Технология программирования

Блок 3. Сложные структуры данных и методы их обработки

Тема 3. Доступ к файлам

Цели изучения темы:

- сформировать у студентов навыки работы с внешними файлами с помощью VBA.

Задачи изучения темы:

- ознакомить студентов с видами внешних файлов и способами работы с ними.

В результате изучения данной темы Вы будете
знать:

- виды внешних файлов и их особенности;
- синтаксические особенности работы с внешними файлами в VBA;

уметь:

- программировать задачи с использованием внешних файлов;
- анализировать и применять соответствующие операторы, функции и объектные методы;

владеть:

- набором необходимых функций, операторов и методов управления файлами.

Учебные вопросы темы:

1. Работа с внешними файлами
2. Работа с файлом произвольного доступа
3. Работа с файлом последовательного доступа
4. Объект FileSearch

Основные термины и понятия, которые Вам предстоит изучить:

файл произвольного доступа, файл произвольного доступа, бинарный файл, константы атрибутов файла.

Вопрос 1. Работа с внешними файлами

Программы, написанные на VBA, умеют работать с внешними файлами.

Управление файлами включает действия, такие как копирование файлов, удаление неиспользуемых файлов для освобождения области дисковой памяти, перемещение файлов с одного диска на другие и создание или удаление каталогов диска. Управление файлами включает также такие виды обработки, как просмотр списка файлов в папке для определения размера файла или даты и времени, когда этот файл был модифицирован в последний раз.

В частности, на практике могут возникнуть задачи по поиску файлов в директориях, по открытию, обработке, сохранению файлов. Открытие, обработка, сохранение – дело отдельных приложений (например, MS Word, MS Excel) – то есть эти задачи решаются с помощью объектных моделей этих приложений. А вот поиск файлов осуществляется общими для всех методами VBA.

Операторы, функции и объектные методы, имеющиеся в VBA, делятся на шесть различных функциональных частей:

- Получение или изменение атрибутов файла;
- Выборка или нахождение имен файлов;
- Получение или изменение текущего диска и папки или создание и удаление папок диска;
- Копирование или удаление файлов;
- Переименование или перемещение файлов;
- Получение информации о файлах, такой как длина файла, дата и время, когда этот файл был модифицирован последний раз.

В VBA допустима работа с тремя типами текстовых файлов.

Файл последовательного доступа	Рассматривается как последовательность строк произвольной длины, разделенных специальными символами. Чтение и запись в файл производится построчно
Файл произвольного доступа	Состоит из записей фиксированной длины и размер записи указывается при его открытии. Это позволяет локализовать любую запись в файле по ее номеру
Бинарный файл	Является частным случаем файла произвольного доступа. Размер записи в бинарном файле считается равным 1 байту.

Открытие и закрытие файла

Open – разрешает выполнение операций ввода/вывода при работе с файлом.

Open Путь For Режим [Access Доступ] [Блокировка] As [#] НомерФайла [Len=Длина]

- *Путь* — строковое выражение, указывающее имя файла
- *Режим* — устанавливает режим работы с файлом. Допустимые Значения: Append, Binary, Input, Output или Random
- *Доступ* — устанавливает операции, разрешенные с открытым файлом. Допустимые значения: Read, Write или Read Write
- *Блокировка* — устанавливает операции, разрешенные с открытым файлом другим процессам. Допустимые значения:Shared, Lock Read, Lock Write и Lock Read Write и
- *Номер файла* — допустимый номер файла. Число в интервале от 1 до 255. Обратите внимание на то, что параметру Номер-Файла предшествует символ#. Значение номерФайла нельзя изменять, пока файл открыт. Но при следующем открытии файла номер Файла может быть другим числом длина — число, меньшее либо равное 32 767 (байт). Для файлов, открытых в режиме Random, это значение является длиной записи. Для файлов с последовательным доступом это значение является числом буферизуемых символов.

