Редькин Ю.В., Бузенков И.И., Чернышева Е.М. Организация низкоуровневого доступа к файлам данных информационно-вычислительных систем. // Проблемы информатики в образовании, управлении, экономике и технике: Сб. статей XVIII Междунар. научно-техн. конф. – Пенза: ПДЗ, 2018. – С. 168-174.

УДК 004.4
ББК 32.973.26-018

ОРГАНИЗАЦИЯ НИЗКОУРОВНЕВОГО ДОСТУПА К ФАЙЛАМ ДАННЫХ ИНФОРМАЦИОННО-ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

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

ORGANIZATION OF LOW-LEVEL ACCESS TO DATA FILES OF COMPUTER INFORMATION SYSTEMS

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

Аннотация. Рассмотрены вопросы создания прикладного программного обеспечения для информационно-вычислительных систем. Основное внимание уделено организации эффективного низкоуровневого доступа к файлам базы данных с помощью средств языка программирования VBA.

Ключевые слова: информационно-вычислительная система, программное обеспечение, низкоуровневый доступ, файл данных, база данных.

Abstract. The questions of creation of applied software for information-computing systems are considered. Particular attention is paid to the organization of effective low-level access to database files using the VBA programming language tools.

Keywords: computer information system, software, low-level access, data file, database.

Основная сложность в создании современных информационно-вычислительных систем (ИВС) заключается в разработке специализированного программного обеспечения, способного осуществлять непрерывную обработку огромного количества информации в соответствии с назначением данного программного обеспечения. Универсальная среда языка *Visual Basic for Applications* (*VBA*), уже предустановленная на большинстве персональных компьютеров, позволяет упростить разработку прикладных программ, поскольку включает в себя большую часть инструментов, необходимых для работы с аппаратными средствами ИВС [1]. Однако, вследствие малой популярности языка *VBA* на профессиональном рынке, его потенциал так и остается еще не раскрытым и, соответственно, не востребованным [2].

Особенно наглядно преимущества *VBA* проявляются при разработке систем сбора и обработки данных различного назначения. Программное обеспечение такой системы строится по модульному принципу: процессы сбора данных от датчиков, их запоминания в файлах, чтения данных из файлов, обработки и визуализации результатов реализуются различными частями (модулями) главной (основной) *VBA*-программы, являющейся приложением *MS Excel* [3].

Одной из сложностей, возникающей при разработке модулей программного обеспечения системы сбора и обработки данных является организация работы с данными, размещаемыми на диске для хранения и последующего чтения и обработки [4]. Проблема заключается в том, что в основе *VBA* лежит язык *Basic*, разработанный тогда, когда еще не существовало объектно-ориентированных средств разработки. Поэтому он поддерживает низкоуровневые средства ввода-вывода в дисковые файлы. Такой подход обладает определенными достоинствами. Так, все операторы и функции, необходимые для работы с файлами, встроены в *VBA* и не требуют внешних библиотек. В отличие от объектно-ориентированного подхода, здесь невозможно работать со свойствами файла и методами, но зато достигается более полный контроль над тем, как данные организуются в файле, и над тем, какие данные читаются и записываются.

К сожалению, в современной литературе по языку *VBA* вопрос использования низкоуровневых функций доступа к файлам изложен недостаточно полно, авторы обычно ссылаются на более ранние источники, например, [5]. Именно поэтому представляет интерес рассмотреть более детально процедуру обмена данными с файлами, особенно важную при разработке программного обеспечения систем сбора и обработки данных, работающих в реальном масштабе времени.

Для работы с содержимым файла данных прежде всего необходимо его открыть. Делается это с помощью оператора *Open*, синтаксис которого имеет вид:

Open "fName" For Mode As #FileNum,

где *fName* (имя файла) – полное имя файла, включая путь к нему с указанием логического диска, каталога и подкаталогов; *Mode* (режим) – ключ, определяющий то, каким образом планируется работать с открываемым файлом (*Input* – для чтения, *Output* – для записи, *Append* – для добавления данных в файл, *Random* – произвольный); *FileNum* (номер файла) – уникальный номер открываемого файла (целое число в диапазоне от 1 до 511).

После того как работа с файлом завершена, его обязательно следует закрыть, поскольку закрытие файла гарантирует, что все изменения, внесенные в файл, действительно будут записаны на диск, а ресурсы, занимаемые файлом, освободятся и станут доступными для других приложений. Для этого используется оператор *Close*, синтаксис которого имеет вид:

Close #FileNum,

где *FileNum* – номер открытого файла.

