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 Д. С. Кончик

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**

Изучить структуру и принципы работы с реестром и журналами операционной системы *Windows*, получить практические навыки доступа к реестру и управления журналами с помощью встроенных и доступных инструментов, а также ознакомиться с другими вспомогательными средствами управления операционной системой.

1. **ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Реестр – это иерархическая база данных, содержащая данные, критически важные для работы *Windows*, а также приложений и служб, работающих в *Windows*. Данные структурированы в формате дерева. Каждый узел в дереве называется ключом. Каждый ключ может содержать как подразделы, так и записи данных, называемые значениями. Иногда наличие ключа – это все данные, необходимые приложению; в других случаях приложение открывает ключ и использует значения, связанные с ключом. Ключ может иметь любое количество значений, а значения могут быть в любой форме.

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 [1].

Хотя существует несколько технических ограничений на тип и размер данных, которые приложение может хранить в реестре, существуют определенные практические рекомендации по повышению эффективности системы. Приложение должно хранить данные конфигурации и инициализации в реестре, а также другие типы данных в другом месте.

Как правило, данные, состоящие из более чем одного или двух килобайт, должны храниться в виде файла и ссылаться на них с помощью раздела в реестре, а не как значение. Вместо того чтобы дублировать большие фрагменты данных в реестре, приложение должно сохранить данные в виде файла и ссылаться на него. Исполняемый двоичный код никогда не должен храниться в реестре [2].

Данные об ошибках и событиях из разных приложений сложно объединить в один полный отчет. Ведение журнала событий обеспечивает стандартный централизованный способ записи важных программных и аппаратных событий приложениями (и операционной системой). Служба ведения журнала событий записывает события из различных источников и сохраняет их в одной коллекции, называемой журналом событий. Просмотр событий позволяет просматривать журналы; программный интерфейс также позволяет просматривать журналы [3].

1. **РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ**

В результате выполнения лабораторной работы было создано графическое приложение стрелочных часов.

Для изменения часового пояса используется обработка нажатия клавиш *VK\_LEFT* и *VK\_RIGHT*. Информация об изменении часового пояса, а также о запуске и остановке приложения записывается в журнал событий (рисунок 1).

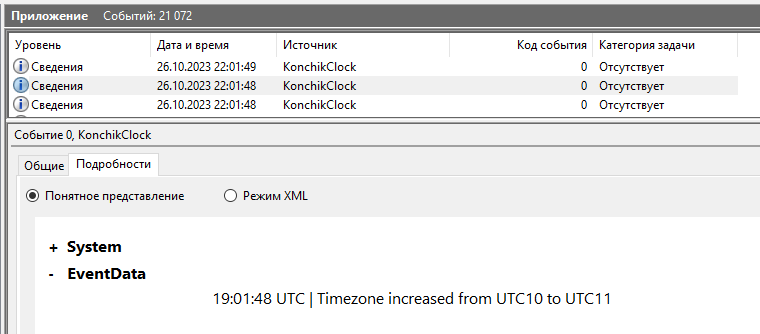


Рисунок 1– Журнал событий

Также с помощью реестра была реализована возможность автоматического сохранения при изменении и загрузки при запуске приложения часового пояса. (рисунок 2).

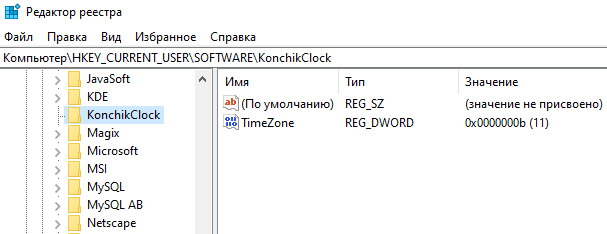


Рисунок 2 – Реестр

# 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 и редактирования ключей реестра, а также записи в системный журнал.

С помощью разработанного приложения «Стрелочные часы» была продемонстрирована возможность сохранения настроек часового пояса в реестре и ведение журнала с информацией о включении и выключении приложения, а также изменении часового пояса.

