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 Е. А. Сироткин

Преподаватель Н. Ю. Гриценко

Минск 2023

СОДЕРЖАНИЕ

[1 Цель работы 3](#_Toc146752068)

[2 Теоретические сведения 4](#_Toc146752069)

[3 Результат выполнения 5](#_Toc146752070)

[Заключение 7](#_Toc146752071)

[Список использованных источников 8](#_Toc146752072)

[Приложение А (обязательное) Листинг кода 9](#_Toc146752073)

1. **ЦЕЛЬ РАБОТЫ**

Цель лабораторной работы заключается в изучении структуры и принципов функционирования реестра и журналов операционной системы *Windows*, приобретении практических навыков доступа к реестру и управления журналами с использованием имеющихся инструмен­­тов, а также знакомстве с другими доступными средствами управления операционной системой.

1. **ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Реестр *Windows* (*Windows Registry*) – это централизованное иерархическое хранилище информации и настроек, которое используется операционной системой *Microsoft Windows* для хранения конфигурационных данных, настроек приложений, параметров системы и другой важной информации. Реестр является важной частью операционной системы *Windows* и используется для управления множеством параметров, опций и настроек, необходимых для работы *Windows* и установленных приложений. В реестре *Windows* информация организована в виде иерархии ключей и подключей, а каждый ключ может содержать значения или поддеревья.

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 печатный 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. Имена ключей не локализованы в других языках, хотя значения могут быть [1].

Журналы событий – это специальные файлы, в которые система и приложения записывают значимые для компьютера события, такие как события входа пользователей в систему или ошибки, возникающие при работе приложений. Когда возникают подобные типы событий, система *Windows* создает записи в журналах событий. В детальных описаниях событий пользователи могут найти информацию, полезную для устранения неисправностей, обнаружения причин проблем с системой, приложениями, оборудованием компьютера [2].

Для создания записей в системном журнале (*Event Log*) *Windows* была использованя функция *ReportEvent*. Она позволяет создавать пользовательские записи в журнале событий, указывая тип события, идентификатор ресурса, текстовое описание и другие параметры.

*RegSetValueEx* – это функция для работы с реестром *Windows*. Она используется для установки значения (или создания нового значения) в указанном ключе реестра. Функция принимает параметры, такие как дескриптор ключа, имя значения, тип данных (например, строка или число), и данные, которые вы хотите установить. *RegSetValueEx* позволяет программам изменять настройки и параметры в реестре.

1. **РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ**

В ходе выполнения данной лабораторной работы было разработано приложение для записи списка дел. Предоставляется возможность выбора заголовка и текста заметки, а также очистки поля заметки (рисунок 1).

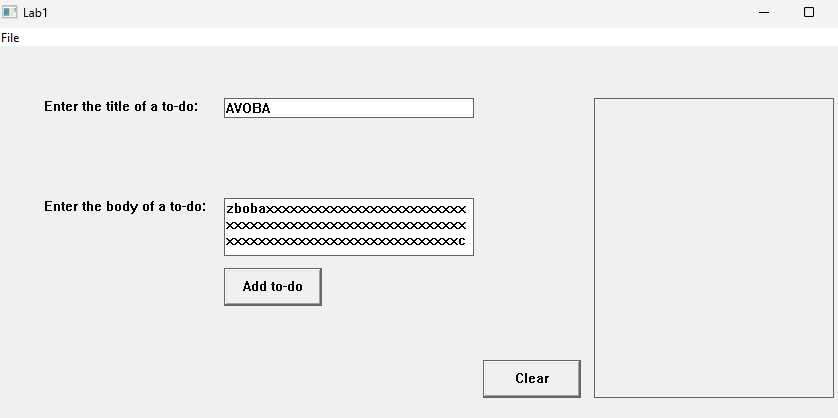


Рисунок 1– Главное окно

При добавлении заметки информация о данном действии заносится в журнал событий (рисунок 2).

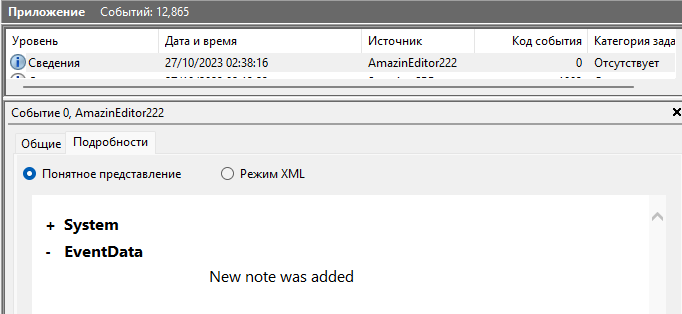


Рисунок 2 – Журнал событий

Благодаря использованию реестра, имеется возможность сохранения введенных данных при изменении и загрузке при запуске данного приложения (рисунок 3).