После открытия файла в соответствии с режимом доступа, определенным при его открытии, с данными, хранящимися в нем, можно выполнять определенные действия. Для этого в *VBA* существует множество операторов чтения и записи файлов [5].

Так, для работы с текстовыми файлами рекомендуется использовать операторы *Input* и *Line Input* (для последовательного чтения), *Print* и *Write*

(для последовательной записи). Однако для организации произвольного доступа к двоичным файлам данных удобнее использовать пару функций *Put* (запись) и *Get* (чтение), синтаксис которых имеет вид:

Put #FileNum, RecNum, usrData

– записывает переменную *usrData* в файле *FileNum* с произвольным доступом (*Random*) в запись *RecNum*);

Get *#FileNum, RecNum,* usrData

– читает определенные пользователем данные, хранимые в записи *RecNumr* файла *FileNum* с произвольным доступом (*Random*) в переменную *usrData.*

Рассмотрим возможности операторов *Get* и *Put* на примере организации хранения записей о показаниях аналоговых датчиков системы сбора и обработки данных. Для этого введем пользовательский тип данных *Unit*, служащий для хранения показаний датчика, даты и времени получения показаний и описания датчика:

```
Type Unit                    ' Описание структуры записи
RecNum As Integer            ' Номер записи
RecDate As Date              ' Дата записи
UnitName As String * 22      ' Наименование устройства
UnitValue As Byte            ' Показания устройства
UnitScale As Integer         ' Масштаб устройства
End Type
```

Отметим, что при описании пользовательского типа данных для организации записей используются строковые переменные фиксированной длины. Это необходимо, чтобы избежать ошибок при обработке записей. Так, для типа *Unit* размер записи составляет 43 байта. Этот размер обязательно необходимо указывать при открытии файла данных (для чтения, записи или модификации). Если запись окажется переменной длины, то возможно возникновение ошибок. В качестве примера в Листингах 1 и 2 приведены две функции *PutRecord* и *GetRecord*, предназначенные для размещения записей (в заданном формате *Unit*) в файл *FileName* и чтения из него. В каждой из них используется один и тот же оператор *Open*:

Open FileName For Random As #hFile Len = Len(*FlashUnit*).

Этим оператором файл *FileName* открывается в режиме произвольного доступа, при этом длина записи *FlashUnit* передается ему с помощью функции *Len*.

Листинг 1. Получение показаний датчика и их запись в файл

```
Sub PutRecord(RecNumber As Integer, FlashUnit As Unit)
Dim hFile As Integer  'Описываем дескриптор файла
hFile = FreeFile      'Получаем дескриптор файла для записи
```

```
Open FileName For Random As #hFile Len = Len(FlashUnit)  'Открываем файл
FlashUnit.RecNum = RecNumbers          'Запоминаем номер записи
FlashUnit.UnitValue = AIN(1)       'Считываем показания датчика
FlashUnit.RecDate = Now()          'Считываем текущую дату и время
Put #hFile, RecNumber, FlashUnit 'Делаем запись RecNumber в файл
Close #hFile                       'Закрываем файл после записи
End Sub
```

Листинг 2. Чтение показаний датчика из файла

```
Sub GetRecord(RecNumber As Integer, FlashUnit As Unit)
Dim hFile As Integer  'Описываем дескриптор файла
hFile = FreeFile   'Получаем дескриптор файла для чтения
Open FileName For Random As #hFile Len = Len(FlashUnit)    'Открываем файл
Get #hFile, RecNumber, FlashUnit 'Читаем запись RecNumber из файла
Close #hFile                       'Закрываем файл после чтения
End Sub
```

Обе приведенные функции (*PutRecord* и *GetRecord*) обеспечивают произвольный доступ к записям *RecNumber* файла *FileName*, не просматривая содержание других записей. Если запись найдена, то она считывается (при выполнении операции чтения) или заменяется новой (при выполнении операции записи). Если запрашиваемой записи не существует, то она создается (при выполнении операции записи). Такой подход позволяет считывать из файла и записывать в него любую информацию, основываясь лишь на номере записи. Это главная причина высокой эффективности рассматриваемого низкоуровневого режима доступа к данным.

