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»

Выполнил:

студент гр. 153504

Михалевич M.П.

Проверил:

Гриценко Н.Ю.

Минск 2023

СОДЕРЖАНИЕ

[1 Цели работы 3](#_Toc147863804)

[2 Kраткие теоретические сведения 4](#_Toc147863805)

[3 Результаты выполнения лабораторной работы 5](#_Toc147863806)

[Вывод 6](#_Toc147863807)

[Список использованных источников 7](#_Toc147863808)

[Приложение А 8](#_Toc147863809)

## 1 ЦЕЛИ РАБОТЫ

Целью данной лабораторной работы является разработка утилиты для создания и управления реестровыми записями Windows, включая создание, изменение и удаление ключей и значений.

## 2 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Центральная база данных настроек и опций в ОС Microsoft Windows, известная как Реестр, играет ключевую роль в настройке и контроле системы. Она собирает в себе всю информацию о настройках системы, установленном программном обеспечении, драйверах и других важных аспектах системы, облегчая управление и настройку операционной системы для администраторов и программ.

Журналы событий Windows являются инструментом записи и отслеживания различных системных событий, ошибок и действий, связанных с безопасностью. Эти логи предоставляют ценную информацию для диагностики и отслеживания работы ОС, приложений и системных служб, помогая в обнаружении и решении проблем.

Существует множество специализированных инструментов и программных интерфейсов (API), которые предлагают возможности по управлению Реестром и анализу журналов событий. Они предоставляют функции для просмотра, редактирования и изучения данных в Реестре, а также для отслеживания и сортировки событий в журналах. Правильное и безопасное использование этих инструментов критично для защиты системы от неавторизованного доступа и для своевременного выявления технических неполадок в работе Windows.

В процессе выполнения данной лабораторной работы, были применены следующие теоретические сведения:

1. Программирование под Windows API: создание оконных приложений, обработка сообщений окна и создание графического интерфейса пользователя.
2. Работа с реестром Windows: использование функций API для открытия, создания, чтения, модификации и удаления ключей реестра.
3. Управление доступом к реестру: использование различных уровней доступа, таких как KEY\_READ и KEY\_ALL\_ACCESS.
4. Обработка ошибок: применение проверок на успешность выполнения функций и информирование пользователя с помощью диалоговых окон.
5. Работа со строками в C++: использование широких символов (wchar\_t), строковых литералов и функций для работы со строками, таких как wsprintf и wcslen.
6. Интерфейс программирования приложений (API) для работы с журналами событий Windows: чтение и перечисление значений ключей реестра.
7. Взаимодействие с элементами управления Windows: создание и использование текстовых полей (EDIT), кнопок (BUTTON) и списков (LISTBOX).

## 3 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

В ходе лабораторной работы было разработано приложение для управления реестром Windows, позволяющий пользователю через графический интерфейс добавлять, модифицировать и удалять ключи реестра. Пользовательский интерфейс состоит из текстовых полей для ввода названия ключа, значения и данных, и включает в себя три кнопки: "Create Key", "Modify Key" и "Delete Key". При нажатии на "Create Key" можно создать ключ, а при нажатии на "Delete Key" можно его удалить. Также есть "Modify Key", которая добавляет value и data для ключа, а еще можно изменить data на другое значение. Результат работы показан на рис. 3.1.

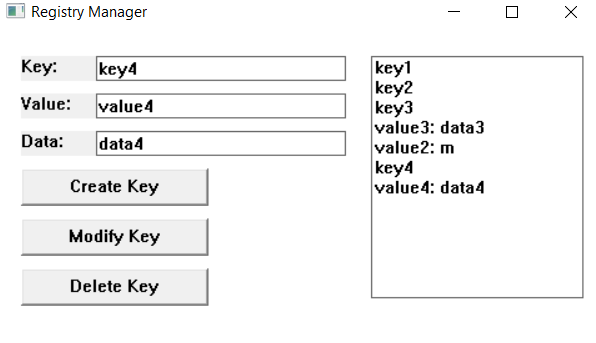


Рисунок 3.1 – Результат работы программы

## ВЫВОД

В ходе выполнения лабораторной работы была разработана утилита для работы с реестром Windows, позволяющая пользователю создавать, изменять и удалять ключи и значения реестра.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Щупак Ю. Win32 API. Разработка приложений для Windows. – СПб: Питер, 2008. – 592 с.: ип.
2. [Электронный ресурс]. – Режим доступа: [https://learn.microsoft.com/ru-ru/windows/win32/api – Дата доступа 23.10.2023](https://learn.microsoft.com/ru-ru/windows/win32/api%20–%20Дата%20доступа%2023.10.2023)
3. [Электронный ресурс]. – Режим доступа:  [https://learn.microsoft.com/ru-ru/cpp/parallel/multithreading-with-c-and-win32?view=msvc-170 – Дата доступа 24.10.2023](%20https://learn.microsoft.com/ru-ru/cpp/parallel/multithreading-with-c-and-win32?view=msvc-170%20–%20Дата%20доступа%2024.10.2023)

