МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра вычислительной техники

УТВЕРЖДАЮ
Проректор по учебной работе
О.Г. Локтионова
«16» 12 2017г.

РАБОТА С РЕЕСТРОМ WINDOWS

Методические указания к лабораторным и практическим занятиям
для студентов направления 09.03.01

Курск 2017

УДК 004.451.8

Составители: В.С. Панищев, Е.Ю. Емельянова

Рецензент
Кандидат технических наук, доцент *Халин Ю.А.*

Работа с реестром Windows: методические указания к лабораторным и практическим занятиям для студентов направления 09.03.01/ Юго-Зап. гос. ун-т; сост.; В.С. Панищев, Е.Ю. Емельянова. – Курск, 2017. - 23 с.: табл. 2.– Библиогр.: с. 23

Содержат сведения по вопросам назначения, структуры реестра Windows, функции Windows API для программной работы с реестром. Теоретические материалы рассмотрены на примере. Излагаются указания по подготовке и выполнению задания.

Методические указания соответствуют рабочей программе дисциплины «Операционные системы».

Предназначены для студентов направления 09.03.01 очной и заочной форм обучения.

Текст печатается в авторской редакции

Подписано в печать 15.12.17. Формат 60*84 1/16.
Усл. печ. л. 1,34. Уч.-изд. л. 1,21. Тираж 50 экз. Заказ 423 Бесплатно.
Юго-Западный государственный университет.
305040 Курск, ул. 50 лет Октября, 94.

Оглавление

1 ЦЕЛЬ РАБОТЫ

Изучить назначение и устройство реестра Windows; научиться программно работать с реестром, используя функции Windows API или библиотеки языков высокого уровня.

2 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

2.1 Назначение реестра

Данные о конфигурации системы и текущие настройки программ в Windows 3.x были рассредоточены в разных ini-файлах. Такая рассредоточенность не позволяла операционной системе контролировать установленные программы, а программам – знать друг о друге. Кроме того, размер ini-файлов ограничивался 64К.

В связи с внедрением, начиная с Windows 3.x, технологии OLE было решено собрать все сведения о серверах OLE в единой регистрационной базе данных, от которой и произошел реестр. Итак, **реестр** (***Registry***) – это централизованная база данных, которая содержит конфигурационную информацию Windows. В ней хранятся данные об аппаратных средствах компьютера, о пользователях системы, их правах и настройках, информация об установленных программах, что позволяет легко организовать операции OLE и DDE между приложениями.

2.2 Файлы реестра

Физически вся информация реестра разбита на несколько файлов. Реестры Windows 9x и NT частично различаются. В Windows 95/98 реестр содержится в двух файлах SYSTEM.DAT и USER.DAT, находящися в каталоге Windows. В Windows Me был добавлен еще один файл CLASSES.DAT. В Windows XP реестр хранится во многих файлах. Основная часть хранится в файлах *sam, security, software, system, default* (все файлы без расширения).

Файлы реестра считываются в оперативную память при загрузке Windows. Все изменения, вносимые в реестр, сначала сохраняются в оперативной памяти, а затем периодически сбрасываются на диск (также перезапись реестра делается перед завершением работы). При каждом удачном запуске операционная система сохраняет копию реестра в CAB-файле, который записывается в скрытый каталог

SYSBCKUP каталога Windows. По умолчанию, хранятся последние пять копий.

Для редактирования реестра вручную в комплект Windows входит программа *regedit.exe* или *regedit32.exe*, обладающая расширенными возможностями.

2.3 Резервное копирование и восстановление реестра

При работе с реестром надо соблюдать осторожность. Удаление каких-либо важных данных случайно или по незнанию может привести к краху операционной системы. Тогда спасти положение может только восстановление последней работоспособной копии.

Способ 1. Предварительно сохранить на диске файлы SYSTEM.DAT и USER.DAT. Они находятся в каталоге, куда была установлена операционная система и имеют атрибуты "только для чтения" и "скрытый". Если реестр окажется поврежденным, придется загрузить систему с дискеты (DOS), и скопировать сохраненные файлы SYSTEM.DAT и USER.DAT в папку Windows. Перед копированием следует снять атрибуты «скрытый» и «системный», а затем – восстановить, после чего перезагрузить систему обычным образом.

Способ 2. С помощью утилиты *scanreg.exe*. Надо перезагрузиться в DOS и создать резервную копию реестра командой `SCANREG/BACKUP`. Чтобы восстановить реестр, надо перезагрузиться в DOS и выполнить команду `SCANREG/RESTORE`. Появится список доступных резервных копий реестра, отсортированных по времени их создания. После выбора нужной копии данные будут благополучно восстановлены.