В качестве примера на рисунке представлен результат использования функций *PutRecord* и *GetRecord* в модуле чтения/записи программного обеспечения системы сбора и обработки данных.

	A	B	C	D	E
1	Номер записи	Дата	Время	Название датчика	Показание датчика
2	1	29.09.2018	22:50:33	Датчик температуры N1	43,12
3	2	29.09.2018	22:50:34	Датчик температуры N1	41,96
4	3	29.09.2018	22:50:35	Датчик температуры N1	40,80
5					

Результат опроса датчика системы сбора и обработки данных

В данном случае модуль *VBA*, работающий как приложение *MS Excel*, обеспечивает циклический опрос одного датчика температуры и последовательную запись получаемых данных в файл. После окончания заданной серии опросов датчика производится чтение записей из файла данных, их окончательная обработка и отображение на листе программы *MS Excel*.

Как видно из рисунка, использование низкоуровневых функций (*PutRecord* и *GetRecord*) позволяет организовать эффективный и быстрый до-

ступ к записям данных в реальном времени, что затруднительно обеспечить стандартными средствами работы с файлами базы данных. В то же время описанный способ доступа к записям имеет существенные ограничения – так, он не позволяет организовать полноценную поддержку настоящей базы данных. Однако, используя структуры данных, можно создавать довольно сложные базы данных, обеспечивающие быстрый поиск нужной информации и оперативную запись в них новых или модифицированных данных, поступающих в систему в реальном масштабе времени (например, от датчиков).

Такой подход к организации доступа к файлам данных позволяет обеспечить разработку программного обеспечения различных ИВС, решающих достаточно сложные задачи, связанные со сбором, сохранением, обработкой и визуализацией результатов в реальном времени, с помощью стандартных средств языка *VBA*. Это позволяет существенно снизить сложность разработки программного обеспечения ИВС и, как следствие, сократить стоимость и сроки разработки системы и упростить ее последующее сопровождение и модернизацию.

Библиографический список

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

2. Уокенбах Джон. Excel 2013. Профессиональное программирование на VBA / пер. с англ. М.: ИД Вильямс, 2014. 960с.: ил.

3. Редькин Ю.В., Бузенков И.И. Технология создания программ для систем сбора данных на основе персонального компьютера и универсального программного обеспечения // Проблемы информатики в образовании, управлении, экономике и технике: сборник статей XVII Международной научно-техн. конф. Пенза: ПДЗ, 2017. С. 145-149.

4. 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-практ. конф. Пенза: ПДЗ, 2018. С. 114-119.

5. Кетков Ю.Л., Кетков А.Ю. Практика программирования: Visual Basic, C++ Builder, Delphi. СПб.: БХВ-Петербург, 2002. 464с.

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

УДК 681.5

ВОЗМОЖНОСТИ СЕРВИС-ОРИЕНТИРОВАННОГО ВЗАИМОДЕЙСТВИЯ ДАТЧИКОВ

И.И. Смирнов, П.С. Чернов

THE CAPABILITIES OF A SERVICE-ORIENTED INTERFACE FOR SMART SENSORS

I.I. Smirnov, P.S. Chernov

Аннотация. На примере спецификаций SWE рассмотрены преимущества и недостатки сервис-ориентированного информационного взаимодействия датчиков в сети. Тенденция к переходу на использование RESTful-сервисов жестко привязывает методы обмена данными к интернет-протоколам. С одной стороны, это упрощает разработку и отладку, с другой – накладывает определенные ограничения.

Ключевые слова: интеллектуальный датчик, цифровой интерфейс, сервис-ориентированная архитектура.

Abstract. We discuss the advantages and disadvantages of a service-oriented interaction in the sensornetwork on the exampleof SWE specifications. The transition to the RESTful services limits data exchange methods to Internet protocols. On the one hand, it simplifies development and debugging, on the other – imposes certain restrictions.

Keywords: smartsensors, digital interface, service-oriented architecture.

В современных интеллектуальных датчиках и актюаторах передача сигнала, несущего информацию об измеряемой величине, и служебные данные, осуществляется посредством цифрового интерфейса с использованием того или иного коммуникационного протокола или стандарта промышленных сетей (ProfiBus, DeviceNet, Interbus, Fieldbus, CANbus, LIN, Modbus и др.) [1].

Несмотря на большое разнообразие стандартов, большинство из них не отражает тенденции к переходу на сервис-ориентированную архитек-