Про инструкцию ***open*** важно также знать, что во время ее работы VBA также резервирует файловый буфер в памяти компьютера для ускорения процесса записи и считывания (прямое записывание информации на диск может существенно замедлить выполнение программы, что особенно заметно при работе с большими файлами). Максимальное число файловых буферов устанавливается в системном файле Config.sys

Close – завершает операции ввода/вывода с файлом, открытым с помощью инструкции ***open***. Эта инструкция 'очищает буфер и указывает операционной системе обновить FAT (таблицу размещения файлов). Важно, чтобы каждый файл по завершении работы с ним был закрыт, иначе это может привести к частичной потере информации.

Close [СписокНомеровФайлов]

СписокНомеровФайлов может представлять один или несколько номеров файлов. При этом используется следующий синтаксис, где номерФайла представляет любой допустимый номер файла: [[#] номерФайла] [, [#] номерФайла]

Reset – закрывает все активные файлы, открытые с помощью инструкции open, и записывает содержимое всех буферов файлов на диск, открытых с помощью инструкции *Open*

FreeFile – функция возвращает доступный номер, который может использоваться в инструкции *Open*.

Приведем пример инструкции, открывающей для записи бинарный файл первый из рабочей папки, и затем инструкции, закрывающей этот бинарный файл.

```
Open "Первый" For Binary Access Write As #1
Close #1
```

Вопрос 2. Работа с файлом произвольного доступа

Приведем инструкции ввода/вывода информации при работе с файлом произвольного доступа, а также инструкции определения длины файла и текущей позиции указателя в файле.

Put Записывает содержимое переменной в файл произвольного доступа.

```
Put [#] НомерФайла, [НомерЗаписи] , ИмяПеременной
```

- НомерФайла — номер файла
- НомерЗаписи — номер записи (режим Random) или номер байта (режим Binary), с которого следует начать запись. Если аргумент НомерЗаписи опущен, то записывается на то место, где был установлен указатель после выполнения последней инструкции Get или Put, либо куда он переведен с помощью функции
- ИмяПеременной — имя переменной, содержащей данные, которые следует записать в файл

Get Читает данные из открытого файла произвольного доступа в переменную.

```
Get [#] НомерФайла, [НомерЗаписи] , ИмяПеременной
```

- НомерФайла — номер файла
- НомерЗаписи — номер записи (для файлов в режиме Random) или номер байта (для файлов в режиме Binary), с которого следует начать чтение
- Имяпеременной — имя переменной, в которую следует поместить считанные данные

Seek Функция возвращает значение типа *Long*, определяющее текущее положение указателя чтения/записи внутри файла, открытого с помощью инструкции *Open*

```
Seek (НомерФайла)
```

LOF Функция возвращает значение типа *Long*, представляющее размер файла в байтах, открытого с помощью инструкции *Open* . Для определения размера закрытого файла следует использовать функции *FileLen*

```
LOF (НомерФайла)
```

FileLen Возвращает значение типа *Long*, содержащее размер файла в байтах

```
FileLen (Путь )
```

Вопрос 3. Работа с файлом последовательного доступа

Файл последовательного доступа иногда удобно задавать как последовательность записей (например, записей о студентах), причем каждая из записей формируется из группы полей (например из полей Фамилия и оценка). Отметим, что такая группировка по записям не является чем-то присущим файлам последовательного доступа. Это просто подход, позволяющий упростить процесс последовательного считывания записей. В файле последовательного доступа существует только одна внутренняя структура, образованная разделителями (запятыми или специальными символами, обозначающими переход на новую строку).

В противоположность файлам последовательного доступа, в файлах произвольного доступа запись является встроенным элементом. Файл произвольного доступа — это файл, упорядоченный по записям, что позволяет быстро переместиться на любую запись, минуя предыдущие.

При создании файла произвольного доступа указывается максимальная длина каждой записи. Само собой разумеется, что в любую запись можно вводить данные, занимающие не все место, выделенное для записи, но нельзя ввести данные, требующие больше места, чем допустимая длина записи. Лишняя информация будет просто усекаться.

