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. А. Тиханёнок

Преподаватель Н. Ю. Гриценко

Минск 2023

СОДЕРЖАНИЕ

[1 Цель работы 3](#_Toc146752068)

[2 Теоретические сведения 4](#_Toc146752069)

[3 Результат выполнения 5](#_Toc146752070)

[Заключение 6](#_Toc146752071)

[Список использованных источников 7](#_Toc146752072)

[Приложение А (обязательное) Листинг кода 8](#_Toc146752073)

1. **ЦЕЛЬ РАБОТЫ**

Цель работы изучить и применить на практике знания об реестрах и журналах *Windows*, научиться получать доступ к реестру *Windows*, работа с журналами *Windows*, изменять другие вспомогательные средства управления. Для достижения цели будет создано приложение для мониторинга изменений в системных параметрах и автоматической реакции на изменения.

1. **ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Реестр и журналы *Windows* являются важными компонентами операционной системы *Microsoft* *Windows*, предназначенными для хранения системной информации и управления событиями. Давайте рассмотрим их более подробно.

Реестр *Windows* - это централизованное хранилище системных настроек, информации об установленных приложениях, аппаратных компонентах и конфигурации операционной системы. Это база данных, используемая Windows для управления системой.

Реестр организован в виде древовидной структуры, где каждый узел представляет собой ключ и содержит пары "имя-значение". Он разделен на пять основных разделов: *HKEY\_CLASSES\_ROOT*, *HKEY\_CURRENT\_USER*, *HKEY\_LOCAL\_MACHINE*, *HKEY\_USERS* и *HKEY\_CURRENT\_CONFIG*.

Реестр используется для хранения настроек системы, информации об установленных приложениях, путей к исполняемым файлам, ассоциаций файлов, параметров устройств и других системных данных.

Доступ к реестру *Windows* можно получить через *API* *Windows* (набор функций для взаимодействия с операционной системой) или с помощью встроенных инструментов, таких как *Regedit* (Редактор реестра).

Журналы *Windows* это механизм записи и отслеживания событий, происходящих в операционной системе. Они предоставляют информацию о работе приложений, ошибках, предупреждениях и других важных событиях.

Журналы *Windows* используются для диагностики и мониторинга работы операционной системы. Они позволяют администраторам и разработчикам отслеживать события, связанные с безопасностью, работой приложений и состоянием системы.

Для доступа к журналам *Windows* можно использовать специализированные инструменты, такие как "Просмотр событий" (*Event Viewer*), а также программные интерфейсы (*API*) для анализа и мониторинга событий.

Кроме реестра и журналов, *Windows* также предоставляет другие средства управления, такие как диспетчер задач (*TaskManager*), службы (*Services*), системные настройки (*ControlPanel*) и другие инструменты, которые позволяют администраторам и пользователям управлять и настраивать различные аспекты операционной системы.

1. **РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ**

В результате выполнения лабораторной работы было создано приложение для мониторинга изменений в системных параметрах и автоматической реакции на изменения (Рисунок 1).

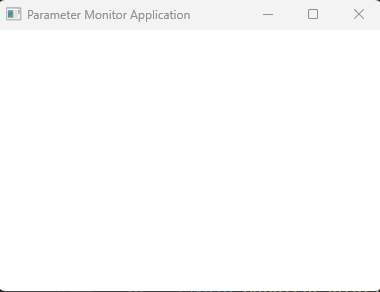


Рисунок 1 – Стартовое окно приложения

При изменении параметров системы выводится соответствующее сообщение в виде окна уведомления, в данном случае изменялось расширение экрана (Рисунок 2).

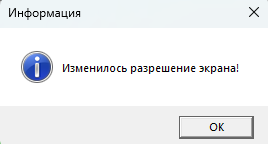


Рисунок 2– Сообщение при изменении параметров экрана

При изменении параметров системы выводится соответствующее сообщение в виде окна уведомления, в данном случае изменялся язык системы (Рисунок 3).

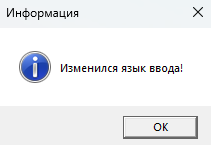


Рисунок 3– Сообщение при изменении языка системы

# ЗАКЛЮЧЕНИЕ

В ходе данной лабораторной работы были получены знания по теме лабораторной работы и были изучены различные аспекты.

Реестр Windows представляет собой важный инструмент для хранения системных настроек и информации о конфигурации операционной системы. Он структурирован в виде древовидной структуры с парами "имя-значение" и используется для хранения настроек, информации об установленных приложениях и других системных данных.

