*Министерство образования Республики Беларусь*

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

*Факультет информационных технологий и управления*

*Кафедра информационных технологий автоматизированных систем*

К защите допустить

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Шаплыко

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2015 г.

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

НА ТЕМУ

**РАБОТА С РЕЕСТРОМ**

Выполнил ст. гр. 220602 \_\_\_\_\_\_\_\_\_\_\_\_\_\_ /

Принял \_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Шаплыко В.В. /

МИНСК 2015

СОДЕРЖАНИЕ

[Введение 4](#_Toc437908987)

[1. Анализ предметной области 5](#_Toc437908988)

[1.1 Общие сведения 5](#_Toc437908989)

[2. Описание реализации 6](#_Toc437908990)

[2.1 Выбор и обоснование средств реализации 6](#_Toc437908991)

[2.2 Обзор доступных инструментальных средств 6](#_Toc437908992)

[2.3 Реализация программы 7](#_Toc437908993)

[3. Руководство пользователя 8](#_Toc437908994)

[Заключение 9](#_Toc437908995)

[Список использованных источников 10](#_Toc437908996)

[[5] Приемы программирования в С++ / А.Я. Архангельский. – Москва, 2010. -857 с. 10](#_Toc437908997)

[Приложение А (справочное) Программный код 11](#_Toc437908998)

# Введение

Система - это четко упорядоченная, структурированная совокупность двух или нескольких элементов (число их может быть бесконечным). Элементы находятся во взаимосвязи и подчиняются законам, установленным в данной системе. Система может быть составляющей другой, более объемной мощной системы, т.е. представлять собой отдельный элемент.

В данной работе мы будем ссылаться на отдельный, наиболее динамично развивающийся тип системы – информационной.

Информационная система представляет собой любую систему обработки информации. Информационная система в более узком смысле – совокупность аппаратно-программных средств, задействованных для решения некоторой прикладной задачи. Информационные системы предназначены для хранения, обработки, поиска, распространения, передачи и предоставления информации. Для создания и поддержания таких систем, соответственно, требуется отдельный вид деятельности человека – системное программирование.

Системное программирование (или программирование систем) - это вид программирования, который заключается в работе с системным программным обеспечением. Главным отличием системного программирования по сравнению с прикладным программированием является то, что прикладное программное обеспечение предназначено выпускать (создавать и обновлять) программы для пользователей (например, текстовые процессоры), тогда как системное программирование предназначено выпускать программы, обслуживающие аппаратное обеспечение (например, дефрагментация диска ) что обусловливает значительную зависимости такого типа ПО от аппаратной части.

Определение «системное» подчеркивает тот факт, что результаты этого вида программирования существенно меняют свойства и возможности вычислительной системы. В то же время бесспорным остаётся тот факт, что в определенной степени этот результат имеет место при применении любых программ, выполняемых в вычислительной системе. Подводя итог, можно утверждать, что системным следует называть подраздел программирования как вида инженерной деятельности, в котором программист использует специфические и часто уникальные свойства и возможности внутренних уровней вычислительной системы.

Целью данной работы является изучение функционирования и реализации работы с реестром.

# Анализ предметной области

## Общие сведения

Словарь «Microsoft Computer Dictionary» дает такое определение реестра: «Иерархически построенная, централизованная база данных в составе операционных систем Microsoft Windows 9x/NT/2000/XP/2003/Vista/7/8, содержащая сведения, которые используются операционной системой для работы с пользователями, программными продуктами и устройствами».

В Реестре хранятся данные, которые необходимы для правильного функционирования Windows. К ним относятся профили всех пользователей, сведения об установленном программном обеспечении и типах файлов, которые могут быть созданы каждой программой, информация о свойствах папок и значках приложений, а также установленном оборудовании и используемых портах. Системный Реестр заменяет собой большинство текстовых ini-файлов, которые использовались в Windows 3.x, а также файлы конфигурации MS-DOS (например, Autoexec.bat и Config.sys). Немного углубимся в историю и узнаем, что из себя представляет ini-файл.