На практике бывает удобно создавать отдельный файл, в котором хранится информация о структуре файла произвольного доступа: структура записи, ее длина и заголовки полей. Отметим, что при открытии файла произвольного доступа, в отличие от файла последовательного доступа, не надо специально указывать, открывается он для ввода или вывода информации. Ввод и вывод информации определяют команды *Put* и *Get*.

Интересной особенностью файла произвольного доступа является то, что при работе с ним можно определить число записей не пересчитывая их. Число записей равно отношению размера файла к длине одной записи. Длина записи устанавливается при создании файла произвольного доступа и определяется типом переменной, при помощи которой файл был создан, размер открытого файла возвращается функцией *LOF*, а еще не открытого — функцией *FileLen*. В таком случае число записей в файле равно **`LOF(Номер файла) / Len(ДлинаЗаписи).`**

В нижеприведенной таблице указаны функции, операторы и методы управления файлами. В первом столбце таблицы находится ключевое слово VBA, во втором - указывается, предназначено ли ключевое слово для функции, оператора или объектного метода. В третьем столбце содержится краткое описание назначения каждой функции, оператора или метода.

Имя	Категория	Назначение
ChDir	Оператор	Изменяет текущий каталог. `ChDir путь`
ChDrive	Оператор	Изменяет текущий диск. `ChDrive диск` Например, `ChDrive "D"`
CurDir	Функция	Возвращает текущий каталог
Dir	Функция	Возвращает имя каталога или файла, совпадающее с определенным именем файла, передаваемым как строковый аргумент. Предназначена для нахождения одного или нескольких файлов на диске.
FileCopy	Оператор	Копирует файл `FileCopy source, destination` • source — строковое выражение, указывающее имя копируемого файла • destination — строковое выражение, указывающее имя результирующего файла. Аргумент destination может содержать имя каталога или папки и диска
FileDateTime	Функция	Возвращает значение типа Date, содержащее дату и время, когда этот файл был изменен последний раз. `FileDateTime (путь )`
FileLen	Функция	Возвращает длину файла в байтах
FileAttr	Функция	Возвращает значение типа Long, представляющее режим файла, открытого с помощью инструкции open. Возвращаемые значения: 1 (для режима input), 2 (output), 4 (Random), 8 (Append) и 32 (Binary) `FileAttr (НомерФайла, Тип)` • НомерФайла — допустимый номер файла

		• Тип — число, указывающее характер возвращаемых данных. Если тип установлен равным 1 , то функция FileAttr возвращает значение, указывающее режим работы файла
GetAttr	Функция	Возвращает число, представляющее объединенные атрибуты файла или каталога диска, такие как System, Hidden и т.д. `GetAttr (путь)`
GetOpenFileName	Метод	Отображает Excel-диалоговое окно Open и возвращает имя файла, выбранное пользователем. В Word не имеется.
GetSaveAsFileName	Метод	Отображает Excel-диалоговое окно Save As и возвращает имя файла, выбранное пользователем. В Word не имеется.
Kill	Оператор	Удаляет файлы с диска. `Kill путь` В аргументе путь допустимо использование символов (*) и (?) для удаления нескольких файлов по маске
MkDir	Оператор	Создает каталог диска.
Name	Оператор	Переименовывает или перемещает файл.
RmDir	Оператор	Удаляет каталог диска.
SetAttr	Оператор	Устанавливает атрибуты файла. `SetAttr pathname, attributes` Атрибуты в аргументе attributes определяются как сумма констант

