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 А. С. Сорокин

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**

Исследовать и овладеть пониманием структуры, принципов работы с реестром и журналами операционной системы Windows. Получить практические навыки доступа к реестру, управления журналами с использованием встроенных инструментов, а также освоить другие вспомогательные средства управления операционной системой *Windows*.

1. **ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Реестр Windows это централизованное хранилище системной информации и настроек операционной системы *Microsoft Windows*. Он содержит информацию о конфигурации, параметрах, настройках и установках, необходимых для работы операционной системы и приложений. Реестр можно рассматривать как базу данных, где хранятся ключи и значения, организованные в иерархической структуре. [1]

Для доступа к реестру *Windows* используется Редактор реестра (*Registry Editor),* который позволяет просматривать, редактировать и создавать записи в реестре. Важно помнить, что неправильное редактирование реестра может привести к серьезным проблемам с операционной системой, поэтому необходимо быть осторожным и резервировать реестр перед внесением изменений.

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. [2]

Журналы событий *Windows* (*Event Logs*) представляют собой инструмент для записи и отслеживания событий, происходящих в операционной системе. Они используются для диагностики и мониторинга работы системы, выявления ошибок, а также для обеспечения безопасности.

В *Windows* существуют три основных журнала событий: Журнал системы (*System*), Журнал безопасности (*Security*) и Журнал приложений (*Application*). Каждый из них содержит записи о событиях, произошедших в соответствующей области операционной системы. [3]

Для работы с журналами Windows используются инструменты администрирования, такие как "Просмотр событий" (*Event Viewer*), которые позволяют фильтровать, анализировать и экспортировать записи из журналов для последующего анализа и решения проблем.

Обращение к реестру и работа с журналами являются важной частью администрирования и обслуживания операционной системы *Windows*, их корректное использование помогает обеспечить стабильность и безопасность системы.

1. **РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ**

В результате выполнения лабораторной работы было создана программа для создания и управления реестровыми записями *Windows*, включая создание, изменение и удаление ключей и значений (рисунок 1).

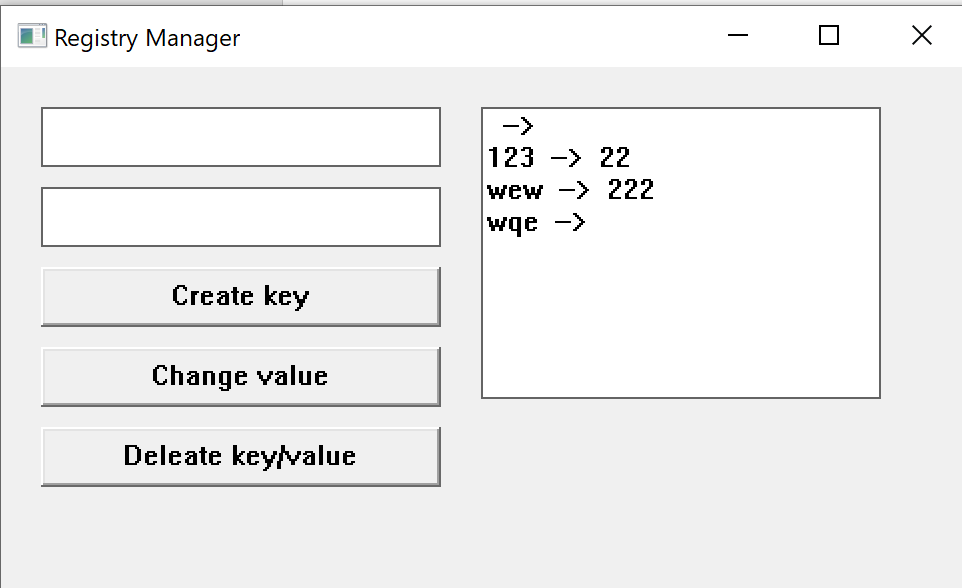


Рисунок 1– Главное окно

В реестр сохраняются созданные ключи и значения, а также имеется возможность их изменить или удалить. (рисунок 2).

