Редькин Ю.В., Бузенков И.И., Чернышова Е.М. О сохранении настроек VBA-программ информационно-вычислительных систем. // Проблемы информатики в образовании, управлении, экономике и технике: Сб. статей XIX Междунар. научно-техн. конф. – Пенза: ПДЗ, 2019. – С. 207-213.

УДК 004.4
ББК 32.973.26-018

О СОХРАНЕНИИ НАСТРОЕК VBA-ПРОГРАММ ИНФОРМАЦИОННО-ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

Ю.В. Редькин, И.И. Бузенков, Е.М. Чернышева

ABOUT WAYS TO SAVE SETTINGS OF VBA-PROGRAM OF INFORMATION-COMPUTING SYSTEMS

Yu.V. Redkin, I.I. Buzenkov, E.M. Chernysheva

Аннотация. Классифицированы способы сохранения настроек прикладных VBA-программ для информационно-вычислительных систем. Указаны их достоинства, недостатки и основные области применения.

Ключевые слова: информационно-вычислительная система, локальный компьютер, регистр, прикладное программное обеспечение, файл данных, база данных.

Abstract. The methods for saving the settings of applied VBA programs are classified for information-computing systems. Their advantages, disadvantages and main areas of application are indicated.

Keywords: computer information system, local computer, register, application software, data file, database.

Язык *Visual Basic for Applications* (*VBA*), встроенный в *MS Office*, не только является простым средством для изучения ООП, но и с успехом может использоваться для создания достаточно сложных приложений, взаимодействующих как с аппаратными средствами информационно-вычислительных систем (ИВС) [1], так и их сетевыми ресурсами (дисками, принтерами и т.д.) [2].

Однако разработка сложных приложений предполагает наличие большого числа настроек, таких как настройки пользовательского интерфейса, которые требуется сохранять при закрытии приложения. Например, это может быть текстовое окно, хранящее имя выбранной директории; в него можно ввести и другой путь, но при повторном запуске программы имя директории примет исходное состояние, что не всегда удобно. Сохранение настроек особенно важно для сетевых приложений, для которых обязательно запоминание: доступных сетевых дисков и принтеров, расположение баз данных, пути для чтения и сохранения данных по умолчанию и иных путей к сетевым ресурсам.

Существуют различные способы сохранения настроек с целью их последующего восстановления при новом запуске программы. Простейшим из них является использование самого документа *MS Office*, например, таблицы *MS Excel*. Для хранения параметров можно отвести часть таблицы, но есть риск, что сохраненные настройки не будут видны из-за других данных.

Универсальным способом хранения настроек программы является использование *INI*-файлов, которые хранят информацию о параметрах на локальном компьютере. Для работы с *INI*-файлами удобно использовать вызовы *Win API*. Так, для доступа к строкам *INI*-файлов можно применять *API*-функции *WritePrivateProfileString*() и *GetPrivateProfileString*() из библиотеки *kernel32.dll*. Пример простой программы записи параметров в *INI*-файл и чтения их из него с использованием этих функций представлен на рис. 1. В ней для работы с *API*-функциями в раздел декларации модуля добавлены их описания, которые дают ссылку на файл *kernel32.dll* и определяют порядок использования параметров. Само обращение к *INI*-файлу выполняется простым вызовом этих функций и передачей им предварительно инициализированных параметров.

```
Private Declare Function GetPrivateProfileString Lib "kernel32" Alias _
"GetPrivateProfileStringA" (ByVal lpApplicationName As String, ByVal _
lpKeyName As Any, ByVal lpDefault As String, ByVal lpReturnedString As _
String, ByVal nSize As Long, ByVal lpFileName As String) As Long
Private Declare Function WritePrivateProfileString Lib "kernel32" Alias _
"WritePrivateProfileStringA" (ByVal lpApplicationName As String, _
ByVal lpKeyName As Any, ByVal lpString As Any, ByVal lpFileName As String) As Long

Sub WrGet_IniFile()
Dim n As Long: Dim s As String: s$ = Space$(256)
'Запись параметра в файл
n = WritePrivateProfileString("Parameters", "Path", "c:\tmp", "c:\tmp\myini.ini")
'Чтение параметра из файла
n = GetPrivateProfileString("Parameters", "Path", "", s$, 256, "c:\tmp\myini.ini")
'Отображение считанного параметра
If (n > 0) Then MsgBox "Path=" & s$
End Sub
```

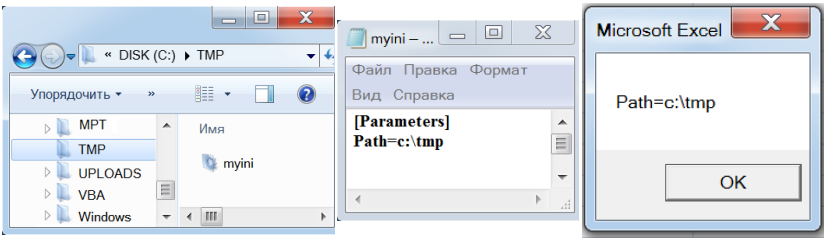
Рис. 1. Программа записи и чтения INI-файла с помощью API-функций