Способ 3. Экспортирование раздела с помощью редактора *regedit*. Если вы планируете менять какой-то раздел или ветвь реестра, можно предварительно сохранить его в REG–файле. Для этого в *regedit* выделите нужный раздел и выберите пункт меню *Реестр/Экспорт файла реестра*. Для восстановления раздела из REG-файла достаточно в проводнике Windows дважды на нем щелкнуть.

Примечание: в Windows XP 1-й и 2-й способы не работают.

2.4 Внутренняя организация реестра

Реестр имеет иерархическую древовидную структуру, сходную с системой каталогов диска. Он состоит из двух типов объектов:

разделов (или ключей (*Keys*)), и параметров (*Values*). **Разделы** содержат параметры и вложенные **подразделы** (*Sybkeys*).

Каждый раздел и параметр имеет имя. Имя может состоять из любых печатных символов, включая пробелы, символ подчеркивания и спецсимволы. Не допускается использование в именах обратной наклонной черты (\), поскольку она воспринимается как разделитель при обозначении путей к разделам реестра. Заглавные и строчные буквы в именах различаются только при отображении имен. При поиске разделов и параметров регистр не учитывается. Например, при отображении в браузере имени раздела GraphicsControl буквы G и C будут заглавными. Однако при поиске имена GraphicsControl, graphicscontrol и GRAPHICSCONTROL будут считаться одинаковыми. Имя должно быть уникальным в пределах «родительского» раздела.

Параметры состоят из трех частей: имя, тип и значение.

В Win95/98 допустимы три типа параметров: REG_SZ, REG_BINARY и REG_DWORD. Win2000/XP использует 11 типов (таблица 1). Значение параметра в Win95/98 не может превышать 64 Кбайта, а в Win2000/XP до 1 Мб.

Любой раздел имеет один **параметр по умолчанию**, типа REG_SZ. У него нет имени (имя – пустая строка; в *regedit* выводится как [Default]). Этот параметр может не содержать никаких данных.

Разделы, находящиеся в «корне» реестра, со всеми своими подразделами образуют **ветви** (*Hive keys*). В зависимости от версии Windows определено 5 или 6 ветвей (таблица 2). Из них две ветви являются основными: *HKEY_LOCAL_MACHINE* (подгружается из файла SYSTEM.DAT) и ветвь *HKEY_USERS* (из файла USER.DAT). Остальные ветви представляют собой ссылки на подразделы основных ветвей (это значит, что внесенная в них информация автоматически отображается в соответствующих подразделах основных ветвей). Ссылки сделаны для быстроты обращения к часто востребуемым разделам реестра, и для совместимости с ранними версиями Windows.

Реестр содержит тысячи подразделов и параметров, и только часть их них стандартизована. Параметры с предопределенными именами и значениями воспринимаются Windows и влияют на ее работу. Прикладные программы могут добавлять в реестр собственные подразделы с произвольными параметрами для хранения своих текущих настроек.

Таблица 1 – **Типы данных реестра**

Тип параметра	Способ хранения	Назначение
REG_BINARY	двоичные данные	Массив байт (длина произвольная).
REG_DWORD	32-разряд. число	Целое число со знаком.
REG_SZ	строка	Строка текста произвольной длины (один байт/символ). Завершается символом с кодом 0.
REG_EXPAND_SZ	строка	«Расширяемая» строка. То же, что REG_SZ. Может содержать ссылки на переменные окружения (в строке имена переменных записываются между символами %, например %PATH%)
REG_MULTI_SZ	массив строк	Массив строк. Каждая строка завершается нулем, в конце массива располагаются два нулевых символа
REG_NONE	не определён	Зашифрованные пользователем данные
REG_DWORD_ _LITTLE_ENDIAN	32-разряд. число	Число с порядком байтов, принятом в процессорах Intel (то же, что и REG_DWORD): младшие разряды хранятся по младшим адресам
REG_DWORD_ _BIG_ENDIAN	32-разряд. число	Число с обратным порядком байтов (как в процессорах Motorola) (младшие разряды по старшим адресам)
REG_LINK	строка	Путь к файлу, в кодировке Unicode (2 байта/символ)
REG_RESOURCE_LIST	строка	Список ресурсов устройств
REG_FULL_RESOURCE_ _DESCRIPTOR	строка	Идентификатор ресурса устройства
REG_RESOURCE_ _REQUIREMENTS_LIST	строка	Идентификатор ресурса устройства

При удалении параметров и разделов соответствующие объекты помечаются как удаленные, а физического удаления данных из файлов реестра не происходит. Это связано с тем, что после физического удаления необходима перестройка бинарного дерева, что связано с большими накладными расходами. Поэтому файлы реестра при частой установке и удалении программ «распухают».