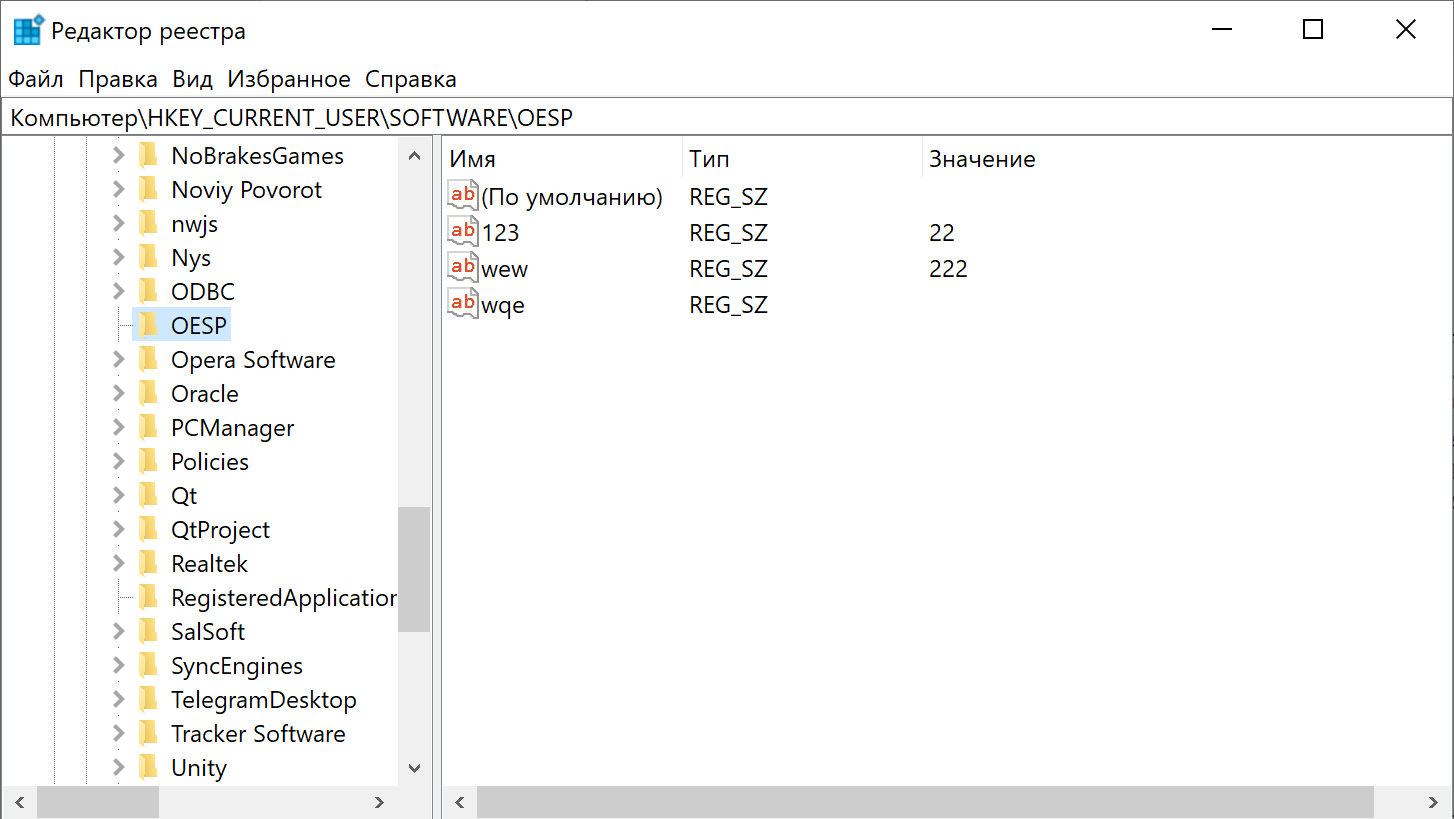


Рисунок 2 – Реестр

Все действия с реестром записываются в журнал событий (рисунок 3).