**ini-файл** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Initialization file*) — это [файл конфигурации](https://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB_%D0%BA%D0%BE%D0%BD%D1%84%D0%B8%D0%B3%D1%83%D1%80%D0%B0%D1%86%D0%B8%D0%B8), который содержит данные настроек для [Microsoft Windows](https://ru.wikipedia.org/wiki/Microsoft_Windows" \o "Microsoft Windows), [Windows NT](https://ru.wikipedia.org/wiki/Windows_NT" \o "Windows NT) и некоторых приложений.

Появились с самых первых версий Windows. В версии Windows 1.01 это был только файл [WIN.INI](https://ru.wikipedia.org/w/index.php?title=WIN.INI&action=edit&redlink=1). В Windows 3.0 добавился файл SYSTEM.INI. А затем их количество начало расти быстро и бесконтрольно.

Не существует подробной официальной [спецификации](https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B5%D1%86%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F) формата. Начиная с [Windows 95](https://ru.wikipedia.org/wiki/Windows_95" \o "Windows 95), INI файлы считаются устаревшими и в качестве замены им [Microsoft](https://ru.wikipedia.org/wiki/Microsoft" \o "Microsoft) предлагает использовать [системный реестр](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B5%D1%81%D1%82%D1%80_Windows) ([Registry](http://msdn.microsoft.com/library/en-us/sysinfo/base/registry.asp)). Тем не менее INI файлы продолжают использоваться как приложениями других производителей, так и компонентами ОС от Microsoft. Например, файл boot.ini используется в [Windows NT4/2000/XP](https://ru.wikipedia.org/wiki/Windows_NT" \o "Windows NT) при загрузке для выбора из нескольких [ОС](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0).

Хотя INI файлы и приобрели популярность в [Windows](https://ru.wikipedia.org/wiki/Microsoft_Windows" \o "Microsoft Windows), они могут использоваться в любой ОС. Несложная структура этого формата позволяет легко обрабатывать их программно и имеет достаточно понятный вид для чтения и изменения человеком. Ниже предоставлены некоторые из причин отказа от ini-файлов в пользу реестра:

1. на некоторых платформах размер INI-файла не может превышать 64 Кбайт;
2. по мере эксплуатации системы количество записей в файлах win.ini и system.ini быстро увеличивается, что приводит к их чрезмерному «засорению» данными. К тому же размер этих файлов зачастую превышает 64 Кбайт;
3. при обращении к INI-файлу большого размера производительность Windows может существенно снизиться, так как для обнаружения конкретной записи операционная система использует линейный поиск. Таким образом, даже если ваша система поддерживает работу с большими INI-файлами, использование таких файлов не рекомендуется;
4. организовать защиту INI-файлов или управление этими файлами через сеть достаточно сложно;
5. в INI-файле можно хранить только текстовую информацию. Хранение данных других типов не поддерживается. Чтобы сохранить числовое значение, вы должны записать его в виде строки, а позже, при чтении, осуществить обратное преобразование.

Во время запуска операционной системы происходит до тысячи обращений к реестру, а во время работы на ПК в течение одного сеанса работы – до 10 тысяч! Отдельные компоненты реестра хранятся в оперативной памяти ПК в течение всего сеанса работы. Запись (считывание) информации в реестр (из реестра) происходит постоянно: например, если мы устанавливаем какую-нибудь программу, вся информация, необходимая для запуска и работы этой программы, записывается в реестр. Если мы устанавливаем новое устройство, в реестре будет отмечено, где находится его драйвер и т.д.

Если же мы запускаем какую-либо программу или устройство, то из реестра считывается вся необходимая для запуска программы (устройства) информация.