Вызов *API*-функции *WritePrivateProfileString*() записывает строку в *INI*-файл. Первый параметр функции *WritePrivateProfileString*(), имеющий значение "*Parameters*", указывает секцию *INI*-файла для записи данных. Второй параметр (“*Path*”) – это имя ключа в секции, а третий параметр ("*c:\tmp*") – значение ключа, которое надо записать в *INI*-файл. Четвертый параметр – это имя *INI*-файла; по умолчанию *INI*-файл находится в директории *Windows*, но можно сохранить его в ином месте, указав, например, "*c:\tmp\myini.ini*". Заметим, что можно скрыть сохраненные настройки, используя иное расширение файла.

Вызов *API*-функции *GetPrivateProfileString*() считывает сохраненный в *INI*-файле путь в строковую переменную s$ длиной 256 байт, созданную ранее командой *Space*$. Функция *GetPrivateProfileString*() работает аналогично *WritePrivateProfileString*(), но имеет больше параметров. Первый, второй и последний параметры те же, что и в функции записи, и передают ей имя секции, ключа и имя файла, из которого будет производиться чтение.

Третий параметр содержит значение по умолчанию, если ключ не найден. Четвертый параметр содержит имя строковой переменной, в которой будет сохранен результат, а пятый параметр указывает ее максимальную длину.

Результат работы программы *WrGet_IniFile*() представлен на рис. 2: а – файл *myinin.ini* в папке *c:\tmp*; б – содержание файла *myinin.ini* в программе «Блокнот»; в – результат чтения параметра *Path* из *INI*-файла *VBA*-программой.



а) записанный файл б) содержание файла в) результат чтения

Рис. 2. Результат работы программы записи INI-файла с помощью API-функций

Указанная пара *API*-функций обеспечивает удобный и безопасный доступ к ключам секций *INI*-файлов: если *INI*-файл не существует, то он будет создан; если ключа не существует, то он тоже будет создан. Как следствие, данный способ рекомендуется использовать в простых программах для сохранения информации о настройках, введенных пользователем.

В настоящее время *Microsoft* больше не использует *INI*-файлы и для записи такого рода информации рекомендуется применять ключи в реестре – специальной базе данных о компьютере, установленных аппаратных и программных средствах, данных о пользователях и т. д. Реестр имеет структуру дерева, в верхней части которого имеются пять основных ключей. Для хранения данных о текущем пользователе предназначен ключ *HKEY_CURRENT_USER*, который может иметь подключи, структурированные по уровням. Для доступа к ключам реестра из *VBA*-программы используются функции *WinAPI* [3]. С помощью этих функций можно: прочитать содержимое ключей реестра; создать новые ключи в реестре; прочитать и изменить значения параметров под ключами.

Функция *API* позволяет сохранять данные в реестре даже небольшой *VBA*-программе. При этом понадобится создать свой ключ, выбрав для его хранения любой из основных ключей. Для этих целей используется функция *RegCreateKey*(), с помощью которой можно сформировать структуру из ключей, как это показано в программе (рис. 2), на примере создания структуры ключей *SOFTWARE/VBA/Win32 API/Demo* под ключом *HKEY_CURRENT_USER*.

```
Declare Function RegCreateKey Lib "advapi32.dll" Alias "RegCreateKeyA" _
(ByVal hKey As Long, ByVal lpSubKey As String, phkResult As Long) As Long

Public Sub CreateRegistryKeyDemo()
Dim lReturn As Long: Dim sSubKey As String
Dim hSubKeyHandle As Long: sSubKey = "SOFTWARE\VBA\Win32 API\Demo"
lReturn = RegCreateKey(HKEY_CURRENT_USER, sSubKey, hSubKeyHandle)
MsgBox lReturn
End Sub
```

Рис. 3. Программа создания ключа с помощью API-функций

Для установки значений параметров реестра по умолчанию используется функция *RegSetValue*(), а для задания величин именованных параметров – функция *RegSetValueEx*(). В программе (рис. 4) показано использование этой функции для установки значений именованных параметров ключа, созданного предыдущей программой.

```
Declare Function RegOpenKey Lib "advapi32.dll" Alias "RegOpenKeyA" _
(ByVal hKey As Long, ByVal lpSubKey As String, phkResult As Long) As Long
Declare Function RegSetValueEx Lib "advapi32.dll" Alias "RegSetValueExA" _
(ByVal hKey As Long, ByVal lpValueName As String, ByVal Reserved As Long, _
ByVal dwType As Long, lpData As Any, ByVal cbData As Long) As Long

Public Sub SetRegistryValue()
Dim lReturn As Long: Dim hSubKeyHandle As Long
Dim sSubKeyName As String: Dim sValueName As String: Dim sValue As String
sSubKeyName = "SOFTWARE\VBA\Win32 API\Demo": Dim lValueSize As Integer
'Открытие ключа и получение его дескриптора
lReturn = RegOpenKey(HKEY_CURRENT_USER, sSubKeyName, hSubKeyHandle)
If lReturn <> 0 Then Exit Sub
'Установка первого значения
sValueName = "Path": sValue = "C:\tmp": lValueSize = Len(sValue)
lReturn = RegSetValueEx(hSubKeyHandle, sValueName, 0, REG_SZ, sValue, lValueSize)
'Установка второго значения
sValueName = "Copy": sValue = "Yes": lValueSize = Len(sValue)
lReturn = RegSetValueEx(hSubKeyHandle, sValueName, 0, REG_SZ, sValue, lValueSize)
End Sub
```