# 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*

#define \_USE\_MATH\_DEFINES

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <windows.h>

#include <string>

#include <string>

#include <ctime>

#include <map>

#include <thread>

#include "Registry.h"

#include "Constants.h"

#include "EventLogging.h"

using namespace std;

int utcOffset = DEFAULT\_TIMEZONE;

HHOOK keyboardHook;

LRESULT CALLBACK KeyboardProc(int nCode, WPARAM wParam, LPARAM lParam);

LRESULT CALLBACK MainWindowProcedure(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam);

void PlaySoundAsync(wstring soundFile);

RECT reduceRect(RECT rect, const double alpha);

void DrawClock(HDC hdc, RECT rect, int hour, int minute, int second);

void DisplayTimeZone(HWND hWnd);

void UpdateClock(HWND hWnd);

int WINAPI WinMain(HINSTANCE hInst, HINSTANCE, LPSTR cmdline, int ss) {

utcOffset = LoadTimeZoneFromRegistry();

keyboardHook = SetWindowsHookEx(WH\_KEYBOARD\_LL, KeyboardProc, hInst, 0);

// Создание класса главного окна

WNDCLASS wc;

wc.style = NULL;

wc.lpfnWndProc = MainWindowProcedure;

wc.cbClsExtra = NULL;

wc.cbWndExtra = NULL;

wc.hInstance = hInst;

wc.hIcon = NULL;

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

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

wc.lpszMenuName = NULL;

wc.lpszClassName = L"MainWindowClass";

if (!RegisterClass(&wc))

return EXIT\_FAILURE;

// Создание главного окна

HWND hWnd = CreateWindow(

L"MainWindowClass", L"Clock",

WS\_OVERLAPPEDWINDOW,

0, 0, 500, 500,

NULL, NULL, hInst, NULL

);

if (!hWnd)

return EXIT\_FAILURE;

ShowWindow(hWnd, ss);

SetTimer(hWnd, NULL, 1000, NULL);

// Цикл обработки сообщений

MSG msg;

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

TranslateMessage(&msg);

DispatchMessage(&msg);

}

UnhookWindowsHookEx(keyboardHook);

return msg.wParam;

}

LRESULT CALLBACK KeyboardProc(int nCode, WPARAM wParam, LPARAM lParam) {

if (nCode >= 0) {

if (wParam == WM\_KEYDOWN) {

KBDLLHOOKSTRUCT\* pKeyInfo = (KBDLLHOOKSTRUCT\*)(lParam);

if (pKeyInfo->vkCode == VK\_SPACE) {

thread soundThread(PlaySoundAsync, L"audio/kukushka.wav");

soundThread.detach();

WriteToEventLog(L"Kukushka played");

}

}

}

return CallNextHookEx(keyboardHook, nCode, wParam, lParam);

}

// Функция обработки сообщений

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

switch (msg)

{

case WM\_CREATE:

{

thread soundThread(PlaySoundAsync, L"audio/start-windows.wav");

soundThread.detach();

WriteToEventLog(L"Application launched");

break;

}

case WM\_SIZE:

{

UpdateClock(hWnd);

break;

}

case WM\_GETMINMAXINFO: {

MINMAXINFO\* minsize = (MINMAXINFO\*)lParam;

minsize->ptMinTrackSize.x = 400;

minsize->ptMinTrackSize.y = 400;

break;

}

case WM\_TIMER: {

UpdateClock(hWnd);

break;

}

case WM\_KEYDOWN:

switch (wParam) {

case VK\_LEFT:

if (utcOffset <= -11) {

MessageBox(NULL, L"Minimum time zone: UTC-11", L"Warning", MB\_ICONWARNING | MB\_OK);

}

else {

utcOffset -= 1;

UpdateClock(hWnd);

SaveTimeZoneToRegistry(utcOffset);

wstring log = wstring(L"Timezone decreased from UTC") + to\_wstring(utcOffset) + L" to UTC" + to\_wstring(utcOffset - 1);