Таблица 2 – **Корневые разделы (ветви) реестра**

Имя ветви	Описание
HKEY_LOCAL_MACHINE	Информация об аппаратных средствах, возможных конфигурациях оборудования и установленном программном обеспечении
HKEY_CURRENT_CONFIG	Ссылка на один из подразделов HKEY_LOCAL_MACHINE\Config соответствующий текущей конфигурации
HKEY_CLASSES_ROOT	Ссылка на HKEY_LOCAL_MACHNE\Software\Classes . Зарегистрированные классы OLE и известные типы файлов. Для классов OLE указываются пути к серверам, для файлов – программы, обрабатывающие их по умолчанию.
HKEY_USERS	Содержит профили всех зарегистрированных пользователей. Каждый профиль сохраняется в отдельном подразделе. Подраздел с именем .DEFAULT используется для создания профиля по умолчанию.
HKEY_CURRENT_USER	Ссылка на один из подразделов HKEY_USERS, соответствующий текущему пользователю
HKEY_DYN_DATA	Только Win95/98. Временный раздел для хранения динамических данных, уничтожается после завершения работы.
HKEY_PERFOMANCE_DATA	Только WinNT. Временный раздел для хранения динамических данных, уничтожается после завершения работы

2.5 Функции Windows API для работы с реестром

API (*Application Program Interface,* Интерфейс прикладного программирования) – множество функций операционной системы, которые можно вызывать из прикладных программ, и с помощью которых можно управлять операционной системой, устройствами, другими программами, и пр. То есть все средства операционной системы, реализованные в служебных программах и утилитах, доступны с помощью функций API. Функции Windows API можно вызывать из ЯВУ, подключив в C++ Builder разные включаемые файлы `#include <win….h>`, а в Delphi – модуль `USES Windows`. Документация по функциями Windows API поставляется с языками программирования, обычно в разделе Help/Windows SDK. Там можно по имени найти любую функцию API и посмотреть, в какой библиотеке она находится.

В Windows XP для работы с реестром предназначены 26 функций (таблица 3). Функции, имена которых оканчиваются на "Ex", используются в 32-разрядных версиях Windows, их двойники с именами, не оканчивающимися на "Ex", оставлены для совместимости с 16-рязрядными версиями Windows 3.x, и пр выполнении работы их использовать не следует.

Таблица 3 – **Функции WinAPI для работы с реестром**

Название	Описание
RegCreateKeyEx, RegCreateKey	Создать новый или открыть существующий раздел
RegOpenKeyEx, RegOpenKey	Открыть существующий раздел
RegCloseKey	Закрыть раздел
RegSetValueEx, RegSetValue	Добавить новый параметр или изменить значение существующего параметра
RegDeleteKey	Удалить раздел
RegDeleteValue	Удалить параметр
RegEnumKeyEx, RegEnumKey	Получить информацию о вложенных подразделах (имя, класс)
RegEnumValue	Получить информацию о параметрах раздела (имя, тип, значение)

Название	Описание
RegQueryInfoKey	Получить характеристики раздела (количество подразделов и параметров, время последнего обращения и др.)
RegQueryValueEx, RegQueryValue	Получить тип и значение параметра с заданными именем
RegQueryMultipleValues	Получить тип и значения параметров из заданного списка
RegFlushKey	Немедленно сохранить содержимое реестра из оперативной памяти в файлы SYSTEM.DAT, USER.DAT
RegSaveKey	Сохранить раздел реестра в указанный файл
RegLoadKey	В разделе HKEY_LOCAL_MACHINE или HKEY_USERS создать новый подраздел и загрузить в него данные из указанного файла
RegUnLoadKey	Удалить указанный подраздел ветви HKEY_LOCAL_MACHINE или HKEY_USERS
RegRestoreKey	Заменить содержимое раздела данными из файла
RegReplaceKey	Сохранить раздел в файл резервной копии, и подгрузить в него данные из другого файла
RegGetKeySecurity	Получить дескриптор безопасности раздела
RegSetKeySecurity	Установить дескриптор безопасности раздела
RegConnectRegistry	Подключиться по сети к реестру удаленного компьютера
RegNotifyChangeKeyValue	Связать событие (event) с изменением состояния раздела: закрытие / добавление / удаление / изменение параметров и вложенных подразделов.

В Delphi и Builder определен класс **TRegistry** для работы с реестром (описание есть в справочной системе ЯВУ). Все его методы, в конечном счете, вызывают перечисленные функции API.

2.6 Порядок работы с реестром средствами WinAPI

В чем-то похож на работу с файлами: сначала выбранный раздел реестра необходимо открыть (или создать и открыть), получив после открытия *дескриптор* (*Handle*) раздела. Далее в открытом разделе делаются любые операции (создание, удаление, опрос подразделов и параметров, сохранение/восстановление разделов и пр.). После окончания работы раздел нужно закрыть.
Корневые разделы открыты всегда. Их дескрипторы – предопределенные константы:

HKEY_CLASSES_ROOT
HKEY_CURRENT_USER
HKEY_CURRENT_CONFIG
HKEY_LOCAL_MACHINE
HKEY_USERS
HKEY_PERFORMANCE_DATA (только Windows NT)
HKEY_DYN_DATA (только Windows 95/98)

Функции WinAPI (таблица 3) при успешном выполнении возвращают **0** (константа ERROR_SUCCESS); или ≠0 код ошибки, который можно преобразовать в строковое сообщение функцией *FormatMessage* с параметром FORMAT_MESSAGE_FROM_SYSTEM. Функцию *GetLastError* использовать не следует, так как она об ошибках реестра не сообщает.