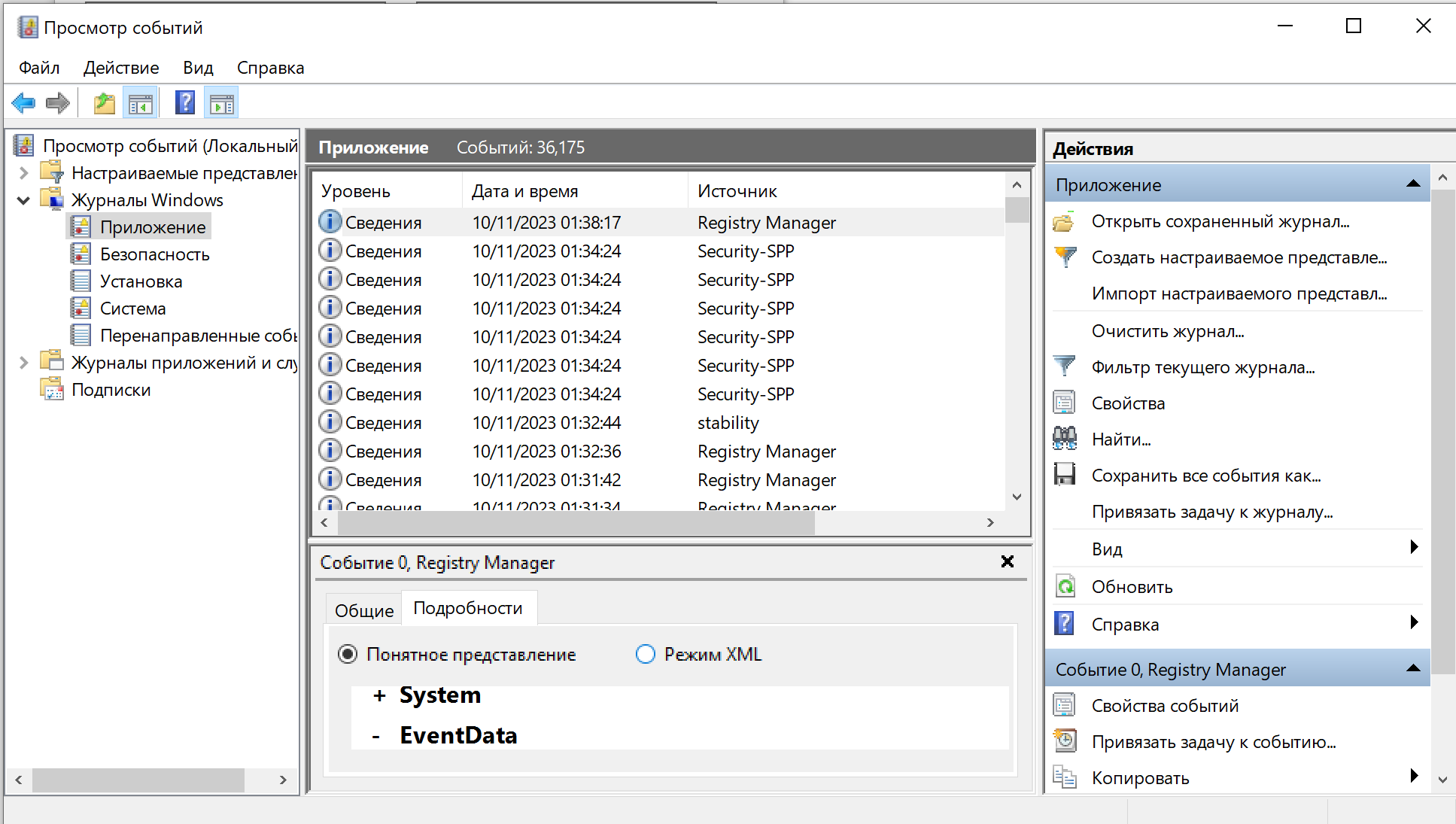


Рисунок 3 – Журнал событий

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

С помощью разработанного приложения была продемонстрирована возможность сохранения данных в реестре и ведение журнала с информацией о каждом действии с реестром.