Системный реестр Windows – это иерархическая база данных. Информация хранится в ключах. Каждый ключ может содержать несколько подключей. Помимо подключей, каждый ключ может содержать также несколько значений. Такая структура сильно напоминает файловую систему. Можно рассматривать ключи как каталоги, а значения ключей – как файлы. Реестр поддерживался и в ранних версиях Windows, однако до последнего времени большинство программ его не использовало. Традиционно реестр использовался только приложениями OLE, а также старым диспетчером программ (Program Manager). В настоящее время реестр используется многими современными механизмами Windows (такими как, например, новая графическая оболочка и ActiveX).

На рисунке 1.1 представлена рабочее окно редактора реестра входящего в комплект семейства Windows.

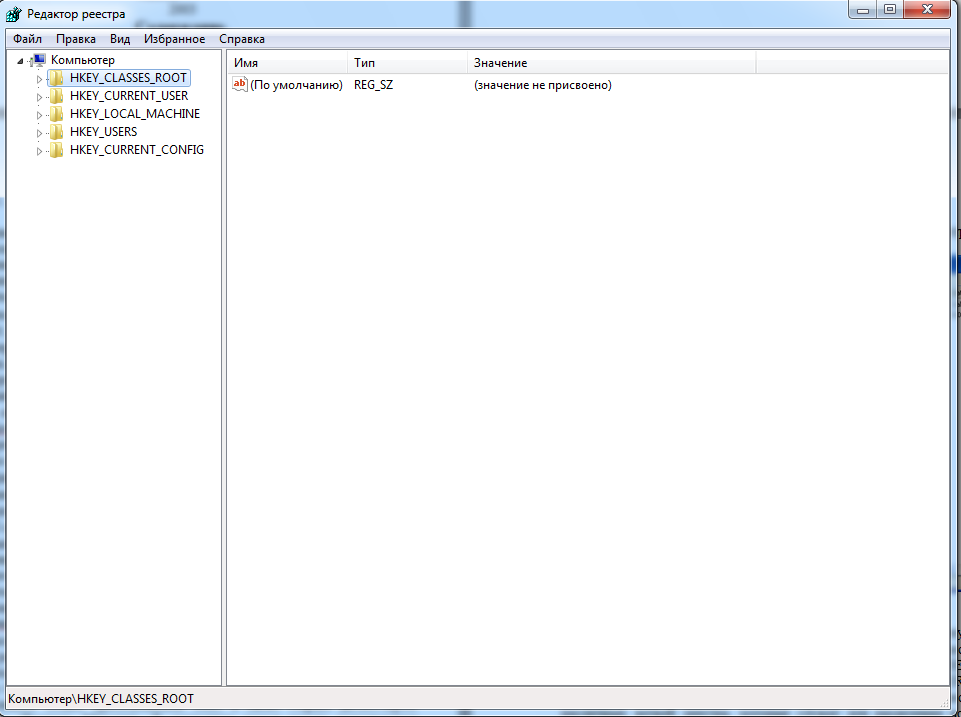


Рис.1.1 – Рабочее окно программы Regedit

Куст реестра – это группа разделов, подразделов и параметров реестра с набором вспомогательных файлов, содержащих резервные копии этих данных.

Реестр состоит из 5-ти ветвей:

1. HKEY\_CURRENT\_USER (HKCU) – данный раздел является корневым для данных конфигурации пользователя, вошедшего в систему в настоящий момент. Здесь хранятся папки пользователя, цвета экрана и параментры панели управления. Эти сведения сопоставлены с профилем пользователя.
2. HKEY\_USERS (HKU) – данный раздел содержит все активные загруженные профили пользователей компьютера.
3. HKEY\_LOCAL\_MACHINE (HKLM) – раздел содержит параметры конфигурации, относящиеся к данному компьютеру (для всех пользователей).
4. HKEY\_CLASSES\_ROOT (HKCR) – является подразделом HKEY\_LOCAL\_MACHINE\Software. Хранящиеся здесь сведения обеспечивают выполнение необходимой программы при открытии файла с использованием проводника.
5. HKEY\_CURRENT\_CONFIG (HKCG) – данный раздел содержит сведения о профиле оборудования, используемом локальным компьютером при запуске системы.