Константы атрибутов файла

Константа	Значение	Описание
vbNormal	0	Обычный
vbReadOnly	1	Только чтение
vbHidden	2	Скрытый
vbSystem	4	Системный
vbDirectory	16	Каталог или папка

vbArchive	32	Файл был изменен после последнего резервирования

Как правило, чтобы открыть файл, нужно знать его имя. Иными словами, поиск файлов заключается в получении имен файлов, находящихся в определенной директории. Для этого можно использовать команду Dir. Она возвращает строку, содержащую имя файла, используя путь, заданный при вызове. Давайте рассмотрим конструкцию, которая позволяет найти все файлы, находящиеся в корневой директории диска C.

```
var_Doc = Dir("C:\*.*")
    Do While var_Doc <> ""
        MsgBox var_Doc
        var_Doc = Dir()
    Loop
```

Сначала мы присваиваем переменной *var_Doc* первое найденное имя файла. Очевидно, что узнав имя файла, мы можем сказать, что нашли этот файл на диске. В самом простом варианте использования функции *Dir* в качестве параметров мы передаем ей путь и маску имени файла. Знак ***** в маске означает любое количество любых символов. Следовательно, ***.*** означает "все файлы" - то есть файлы с любыми именами и любыми расширениями. В маске можно так же использовать знак **?** - он символизирует один любой символ. Если не указать путь к файлам, а лишь маску - *Dir* будет искать их в текущей директории. Например, для Microsoft Word по умолчанию это папка Мои документы.

Помимо пути и маски при поиске файлов можно указать некоторые дополнительные параметры. Так, по умолчанию функция ищет лишь обычные файлы, не обращая внимания на папки, скрытые и системные файлы. Чтобы функция нашла по заданному пути не только файлы, но и папки, ее нужно вызвать так:

```
var_Doc = Dir("C:\*.*", vbDirectory)
```

Обратите внимание на то, что после пути и маски указан параметр *vbDirectory* - он указывает функции, что она должна включить в поиск и директории.

После того, как первое найденное имя присвоено переменной, мы запускаем цикл с предусловием, в котором проверяем, не пуста ли эта переменная. Если *Dir* не обнаружил по указанному пути ничего подходящего под заданную маску, он возвратит, пустую строку. Следовательно, цикл в таком случае не выполнится ни разу.

В цикле есть две строки. Первая выводит найденное имя на экран, а вторая - вызывает функцию *Dir* еще раз - без параметров. Такой вызов

возвращает следующее имя файла, подходящее под заданный при первом вызове *Dir* шаблон. После этого все повторяется. В реальной программе в такой цикл можно вставить команды для работы с найденными файлами.

Помимо *Dir* полезной может оказаться команда *ChDir*. Она позволяет перейти в указанную при ее вызове директорию, которая будет использоваться в качестве директории по умолчанию. Такая конструкция, предшествующая циклу из предыдущего примера позволит найти все файлы в папке "Документы", которая расположена по пути " C:\Документы ":

```
ChDir ("C:\Документы")
var_doc = Dir("*.*")
```

Ввод данных в файл последовательного доступа

Print – записывает форматированные данные в файл последовательного доступа

```
Print #НомерФайла, [СписокВывода]
```

НомерФайла — номер файла

СписокВывода — выражение (или список выражений), записываемое в файл.

В аргументе СписокВывода разделителем списка выводимых выражений является ";" (данные выводятся подряд) или "," (данные выводятся по зонам). Кроме того, в аргументе СписокВывода допускается использование функций Spc и Tab:

Spc(n) — используется для вставки n пробелов в файл

Tab(n) — устанавливает курсор в столбец с номером n

Write – записывает неформатированные данные в файл последовательного доступа. В отличие от инструкции *Print*, инструкция *Write* вставляет запятые между элементами и заключает строки в кавычки по мере записи их в файл.

```
Write #НомерФайла, [СписокВывода]
```

НомерФайла — номер файла

СписокВывода — выражение или список выражений, записываемых в файл

Данные, записанные с помощью инструкции *Write*, обычно считываются из файла с помощью инструкции *Input*

