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 И. В. Бобко

Преподаватель Н. Ю. Гриценко

Минск 2023

СОДЕРЖАНИЕ

[1 Цель работы 3](#_Toc146752068)

[2 Теоретические сведения 4](#_Toc146752069)

[3 Результат выполнения 5](#_Toc146752070)

[Заключение 7](#_Toc146752071)

[Список использованных источников 8](#_Toc146752072)

[Приложение А (обязательное) Листинг кода 9](#_Toc146752073)

1. **ЦЕЛЬ РАБОТЫ**

Исследовать и овладеть пониманием структуры, принципов работы с реестром и журналами операционной системы Windows. Получить практические навыки доступа к реестру, управления журналами с использованием встроенных инструментов, а также освоить другие вспомогательные средства управления операционной системой *Windows*.

1. **ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Реестр Windows это централизованное хранилище системной информации и настроек операционной системы *Microsoft Windows*. Он содержит информацию о конфигурации, параметрах, настройках и установках, необходимых для работы операционной системы и приложений. Реестр можно рассматривать как базу данных, где хранятся ключи и значения, организованные в иерархической структуре. [1]

Для доступа к реестру *Windows* используется Редактор реестра (*Registry Editor),* который позволяет просматривать, редактировать и создавать записи в реестре. Важно помнить, что неправильное редактирование реестра может привести к серьезным проблемам с операционной системой, поэтому необходимо быть осторожным и резервировать реестр перед внесением изменений.

У каждого ключа есть имя, состоящее из одного или нескольких печатных символов. В именах ключей регистр не учитывается. Имена ключей не могут содержать символ обратной косой черты, но можно использовать любой другой символ. Имена значений и данные могут содержать символ обратной косой черты. Имя каждого подраздела уникально по отношению к ключу, который находится непосредственно над ним в иерархии. [2]

Журналы событий *Windows* (*Event Logs*) представляют собой инструмент для записи и отслеживания событий, происходящих в операционной системе. Они используются для диагностики и мониторинга работы системы, выявления ошибок, а также для обеспечения безопасности.

В *Windows* существуют три основных журнала событий: Журнал системы (*System*), Журнал безопасности (*Security*) и Журнал приложений (*Application*). Каждый из них содержит записи о событиях, произошедших в соответствующей области операционной системы. [3]

Для работы с журналами Windows используются инструменты администрирования, такие как "Просмотр событий" (*Event Viewer*), которые позволяют фильтровать, анализировать и экспортировать записи из журналов для последующего анализа и решения проблем.

Обращение к реестру и работа с журналами являются важной частью администрирования и обслуживания операционной системы *Windows*, их корректное использование помогает обеспечить стабильность и безопасность системы.

1. **РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ**

В результате выполнения лабораторной работы было создана программа для создания и управления реестровыми записями *Windows*, включая создание, изменение и удаление ключей и значений (рисунок 1).

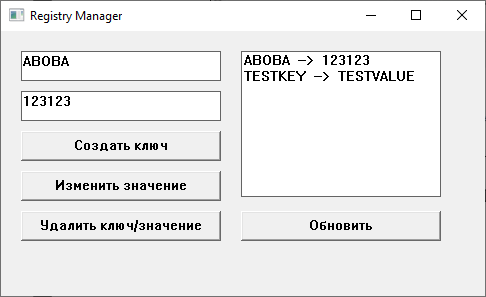


Рисунок 1– Главное окно

В реестр сохраняются созданные ключи и значения, а также имеется возможность их изменить или удалить. (рисунок 2).

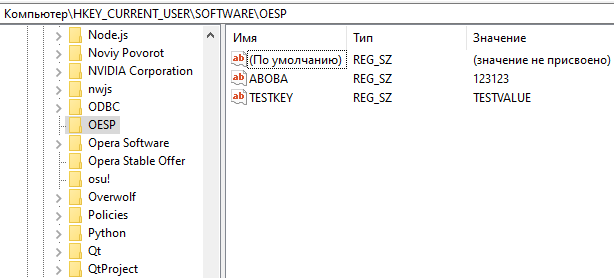


Рисунок 2 – Реестр

Все действия с реестром записываются в журнал событий (рисунок 3).