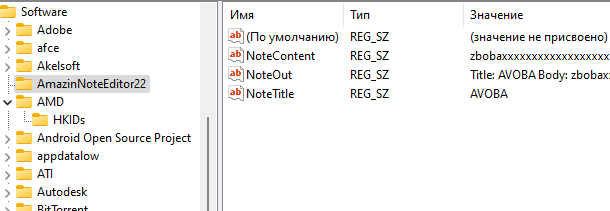


Рисунок 3 – Реестр

# ЗАКЛЮЧЕНИЕ

В ходе выполнения данной лабораторной работы были освоены фундаментальные аспекты пользования реестром и журналами в операционной системе *Windows*. Это включало в себя навыки доступа и работы с реестром и журналами. В рамках работы над приложением была добавлена возможность сохранения данных в реестре для последующего их восстановления, а также записи информации о событиях приложения в журнал.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Структура реестра [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/windows/win32/sysinfo/structure-of-the-registry/.
2. Журналы событий Windows [Электронный ресурс]. – Режим доступа: https://eventlogxp.com/rus/essentials/windowseventlog.html.

# ПРИЛОЖЕНИЕ А (обязательное) Листинг кода

Листинг 1 – Файл *Lab5.cpp*

#include <windows.h>

#include <stdlib.h>

#include <string>

#define FILE\_MENU\_EXIT 3

#define GENERATE\_BUTTON 4

#define CLEA-R\_BUTTON 5

LRESULT CALLBACK WindowProcedure(HWND, UINT, WPARAM, LPARAM);

void AddMenus(HWND);

void AddControls(HWND);

void SaveDataToRegistry(const TCHAR\* keyName, const TCHAR\* valueName, const TCHAR\* data);

bool LoadDataFromRegistry(const TCHAR\* keyName, const TCHAR\* valueName, TCHAR\* buffer, DWORD bufferSize);

void WriteToEventLog(const std::wstring& message);

HWND hTitle, hBody, hOut;

HMENU hMenu;

wchar\_t title[31], body[200], out[250];

int titleLength = 0;

int bodyLength = 0;

int WINAPI WinMain(HINSTANCE hInst, HINSTANCE hPrevInst, LPSTR args, int ncmdshow) {

WNDCLASS wc = { 0 };

wc.hbrBackground = (HBRUSH)COLOR\_WINDOW;

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

wc.hInstance = hInst;

wc.lpszClassName = L"Lab1";

wc.lpfnWndProc = WindowProcedure;

if (!RegisterClassW(&wc)) {

return -1;

}

CreateWindowW(L"Lab1", L"Lab1", WS\_OVERLAPPEDWINDOW | WS\_VISIBLE, 100, 100, 900, 450, NULL, NULL, NULL, NULL);

MSG msg = { 0 };

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

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return 0;

}

LRESULT CALLBACK WindowProcedure(HWND hWnd, UINT msg, WPARAM wp, LPARAM lp) {

switch (msg) {

case WM\_COMMAND:

switch (wp) {

case FILE\_MENU\_EXIT:

MessageBeep(MB\_ICONINFORMATION);

GetWindowText(hTitle, title, 31);

GetWindowText(hBody, body, 200);

GetWindowText(hOut, out, 250);

SaveDataToRegistry(L"Software\\AmazinNoteEditor22", L"NoteTitle", title);

SaveDataToRegistry(L"Software\\AmazinNoteEditor22", L"NoteContent", body);

SaveDataToRegistry(L"Software\\AmazinNoteEditor22", L"NoteOut", out);

DestroyWindow(hWnd);

break;

case GENERATE\_BUTTON:

titleLength = GetWindowTextLength(hTitle);

bodyLength = GetWindowTextLength(hBody);

if (titleLength == 0 || bodyLength == 0) {

MessageBox(hWnd, L"Both title and body must be filled with non-space characters!", L"Validation Error", MB\_ICONERROR);

}

else {

GetWindowText(hTitle, title, 31);

GetWindowText(hBody, body, 200);

if (title[0] == L' ' || body[0] == L' ') {

MessageBox(hWnd, L"Title and body cannot start with a space character!", L"Validation Error", MB\_ICONERROR);

}

else {

wsprintf(out, L"Title: %s\r\n\r\nBody: %s", title, body);

SetWindowText(hOut, out);

WriteToEventLog(L"New note was added");

}

}

break;

case CLEAR\_BUTTON:

SetWindowText(hOut, L"");

break;

}

break;

case WM\_CREATE:

AddMenus(hWnd);

AddControls(hWnd);

TCHAR titleBuffer[31];

if (LoadDataFromRegistry(L"Software\\AmazinNoteEditor22", L"NoteTitle", titleBuffer, sizeof(titleBuffer))) {