WriteToEventLog(log);

}

break;

case VK\_RIGHT:

if (utcOffset >= 12) {

MessageBox(NULL, L"Maximum time zone: UTC+12", L"Warning", MB\_ICONWARNING | MB\_OK);

}

else {

utcOffset += 1;

UpdateClock(hWnd);

SaveTimeZoneToRegistry(utcOffset);

wstring log = wstring(L"Timezone increased from UTC") + to\_wstring(utcOffset - 1) + L" to UTC" + to\_wstring(utcOffset);

WriteToEventLog(log);

}

break;

}

break;

case WM\_DESTROY:

{

WriteToEventLog(L"Application stopped");

PostQuitMessage(0);

break;

}

}

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

}

void PlaySoundAsync(wstring soundFile) {

PlaySound(soundFile.c\_str(), NULL, SND\_FILENAME);

}

// Уменьшить прямоугольик отрисовки

RECT reduceRect(RECT rect, const double alpha) {

RECT result;

// Исходные размеры

int width = rect.right - rect.left;

int height = rect.bottom - rect.top;

// Новые размеры

int newWidth = (int)(width \* alpha);

int newHeight = (int)(height \* alpha);

// Смещение по X и Y для центрирования уменьшенного прямоугольника

int xOffset = (width - newWidth) / 2;

int yOffset = (height - newHeight) / 2;

// Новые координаты всех границ прямоугольника

result.left = rect.left + xOffset;

result.top = rect.top + yOffset;

result.right = result.left + newWidth;

result.bottom = result.top + newHeight;

return result;

}

// Отрисовка часов

void DrawClock(HDC hdc, RECT rect, int hour, int minute, int second) {

// Вычисление в соответствии с размером окна

int width = rect.right - rect.left;

int height = rect.bottom - rect.top;

int diameter = min(width, height);

int radius = diameter / 2;

int centerX = rect.left + diameter / 2;

int centerY = rect.top + diameter / 2;

// Отрисовка большого круга

int penWidth = diameter / 100;

HPEN hPen = CreatePen(PS\_SOLID, penWidth, RGB(0, 0, 0));

HPEN hOldPen = (HPEN)SelectObject(hdc, hPen);

Ellipse(hdc, rect.left, rect.top, rect.left + diameter, rect.top + diameter);

// Установка размера шрифта в зависимости от диаметра

int fontSize = diameter / 10;

HFONT hFont = CreateFont(fontSize, 0, 0, 0, FW\_NORMAL, FALSE, FALSE, FALSE, DEFAULT\_CHARSET,

OUT\_OUTLINE\_PRECIS, CLIP\_DEFAULT\_PRECIS, CLEARTYPE\_QUALITY, DEFAULT\_PITCH, L"Arial");

HFONT hOldFont = (HFONT)SelectObject(hdc, hFont);

// Рисование цифр

for (int i = 1; i <= 12; i++) {

// Расчет координаты (x, y) для цифры

double angle = i \* (360.0 / 12.0);

double radian = angle \* M\_PI / 180.0;

int textX = centerX + (radius \* sin(radian));

int textY = centerY - (radius \* cos(radian));

// Создание кисти для заливки окружности

HBRUSH hBrush = CreateSolidBrush(RGB(255, 255, 255));

HBRUSH hOldBrush = (HBRUSH)SelectObject(hdc, hBrush);

// Отрисовка окружности для цифры

int radius = diameter \* 0.08;

Ellipse(hdc, textX - radius, textY - radius, textX + radius, textY + radius);

// Восстановка предыдущей кисти

SelectObject(hdc, hOldBrush);

DeleteObject(hBrush);

// Отрисовка цифры

SetBkMode(hdc, TRANSPARENT);

wstring digit = to\_wstring(i);

SIZE textSize;

GetTextExtentPoint32W(hdc, digit.c\_str(), digit.length(), &textSize);

SetTextColor(hdc, RGB((i \* 25) % 256, (i \* 45) % 256, (i \* 65) % 256));