Рис. 4. Программа модификации ключа с помощью API-функций

Перед установкой значений с помощью функции *RegOpenKey*() требуется открыть ключ, под которым находится требуемый параметр. Имя параметра и устанавливаемое значение передаются в функцию *RegSetValueEx*(); тип устанавливаемого значения (*REG_SZ*) задает четвертый аргумент функции, а переменная *lValueSize* задает длину присваиваемого строкового значения.

Результат работы программы показан на рис. 5. В левом окне представлена структура ключей локального компьютера с раскрытым ключом *HKEY_CURRENT_USER*, а в правом – сохраненные записи в созданном ранее ключе *Software/VBA/Win32 API/Demo*.

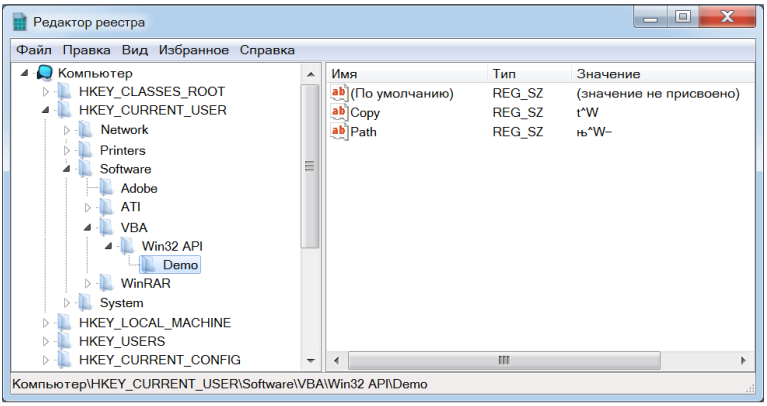


Рис. 5. Результат редактирования реестра с использованием API-функций

Таким образом, самым простым способом сохранения настроек прикладной *VBA*-программы является использование базы данных самого приложения, в которое встроен код программы. Например, в случае *MS Excel* – это ячейки таблицы. Однако этот способ не всегда удобен, поскольку настройки могут оказаться скрытыми основной таблицей приложения. Альтернативным простым и безопасным способом сохранения настроек является применение *INI*-файлов, позволяющих с помощью несложных средств организовать доступ к требуемым настройкам даже в небольшой программе. Универсальным является сохранение текущих настроек *VBA*-программы в реестре локального компьютера под одним из основных ключей, например, *HKEY_CURRENT_USER*. Однако этот способ более сложен в реализации и считается менее безопасным, чем *INI*-файлы, поэтому его можно рекомендовать для сложных приложений, например, взаимодействующих с аппаратными и сетевыми ресурсами ИВС.

Библиографический список

1. Бернд Г., Каинка Б. Измерение, управление и регулирование с помощью макросов VBA в Word и Excel. СПб.: КОРОНА-ВЕК, 2014. 256 с.

2. Редькин Ю.В., Бузенков И.И., Чернышева Е.М. Использование VBA для обеспечения доступа к ресурсам информационных систем // Информационные ресурсы и системы в экономике, науке и образовании: сборник статей VIII Международной научно-практической конференции. Пенза: ПДЗ, 2018. С. 114-119.

3. Джонсон М. Харт. Системное программирование в среде Windows, 3-е изд. / пер с англ. М.: Вильямс, 2005. 592 с.

Редькин
Юрий Викторович
Государственный морской
университет имени
адмирала Ф.Ф. Ушакова,
г. Новороссийск, Россия

Redkin Yu.V.
Admiral Ushakov Maritime
State University,
Novorossiysk, Russia

Бузенков
Игорь Иванович
Государственный морской
университет имени
адмирала Ф.Ф. Ушакова,
г. Новороссийск, Россия
E-mail: igor.buzenkov@mail.ru

Buzenkov I.I.
Admiral Ushakov Maritime
State University,
Novorossiysk, Russia

Чернышева Елена Михайловна
Государственный морской
университет имени
адмирала Ф.Ф. Ушакова,
г. Новороссийск, Россия

Chernysheva E.M.
Admiral Ushakov Maritime
State University,
Novorossiysk, Russia