valueData;
                        _stprintf_s(outputText, 1024, _T("%u"), value);
                    }
                    else
                    {
                        _stprintf_s(outputText, 1024, _T("Type 0x%X (not
supported)"), valueType);
                    }

                    _tcscat_s(outputText, 1024, _T("\r\n"));
```

```
                }
                free(valueData);
            }
            valueNameSize = sizeof(valueName);
            index++;
        }

        if (_tcslen(outputText) > 0)
        {
            SetWindowText(hOutputEdit, outputText);
        }
        else
        {
            SetWindowText(hOutputEdit, _T("No values found in the registry
key."));
        }
        RegCloseKey(hKey);
    }
    else
    {
        TCHAR outputText[1024];
        wsprintf(outputText, _T("Registry key not found: %s"), keyName);
        SetWindowText(hOutputEdit, outputText);
    }
}
```