Дескриптор ключа (key handle) хранится в переменной типа HKEY. Чтобы получить дескриптор ключа HKEY, необходимо использовать функции RegOpenKeyEx или RegCreateKeyEx (системные вызовы, относящиеся к работе с реестром, перечислены в табл. 1.1). Первый аргумент этих функций имеет тип HKEY. Откуда его взять? К счастью, при вызове любой из этих функций в качестве первого аргумента можно указать константу, например имя одного из корневых ключей реестра (например, HKEY\_CLASSES\_ROOT).

Таблица 1.1. Некоторые системные вызовы для работы с реестром

|  |  |
| --- | --- |
| Функция | Описание |
| RegCloseKey | Закрыть открытый ключ реестра |
| RegConnectRegistry | Подключиться к удаленному реестру |
| RegCreateKeyEx | Создать новый ключ |
| RegDeleteKey | Удалить ключ |
| RegDeleteValue | Удалить значение |
| RegEnumKeyEx | Перейти к следующему подключу (каждый раз возвращает новый ключ) |
| RegEnumValue | Перейти к следующему значению (каждый раз возвращает новое  значение) |
| RegFlushKey | Внести значения ключа в реестр (чтобы убедиться в том, что все  изменения сохранены на диске) |
| RegLoadKey | Загрузить ключ из специального файла (см. также RegSaveKey) |
| RegOpenKeyEx | Открыть ключ |
| RegQuerylnfoKey | Запросить информацию о ключе |
| RegQueryValueEx | Прочитать значение |
| RegReplaceKey | Заменить ключ после перезапуска системы |
| RegSaveKey | Записать ключ в файл |
| RegSetKeySecurity | Установить разрешения на доступ к ключу |

После того как ключ открыт, можно прочитать значение (RegQuery ValueEx), изменить его (RegSetValueEx), удалить ключ (RegDeleteKey) или перебрать подключи этого ключа (RegEnumKeyEx). Вызов RegEnumValue позволяет перебрать все именованные значения данного ключа. После того как работа с ключом завершена, ключ необходимо закрыть при помощи вызова RegCloseKey. В табл. 1.1 упомянуты некоторые другие функции, предназначенные для работы с ключами реестра.

В Windows 2000 для защиты разных элементов реестра используются специальные механизмы безопасности. Каждая программа обладает индивидуальным набором разрешений на доступ к различным ключам. Если программа обладает необходимым уровнем доступа, она не сможет изменить значение ключа или получить доступ к ветви реестра. Работать с реестром можно удаленно, через сеть.

Не следует использовать функцию GetLastError для того, чтобы получить информацию об ошибке, которая произошла в результате обращения к одной из функций работы с реестром. Все эти функции возвращают код ошибки непосредственно. Функция GetLastError поможет узнать об ошибках, которые произошли в результате обращения к другим системным вызовам API ранее в программе.

Реестр способен содержать в себе данные многих экзотических типов (табл. 1.2). Однако на самом деле система никогда не обращает внимания на эти типы данных. Например, переменная типа REG\_EXPAND\_SZ – это строка, содержащая имена переменных окружения, которые необходимо заменить значениями этих переменных самостоятельно. Реестр не предоставляет доступ к значениям переменных окружения автоматически. Вместо этого надо получить необходимые значения собственными силами, например, при помощи функции ExpandEnvironmentStrings.

