# Лекция 7
# Оглавление

# Файлы

Архитектура современных компьютеров предусматривает наличие оперативной и внешней памяти. В оперативной памяти находятся выполняемая в данный момент программа и обрабатываемые данные. Объем оперативной памяти ограничен, ее стоимость относительно высока, информация не сохраняется при выключении питания компьютера – все это не позволяют ее применять для постоянного хранения больших объемов информации. Для этой цели используется внешняя память. Это жесткие и гибкие магнитные диски, оптические диски, позволяющие хранить сотни мегабайт и гигабайты информации.

Данные на внешних носителях информации хранятся в виде файлов. Под файлом понимается именованная часть внешней памяти, предназначенная для хранения совокупность данных. Файл состоит из записей. Запись характеризуется длиной, выраженной в байтах.

Запись состоит из данных, которые передаются между оперативной и внешней памятью за одну операцию чтения или записи данных.

Чтение данных – это передача данных из внешней памяти в оперативную память, запись данных – это передача данных из оперативной памяти во внешнюю память.

Любое приложение, связанное с необходимостью хранения и обработки значительных объемов данных, неизбежно вынуждено хранить эти данные в файлах, а значит и с записью данных в файл и чтением их из файла. VB поддерживает три вида файлов:

- файлы с последовательным доступом,
- файлы с произвольным доступом,
- двоичные файлы.

Файлы с последовательным доступом – это в основном текстовые файлы, которые можно открывать с помощью текстового редактора. Текстовый файл может содержать коды символов, признак перевода строки `vbCrLf`, признак табуляции `vbTab` и признак конца файла. Здесь записи – это строки переменной длины, отделенные друг от друга символом перевода строки. Такие файлы обычно создаются приложениями для обработки и хранения текстовой информации (но не числовой).

Файлы с последовательным доступом читаются от начала к концу, поэтому невозможно одновременно и считывать из них данные, и записывать таковые. Обычно информация из текстового файла считывается вся в память и сохраняется вся в файле после окончания работы с ней. Чтобы изменить одну запись файла последовательного доступа, его нужно весь записать заново.

Если же приложению требуется частый доступ к данным, хранящимся в некотором файле, следует использовать файлы с произвольным доступом.

Как и в файлах с последовательным доступом, текстовые данные хранятся в них в виде символов, по два байта на символ. Однако числа хранятся в своем естественном формате (как `Integer`, `Double`, `Single` и т. д.).

Файлы с произвольным доступом в любой момент позволяют обработать любую запись. В предыдущих версиях VB требовалось, чтобы длина всех записей файла произвольного доступа была одинакова. В VB.NET записи файла произвольного доступа могут иметь разную длину. В этом случае информация о длине записи содержится в самой записи. Любую из них легко найти в файле по ее индексу. Более того, в отличие от файлов с последовательным доступом, файлы с произвольным доступом можно открывать для одновременного чтения и записи.

Двоичные файлы подобны файлам с последовательным доступом, но длина записи у этих файлов равна одному байту.

Все внешние устройства позволяют использовать последовательный доступ к записям файла. Произвольный доступ позволяют использовать только магнитные и оптические диски.

Процесс работы с файлом почти не зависит от его типа и условно делится на этапы:
1. Получение номера свободного канала ввода-вывода (номера файла). Применяется функция `FreeFile()`.
2. Открытие файла. Операционная система резервирует некоторое количество памяти для хранения данных файла. Если файл не существует, он создается, а затем открывается. Для открытия файла (при необходимости и для его создания) используется функция `FileOpen()`.
3. Чтение или запись данных. Файл может быть открыт для чтения, для записи или для чтения и записи. Данные считываются, обрабатываются, а затем снова сохраняются в том же или другом файле.
4. Закрытие файла. Когда файл закрывается, операционная система освобождает занимаемую им память. Для выполнения этой операции используется функция `FileClose()`.

В следующих разделах будут рассмотрены некоторые функции VB, используемые для обработки файлов. С полным перечнем этих функций можно ознакомиться с помощью справочной системы VB. К оглавлению

## 1. Структуры

Кроме базовых типов данных, таких как `Integer`, `Long` и т.п., VB поддерживает также типы данных, определяемые пользователем. Они могут быть созданы как на основе базовых типов данных, так и на основе типов ранее определенных пользователем. Пользовательский тип данных, называемый структурой, широко применяются при построении файлов произвольного доступа.

Для определения пользовательского типа (структуры) данных используется ключевое слово `Structure`:

```
[Public/Private]Structure Имя_типа
    Dim/Public/Private Элемент1 As Тип
    Dim/Public/Private [Элемент2 As Тип]
.
.
.
End Structure
```

Структура не может быть объявлена внутри процедуры или в функции. Она может быть объявлена только в начале проекта, а также в форме или в модуле до первой процедуры. Определив собственный тип данных, Вы можете использовать его для объявления переменных этого типа. Эти переменные могут быть локальными, переменными области формы или модуля, а также глобальными. Переменную пользовательского типа называют записью. Отдельные компоненты этой переменной называют полями записи. Переменная пользовательского типа является структурированной. Она подобно массиву включает в свой состав отдельные элементы, но в отличие от массива, ее элементы могут иметь разный тип:

```
Public Class Form1
    Structure Товар
        Dim Название As String
        Dim Цена As Decimal
        Dim Номер As Long
    End Structure
    Dim Инструмент As Товар

    Private Sub Button1_Click()
.
```

```
.
.
        Инструмент.Название = "Отвертка"
        Инструмент.Цена = 120
.
.
.
    End Sub
```

В этом примере определяется тип данных `Товар`. Затем объявляется переменная `Инструмент` типа `Товар`, а конкретные значения составляющих этой переменной устанавливаются в процедуре `Button1_Click`.