Все функции WinAPI работают со строками в формате *null-terminated* (строки, оканчивающиеся кодом 0). Это нужно учитывать при выборе строковых типов данных в Builder и Delphi.

Открытие раздела реестра

```
LONG RegOpenKeyEx(
  HKEY hKey,          // дескриптор открытого раздела верхнего уровня
  LPCTSTR lpszSubKey,// адрес строки с именем раздела
  DWORD dwReserved,   // резерв, равно 0
  REGSAM samDesired,  // маска доступа (см. ниже)
  PHKEY phkResult     // указатель на переменную типа HKEY, куда
                      // передается дескриптор открытого раздела
);
```

Маска доступа (параметр *samDesired*) определяет тип операций, которые планируется проводить над разделом, и получается как (+ или OR) комбинация флагов:

KEY_CREATE_LINK	создать символическую ссылку;
KEY_CREATE_SUB_KEY	создать подразделы;
KEY_ENUMERATE_SUB_KEYS	просмотреть подразделы;
KEY_QUERY_VALUE	просмотреть параметры;
KEY_EXECUTE	прочитать параметр по умолчанию;
KEY_NOTIFY	изменить параметр по умолчанию;
KEY_SET_VALUE	изменить значения параметров.

Также определены стандартных комбинации флагов:

KEY_READ = KEY_QUERY_VALUE + KEY_ENUMERATE_SUB_KEYS + + KEY_NOTIFY;
KEY_WRITE = KEY_SET_VALUE + KEY_CREATE_SUB_KEY;
KEY_ALL_ACCESS = комбинация всех флагов, кроме KEY_EXECUTE.

Открывать разделы реестра можно в несколько этапов. К примеру, для доступа к разделу *HKEY_CURRENT_USER\Software\Far* сначала можно открыть подраздел *Software* (*HKEY_CURRENT_USER* открыт всегда), а потом подраздел *Far*. Другой способ – сразу открыть нужный подраздел, передав в `lpszSubKey` полный путь "*HKEY_CURRENT_USER\Software\Far*". В Windows поэтапное открытие может вызывать ошибки, связанные с доступом. Поэтому лучше открывать подразделы, сразу указывая полные пути к ним.

Создание нового раздела

Функция **RegCreateKeyEx** создает новый раздел, или открывает существующий, если раздел с таким именем уже есть:

```
LONG RegCreateKeyEx(
  HKEY hKey,              // дескриптор открытого раздела верхнего уровня
  LPCTSTR lpszSubKey,     // адрес строки с именем раздела
  DWORD dwReserved,       // резерв, равно 0
  LPTSTR lpClass,         // адрес строки с именем класса раздела (можно
  DWORD dwOptions,        // тип раздела (см. ниже)                  NULL)
  REGSAM samDesired,      // маска доступа (см. выше)
  LPSECURITY_ATTRIBUTES lpSecurityAttributes,
```

```
                    // адрес структуры с атрибутами защиты (можно NULL)
  PHKEY phkResult       // указатель на переменную типа HKEY, куда
                    // передается дескриптор созданного / открытого раздела
  LPDWORD lpdwDisposition  // адрес приемника кода результата
);
```

Параметр *dwOptions* задает тип раздела:

REG_OPTION_VOLATILE – временный (несохраняемый) раздел (существует до первой перезагрузки системы);

REG_OPTION_NON_VOLATILE – обычный, сохраняемый, раздел.

В переменную *lpdwDisposition* функция помещает код, показывающий результат выполнения: REG_CREATED_NEW_KEY – был создан новый раздел; REG_OPENED_EXISTING_KEY – был открыт существующий раздел.

Закрытие раздела

```
LONG RegCloseKey( HKEY hKey );
```

Все изменения, сделанные в этом разделе реестра, остаются пока в оперативной памяти, и будут зафиксированы на диске при очередной перезаписи файлов реестра. Чтобы немедленно записать изменения в файлы (например, в следующую секунду может внезапно отключиться питание ПК и ваши изменения не «доживут» до перезаписи) вызывается функция

```
LONG RegFlushKey( HKEY  hKey );
```

Запись в файлы занимает несколько секунд и требует дополнительных системных ресурсов на хеширование, поэтому функцию *RegFlushKey* рекомендуется применять в крайних случаях. После использования *RegFlushKey* ключ надо закрыть функцией *RegCloseKey*.

