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. 153503

Филипеня А.Д.

Проверил:

Гриценко Н.Ю.

Минск 2023

СОДЕРЖАНИЕ

[1 Постановка задачи 3](#_Toc146631498)

[2 Краткие теоретические сведения 4](#_Toc146631499)

[3 Результаты выполнения лабораторной работы 6](#_Toc146631500)

[Выводы 8](#_Toc146631501)

[Список использованных источников 9](#_Toc146631502)

[Приложение А (обязательное) Листинг кода 10](#_Toc146631503)

## 1 ПОСТАНОВКА ЗАДАЧИ

Целью выполнения данной лабораторной работы является разработка приложения для мониторинга изменений в системных параметрах и автоматической реакции на изменения.

## 2 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Win32 API (Windows API) представляет собой набор функций и интерфейсов, предоставляемых операционной системой Windows для разработки приложений. Этот мощный набор инструментов обеспечивает доступ к различным функциональным возможностям Windows, включая создание и управление окнами, обработку сообщений, работу с файлами и реестром, а также многие другие операции. Win32 API играет ключевую роль в разработке приложений для Windows и обеспечивает высокую степень контроля над поведением приложений.[1]

Реестр Windows и журналы событий играют важную роль в управлении и мониторинге операционной системы Microsoft Windows. Реестр – это централизованное хранилище системных настроек, параметров и данных о приложениях. Он используется для хранения информации о конфигурации операционной системы, установленных приложениях, драйверах и других системных ресурсах. Реестр предоставляет доступ к этим данным и позволяет как системным администраторам, так и приложениям управлять системными параметрами.

Журналы событий Windows служат для регистрации и мониторинга событий и ошибок в операционной системе. Они предоставляют информацию о работе системы, приложениях и службах, а также могут содержать данные о безопасности и аудите. Журналы событий помогают администраторам и разработчикам выявлять проблемы, мониторить производительность и обнаруживать события, связанные с безопасностью.[2]

Для управления реестром Windows и анализа журналов событий доступны различные инструменты и API. Они позволяют выполнять операции чтения, записи и анализа данных в реестре, а также мониторить и фильтровать события в журналах. Для обеспечения безопасности и надежности системы важно правильно использовать эти средства управления, чтобы предотвратить несанкционированный доступ и обнаруживать проблемы в работе операционной системы.

Для выполнения данной лабораторной работы, были использованы следующие теоретические сведения и концепции:

1 Работа с реестром Windows: при написании кода использовались функции из Windows API (Application Programming Interface) для работы с реестром, такие как RegOpenKeyEx, RegCreateKey, RegSetValueEx, RegQueryValueEx, и RegCloseKey. Эти функции позволяют создавать, читать и записывать данные в реестр.

2 Функция GetKeyboardLayout() получает текущую раскладку клавиатуры.

3 Функция SetTimer() устанавливает таймер для периодического обновления.

4 Приложение управляет окном, его размерами, заголовком и стилями с использованием функций Win32 API, таких как CreateWindow и ShowWindow.

5 Работа с строками: при написании кода были использованы операции с C++ строками.

## 3 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

В ходе выполнения лабораторной работы было разработано приложения для мониторинга изменений раскладки клавиатуры и автоматической реакции на изменения реализованное по средством изменения цвета окна. Результат работы программы при английской раскладке клавиатуре показан на рисунке 3.1.

