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

Скроба А.П.

Проверил:

Гриценко Н.Ю.

Минск 2023

**СОДЕРЖАНИЕ**

[1 Цель работы 3](#_Toc146675523)

[2 Краткие теоретические сведения 4](#_Toc146675524)

[3 Результат выполнения программы 5](#_Toc146675525)

[Список использованных источников 9](#_Toc146675526)

[Приложение А 10](#_Toc146675527)

# 1 ЦЕЛЬ РАБОТЫ

Разработать приложение для мониторинга изменений в системных параметрах и автоматической реакции на изменения разрешения экрана, раскладки клавиатуры, фона рабочего стола, краткого формата времени, подключения и отключения питания компьютера.

**2 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Реестр Windows или системный реестр — иерархически построенная база данных параметров и настроек в большинстве операционных систем Microsoft Windows. Реестр содержит информацию и настройки для аппаратного обеспечения, программного обеспечения, профилей пользователей, предустановки. Большинство изменений в Панели управления, ассоциации файлов, системные политики, список установленного ПО фиксируются в реестре.

Физически реестр состоит из группы файлов, которые хранятся в папке System32\config. При загрузке ОС из этих файлов собирается база данных с текущими параметрами, которые и применяются в работе Windows.

Структура реестра Windows строго иерархична и имеет четкое построение. Основная его составная часть – это ключи (или параметры), в которых и хранится вся информация. Каждый параметр реестра Windows отвечает за определенное свойство системы. Ключи с данными о смежных настройках компьютера объединены в разделы, которые, в свою очередь, являются подразделами более крупных разделов и т.д.

В реестре могут быть объекты двух типов: разделы и параметры. Параметры или ключи реестра имеют имена, представленные в обычном текстовом виде и значения, которые хранятся в виде стандартизированных записей определенного типа.

# 3 РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОГРАММЫ

Результатом выполнения лабораторной работы является приложение для мониторинга изменений в системных параметрах и автоматической реакции на различные изменения, например, изменение разрешения экрана (рисунок 3.1).

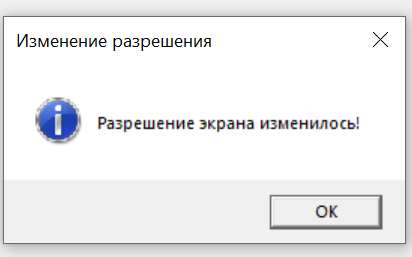
****

Рисунок 3.1 – Сообщение программы об изменении разрешении экрана

Так же программа реагирует на изменение языка раскладки (рисунок 3.2)

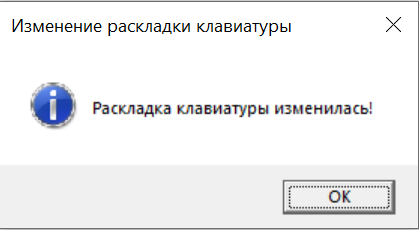


Рисунок 3.2 – Сообщение программы при изменении языка раскладки

Программа также выведет сообщение если компьютер подключен или отключен от питания (рисунок 3.3, рисунок 3.4).

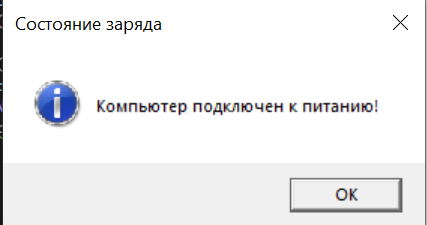


Рисунок 3.3 – Сообщение программы при подключении компьютера к питанию

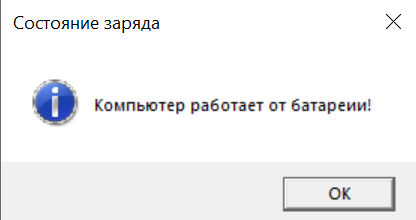


Рисунок 3.4 – Сообщение программы при отключении компьютера от питания

Так же программа выводит сообщение на экран при изменении фона рабочего стола и изменении краткого формата отображения времени (рисунок 3.5, рисунок 3.6).

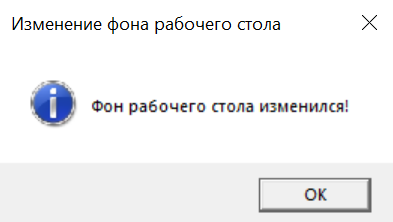
****

Рисунок 3.5 – Сообщение об изменении фона рабочего стола

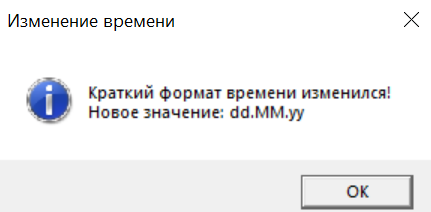


Рисунок 3.6 – Сообщение об изменении краткого формата отображения времени

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] 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.