# 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 *Lab\_5.cpp*

#include <windows.h>

#include "SoftwareDefinitions.h"

#include <string>

#include <iostream>

#include <thread>

#include <chrono>

#include <Pdh.h>

#include <cstdio>

#define ID\_TIMER 1

#define WINDOW\_WIDTH 300

#define WINDOW\_HEIGHT 200

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

WNDCLASS SoftwareMainClass = NewWindowClass((HBRUSH)COLOR\_WINDOW, LoadCursor(NULL, IDC\_HAND), hInst, LoadIcon(NULL, IDI\_ERROR),

L"MainWndClass", SoftwareMainProcedure);

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

CreateWindow(L"MainWndClass", L"Lab\_4 window", WS\_OVERLAPPEDWINDOW | WS\_VISIBLE, 100, 100, 550, 450, NULL, NULL, NULL, NULL);

MSG msg = { };

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

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return 0;

}

WNDCLASS NewWindowClass(HBRUSH BGColor, HCURSOR Cursor, HINSTANCE hInst, HICON Icon, LPCWSTR Name, WNDPROC Procedure) {

WNDCLASS NWC = { 0 };

NWC.hIcon = Icon;

NWC.hCursor = Cursor;

NWC.hInstance = hInst;

NWC.lpszClassName = Name;

NWC.hbrBackground = BGColor;

NWC.lpfnWndProc = Procedure;

return NWC;

}

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

switch (msg)

{

case WM\_COMMAND:

switch (wp) {

case OnMenuClicked1:

MessageBoxA(hWnd, "menu 1 was klicked", "Menu worked", MB\_OK);

break;

case OnCalculate:

break;

case OnExitSoftware:

PostQuitMessage(0);

break;

}

break;

case WM\_CREATE:

MainWndAddMenus(hWnd);

MainWndAddWidgets(hWnd);

AuditRegistryChanges(hWnd);

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

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

}

return 0;

}

void MainWndAddMenus(HWND hwnd) {

HMENU RootMenu = CreateMenu();

HMENU SubMenu = CreateMenu();

//HMENU SubActionMenu = CreateMenu();

//AppendMenu(SubActionMenu, MF\_STRING, OnMenuClicked3, L"Menu 3");

//AppendMenu(SubMenu, MF\_POPUP, (UINT\_PTR)SubActionMenu, L"Action");

AppendMenu(SubMenu, MF\_SEPARATOR, NULL, NULL);

AppendMenu(SubMenu, MF\_STRING, OnExitSoftware, L"Exit");

AppendMenu(RootMenu, MF\_POPUP, (UINT\_PTR)SubMenu, L"File");

AppendMenu(RootMenu, MF\_STRING, (UINT\_PTR)SubMenu, L"Help");

SetMenu(hwnd, RootMenu);

}

void MainWndAddWidgets(HWND hWnd) {

hStaticControl = CreateWindowA("static", "Main Window!", WS\_VISIBLE | WS\_CHILD | ES\_CENTER, 200, 5, 100, 30, hWnd, NULL, NULL, NULL);

hEditControl = CreateWindowA("edit", "", WS\_VISIBLE | WS\_CHILD | ES\_MULTILINE | WS\_VSCROLL, 5, 40, 470, 60, hWnd, NULL, NULL, NULL);

//CreateWindowA("button", "Calculate", WS\_VISIBLE | WS\_CHILD | ES\_CENTER, 130, 200, 100, 50, hWnd, (HMENU)OnCalculate, NULL, NULL);

/\*g\_hTextBox = CreateWindowA("edit", "EDIT",

WS\_CHILD | WS\_VISIBLE | WS\_VSCROLL | WS\_HSCROLL | ES\_MULTILINE | ES\_AUTOVSCROLL | ES\_AUTOHSCROLL,

5, 110, 470, 280, hWnd, NULL, NULL, NULL);\*/

}

void AuditRegistryChanges(HWND hwnd)

{

// Open the registry key to monitor changes

HKEY hKey;

DWORD result = RegOpenKeyEx(HKEY\_LOCAL\_MACHINE, L"Software\\Microsoft\\Windows\\CurrentVersion", 0, KEY\_NOTIFY | KEY\_READ, &hKey);

if (result != ERROR\_SUCCESS)