TextOutW(hdc, textX - textSize.cx / 2, textY - textSize.cy / 2, digit.c\_str(), digit.length());

SetBkMode(hdc, OPAQUE);

}

// Восстановление предыдущего пера

SelectObject(hdc, hOldPen);

DeleteObject(hPen);

// Восстановление предыдущего шрифта

SelectObject(hdc, hOldFont);

DeleteObject(hFont);

// Отрисовка часовой стрелки

int hourAngle = (hour % 12) \* 30 + (minute / 2);

int hourLength = radius \* 0.25;

int hx = centerX + hourLength \* sin(hourAngle \* M\_PI / 180.0);

int hy = centerY - hourLength \* cos(hourAngle \* M\_PI / 180.0);

int hourPenWidth = diameter / 100;

HPEN hHourPen = CreatePen(PS\_SOLID, hourPenWidth, RGB(0, 0, 0));

hOldPen = (HPEN)SelectObject(hdc, hHourPen);

MoveToEx(hdc, centerX, centerY, NULL);

LineTo(hdc, hx, hy);

SelectObject(hdc, hOldPen);

DeleteObject(hHourPen);

// Отрисовка минутной стрелки

int minuteAngle = minute \* 6 + (second / 10);

int minuteLength = radius \* 0.45;

int mx = centerX + minuteLength \* sin(minuteAngle \* M\_PI / 180.0);

int my = centerY - minuteLength \* cos(minuteAngle \* M\_PI / 180.0);

int minutePenWidth = diameter / 150;

HPEN hMinutePen = CreatePen(PS\_SOLID, minutePenWidth, RGB(128, 128, 128));

hOldPen = (HPEN)SelectObject(hdc, hMinutePen);

MoveToEx(hdc, centerX, centerY, NULL);

LineTo(hdc, mx, my);

SelectObject(hdc, hOldPen);

DeleteObject(hMinutePen);

// Отрисовка секунднрй стрелки

int secondAngle = second \* 6;

int secondLength = radius \* 0.55;

int sx = centerX + secondLength \* sin(secondAngle \* M\_PI / 180.0);

int sy = centerY - secondLength \* cos(secondAngle \* M\_PI / 180.0);

int secondPenWidth = diameter / 200;

HPEN hSecondPen = CreatePen(PS\_SOLID, secondPenWidth, RGB(255, 0, 0));

hOldPen = (HPEN)SelectObject(hdc, hSecondPen);

MoveToEx(hdc, centerX, centerY, NULL);

LineTo(hdc, sx, sy);

SelectObject(hdc, hOldPen);

DeleteObject(hSecondPen);

HBRUSH hBrush = CreateSolidBrush(RGB(0, 0, 0));

HBRUSH hOldBrush = (HBRUSH)SelectObject(hdc, hBrush);

int r = radius / 30;

Ellipse(hdc, centerX - r, centerY - r, centerX + r, centerY + r);

SelectObject(hdc, hOldBrush);

DeleteObject(hBrush);

}

void DisplayTimeZone(HWND hWnd) {

wstring new\_title = BASE\_TITLE;

if (utcOffset < 0)

new\_title += to\_wstring(utcOffset);

else if (utcOffset > 0)

new\_title += L"+" + to\_wstring(utcOffset);

new\_title += L" | " + TIME\_ZONE\_MAP[utcOffset];

SetWindowText(hWnd, new\_title.c\_str());

}

void UpdateClock(HWND hWnd) {

HDC hdc = GetDC(hWnd);

RECT clientRect;

GetClientRect(hWnd, &clientRect);

HBRUSH bgBrush = CreateSolidBrush(GetSysColor(COLOR\_WINDOW));

FillRect(hdc, &clientRect, bgBrush);

DeleteObject(bgBrush);

time\_t now = time(0);

tm\* gmtm = gmtime(&now);

DrawClock(hdc, reduceRect(clientRect, 0.8), gmtm->tm\_hour + utcOffset, gmtm->tm\_min, gmtm->tm\_sec);

DisplayTimeZone(hWnd);