Таблица 1.2. Типы данных, хранящихся в реестре

|  |  |
| --- | --- |
| Тип | Описание |
| REG\_BINARY | Бинарные данные в любой форме |
| REG\_DWORD | 32-битное число (часто используется для хранения байта, слова или даже бита) |
| REG\_DWORD\_LITTLE\_ENDIAN | 32-битное число в формате аппаратных платформ Intel |
| REG\_DWORD\_BIG\_ENDIAN | 32-битное число в формате аппаратных платформ Motorola |
| REG\_EXPAND\_SZ | Строка, завершающаяся нулем, содержащая ссылки на переменные окружения (например: "%РАТН%") |
| REG\_LINK | Символическая ссылка в формате UNICODE |
| REG\_MULTI\_SZ | Массив строк, каждая из которых завершается нулем. В конце массива располагаются два нулевых символа |
| REG\_NONE | Неопределенный тип |
| REG\_RESOURCE\_LIST | Список ресурсов драйвера устройства |
| REG\_SZ | Строка, завершающаяся нулем |

**1.2 Постановка задачи**

Требуется разработать программу, который будет работать с системным реестром Windows. Программа будет уметь:

* Открывать ключ реестра
* Отслеживать какие-либо изменения в подразделе реестра
* Выводить соответствующее сообщение об изменениях

# Описание реализации

## Выбор и обоснование средств реализации

## Обзор доступных инструментальных средств

В настоящее время существует большой выбор средств разработки приложений на платформе операционной системы Windows. Ниже приведен обзорнаиболее популярных средств.

*BorlandDelphi*

Среда быстрой разработки Delphi – продукт компании Borland, основана на объектно-ориентированном языке программирования Pascal, В среде имеются удобные средства для создания оконных приложений. Также, в Delphi есть стандартные компоненты для работы с базами данных и веб-сервисами.

*MicrosoftVisualBasic*

VisualBasic от компании Microsoft представляет собой систему быстрой разработки приложений RAD (RapidApplicationDevelopment), похожую на среду Delphi. Скорее всего, это объясняется конкуренцией между компаниями Microsoft и BorlandInternational. Однако, не смотря на схожесть интерфейсов, разница в языках существенна. VisualBasic – объектно-ориентированный язык программирования, как правило встроенный во многие приложения MicrosoftOffice. К достоинствам можно отнести простоту создания не сложных приложений, а также возможность редактирования и создания компонент MicrosoftOffice.

*MicrosoftVisual C++*

Язык программирования Visual C++ из среды разработки VisualStudio компании Microsoft наиболее мощное средство разработки системных программ. Однако, разработка программ на этом языке - трудоемкий и сложный процесс.

*Java*

В языке Java используется технология объектно-ориентированного программирования, которая позволяет сократить общее время разработки и писать повторно используемый код. Java-приложения являются независимыми от платформы. Это достигается путем совмещения в языке свойств компилятора и интерпретатора. Платформонезависимость байт-кода обеспечивается наличием виртуальных java-машин для всех основных платформ. В комплект поставки Java входят стандартные классы, которые обладают достаточной функциональностью для быстрой разработки приложений. Развитые средства безопасности позволяют использовать Java для разработки

приложений, работающих в Интернете. Единственным недостатком Java является медленная скорость работы, обусловленная использованием ЛТ-компиляторов – цена кроссплатформенности.

*Borland C++ Builder*

Визуальная среда разработки Borland C++ Builder объединяет мощь индустриального стандарта C++ и компонентно-ориентированный подход к разработке программных систем, 6-я версия C++ Builder обладает развитыми средствами для разработки интернет-ориентированных бизнес\* приложений и всеми необходимыми технологиями разработки интегрированных решений на платформе Windows. Средства поддержки Windows 2000/XP позволяют визуально создавать серверные и клиентские приложения с использованием всех возможностей СОМ+.

Руководствуясь техническим заданием к программно-инструментальному комплексу, среда разработки приложений Borland C++ Builder было выбрана по следующим причинам:

В этой среде сочетается гибкость и мощь языка C++ и средства для быстрого построения многофункционального интерфейса;

В последней шестой версии C++ Builder предлагаются специализированные компоненты, облегчающие создание приложений работающих с Интернетом;

Данная среда является одним из самых популярных средств быстрой разработки приложений во всем мире. Компания Borland регулярно выпускает новые версии данной среды и предполагает ее дальнейшую поддержку.