## ПРИЛОЖЕНИЕ А

**(обязательное)**

**Листинг кода**

#include <windows.h>

#include <string>

const wchar\_t CLASS\_NAME[] = L"Sample Window Class";

const wchar\_t WINDOW\_NAME[] = L"Registry Manager";

const wchar\_t\* subkey = L"MyProject";

HWND hKeyEdit;

HWND hValueEdit;

HWND hDataEdit;

HWND hListBox;

HKEY OpenMainKey() {

HKEY hKey;

if (RegOpenKeyEx(HKEY\_CURRENT\_USER, subkey, 0, KEY\_ALL\_ACCESS, &hKey) != ERROR\_SUCCESS) {

MessageBox(NULL, L"Ошибка открытия ключа.", L"Ошибка", MB\_OK | MB\_ICONERROR);

}

return hKey;

}

void ListSubKeys() {

HKEY hMainKey = NULL;

if (RegOpenKeyEx(HKEY\_CURRENT\_USER, subkey, 0, KEY\_READ, &hMainKey) == ERROR\_SUCCESS) {

TCHAR achKey[MAX\_PATH];

TCHAR achValue[MAX\_PATH];

DWORD cbName;

DWORD cSubKeys = 0, cValues = 0;

DWORD cbValueName, cbValueData, valueType;

BYTE dataBuffer[512];

RegQueryInfoKey(hMainKey, NULL, NULL, NULL, &cSubKeys, NULL, NULL, &cValues, NULL, NULL, NULL, NULL);

for (DWORD i = 0; i < cSubKeys; i++) {

cbName = MAX\_PATH;

if (RegEnumKeyEx(hMainKey, i, achKey, &cbName, NULL, NULL, NULL, NULL) == ERROR\_SUCCESS) {

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

HKEY hSubKey;

if (RegOpenKeyEx(hMainKey, achKey, 0, KEY\_READ, &hSubKey) == ERROR\_SUCCESS) {

DWORD cSubKeyValues = 0;

RegQueryInfoKey(hSubKey, NULL, NULL, NULL, NULL, NULL, NULL, &cSubKeyValues, NULL, NULL, NULL, NULL);

for (DWORD j = 0; j < cSubKeyValues; j++) {

cbValueName = MAX\_PATH;

cbValueData = sizeof(dataBuffer);

if (RegEnumValue(hSubKey, j, achValue, &cbValueName, NULL, &valueType, dataBuffer, &cbValueData) == ERROR\_SUCCESS) {

if (valueType == REG\_SZ) {

SendMessage(hListBox, LB\_ADDSTRING, 0, (LPARAM)(std::wstring(achValue) + L": " + (wchar\_t\*)dataBuffer).c\_str());

}

}

}

RegCloseKey(hSubKey);

}

}

}

RegCloseKey(hMainKey);

}

}

void RefreshListBox() {

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

ListSubKeys();

}

void CreateKey(const wchar\_t\* keyName) {

HKEY hMainKey = OpenMainKey();

HKEY hSubKey;

if (RegCreateKeyEx(hMainKey, keyName, 0, NULL, 0, KEY\_ALL\_ACCESS, NULL, &hSubKey, NULL) == ERROR\_SUCCESS) {

MessageBox(NULL, L"Ключ создан.", L"Успех", MB\_OK | MB\_ICONINFORMATION);

RegCloseKey(hSubKey);

}

else {

MessageBox(NULL, L"Ошибка создания ключа.", L"Ошибка", MB\_OK | MB\_ICONERROR);

}

RegCloseKey(hMainKey);

}

void SetValue(const wchar\_t\* keyName, const wchar\_t\* valueName, const wchar\_t\* data) {

HKEY hMainKey = OpenMainKey();

HKEY hSubKey;

wchar\_t fullPath[256]; // Выделяем достаточно места под полный путь

wsprintf(fullPath, L"MyProject\\%s", keyName);

if (RegOpenKeyEx(hMainKey, keyName, 0, KEY\_SET\_VALUE, &hSubKey) == ERROR\_SUCCESS) {

if (RegSetValueEx(hSubKey, valueName, 0, REG\_SZ, (BYTE\*)data, (wcslen(data) + 1) \* sizeof(wchar\_t)) == ERROR\_SUCCESS) {

MessageBox(NULL, L"Значение установлено.", L"Успех", MB\_OK | MB\_ICONINFORMATION);