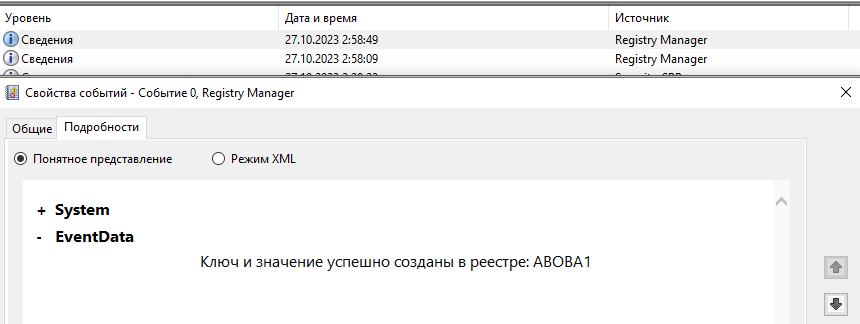


Рисунок 3 – Журнал событий

# ЗАКЛЮЧЕНИЕ

В ходе выполнения данной лабораторной работы были приобретены необходимые теоретические знания и практические навыки по работе с реестром и журналами в операционной системе *Windows*. Была реализована возможность создания, удаления и редактирования ключей реестра, а также записи в системный журнал.

С помощью разработанного приложения была продемонстрирована возможность сохранения данных в реестре и ведение журнала с информацией о каждом действии с реестром.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Структура реестра [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/windows/win32/sysinfo/structure-of-the-registry/.
2. Пространство реестра [Электронный ресурс]. – Режим доступа: <https://learn.microsoft.com/ru-ru/windows/win32/sysinfo/registry-storage-space>.
3. Ведение журнала событий [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/windows/win32/eventlog/event-logging/.

# ПРИЛОЖЕНИЕ А (обязательное) Листинг кода

Листинг 1 – Файл *Source.cpp*

#include <windows.h>

#include <tchar.h>

#include <string>

HWND hKeyEdit;

HWND hValueEdit;

HWND hListBox;

HWND hCreateButton;

HWND hChangeButton;

HWND hDeleteButton;

HWND hRefreshButton;

void SaveDataToRegistry(HWND hWnd);

void DeleteDataFromRegistry(HWND hWnd);

void RefreshList(HWND hWnd);

void ChangeDataInRegistry(HWND hWnd);

void WriteToEventLog(const std::wstring& message);

LRESULT CALLBACK WndProc(HWND hwnd, UINT msg, WPARAM wParam, LPARAM lParam);

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

MSG msg;

HWND hWnd;

WNDCLASSEX wc;

wc.cbSize = sizeof(WNDCLASSEX);

wc.cbClsExtra = 0;

wc.cbWndExtra = 0;

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

wc.hCursor = LoadCursor(NULL, IDC\_ARROW);

wc.hIcon = LoadIcon(NULL, IDI\_APPLICATION);

wc.hIconSm = LoadIcon(NULL, IDI\_APPLICATION);

wc.hInstance = hInstance;

wc.lpfnWndProc = WndProc;

wc.lpszClassName = L"RegistryManager";

wc.lpszMenuName = NULL;

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

if (!RegisterClassEx(&wc))

{

MessageBox(NULL, L"Failed to register window class.", L"Error", MB\_ICONERROR | MB\_OK);

return EXIT\_FAILURE;

}

hWnd = CreateWindow(wc.lpszClassName, L"Registry Manager", WS\_OVERLAPPEDWINDOW, CW\_USEDEFAULT, CW\_USEDEFAULT, 500, 300, NULL, NULL, wc.hInstance, NULL);

if (hWnd == INVALID\_HANDLE\_VALUE)