Добавление, изменение, удаление подразделов и параметров

Если предполагается выполнять эти операции над параметрами, раздел должен быть открыт с флагом KEY_SET_VALUE. Если предполагается создавать новые подразделы, то раздел должен быть открыт с флагом доступа KEY_CREATE_SUB_KEY. Если и то, и другое, то с флагом KEY_WRITE (KEY_SET_VALUE + KEY_CREATE_SUB_ KEY).

Добавляет подразделы функция *RegCreateKeyEx* (см. выше).

Удаление вложенных подразделов выполняет функция **RegDeleteKey.** В Windows 95/98 она удаляет раздел со всеми вложенными подразделами и параметрами. В Windows NT (2000, XP) удаляет, только если раздел пуст.

```
LONG RegDeleteKey(
  HKEY   hKey,          // дескриптор открытого раздела верхнего уровня
  LPCTSTR   lpszSubKey  // адрес строки с именем удаляемого подраздела.
);
```

Если строка *lpszSubKey* пустая, или её адрес указан как NULL, удаляется раздел, заданный дескриптором *hKey* (т.е. текущий).

Функция **RegSetValueEx** предназначена для создания новых параметров и изменения значения существующих:

```
LONG RegSetValueEx(
  HKEY hKey,              // дескриптор открытого раздела
  LPCTSTR lpValueName,    // адрес строки с именем параметра
  DWORD Reserved,         // резерв, равно 0
  DWORD dwType,  // тип параметра (константа из первого столбца
                 // таблицы 1)
  CONST BYTE *lpData,     // адрес буфера с новым значением параметра
  DWORD cbData            // размер буфера в байтах
);
```

Изменить значение параметра "по умолчанию" можно, передав функции в качестве имени параметра пустую строку, или указав *lpValueName* = NULL.

Функция **RegDeleteValue** удаляет параметры из указанного раздела реестра:

```
LONG RegDeleteValue(
    HKEY   hKey,            // дескриптор открытого раздела
    LPTSTR  lpszValue       // адрес строки с именем параметра
);
```

Если строка *lpszValue* пустая или ее адрес равен NULL, удаляется значение параметра по умолчанию (сам параметр при этом сохраняется, но его значение становится "не определено").

Опрос подразделов и параметров

Если имена подразделов и параметров известны, для опроса используются функции *RegQueryInfoKey, RegQueryValueEx, RegQueryMultipleValues.*

Если имена не известны, опрос производится по порядковому номеру подраздела или параметра функциями *RegEnumKeyEx и RegEnumValue.*

Функция **RegQueryValueEx** ищет параметр с заданным именем, и, если поиск успешен, возвращает тип и значение параметра:

```
LONG RegQueryValueEx(
  HKEY hKey,              // дескриптор открытого раздела
  LPTSTR lpValueName,     // адрес строки с именем параметра
  LPDWORD lpReserved,     // резерв, равен NULL
  LPDWORD lpType,         // адрес приемника кода типа параметра
  LPBYTE lpData,          // адрес буфера для приема значения
                          // параметра
  LPDWORD lpcbData        // адрес переменной с max длиной буфера
);
```

Перед вызовом в *lpcbData* помещается адрес переменной типа INT, хранящей максимальную длину буфера *lpData*. После вызова функции *RegQueryValueEx* значение опрашиваемого параметра возвращается в буфере *lpData*, а фактическая длина данных – в переменной *lpcbData*. Если буфер, определенный *lpData*, слишком мал, функция возвращает код ошибки ERROR_MORE_DATA, и в *lpcbData* возвращается требуемый размер буфера. Тип параметра передается через параметр *lpType* и принимает значение константы из первого столбца таблицы 1. Для опроса параметра по умолчанию в качестве имени параметра передается пустая строка или устанавливается *lpValueName*=NULL.

Если функция *RegQueryValuEx* вызывается только для того, чтобы проверить, существует параметр с заданным именем, или нет, а значение его получать не требуется, то функцию следует вызывать с *lpData* = *lpcbData* = NULL.

Для опроса именованных параметров раздел должен быть открыт с доступом KEY_QUERY_VALUE, для опроса параметра по умолчанию – с доступом KEY_EXECUTE.

Для получения информации о разделе используется функция

```
LONG RegQueryInfoKey(
 HKEY hKey,                          // дескриптор открытого раздела
 LPTSTR lpClass,                  // адрес строки-приемника имени класса
 LPDWORD lpcbClass,          // длина строки с именем класса
 LPDWORD lpReserved,       // резерв, равно NULL
 LPDWORD lpcSubKeys,       // приемник количества подразделов
 LPDWORD lpcbMaxSubKeyLen,  // приемник максимальной длины
                                  // имени подраздела
 LPDWORD lpcbMaxClassLen,    // приемник макс. длины названия
                                  //класса
 LPDWORD lpcValues,          // приемник количества параметров текущего
                                  // подраздела
 LPDWORD lpcbMaxValueNameLen, // приемник максимальной длины
                                  // имени параметра
 LPDWORD lpcbMaxValueLen, // приемник макс. длины значения
                                  //параметра
 LPDWORD lpcbSecurityDescriptor,// адрес структуры для
                                  // размещения дескриптора безопасности
 PFILETIME lpftLastWriteTime // время последнего изменения
                                  // раздела
);
```

Функция возвращает информацию о количестве вложенных подразделов и параметров, максимальной длине имён подразделов и параметров. Если какая-то информация не нужна, вместо адресов передается NULL. Чтобы функция работала, раздел должен быть открыт с доступом KEY_QUERY_VALUE или KEY_READ.