Пример

```
Sub ПримерИспользования Print
Open "C:\Новый"
```

```
For Output As #1
'Печатает текст в файл
Print #1, "Тест"
'Печатает пустую строку в файл
Print #1,
'Печатает в двух зонах печати
Print #1, "Зона 1";
Tab;
"Зона 2" ;
Spc(3) ;
"3 пробела"
Close #1
End Sub
```

Результатом описанных выше инструкций будет файл со следующим содержимым:

Тест

Зона 1 Зона 2 3 пробела

```
Sub ПримерИспользованияwrite
Open "ЕщеПример"
For Output As #1
Write #1, "Пример"; "использования"
Write #1, "инструкции";
Write #1, "Write";
Write #1, "Число";
Close #1
End Sub
```

Результатом описанных выше инструкций будет файл со следующим содержимым:

"Пример","использования" "инструкции","Write" "Число"

Обратите внимание на автоматическое размещение в файле разделителей -запятых, и то, что строковая информация берется в кавычки. В процедуре *ПримерИспользованияWrite* вторая инструкция *write* специально заканчивается знаком ";" Это обеспечивает вывод данных третьей инструкцией write в ту же строку файла, в которую выводила вторая инструкция.

Вывод данных из файла последовательного доступа

Input # - считывает данные из открытого файла последовательного доступа и присваивает их переменным. Данные, считываемые с помощью инструкции Input, обычно записываются в файл с помощью инструкции Write #.

```
Input #НомерФайла, СписокПеременных
```

НомерФайла — номер файла

СписокПеременных — список переменных, которым следует присвоить значения, считанные из файла. Переменные в списке разделяются запятыми

Line Input # - считывает строку из открытого файла последовательного доступа и присваивает ее переменной типа string. Данные, считываемые с помощью инструкции Line input I, как правило, записываются в файл с помощью инструкции Print #.

```
Line Input #НомерФайла, ИмяПеременной
```

НомерФайла — номер файла

ИмяПеременной — имя переменной типа Variant или String

Функция Input - возвращает значение типа *string*, содержащее символы из файла, открытого в режиме *input* или *Binary*. Функция *input* считывает данные, записываемые в файл с помощью инструкции *Print* # или *Put*.

```
Input (Число, [#] НомерФайла)
```

Число задает число возвращаемых символов. Если аргумент Число равен1, то производится посимвольное считывание данных.

Функция EOF - функция возвращает значение *True* при достижении конца файла.

```
EOF (НомерФайла)
```

При последовательном считывании информации из файла часто используется следующий цикл:

```
Do While Not EOF(1) Loop
```

или, для тех пользователей, кто предпочитает инструкцию *While – Wend* инструкции *Do While - Loop*, следующий эквивалентный цикл:

```
While Not EOF (I) Wend
```

Пример использования инструкции input # для считывания данных из файла. Предполагается, что на диске существует файл группа Экономистов, содержащий информацию о студентах. Файл был создан при помощи инструкции write # и состоит из двух столбцов, в первом из которых указывается фамилия, а во втором — оценка студента. Для удобства работы с информацией введен пользовательский тип Студенты. Процедура пример использования Input последовательно считывает фамилии и оценки из файла и выводит их в ячейки первого и второго столбца рабочего листа Excell.

```
Type Students
Name As String * 20
Mark As String * 3
End Type
```

```
Sub UseInput()
Dim Stud As Students

Open "ГруппаЭкономистов"
For Input As 12 i = 1
Do While Not EOF(2)
With Stud
Input #2, .Name, .Mark
Cells(i, 1).Value = .Name
Cells(i, 2).Value = .Mark
End With
i = i + 1
Loop
Close #2
End Sub
```

Пример использования инструкции Line input # для считывания данных из файла группаЭкономистов, имеющего ту же структуру, что и в предыдущем примере, но созданного с помощью инструкции Print #. Инструкция Line input # считывает всю строку из файла в строковую переменную. Поэтому в этом случае уже нет необходимости использовать введенный пользовательский тип, а достаточно ограничиться только обычной строковой переменной. Вся считываемая информация строка за строкой вводится в список диалогового окна.

```
Private Sub UserForm_Initialize()
Dim Student As String
Open "ГруппаЭкономистов"
For Input As #1
i = 1
With ListBoxl
.Clear Do While Not EOF(l)
Line Input 11, Student
.Addltem Sudent
i = i + 1 Loop
Close #l
End With
End Sub
```