{

MessageBox(NULL, L"Failed to create a window.", L"Error", MB\_ICONERROR);

return EXIT\_FAILURE;

}

ShowWindow(hWnd, nCmdShow);

UpdateWindow(hWnd);

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

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return (int)msg.wParam;

}

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

switch (msg)

{

case WM\_CREATE:

{

hKeyEdit = CreateWindow(\_T("EDIT"), \_T(""), WS\_CHILD | WS\_VISIBLE | WS\_BORDER, 20, 20, 200, 30, hWnd, (HMENU)1, NULL, NULL);

hValueEdit = CreateWindow(\_T("EDIT"), \_T(""), WS\_CHILD | WS\_VISIBLE | WS\_BORDER, 20, 60, 200, 30, hWnd, (HMENU)2, NULL, NULL);

hListBox = CreateWindow(\_T("LISTBOX"), \_T(""), WS\_CHILD | WS\_VISIBLE | WS\_BORDER | LBS\_STANDARD, 240, 20, 200, 160, hWnd, (HMENU)3, NULL, NULL);

hCreateButton = CreateWindow(\_T("BUTTON"), \_T("Создать ключ"), WS\_CHILD | WS\_VISIBLE, 20, 100, 200, 30, hWnd, (HMENU)101, NULL, NULL);

hChangeButton = CreateWindow(\_T("BUTTON"), \_T("Изменить значение"), WS\_CHILD | WS\_VISIBLE, 20, 140, 200, 30, hWnd, (HMENU)102, NULL, NULL);

hDeleteButton = CreateWindow(\_T("BUTTON"), \_T("Удалить ключ/значение"), WS\_CHILD | WS\_VISIBLE, 20, 180, 200, 30, hWnd, (HMENU)103, NULL, NULL);

hRefreshButton = CreateWindow(\_T("BUTTON"), \_T("Обновить"), WS\_CHILD | WS\_VISIBLE, 240, 180, 200, 30, hWnd, (HMENU)104, NULL, NULL);

RefreshList(hWnd);

break;

}

case WM\_COMMAND:

{

switch (LOWORD(wParam))

{

case 101:

{

SaveDataToRegistry(hWnd);

break;

}

case 102:

{

ChangeDataInRegistry(hWnd);

break;

}

case 103:

{

DeleteDataFromRegistry(hWnd);

break;

}

break;

case 104:

{

RefreshList(hWnd);

break;

}

}

break;

}

case WM\_DESTROY:

{

PostQuitMessage(0);

break;

}

default:

{

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

}

}

return 0;

}

void SaveDataToRegistry(HWND hWnd) {

TCHAR keyBuffer[256];

TCHAR valueBuffer[256];

GetWindowText(hKeyEdit, keyBuffer, 256);

GetWindowText(hValueEdit, valueBuffer, 256);

HKEY hKey;

if (RegCreateKeyEx(HKEY\_CURRENT\_USER, L"Software\\OESP", 0, NULL, REG\_OPTION\_NON\_VOLATILE, KEY\_WRITE, NULL, &hKey, NULL) == ERROR\_SUCCESS) {

if (RegSetValueEx(hKey, keyBuffer, 0, REG\_SZ, (LPBYTE)valueBuffer, (lstrlen(valueBuffer) + 1) \* sizeof(TCHAR)) == ERROR\_SUCCESS) {

MessageBox(hWnd, L"Ключ и значение успешно созданы в реестре.", L"Успех", MB\_ICONINFORMATION);

TCHAR msg[512];

\_stprintf\_s(msg, L"Ключ и значение успешно созданы в реестре: %s", keyBuffer);

WriteToEventLog(msg);

}

else {

MessageBox(hWnd, L"Не удалось создать значение в реестре.", L"Ошибка", MB\_ICONERROR);

}

RegCloseKey(hKey);

}

else {

MessageBox(hWnd, L"Не удалось открыть или создать ключ в реестре.", L"Ошибка", MB\_ICONERROR);

}

RefreshList(hWnd);