ReleaseDC(hWnd, hdc);

}

Листинг 2 – Файл *Constants.h*

#pragma once

#include <Windows.h>

#include <string>

#include <map>

using namespace std;

CONST INT DEFAULT\_TIMEZONE = 0;

wstring BASE\_TITLE = L"Clock | UTC";

map<int, wstring> TIME\_ZONE\_MAP = {

{-11, L"Midway Island"},

{-10, L"Honolulu"},

{-9, L"Anchorage"},

{-8, L"Los Angeles"},

{-7, L"Denver"},

{-6, L"Chicago"},

{-5, L"New York"},

{-4, L"Caracas"},

{-3, L"Buenos Aires"},

{-2, L"South Georgia"},

{-1, L"Azores"},

{0, L"London"},

{1, L"Paris"},

{2, L"Athens"},

{3, L"Minsk"},

{4, L"Kabul"},

{5, L"Islamabad"},

{6, L"Almaty"},

{7, L"Bangkok"},

{8, L"Beijing"},

{9, L"Tokyo"},

{10, L"Sydney"},

{11, L"Solomon Islands"},

{12, L"Suva"}

};

LPCSTR timeZoneFileName = "timezone.txt";

CONST INT timeZoneFileSize = 4;

LPCSTR logFileName = "log.txt";

CONST INT logFileSize = 1024 \* 10;

Листинг 3 – Файл *EventLogging.h*

#pragma once

#include <string>

#include <Windows.h>

#include <ctime>

using namespace std;

wstring NumberToStringWithLeadingZero(int number) {

if (number < 10)

return L"0" + to\_wstring(number);

else

return to\_wstring(number);

}

void WriteToEventLog(wstring message) {

HANDLE hEventLog = RegisterEventSource(NULL, L"KonchikClock");

if (hEventLog) {

time\_t now = time(0);

tm\* gmtm = gmtime(&now);

const wchar\_t\* messageStrings[1];

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

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

DeregisterEventSource(hEventLog);

}

}

Листинг 4 – Файл *Registry.h*

#include <Windows.h>

#include <string>

#include "Constants.h"

void SaveTimeZoneToRegistry(int timeZone) {

HKEY hKey;

LONG result = RegCreateKeyEx(HKEY\_CURRENT\_USER, L"Software\\KonchikClock", 0, NULL, 0, KEY\_WRITE, NULL, &hKey, NULL);

if (result != ERROR\_SUCCESS) {

MessageBox(NULL, L"Ошибка при создании ключа реестра", L"Ошибка", MB\_ICONERROR | MB\_OK);

RegCloseKey(hKey);

return;

}

result = RegSetValueEx(hKey, L"TimeZone", 0, REG\_DWORD, reinterpret\_cast<BYTE\*>(&timeZone), sizeof(timeZone));

if (result != ERROR\_SUCCESS) {

MessageBox(NULL, L"Ошибка при записи значения в реестр", L"Ошибка", MB\_ICONERROR | MB\_OK);

}

RegCloseKey(hKey);

}

int LoadTimeZoneFromRegistry() {

HKEY hKey;

LONG result = RegOpenKeyEx(HKEY\_CURRENT\_USER, L"Software\\KonchikClock", 0, KEY\_READ, &hKey);

if (result == ERROR\_SUCCESS) {

DWORD timeZone = 0;

DWORD dataSize = sizeof(timeZone);

result = RegQueryValueEx(hKey, L"TimeZone", NULL, NULL, reinterpret\_cast<BYTE\*>(&timeZone), &dataSize);

if (result != ERROR\_SUCCESS) {

RegCloseKey(hKey);

return 0;

}

RegCloseKey(hKey);

return static\_cast<int>(timeZone);

}

else if (result == ERROR\_FILE\_NOT\_FOUND) {

RegCloseKey(hKey);

SaveTimeZoneToRegistry(DEFAULT\_TIMEZONE);

return DEFAULT\_TIMEZONE;

}

else {

RegCloseKey(hKey);

return 0;

}

}