[2] Что такое системный реестр Windows [Электронный ресурс]. – Режим доступа: https://www.bezpk.ru/blog/item/chto-takoe-sistemnyj-reestr-windows.

[3] Функции реестра [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/windows/win32/sysinfo/registry-functions.

# ПРИЛОЖЕНИЕ А

**(обязательное)**

**Исходный код программы**

Листинг 1 – Файл Resource.h:

#pragma once

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

WNDCLASS WindowClass(HBRUSH BGColor, HCURSOR Cursor, HINSTANCE hInst, LPCWSTR Name, WNDPROC wc);

int WINAPI WinMain(\_In\_ HINSTANCE hInstance, \_In\_opt\_ HINSTANCE hPrevInstance, \_In\_ LPSTR lpCmdLine, \_In\_ int nShowCmd);

void CheckWallpaperChange();

void CheckShortTime();

bool GetDesktopWallpaperPath(WCHAR\* wallpaperPath, DWORD bufferSize);

bool GetShortTime(WCHAR\* wallpaperPath, DWORD bufferSize);

int previousScreenWidth = 0;

int previousScreenHeight = 0;

LANGID previousSystemLanguage = 0;

HKL previousKeyboardLayout = nullptr;

HKEY hKey = nullptr;

WCHAR previousWallpaper[MAX\_PATH];

WCHAR previousTime[12];

DWORD bufferSize;

DWORD buff;

HANDLE hRegChangeEvent = nullptr;

Листинг 2 **–** Файл Source.cpp:

#include <Windows.h>

#include <wtypes.h>

#include "Resource.h"

int WINAPI WinMain(\_In\_ HINSTANCE hInstance, \_In\_opt\_ HINSTANCE hPrevInstance, \_In\_ LPSTR lpCmdLine, \_In\_ int nShowCmd) {

WNDCLASS MainClass = WindowClass((HBRUSH)COLOR\_WINDOW, LoadCursor(NULL, IDC\_ARROW), hInstance, L"MainWndClass", WndProc);

if (!RegisterClassW(&MainClass)) { return -1; }

MSG MainMsg = { };

HWND hwnd = CreateWindow(L"MainWndClass", L"Programm", WS\_OVERLAPPEDWINDOW | WS\_VISIBLE, 500, 200, 300, 300, NULL, NULL, NULL, NULL);

if (!hwnd) {

return 1;

}

// Получаем начальное разрешение экрана

previousScreenWidth = GetSystemMetrics(SM\_CXSCREEN);

previousScreenHeight = GetSystemMetrics(SM\_CYSCREEN);

// Получаем начальный язык системы

previousSystemLanguage = GetUserDefaultUILanguage();

// Получаем начальную раскладку клавиатуры

previousKeyboardLayout = GetKeyboardLayout(0);

GUID guid = GUID\_ACDC\_POWER\_SOURCE;

HPOWERNOTIFY hPowerNotify = RegisterPowerSettingNotification(hwnd, &guid, 0);

if (hPowerNotify == NULL) {

return 1;

}

while (GetMessage(&MainMsg, NULL, NULL, NULL)) {

TranslateMessage(&MainMsg);

DispatchMessage(&MainMsg);

// Проверка изменения разрешения экрана

int currentScreenWidth = GetSystemMetrics(SM\_CXSCREEN);

int currentScreenHeight = GetSystemMetrics(SM\_CYSCREEN);

if (currentScreenWidth != previousScreenWidth || currentScreenHeight != previousScreenHeight) {

MessageBox(NULL, L"Разрешение экрана изменилось!", L"Изменение разрешения", MB\_OK | MB\_ICONINFORMATION);

previousScreenWidth = currentScreenWidth;

previousScreenHeight = currentScreenHeight;

}

// Проверка изменения языка системы

LANGID currentSystemLanguage = GetUserDefaultUILanguage();

if (currentSystemLanguage != previousSystemLanguage) {

MessageBox(NULL, L"Язык системы изменился!", L"Изменение языка системы", MB\_OK | MB\_ICONINFORMATION);

previousSystemLanguage = currentSystemLanguage;

}

// Проверка изменения раскладки клавиатуры

HKL currentKeyboardLayout = GetKeyboardLayout(0);

if (currentKeyboardLayout != previousKeyboardLayout) {

MessageBox(NULL, L"Раскладка клавиатуры изменилась!", L"Изменение раскладки клавиатуры", MB\_OK | MB\_ICONINFORMATION);

previousKeyboardLayout = currentKeyboardLayout;