}

void RefreshList(HWND hWnd) {

SendMessage(hListBox, LB\_RESETCONTENT, 0, 0);

HKEY hKey;

if (RegOpenKeyEx(HKEY\_CURRENT\_USER, L"Software\\OESP", 0, KEY\_READ, &hKey) == ERROR\_SUCCESS) {

TCHAR valueName[256];

DWORD valueNameSize = 256;

DWORD index = 0;

while (RegEnumValue(hKey, index, valueName, &valueNameSize, NULL, NULL, NULL, NULL) == ERROR\_SUCCESS) {

TCHAR value[256];

DWORD valueSize = 256;

if (RegQueryValueEx(hKey, valueName, NULL, NULL, (LPBYTE)value, &valueSize) == ERROR\_SUCCESS) {

TCHAR listItem[512];

\_stprintf\_s(listItem, L"%s --> %s", valueName, value);

SendMessage(hListBox, LB\_ADDSTRING, 0, (LPARAM)listItem);

}

valueNameSize = 256;

index++;

}

RegCloseKey(hKey);

}

}

void DeleteDataFromRegistry(HWND hWnd) {

TCHAR keyBuffer[256];

GetWindowText(hKeyEdit, keyBuffer, 256);

HKEY hKey;

if (RegOpenKeyEx(HKEY\_CURRENT\_USER, L"Software\\OESP", 0, KEY\_WRITE, &hKey) == ERROR\_SUCCESS) {

if (RegDeleteValue(hKey, keyBuffer) == ERROR\_SUCCESS) {

MessageBox(hWnd, L"Ключ и значение успешно удалены из реестра.", L"Успех", MB\_ICONINFORMATION);

TCHAR msg[512];

\_stprintf\_s(msg, L"Ключ и значение успешно удалены из реестра: %s", keyBuffer);

WriteToEventLog(msg);

}

else {

MessageBox(hWnd, L"Не удалось удалить значение из реестра.", L"Ошибка", MB\_ICONERROR);

}

RegCloseKey(hKey);

}

else {

MessageBox(hWnd, L"Не удалось открыть ключ в реестре.", L"Ошибка", MB\_ICONERROR);

}

RefreshList(hWnd);

}

void ChangeDataInRegistry(HWND hWnd) {

TCHAR keyBuffer[256];

TCHAR valueBuffer[256];

GetWindowText(hKeyEdit, keyBuffer, 256);

GetWindowText(hValueEdit, valueBuffer, 256);

HKEY hKey;

if (RegOpenKeyEx(HKEY\_CURRENT\_USER, L"Software\\OESP", 0, KEY\_SET\_VALUE, &hKey) == ERROR\_SUCCESS) {

if (RegSetValueEx(hKey, keyBuffer, 0, REG\_SZ, (LPBYTE)valueBuffer, (lstrlen(valueBuffer) + 1) \* sizeof(TCHAR)) == ERROR\_SUCCESS) {

MessageBox(hWnd, L"Значение успешно изменено в реестре.", L"Успех", MB\_ICONINFORMATION);

TCHAR msg[512];

\_stprintf\_s(msg, L"Значение успешно изменено в реестре: %s", keyBuffer);

WriteToEventLog(msg);

}

else {

MessageBox(hWnd, L"Не удалось изменить значение в реестре.", L"Ошибка", MB\_ICONERROR);

}

RegCloseKey(hKey);

}

else {

MessageBox(hWnd, L"Не удалось открыть ключ в реестре.", L"Ошибка", MB\_ICONERROR);

}

RefreshList(hWnd);

}

void WriteToEventLog(const std::wstring& message) {

HANDLE hEventLog = RegisterEventSource(NULL, L"Registry Manager");

if (hEventLog) {

LPCWSTR messageStrings[1];

messageStrings[0] = message.c\_str();

ReportEvent(hEventLog, EVENTLOG\_INFORMATION\_TYPE, 0, 0, NULL, 1, 0, messageStrings, NULL);

DeregisterEventSource(hEventLog);

}

}