Требования к аппаратной конфигурации ПК: компьютер на базе Intel (не младше Pentium, рекомендуется IntelPentiumШ 500 MHz); S мегабайт оперативной памяти (рекомендуется 32), около 2-3 мегабайт свободного пространства на жестком диске; модем или сетевая карта для выхода в сеть Интернет. Требования к программному обеспечению ОС MicrosoftWindows 95/98/ME/20QOttP: Microsoft Data Access Component (MDAC) версии 2.5 и выше. Дополнительно требуется подключение ПК к сети Интернет через модемное или выделенное соединение. Возможно взаимодействие с Интернет через прокси-сервер.

Для разработки приложения использовалась среда разработки C++ Builder 6, т. к. она позволяет быстро решить поставленную задачу без использования дополнительных инструментов[2].

## Реализация программы

# Руководство пользователя

# Заключение

# Список использованных источников

[1] Ревотюк М.П. [Электронный ресурс]. – «Электронный учебно-методический комплекс по дисциплине СИСТЕМНОЕ ПРОГРАММИРОВАНИЕ».

[2] MicrosoftDeveloperNetwork[Электронныйресурс].– Режим доступа: <http://msdn.microsoft.com/ru-ru/>.

[3] Википедия [Электронный ресурс]. – Режим доступа: <http://ru.wikipedia.org/>.

# Приложение А (справочное) Программный код

**Листинг программы**

**Win32Project2.cpp**

#include "stdafx.h"

#include "Win32Project2.h"

#define MAX\_LOADSTRING 100

HINSTANCE hInst; TCHAR szTitle[MAX\_LOADSTRING]; TCHAR szWindowClass[MAX\_LOADSTRING];

ATOM MyRegisterClass(HINSTANCE hInstance);

BOOL InitInstance(HINSTANCE, int);

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

INT\_PTR CALLBACK About(HWND, UINT, WPARAM, LPARAM);

int APIENTRY \_tWinMain(\_In\_ HINSTANCE hInstance,

\_In\_opt\_ HINSTANCE hPrevInstance,

\_In\_ LPTSTR lpCmdLine,

\_In\_ int nCmdShow)

{

UNREFERENCED\_PARAMETER(hPrevInstance);

UNREFERENCED\_PARAMETER(lpCmdLine);

MSG msg;

HACCEL hAccelTable;

LoadString(hInstance, IDS\_APP\_TITLE, szTitle, MAX\_LOADSTRING);

LoadString(hInstance, IDC\_WIN32PROJECT2, szWindowClass, MAX\_LOADSTRING);

MyRegisterClass(hInstance);

if (!InitInstance (hInstance, nCmdShow))

{

return FALSE;

}

hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_WIN32PROJECT2));

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

{

if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

**Продолжение приложения А**

return (int) msg.wParam;

}

ATOM MyRegisterClass(HINSTANCE hInstance)

{

WNDCLASSEX wcex;

wcex.cbSize = sizeof(WNDCLASSEX);

wcex.style = CS\_HREDRAW | CS\_VREDRAW;

wcex.lpfnWndProc = WndProc;

wcex.cbClsExtra = 0;

wcex.cbWndExtra = 0;

wcex.hInstance = hInstance;

wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_WIN32PROJECT2));

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

wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW+1);

wcex.lpszMenuName = MAKEINTRESOURCE(IDC\_WIN32PROJECT2);

wcex.lpszClassName = szWindowClass;

wcex.hIconSm = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI\_SMALL));

return RegisterClassEx(&wcex);

}

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)