Эту функцию можно использовать, чтобы проверить, существует ли раздел с указанным именем; и для получения числа вложенных параметров и подразделов, чтобы потом в цикле организовать их опрос с помощью следующих двух функций – *RegEnumValue* и *RegEnumKeyEx*.

Когда имена подразделов и параметров неизвестны, информацию об этих объектах можно получить с помощью функций перебора **RegEnumValue** и **RegEnumKeyEx**. Доступ к объекту происходит по порядковому номеру *dwIndex*, отсчет ведется с 0.

На практике для перебора параметров и подразделов сначала получают информацию о разделе функцией *RegQueryInfoKey*. и в соответствии с ней резервируют буферы для приема имен и значений нужных объектов. Затем организуется цикл перебора, и в цикле вызываются функции **RegEnum…** для опроса подразделов или параметров.

Опрос параметра по порядковому номеру:

```
LONG RegEnumValue(
  HKEY hKey,              // дескриптор открытого раздела
  DWORD dwIndex,          // порядковый номер параметра (отсчет с 0)
  LPTSTR lpValueName,     // адрес буфера-приемника имени параметра
  LPDWORD lpcbValueName,  // размер буфера-приемника имени
  LPDWORD lpReserved,     // резерв, равно NULL
  LPDWORD lpType,         // адрес приемника кода типа параметра
  LPBYTE lpData,          // адрес буфера-приемника значения
                          // параметра
  LPDWORD lpcbData        // размер буфера-приемника значения
                          // параметра
);
```

Перед вызовом функции в *lpValueName* и *lpData* заносятся адреса буферов приема имени и значения параметра соответственно, в *lpcbValueName* и *lpcbData* – размер буферов. Если размер буферов недостаточен, функция возвращает код ошибки ERROR_MORE_DATA. Если тип или значение параметра не требуется, в параметрах *lpType*, *lpData* и *lpcbData* соответственно, передается NULL. Код типа, возвращаемый в *lpType*, принимает значение константы из таблицы 1.

Перебор подразделов выполняет функция:

```
LONG RegEnumKeyEx(
  HKEY hKey,              // дескриптор открытого раздела
  DWORD dwIndex,          // порядковый номер подраздела (отсчет с 0)
  LPTSTR lpName,          // адрес буфера-приемника имени подраздела
  LPDWORD lpcbName,       // размер буфера-приемника имени
  LPDWORD lpReserved,     // резерв, равно NULL
  LPTSTR lpClass,         // адрес буфера-приемника строки с именем
                          //класса
  LPDWORD lpcbClass,      // размер буфера-приемника строки с именем
                          // класса
  PFILETIME lpftLastWriteTime // приемник времени последнего
);                        // изменения раздела
```

Значение параметров при вызове аналогично *RegEnumValue*. Если какой-либо из параметров не нужен, вместо адреса передается NULL.

3 ПРИМЕР

Задание: консольное приложение. Программе через командную строку передается путь к разделу реестра. Если раздел существует, программа выводит на экран список параметров этого раздела: имя, тип. Если раздела нет – сообщение об ошибке.
Примечание: если путь к разделу содержит пробелы, в командной строке его следует писать в кавычках, например

```
regexample.exe "hkey_current_user\control panel\colors"
```