SetWindowText(hTitle, titleBuffer);

}

TCHAR bodyBuffer[200];

if (LoadDataFromRegistry(L"Software\\AmazinNoteEditor22", L"NoteContent", bodyBuffer, sizeof(bodyBuffer))) {

SetWindowText(hBody, bodyBuffer);

}

TCHAR outBuffer[250];

if (LoadDataFromRegistry(L"Software\\AmazinNoteEditor22", L"NoteOut", outBuffer, sizeof(outBuffer))) {

SetWindowText(hOut, outBuffer);

}

break;

case WM\_DESTROY:

MessageBeep(MB\_ICONINFORMATION);

GetWindowText(hTitle, title, 31);

GetWindowText(hBody, body, 200);

GetWindowText(hOut, out, 250);

SaveDataToRegistry(L"Software\\AmazinNoteEditor22", L"NoteTitle", title);

SaveDataToRegistry(L"Software\\AmazinNoteEditor22", L"NoteContent", body);

SaveDataToRegistry(L"Software\\AmazinNoteEditor22", L"NoteOut", out);

PostQuitMessage(0);

break;

default:

return DefWindowProcW(hWnd, msg, wp, lp);

}

}

void AddMenus(HWND hWnd) {

hMenu = CreateMenu();

HMENU hFileMenu = CreateMenu();

HMENU hSubMenu = CreateMenu();

AppendMenu(hSubMenu, MF\_STRING, NULL, L"Submenu item");

AppendMenu(hFileMenu, MF\_STRING, FILE\_MENU\_EXIT, L"Exit");

AppendMenu(hMenu, MF\_POPUP, (UINT\_PTR)hFileMenu, L"File");

SetMenu(hWnd, hMenu);

}

void AddControls(HWND hWnd) {

CreateWindow(L"static", L"Enter the title of a to-do:", WS\_VISIBLE | WS\_CHILD, 50, 50, 210, 38, hWnd, NULL, NULL, NULL);

hTitle = CreateWindow(L"edit", L"", WS\_VISIBLE | WS\_CHILD | WS\_BORDER | ES\_MULTILINE, 230, 50, 250, 20, hWnd, NULL, NULL, NULL);

CreateWindow(L"static", L"Enter the body of a to-do:", WS\_VISIBLE | WS\_CHILD, 50, 150, 210, 38, hWnd, NULL, NULL, NULL);

hBody = CreateWindow(L"edit", L"", WS\_VISIBLE | WS\_CHILD | WS\_BORDER | ES\_MULTILINE, 230, 150, 250, 58, hWnd, NULL, NULL, NULL);

CreateWindow(L"button", L"Add to-do", WS\_VISIBLE | WS\_CHILD | WS\_BORDER, 230, 220, 98, 38, hWnd, (HMENU)GENERATE\_BUTTON, NULL, NULL);

CreateWindow(L"button", L"Clear", WS\_VISIBLE | WS\_CHILD | WS\_BORDER, 489, 312, 98, 38, hWnd, (HMENU)CLEAR\_BUTTON, NULL, NULL);

hOut = CreateWindow(L"edit", L"", WS\_VISIBLE | WS\_CHILD | WS\_BORDER | ES\_MULTILINE | ES\_READONLY, 600, 50, 240,300 , hWnd, NULL, NULL, NULL);

}

void SaveDataToRegistry(const TCHAR\* keyName, const TCHAR\* valueName, const TCHAR\* data) {

HKEY hKey;

LONG result = RegCreateKeyEx(HKEY\_CURRENT\_USER, keyName, 0, NULL, REG\_OPTION\_NON\_VOLATILE, KEY\_WRITE, NULL, &hKey, NULL);

if (result == ERROR\_SUCCESS) {

RegSetValueEx(hKey, valueName, 0, REG\_SZ, (const BYTE\*)data, (wcslen(data) + 1) \* sizeof(TCHAR));

RegCloseKey(hKey);

}

}

bool LoadDataFromRegistry(const TCHAR\* keyName, const TCHAR\* valueName, TCHAR\* buffer, DWORD bufferSize) {

HKEY hKey;

LONG result = RegOpenKeyEx(HKEY\_CURRENT\_USER, keyName, 0, KEY\_READ, &hKey);

if (result == ERROR\_SUCCESS) {

DWORD dataSize = bufferSize;

result = RegQueryValueEx(hKey, valueName, NULL, NULL, (BYTE\*)buffer, &dataSize);

RegCloseKey(hKey);

if (result == ERROR\_SUCCESS) {

return true;

}

}

return false;

}

void WriteToEventLog(const std::wstring& message) {

HANDLE hEventLog = RegisterEventSource(NULL, L"AmazinEditor222");

if (hEventLog) {

LPCWSTR messageStrings[1];

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

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

DeregisterEventSource(hEventLog);

}

}