{

HWND hWnd;

hInst = hInstance; // Сохранить дескриптор экземпляра в глобальной переменной

hWnd = CreateWindow(szWindowClass, szTitle, WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0, NULL, NULL, hInstance, NULL);

HWND hWnd\_button = CreateWindow(TEXT("button"), TEXT("Начать"), WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON,

10, 50, 80, 30, hWnd, (HMENU)100, hInstance, NULL);

if (!hWnd)

{

return FALSE;

}

ShowWindow(hWnd, nCmdShow);

UpdateWindow(hWnd);

return TRUE;

}

**Продолжение приложения А**

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)

{

int wmId, wmEvent;

PAINTSTRUCT ps;

HDC hdc;

DWORD dwFilter = REG\_NOTIFY\_CHANGE\_NAME |

REG\_NOTIFY\_CHANGE\_ATTRIBUTES |

REG\_NOTIFY\_CHANGE\_LAST\_SET |

REG\_NOTIFY\_CHANGE\_SECURITY;

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

HKEY hMainKey = HKEY\_CURRENT\_USER;

HKEY hKey;

LONG lErrorCode;

switch (message)

{

case WM\_COMMAND:

wmId = LOWORD(wParam);

wmEvent = HIWORD(wParam);

switch (wmId)

{

case 100:

RegOpenKeyEx(hMainKey, TEXT("Software\\Foxit Software"), 0, KEY\_NOTIFY, &hKey);

RegNotifyChangeKeyValue(hKey, TRUE, dwFilter, hEvent, TRUE);

if (WaitForSingleObject(hEvent, INFINITE) == WAIT\_FAILED){}

else

MessageBox(hWnd, TEXT("Изменения"), TEXT("Реестр изменен"), MB\_OK);

CloseHandle(hEvent);

lErrorCode = RegCloseKey(hKey);

break;

case IDM\_ABOUT:

DialogBox(hInst, MAKEINTRESOURCE(IDD\_ABOUTBOX), hWnd, About);

break;

case IDM\_EXIT:

DestroyWindow(hWnd);

break;

default:

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

}

break;

**Продолжение приложения А**

case WM\_CREATE:

SetWindowPos(hWnd, HWND\_TOP, 300, 300,300,300, SWP\_SHOWWINDOW);

break;

case WM\_PAINT:

hdc = BeginPaint(hWnd, &ps);

TextOut(hdc, 10, 20, TEXT("Ожидание изменения реестра"), 26);

EndPaint(hWnd, &ps);

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

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

}

return 0;

}

INT\_PTR CALLBACK About(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)

{

UNREFERENCED\_PARAMETER(lParam);

switch (message)

{

case WM\_INITDIALOG:

return (INT\_PTR)TRUE;

case WM\_COMMAND:

if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)

{

EndDialog(hDlg, LOWORD(wParam));

return (INT\_PTR)TRUE;

}

break;

}

return (INT\_PTR)FALSE;

}

}

**Resource.h**

#define IDS\_APP\_TITLE 103

#define IDR\_MAINFRAME 128

#define IDD\_WIN32PROJECT2\_DIALOG 102

#define IDD\_ABOUTBOX 103

#define IDM\_ABOUT 104

**Продолжение приложения А**

#define IDM\_EXIT 105

#define IDI\_WIN32PROJECT2 107

#define IDI\_SMALL 108

#define IDC\_WIN32PROJECT2 109

#define IDC\_MYICON 2

#ifndef IDC\_STATIC

#define IDC\_STATIC -1

#endif

#ifdef APSTUDIO\_INVOKED

#ifndef APSTUDIO\_READONLY\_SYMBOLS

#define \_APS\_NO\_MFC 130

#define \_APS\_NEXT\_RESOURCE\_VALUE 129

#define \_APS\_NEXT\_COMMAND\_VALUE 32771

#define \_APS\_NEXT\_CONTROL\_VALUE 1000

#define \_APS\_NEXT\_SYMED\_VALUE 110

#endif

#endif