}

else {

MessageBox(NULL, L"Ошибка установки значения.", L"Ошибка", MB\_OK | MB\_ICONERROR);

}

RegCloseKey(hSubKey);

}

else {

MessageBox(NULL, L"Ошибка открытия подключа.", L"Ошибка", MB\_OK | MB\_ICONERROR);

}

RegCloseKey(hMainKey);

}

void DeleteKey(const wchar\_t\* keyName) {

HKEY hMainKey = OpenMainKey();

if (RegDeleteKey(hMainKey, keyName) == ERROR\_SUCCESS) {

MessageBox(NULL, L"Ключ удален.", L"Успех", MB\_OK | MB\_ICONINFORMATION);

}

else {

MessageBox(NULL, L"Ошибка удаления ключа.", L"Ошибка", MB\_OK | MB\_ICONERROR);

}

RegCloseKey(hMainKey);

}

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

switch (uMsg) {

case WM\_CREATE: {

CreateWindow(L"STATIC", L"Key:", WS\_CHILD | WS\_VISIBLE, 20, 20, 60, 20, hwnd, NULL, NULL, NULL);

hKeyEdit = CreateWindow(L"EDIT", L"", WS\_CHILD | WS\_VISIBLE | WS\_BORDER, 80, 20, 200, 20, hwnd, NULL, NULL, NULL);

CreateWindow(L"STATIC", L"Value:", WS\_CHILD | WS\_VISIBLE, 20, 50, 60, 20, hwnd, NULL, NULL, NULL);

hValueEdit = CreateWindow(L"EDIT", L"", WS\_CHILD | WS\_VISIBLE | WS\_BORDER, 80, 50, 200, 20, hwnd, NULL, NULL, NULL);

CreateWindow(L"STATIC", L"Data:", WS\_CHILD | WS\_VISIBLE, 20, 80, 60, 20, hwnd, NULL, NULL, NULL);

hDataEdit = CreateWindow(L"EDIT", L"", WS\_CHILD | WS\_VISIBLE | WS\_BORDER, 80, 80, 200, 20, hwnd, NULL, NULL, NULL);

CreateWindow(L"BUTTON", L"Create Key", WS\_CHILD | WS\_VISIBLE, 20, 110, 150, 30, hwnd, (HMENU)1, NULL, NULL);

CreateWindow(L"BUTTON", L"Modify Key", WS\_CHILD | WS\_VISIBLE, 20, 150, 150, 30, hwnd, (HMENU)2, NULL, NULL);

CreateWindow(L"BUTTON", L"Delete Key", WS\_CHILD | WS\_VISIBLE, 20, 190, 150, 30, hwnd, (HMENU)3, NULL, NULL);

hListBox = CreateWindow(L"LISTBOX", NULL, WS\_CHILD | WS\_VISIBLE | WS\_BORDER | WS\_VSCROLL, 300, 20, 170, 200, hwnd, NULL, NULL, NULL);

RefreshListBox();

break;

}

case WM\_COMMAND: {

int len;

wchar\_t keyName[255];

wchar\_t valueName[255];

wchar\_t data[255];

switch (wParam) {

case 1: // Create Key

len = GetWindowTextW(hKeyEdit, keyName, ARRAYSIZE(keyName));

if (len > 0) {

CreateKey(keyName);

RefreshListBox();

}

break;

case 2: // Modify Key

GetWindowTextW(hKeyEdit, keyName, ARRAYSIZE(keyName));

GetWindowTextW(hValueEdit, valueName, ARRAYSIZE(valueName));

GetWindowTextW(hDataEdit, data, ARRAYSIZE(data));

SetValue(keyName, valueName, data);

RefreshListBox();

break;

case 3: // Delete Key

len = GetWindowTextW(hKeyEdit, keyName, ARRAYSIZE(keyName));

if (len > 0) {

DeleteKey(keyName);

RefreshListBox();

}

break;

}

break;

}

case WM\_DESTROY:

PostQuitMessage(0);

return 0;

}

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

}

int WINAPI wWinMain(HINSTANCE hInstance, HINSTANCE, PWSTR pCmdLine, int nCmdShow) {

WNDCLASS wc = {};

wc.lpfnWndProc = WindowProc;

wc.hInstance = hInstance;

wc.lpszClassName = CLASS\_NAME;

RegisterClass(&wc);

HWND hwnd = CreateWindowEx(0, CLASS\_NAME, WINDOW\_NAME, WS\_OVERLAPPEDWINDOW, CW\_USEDEFAULT, CW\_USEDEFAULT,

500, 300, NULL, NULL, hInstance, NULL);

ShowWindow(hwnd, nCmdShow);

MSG msg = {};

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

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return 0;

}