{

std::cout << "Error opening registry key" << std::endl;

return;

}

// Create an event for waiting for registry changes

HANDLE hEvent = CreateEvent(NULL, FALSE, FALSE, NULL);

if (hEvent == NULL)

{

std::cout << "Error creating event" << std::endl;

RegCloseKey(hKey);

return;

}

// Set the event to interrupt the waiting

result = RegNotifyChangeKeyValue(hKey, TRUE, REG\_NOTIFY\_CHANGE\_LAST\_SET, hEvent, TRUE);

if (result != ERROR\_SUCCESS)

{

std::cout << "Error setting event for interruption" << std::endl;

CloseHandle(hEvent);

RegCloseKey(hKey);

return;

}

// Registry change processing

while (true)

{

// Wait for registry changes

DWORD dwWait = WaitForSingleObject(hEvent, INFINITE);

if (dwWait != WAIT\_OBJECT\_0)

{

std::cout << "Error waiting for registry changes" << std::endl;

continue;

}

// Get information about the changes

DWORD valueType;

BYTE buffer[1024];

DWORD bufferSize = sizeof(buffer);

result = RegQueryValueEx(hKey, L"ProductName", NULL, &valueType, buffer, &bufferSize);

if (result == ERROR\_SUCCESS)

{

// Convert the buffer to a string

std::wstring productName(reinterpret\_cast<wchar\_t\*>(buffer), bufferSize / sizeof(wchar\_t));

// Output the change information

std::wcout << L"Registry change: " << productName << std::endl;

// Display the information in a message box

MessageBox(hwnd, productName.c\_str(), L"Registry Change", MB\_OK);

}

// Reset the event for the next change

result = RegNotifyChangeKeyValue(hKey, TRUE, REG\_NOTIFY\_CHANGE\_LAST\_SET, hEvent, TRUE);

if (result != ERROR\_SUCCESS)

{

std::cout << "Error resetting event for next change" << std::endl;

break;

}

}

// Close the registry key and free resources

RegCloseKey(hKey);

CloseHandle(hEvent);

}

// Функция для добавления информации в реестр

void WriteToRegistry()

{

while (true)

{

std::wstring info = L"Info";

HKEY hKey;

DWORD result = RegCreateKeyEx(HKEY\_CURRENT\_USER, REGISTRY\_KEY\_PATH, 0, NULL, REG\_OPTION\_NON\_VOLATILE, KEY\_WRITE, NULL, &hKey, NULL);

if (result == ERROR\_SUCCESS)

{

result = RegSetValueEx(hKey, REGISTRY\_VALUE\_NAME, 0, REG\_SZ, reinterpret\_cast<const BYTE\*>(info.c\_str()), (info.length() + 1) \* sizeof(wchar\_t));

RegCloseKey(hKey);

}

// Пауза в 10 секунд

std::this\_thread::sleep\_for(std::chrono::seconds(10));

}

}

Листинг 2 – Файл *SoftwareDefinitions.h*

#pragma once

#define OnMenuClicked1 1

#define OnMenuClicked2 2

#define OnMenuClicked3 3

#define OnExitSoftware 4

#define OnCalculate 5

#define IDC\_STATUS\_LABEL 101

// Путь к ключу реестра для записи

#define REGISTRY\_KEY\_PATH L"Software\\MyApp"

// Имя значения реестра для записи

#define REGISTRY\_VALUE\_NAME L"Info"

HWND hEditControl;

HWND hStaticControl;

HWND g\_hTextBox;

HWND hStatusLabel;

double cpuLoad;

double cpuusage;

LRESULT CALLBACK SoftwareMainProcedure(HWND hWnd, UINT msg, WPARAM wp, LPARAM lp);

WNDCLASS NewWindowClass(HBRUSH BGColor, HCURSOR Cursor, HINSTANCE hInst, HICON Icon, LPCWSTR Name, WNDPROC Procedure);

void MainWndAddMenus(HWND hwnd);

void MainWndAddWidgets(HWND hwnd);

void AuditRegistryChanges(HWND hwnd);

void WriteToRegistry();