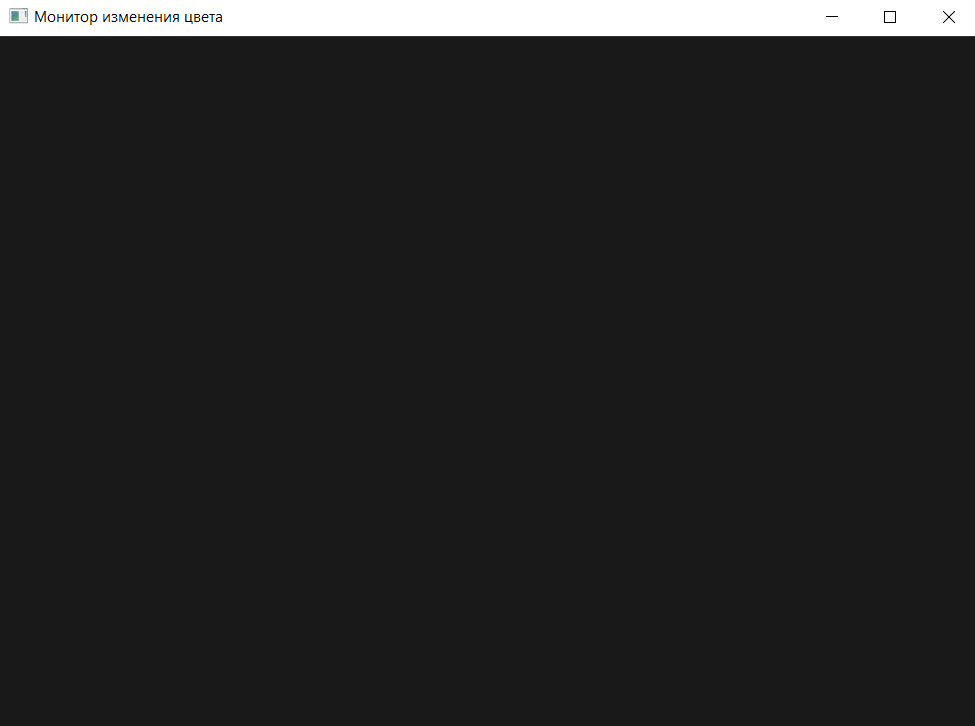


Рисунок 3.1 – Результат работы программы при английской раскладке клавиатуры

При изменении раскладки клавиатуры на русский происходит изменение цвета фона. Результат работы изменения фона окна предоставлен на рисунке 3.2.

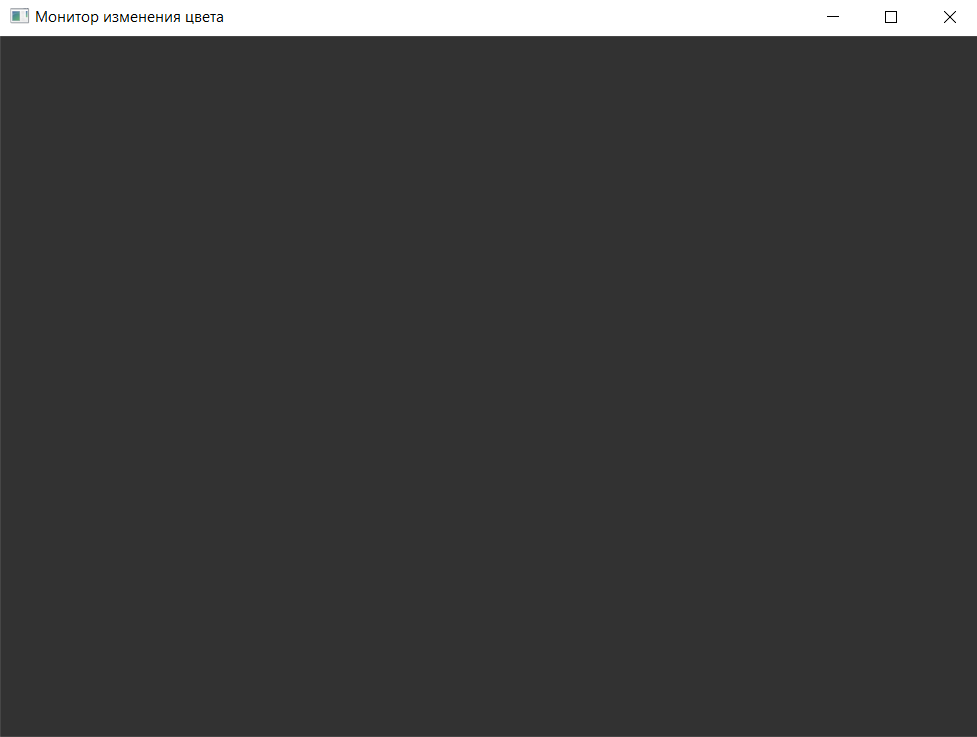


Рисунок 3.2 – Результат работы программы при русской раскладке клавиатуры

Цвета для изменения фона окна в зависимости от раскладки клавиатуры хранятся в реестре. Значение одного из цветов представлено на рисунке 3.3

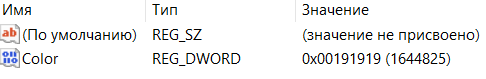


Рисунок 3.3 – Значение цвета

## ВЫВОДЫ

В ходе выполнения данной лабораторной работы было разработано приложения для мониторинга изменения раскладки клавиатуры и автоматической реакции на изменения, реализованное по средством изменения цвета окна.

## 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/debug/system-error-codes--0-499 – Дата доступа 21.10.2023
3. [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/en-us/windows/win32/api/shellapi/nf-shellapi-shellexecutea – Дата доступа: 23.10.2023

**ПРИЛОЖЕНИЕ А**

## (обязательное)

## Листинг кода

**Source.cpp**

#include <windows.h>

#include <string>

// Глобальные переменные

HWND hwnd;

HKEY hRegKey;

COLORREF backgroundColor;

LANGID langId;

// Определение констант для мониторинга изменений в реестре

#define REG\_KEY\_PATH L"SOFTWARE\\MyBrightnessApp"

#define WM\_REGISTRY\_CHANGE (WM\_USER + 1)

void Update();

int get\_lang() {

langId = LOWORD(GetKeyboardLayout(0));

COLORREF backgroundColor;

if (langId == 0x0409) { // en

return 1;

}

else if (langId == 0x0419) { // ru

return 2;

}

else {

return 3;