}

// Проверка изменения фона рабочего стола

CheckWallpaperChange();

// Проверка изменения краткого времени

CheckShortTime();

}

UnregisterPowerSettingNotification(hPowerNotify);

return 0;

}

WNDCLASS WindowClass(HBRUSH BGColor, HCURSOR Cursor, HINSTANCE hInst, LPCWSTR Name, WNDPROC wc) {

WNDCLASS nwc = { };

nwc.hbrBackground = BGColor;

nwc.hCursor = Cursor;

nwc.hInstance = hInst;

nwc.lpszClassName = Name;

nwc.lpfnWndProc = wc;

return nwc;

}

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

if (uMsg == WM\_POWERBROADCAST) {

if (wParam == PBT\_POWERSETTINGCHANGE) {

POWERBROADCAST\_SETTING\* powerSetting = (POWERBROADCAST\_SETTING\*)lParam;

if (powerSetting->PowerSetting == GUID\_ACDC\_POWER\_SOURCE) {

if (powerSetting->Data[0] == 0) {

MessageBox(NULL, L"Компьютер подключен к питанию!", L"Состояние заряда", MB\_ICONINFORMATION);

}

else {

MessageBox(NULL, L"Компьютер работает от батареии!", L"Состояние заряда", MB\_ICONINFORMATION);

}

}

}

}

switch (uMsg) {

case WM\_CREATE:

bufferSize = sizeof(previousWallpaper);

buff = sizeof(previousTime);

GetDesktopWallpaperPath(previousWallpaper, bufferSize);

GetShortTime(previousTime, buff);

break;

case WM\_DESTROY:

if (hRegChangeEvent != nullptr) {

CloseHandle(hRegChangeEvent);

}

PostQuitMessage(0);

break;

default:

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

}

}

void CheckWallpaperChange() {

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

WCHAR currentWallpaper[MAX\_PATH];

bufferSize = sizeof(currentWallpaper);

if (RegQueryValueEx(hKey, L"Wallpaper", nullptr, nullptr, (LPBYTE)currentWallpaper, &bufferSize) == ERROR\_SUCCESS) {

if (lstrcmpW(currentWallpaper, previousWallpaper) != 0) {

MessageBox(NULL, L"Фон рабочего стола изменился!", L"Изменение фона рабочего стола", MB\_OK | MB\_ICONINFORMATION);

lstrcpynW(previousWallpaper, currentWallpaper, MAX\_PATH);

}

}

RegCloseKey(hKey);

}

}

bool GetDesktopWallpaperPath(WCHAR\* wallpaperPath, DWORD bufferSize) {

HKEY hKey = nullptr;

// Открываем ключ реестра, содержащий информацию о фоне рабочего стола

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

DWORD valueType;

// Считываем значение "Wallpaper" из реестра

if (RegQueryValueEx(hKey, L"Wallpaper", nullptr, &valueType, (LPBYTE)wallpaperPath, &bufferSize) == ERROR\_SUCCESS) {

// Проверяем, что тип значения - REG\_SZ (строка)

if (valueType == REG\_SZ) {

RegCloseKey(hKey);

return true;

}

}

RegCloseKey(hKey);

}

return false;

}

void CheckShortTime() {

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

WCHAR currentShortTime[12];

bufferSize = sizeof(currentShortTime);

if (RegQueryValueEx(hKey, L"sShortDate", nullptr, nullptr, (LPBYTE)currentShortTime, &bufferSize) == ERROR\_SUCCESS) {

if (lstrcmpW(currentShortTime, previousTime) != 0) {

WCHAR message[256];

wcscpy\_s(message, L"Краткий формат времени изменился!\nНовое значение: ");

wcscat\_s(message, currentShortTime);

MessageBox(NULL, message, L"Изменение времени", MB\_OK | MB\_ICONINFORMATION);

// Обновляем previousTime

lstrcpynW(previousTime, currentShortTime, MAX\_PATH);

}

}

RegCloseKey(hKey);

}

}

bool GetShortTime(WCHAR\* previousShortTime, DWORD bufferSize) {

HKEY hKey = nullptr;

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

DWORD valueType;

if (RegQueryValueEx(hKey, L"sShortDate", nullptr, &valueType, (LPBYTE)previousShortTime, &bufferSize) == ERROR\_SUCCESS) {

if (valueType == REG\_SZ) {

RegCloseKey(hKey);

return true;

}

}

RegCloseKey(hKey);

}

return false;

}