Листинг программы (на Delphi 7.0):

```
program RegExample;
{$APPTYPE CONSOLE}
uses SysUtils, Windows;
var
  HiveKey : integer;          // дескриптор корневой ветви
  HiveKeyName :string;        // имя ветви
  RegistryPath:string;        // путь к разделу реестра
  RKey : HKEY;                // дескриптор открытого раздела
  RegParamCount : integer;    // число параметров в разделе
  ParamNameBuffer : string;   // статич. буфер для имени пар-ра
  RegParamName : string;      // имя параметра для вывода
  RegParamType : integer;     // код типа параметра
  i,c : cardinal;             // рабочие переменные
```

```
begin
// проверяем параметры командной строки
if ParamCount=0 // нет параметров ?
 then writeln('Ошибка: не указан путь к разделу реестра')
 else begin
   //разделяем параметр на имя ветви и путь:
// имя ветви - символы с 1-го до первого символа '\'
// имя раздела - от '\' до конца строки
HiveKeyName:=copy(ParamStr(1),1,Pos('\',ParamStr(1))-1 );
RegistryPath:=copy(ParamStr(1),Pos('\',ParamStr(1))+1,
                   length(ParamStr(1)));
// идентифицируем ветвь раздела
HiveKeyName:= UpperCase( HiveKeyName );
if HiveKeyName='HKEY_CLASSES_ROOT'
   then HiveKey := HKEY_CLASSES_ROOT
else if HiveKeyName='HKEY_CURRENT_USER'
   then HiveKey:=HKEY_CURRENT_USER
else if HiveKeyName='HKEY_LOCAL_MACHINE'
   then HiveKey:=HKEY_LOCAL_MACHINE
else if HiveKeyName='HKEY_USERS' then HiveKey:=HKEY_USERS
else if HiveKeyName='HKEY_CURRENT_CONFIG'
   then HiveKey:=HKEY_CURRENT_CONFIG
else begin
   writeln('Ошибка: неверное имя корневого раздела');
   exit; // выйти из программы
end;
//пытаемся открыть для чтения указанный раздел
if RegOpenKeyEx(HiveKey,PChar(RegistryPath),0,KEY_READ,RKey)<>0
  then begin
    writeln('Ошибка: не могу открыть раздел '+RegistryPath );
    exit; // выйти из программы
  end
  else begin
    // раздел открыт - узнаём количество параметров
    RegQueryInfoKey( RKey,NIL,NIL,NIL,NIL,NIL,NIL,
                     @RegParamCount,NIL,NIL,NIL,NIL);
    //резервируем буфер на 255 символов
    ParamNameBuffer := StringOfChar(' ',255);
    // цикл опроса параметров (кроме параметра по умолчанию)
    for i:=0 to (RegParamCount-1) do begin
      c:=255; // максимальная длина буфера
      RegEnumValue(RKey, i, PChar(ParamNameBuffer),
                   c,NIL,@RegParamType,NIL,NIL);
      // после вызова в c - длина имени параметра
      RegParamName := copy( ParamNameBuffer, 1, c);
      write( RegParamName+':'); //выводим имя и тип параметра
      case RegParamType of
```

```
        REG_SZ     : writeln('REG_SZ');
        REG_BINARY: writeln('REG_BINARY');
        REG_DWORD : writeln('REG_DWORD');
      end; {case} end; {for}
 RegCloseKey( RKey );      //закрываем реестр
end; {if}
end;
end.
```

4 ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Получить вариант задания.
2. Изучить теоретическую часть методических указаний.
3. Разработать алгоритм решения задачи. Составить программу. Для работы с реестром можно использовать по выбору: функции Windows API или класс TRegistry.
4. Показать программу преподавателю. Оформить отчет.

5 СОДЕРЖАНИЕ ОТЧЕТА

1. Вариант задания.
2. Пояснения к способу решения задачи или алгоритм программы.
3. Листинг программы.
4. Тестовый пример и результаты работы программы.

6 ВАРИАНТЫ ЗАДАНИЙ

1. Пользователь вводит путь к разделу реестра. Программа выводит значение параметра по умолчанию данного раздела.
2. Создать раздел HKEY_CURRENT_USER\Lab2\ и присвоить параметру *Default* введенное пользователем значение.
3. Программа меняет в указанном подразделе реестра значение параметра по умолчанию Пользователь вводит путь к разделу и новое значение параметра *Default*.
4. Дополнить программу из лабораторной работы №1 с динамическим подключением DLL следующими действиями: при первом запуске программа регистрирует в реестре свою DLL, создавая параметр, содержащий с путь к библиотеке. При последующих запусках она ищет в реестре этот параметр, и считывает оттуда путь к библиотеке. Таким образом, EXE-файл можно запускать из любого каталога.