}

}

int GetCurrentKeyboardLayout()

{

HKL keyboardLayout = GetKeyboardLayout(0);

LCID localeId = LOWORD(keyboardLayout);

LANGID langId = MAKELANGID(PRIMARYLANGID(localeId), SUBLANGID(localeId));

if (langId == LANG\_RUSSIAN)

{

return 1;

}

else if (langId == LANG\_ENGLISH)

{

return 2;

}

else if (langId == MAKELANGID(LANG\_BELARUSIAN, SUBLANG\_DEFAULT))

{

return 3;

}

else

{

return 0;

}

}

// Функция для записи цвета в реестр

void WriteColorToRegistry(int brightnessLevel, COLORREF color) {

HKEY hSubKey;

WCHAR subkey[64];

swprintf\_s(subkey, L"Level%d", brightnessLevel);

if (RegCreateKeyEx(hRegKey, subkey, 0, NULL, REG\_OPTION\_NON\_VOLATILE, KEY\_WRITE, NULL, &hSubKey, NULL) == ERROR\_SUCCESS) {

RegSetValueEx(hSubKey, L"Color", 0, REG\_DWORD, (LPBYTE)&color, sizeof(COLORREF));

RegCloseKey(hSubKey);

}

}

// Функция для создания ключей с цветами в реестре

void CreateColorKeysInRegistry() {

for (int i = 0; i < 10; i++) {

COLORREF color = RGB(i \* 25, i \* 25, i \* 25);

WriteColorToRegistry(i, color);

}

}

//KEY\_NOTIFY - Это флаг, указывающий, что мы хотим открыть ключ для мониторинга изменений в реестре

void OpenRegistryKeyForMonitoring() {

LONG result = RegOpenKeyEx(HKEY\_LOCAL\_MACHINE, REG\_KEY\_PATH, 0, KEY\_NOTIFY, &hRegKey);

if (result != ERROR\_SUCCESS) {

MessageBox(NULL, L"Не удалось открыть ключ Реестра", L"Ошибка", MB\_ICONERROR);

}

}

COLORREF ReadColorFromRegistry(int brightnessLevel) {

HKEY hSubKey;

WCHAR subkey\_name[64];

swprintf\_s(subkey\_name, L"Level%d", brightnessLevel);

//KEY\_QUERY\_VALUE чтение значений

if (RegOpenKeyEx(hRegKey, subkey\_name, 0, KEY\_QUERY\_VALUE, &hSubKey) == ERROR\_SUCCESS) {

DWORD valueType;

COLORREF color = RGB(255, 255, 255);

DWORD dataSize = sizeof(COLORREF);

if (RegQueryValueEx(hSubKey, L"Color", NULL, &valueType, (LPBYTE)&color, &dataSize) == ERROR\_SUCCESS) {

RegCloseKey(hSubKey);

return color;

}

RegCloseKey(hSubKey);

}

return RGB(255, 255, 255);

}

void UpdateWindowColor(COLORREF color) {

backgroundColor = color;

// Установка цвета фона окна GCLP\_HBRBACKGROUND - Это флаг, который указывает на свойство класса окна, отвечающее за цвет фона.

SetClassLongPtr(hwnd, GCLP\_HBRBACKGROUND, (LONG)CreateSolidBrush(backgroundColor));

RedrawWindow(hwnd, NULL, NULL, RDW\_ERASE | RDW\_INVALIDATE);

}

// Функция для мониторинга изменений в ключе Реестра

void MonitorRegistryChanges() {

RegNotifyChangeKeyValue(hRegKey, TRUE, REG\_NOTIFY\_CHANGE\_LAST\_SET, hwnd, TRUE);

}

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

{

switch (uMsg)

{

case WM\_CREATE: {

SetTimer(hwnd, 1, 1000, NULL);

Update();

break;

}

case WM\_TIMER:

Update();

break;

case WM\_SETTINGCHANGE:

if (wParam == SPI\_SETDESKWALLPAPER)

{

}

break;

default:

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

}

return 0;

}

void Update() {

int i = get\_lang();

UpdateWindowColor(ReadColorFromRegistry(i));

}

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

WNDCLASS wc = { 0 };

wc.lpfnWndProc = WindowProc;

wc.hInstance = hInstance;

wc.lpszClassName = L"BrightnessColorApp";

RegisterClass(&wc);

hwnd = CreateWindow(L"BrightnessColorApp", L"Монитор изменения цвета", WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, CW\_USEDEFAULT, 800, 600, NULL, NULL, hInstance, NULL);

if (hwnd == NULL) {

MessageBox(NULL, L"Не удалось создать окно", L"Ошибка", MB\_ICONERROR);

return 1;

}

ShowWindow(hwnd, nCmdShow);

OpenRegistryKeyForMonitoring();

MonitorRegistryChanges();

Update();

MSG msg;

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

TranslateMessage(&msg);

DispatchMessage(&msg);

}

RegCloseKey(hRegKey);

return 0;

}