Вопрос 4. Объект FileSearch

Объект FileSearch обладает функциональными возможностями диалогового окна Открытие документа (Open), отображаемого на экране_посредством выбора команды Файл, Открыть (File, Open). Объект FileSearch входит в объект Application и иерархически включает в себя:

- Семейство FoundFiles, которое является списком всех файлов, возвращаемых в результате поиска
- Семейство PropertyTests, которое является списком всех критериев поиска

Объект FileSearch возвращается свойствомFileSearch объектаApplication.

Объект FileSearch имеет следующие два метода.

Execute - поиск специфицированных файлов.

```
Execute (SortBy, SortOrder, AlwaysAccurate)
```

SortBy — устанавливает способ сортировки файлов. Допустимые значения: msoSortbyFileName, msoSortbyFileType, msoSortbyLastModif led и msoSortbySize

SortOrder — устанавливает порядок сортировки файлов.

Допустимые значения: msoSortOrderAscending и msoSort Order Descending

AlwaysAccurate допустимые значения: True (поиск среди измененных файлов) и False (в противном случае)

NewSearch устанавливает критерии, используемые при поиске по умолчанию

Наиболее часто применяемые свойства объекта FileSearch.

FileName	Устанавливает имя файла для поиска. Допустимо использование символов (*) и (?)
FileType	Задает тип файла для поиска. Допустимые значения: msoFileTypeAHFiles, msoFileTypeBinders, msoFile-TypeDatabases, msoFileTypeExcelWorkbooks, msoFileTypeOffice Files, msoFileTyp PowerPointPresentations, msoFileTypeTemplates, msoFileTypeWordDocuments
Lookln	Задает папку для поиска файла
SearchSubFolders	Допустимые значения: True (поиск также проводить в поддиректориях) и False (в противном случае)

Пример выводит список всех файлов текущей папки в поле со списком диалогового окна

```
Private Sub UserForm_Initialize()
ComboBoxl.Clear With Application.FileSearch
.FileName = "*.xls" .SearchSubFolders = False
If              .Execute(SortBy:=msoSortByFileName,
sortorder:=msoSortOrderAscending) > 0 Then
For i = 1 To .FoundFiles.Count
```

```
ComboBoxl.Addltem .FoundFiles(i)
Next i
End If
End With
End Sub
```

Программа отображает в поле со списком полные имена файлов, т. е. имя файла и путь. Для того чтобы в списке отображались только имена файлов (без пути), программу необходимо модифицировать следующим образом:

```
Private Sub UserForm_Initialize()
Dim NameDir As String       ' имя папки
Dim NameFile As String      ' имя файла
Dim LenP As Integer         ' длина пути
ComboBoxl.Clear   NameDir   =   CurDir   LenP   =
Len(ИмяПапки) With Application.FileSearch .FileName
= "*.xls"
.SearchSubFolders = False
If       .Execute       (SortBy:=msoSortByFileName,
sortorder:=msoSortOrderAscending) > 0 Then
For i = 1 To .FoundFiles.Count
NameFile          =          Right(.FoundFiles(i),
Len(.FqundFiles(i)) - LenP - 1)
ComboBoxl.Addltem NameFile
Next i
End If
End With
End Sub
```

Контрольные вопросы:

1. Какие действия относятся к работе с файлами?
2. Какого типа файлы обрабатывает VBA?
3. Какая команда возвращает доступный номер файла,
4. Чем файл последовательного доступа отличается от файла с произвольным доступом?
5. Команды работы с файлом последовательного доступа.
6. Команды работы с файлом произвольного доступа