5. Программа меняет цвет активного окна, перезаписывая соответствующий параметр реестра (HKEY_CURRENT_USER \ Control Panel \ Colors).
6. Пользователь вводит путь к разделу реестра и имя параметра. Если параметр существует, вывести его значение. Если не существует, создать его, запросив у пользователя тип и значение.
7. Пользователь указывает путь к разделу реестра и тип параметров (REG_SZ, REG_DWORD и т.д.). Программа выводит имена и значения параметров заданного типа. Для строк типа REG_EXPAND_SZ заменять переменные окружения (заключенные между символами %) на их фактические значения.
8. Оконное приложение. Пользователь выбирает раздел реестра и вводит имя текстового файла отчета. Программа записывает в файл отчета сначала список подразделов, а следом – список параметров этого радела, включая значение параметра по умолчанию.
9. Пользователь задает путь к подразделу реестра. Программа ищет в указанном подразделе все параметры типа REG_SZ и REG_EXPAND_SZ, значения которых удовлетворяют маске. Маска составляется по способу, принятому для поиска файлов в Windows.
10. Пользователь вводит путь к разделу реестра. Вывести на экран все параметры типа REG_SZ в этом разделе и его подразделах.
11. Написать программу, которая ищет вхождения заданной подстроки в значения параметров типа REG_SZ. Перед поиском пользователь вводит путь к разделу реестра. Поиск вести в указанном разделе и его подразделах.
12. Консольное приложение. В командной строке передается путь к разделу реестра и «маска» имён параметров вида "*s**" (через *s* обозначена произвольная последовательность символов). Вывести на экран все имена подразделов и параметров, имена которых удовлетворяют маске поиска.
13. Консольное приложение. В файл отчета. имя которого передается в командной строке, записываются зарегистрированные в реестре расширения файлов и программы, запускаемые при открытии этих файлов по команде *open*.
14. Зарегистрировать в реестре новое расширение файла, и указать, чтобы при открытии таких файлов автоматически запускалась ваша программа.
15. Пользователь вводит расширение файла. Вывести все зарегистрированные расширения файлов, которые обрабатываются той же программой, что и заданное расширение.
16. Вывести зарегистрированные расширения файлов и программы, запускаемые по умолчанию при открытии этих файлов. Предусмотреть

возможность создания, удаления, изменения связи расширений файлов с зарегистрированными программами.

17. Пользователь вводит маску имени параметра или подраздела. Программа ищет во всем реестре имена подразделов и параметров, удовлетворяющих маске, и выводит результат. Маска может иметь вид: α, α^*, $^*\alpha$, $\alpha^*\beta$. Через α и β обозначены подстроки значащих символов; * является служебным символом, означающим любую подстроку, включая пустую.
18. Программа выполняет две операции: 1) сохранение в файле полного содержимого указанного подраздела реестра (сохраняются имена подразделов и параметры, в том числе содержимое вложенных подразелов); 2) чтение из файла содержимого подраздела, сравнение его с текущим содержимым подраздела реестра, и вывод разницы на экран.
19. Консольное приложение. Программе через командную строку передается имя текстового файла, который состоит из строк вида:
    путь_и_имя_параметра : тип : значение
    Если путь к параметру существует, программа создает в этом подразделе новый параметр. Программа должна выводит диагностические сообщения при ошибках: 1) указан несуществующий путь; 2) недостаточно прав на создание параметров в этом подразделе.
20. Пользователь задает пути к двум разделам реестра. Сравнить содержимое (с точностью до параметров, и содержимого подразделов) и вывести результат: одинаковы разделы или нет.
21. Разработать "редактор реестра" с интерфейсом, подобным *regedit32.exe*. Отличается тем, что на каждый подраздел реестра выводятся права доступа, предоставляемые текущему пользователю. Например, подраздел, редактирование которого разрешено, изображается одним цветом (или значком), подраздел, который можно только просматривать – другим, а раздел, любой доступ к которому запрещен, третьим.
22. "Командный процессор" для работы с реестром. Консольное приложение. Пользователю выводится приглашение, после которого он набирает команды работы с реестром (вроде командной строки DOS). Команды могут быть следующими:
    EXIT – закончить работу с программой;
    OPEN путь_к_разделу_реестра – открыть указанный раздел реестра (сделать его текущим) (по аналогии с DOS-овской команды *CD*);
    SHOW путь_к_разделу_реестра – вывести имена подразделов, имена и значения параметров указанного раздела реестра, SHOW без параметров выводит содержимое текущего раздела (по аналогии с DOS-овской команды *DIR*);
    NEW Имя_параметра:Тип=Значение – создать новый параметр в текущем разделе реестра. Если имя параметра и тип не указаны

(команда имеет вид *NEW =Значение*), присваивается новое значение параметра по умолчанию.

Программа должна сообщать об ошибках в формате команд и выводить сообщения, если команду выполнить невозможно (например, не существует указанного раздела, нарушаются права доступа и пр.)

СПИСОК ЛИТЕРАТУРЫ

1. Пахмурин Д.О. Операционные системы ЭВМ [Электронный ресурс]: Учебное пособие. – Томск: Томский государственный университет систем управоления и радиоэлектроники, 2013. – 254с. Режим доступа: https://biblioclub.ru/index.php?page=author_red&id=175413
2. Тидроу, Р. Управление реестром Windows 95. – СПб.: BHV, 1996 – 280 с.
3. Воллес, Н. Реестр Windows 2000: специальный справочник. – СПб.: Питер, 2001. – 384 с.
4. Вильямс, А. Системное программиров ание в Windows 2000 для профессионалов [Текст] / А. Вильямс. – СПб.: Питер, 2001. – 624 с.
5. Рихтер, Д. Windows для профессионалов: создание эффективных Win32-приложений с учетом специфики 64-разрядной версии Windows [Текст] /Д. Рихтер. – СПб.: Питер: Русск. редакция, 2001. – 752 с.