Доступ к реестру Windows можно получить с использованием API Windows или с помощью специализированных инструментов, таких как Regedit.

Журналы Windows служат для записи и мониторинга событий, происходящих в операционной системе. Они предоставляют информацию о работе приложений, ошибках, предупреждениях и других важных событиях.

Инструменты, такие как "Просмотр событий," позволяют администраторам и разработчикам анализировать и мониторить события в операционной системе.

В дополнение к реестру и журналам, Windows предоставляет другие вспомогательные средства управления, такие как диспетчер задач, службы, системные настройки и другие инструменты, облегчающие управление и настройку различных аспектов операционной системы.

Эти знания о реестре, журналах и других инструментах управления являются важными для администраторов и разработчиков, позволяя им эффективно управлять и мониторить работу операционной системы Windows, обеспечивая её стабильность и производительность.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Щупак Ю. *Win32 API*. Разработка приложений для *Windows*. ─ СПБ: Питер, 2008. ─ 592 с.: ип.
2. Создание классических приложений для *Windows* с использованием *API Win32* [Электронный ресурс]. ─ Режим доступа: https://learn.microsoft.com/ru-ru/windows/win32/api

# ПРИЛОЖЕНИЕ А (обязательное) Листинг кода

Листинг 1 – Файл Lab5.cpp

#include <Windows.h>

#include <tchar.h>

#include <evntrace.h>

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

int mainScreenWidth = -1;

int mainScreenHeight = -1;

HKL mainInputLocale = NULL;

// Функция для создания записи в журнале Windows Event Log

void LogEventToEventLog(LPCTSTR message) {

HANDLE hEventLog = RegisterEventSource(NULL, \_T("ParameterMonitorApp"));

if (hEventLog) {

const WORD eventCategory = 1;

const DWORD eventId = 1001;

ReportEvent(hEventLog, EVENTLOG\_INFORMATION\_TYPE, eventCategory, eventId, NULL, 1, 0, &message, NULL);

DeregisterEventSource(hEventLog);

}

MessageBox(NULL, message, \_T("Информация"), MB\_ICONINFORMATION);

}

void HandleResolutionChange() {

int newScreenWidth = GetSystemMetrics(SM\_CXSCREEN);

int newScreenHeight = GetSystemMetrics(SM\_CYSCREEN);

if (newScreenWidth != mainScreenWidth || newScreenHeight != mainScreenHeight) {

mainScreenWidth = newScreenWidth;

mainScreenHeight = newScreenHeight;

LogEventToEventLog(\_T("Изменилось разрешение экрана!"));

}

}

void HandleInputLocaleChange() {

HKL newInputLocale = GetKeyboardLayout(0);

if (newInputLocale != mainInputLocale) {

mainInputLocale = newInputLocale;

LogEventToEventLog(\_T("Изменился язык ввода!"));

}

}

void HandleSystemParamChange(UINT uiAction) {

if (uiAction == SPI\_SETDESKWALLPAPER) {

LogEventToEventLog(\_T("Изменились обои рабочего стола!"));

}

}

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

WNDCLASSEX wc = { sizeof(WNDCLASSEX), CS\_CLASSDC, WndProc, 0L, 0L, GetModuleHandle(NULL), NULL, NULL, NULL, NULL, \_T("ParameterMonitorApp"), NULL };

RegisterClassEx(&wc);

HWND hwnd = CreateWindow(wc.lpszClassName, \_T("Parameter Monitor Application"), WS\_OVERLAPPEDWINDOW, 100, 100, 400, 300, NULL, NULL, wc.hInstance, NULL);

ShowWindow(hwnd, nCmdShow);

UpdateWindow(hwnd);

MSG msg;

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

TranslateMessage(&msg);

DispatchMessage(&msg);

HandleResolutionChange();

HandleInputLocaleChange();

}

return msg.wParam;

}

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

switch (msg) {

case WM\_CREATE:

SetTimer(hwnd, 1, 1000, NULL);

case WM\_TIMER:

if (wParam == 1) {

HandleResolutionChange();

HandleInputLocaleChange();

}

break;

case WM\_DISPLAYCHANGE:

HandleResolutionChange();

break;

case WM\_INPUTLANGCHANGE:

HandleInputLocaleChange();

break;

case WM\_SETTINGCHANGE:

HandleSystemParamChange(static\_cast<UINT>(wParam));

break;

case WM\_DESTROY:

KillTimer(hwnd, 1);

PostQuitMessage(0);

break;

default:

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

}

return 0;

}