Доступ к элементам переменной пользовательского типа осуществляется, по аналогии с доступом к свойствам, путем указания точки после имени переменной. При этом переменные одинакового типа можно присваивать не поэлементно, а напрямую:

```
    Structure Субъект
        Dim Фамилия_и_Инициалы As String
        Dim ТабельныйНомер As Integer
    End Structure
    Dim Читатель, Пользователь As Субъект
    Private Sub Button1_Click()
        Пользователь.Фамилия_и_Инициалы = "Иванов И.И."
        Пользователь.ТабельныйНомер = 218739
        Читатель = Пользователь
    End Sub
```

Переменные `Читатель` и `Пользователь` относятся к одному типу `Субъект`. Поэтому они присваиваться напрямую, а не поэлементно.

Пользовательские типы данных могут быть составными. В этом случае важна последовательность определения типов. Сначала нужно определить базисный тип, который будет использоваться далее в составных типах. Если не соблюдать это правило, то после запуска программы появится сообщение об ошибке. Ниже приводится пример использования составных пользовательских типов данных:

```
    Structure Персона
        Dim Имя As String
        Dim Фамилия As String
    End Structure
    Structure Клиент
        Dim Идентификатор As Персона
        Dim ДеньРождения As Date
    End Structure

    Dim Покупатель As Клиент

    Private Sub Button1_Click()
        Покупатель.Идентификатор.Имя = "Иван"
        Покупатель.Идентификатор.Фамилия = "Петров"
    End Sub
```

Можно в подпрограмме `Button1_Click` создать массив записей:

```
Dim Персоны(100) As Персона
```

Обращение к полям седьмого элемента этого массива выглядит так:

```
Персоны(6).Имя
Персоны(6).Фамилия
```

В качестве элементов структуры можно использовать массивы. К оглавлению

Данные пользовательского типа рекомендуется использовать при обработке данных неизменной структуры.

## 2. Номер файла

Каждому открытому файлу система VB ставит в соответствие канал ввода-вывода с определенным номером. При выполнении операции ввода и вывода имеет значение не имя файла, а номер связанного с ним канала. Номер свободного канала, который можно использовать для работы с файлом может быть получен с помощью функции FreeFile():

***FreeFile*** ([RangeNumber]).

Необязательный параметр RangeNumber может принимать значение 0 (по умолчанию) и 1. Если его значение равно 0, то возвращается номер канала из диапазона 1– 255, если 1, то из диапазона 256 – 511.

Пример.

```
n = FreeFile()
```

К оглавлению

## 3. Открытие и закрытие файла

Чтобы иметь возможность использовать файл, его нужно открыть или создать, если он еще не существует. Функция FileOpen(), выполняющая эту операцию, принимает множество аргументов, из которых обязательными являются только первые три:

```
FileOpen(number, path, mode _
[, access][, share][, recordLen])
```

В аргументе number задается номер файла, предварительно полученный с помощью функции FreeFile().В аргументе path задается путь к открываемому файлу. Аргумент mode определяет режим открытия файла и может содержать одно из значений, перечисленных в табл. 1.

Таблица 1. Перечисление mode:

| Значение | Описание |
|---|---|
| OpenMode.Input | Файл открывается только для ввода данных (то есть для чтения из файла) |
| OpenMode.Output | Файл открывается только для вывода (то есть для записи в файл) |
| OpenMode.Append | Файл открывается для добавления новых данных |
| OpenMode.Random | Файл открывается для произвольного доступа (чтения и записи, по одной записи зараз) |
| OpenMode.Binary | Файл открывается как двоичный |

Первые три режима характерны для файлов с последовательным доступом. Режим Random предназначен для файлов с произвольным доступом, а Binary – для двоичных файлов. В файле с последовательным доступом изменять отдельные элементы данных нельзя. Можно либо прочитать их (а при желании и сохранить в другом файле), либо записать новые данные. При необходимости получить данные из такого файла нужно открыть его в режиме Input, считать данные и закрыть файл. Для перезаписи данных нужно снова открыть этот файл, на этот раз в режиме Output, и сохранить в нем новые данные.

Если не нужно перезаписывать существующий файл, и Вы хотите просто добавить в него данные (не меняя существующие), откройте файл в режиме Append.

В случае открытия файла в режиме `Output` (и только в этом режиме) VB сотрет его содержимое, даже если Вы ничего в него не запишете. Более того, VB не станет предупреждать Вас о своем намерении стереть файл.
Аргумент `access` определяет, следует ли открыть файл для чтения (`Read`), записи (`Write`) или и чтения, и записи (`ReadWrite`). Если открыть файл с опцией `Read`, программа не сможет модифицировать его даже по ошибке. Данный аргумент не имеет никакой связи с типом файла. Файлы с последовательным доступом можно открывать лишь с опциями `Read` и `Write`, поскольку их можно либо только записывать, либо только считывать. Аргумент `access` предназначен исключительно для защиты данных от случайного изменения. Если Вам нужно прочитать данные из файла, откройте его с опцией `Read`, что позволит исключить риск их изменения.
Аргумент `share` определяет права других приложений Windows на то время, пока ваше приложение держит файл открытым. В Windows может выполняться множество приложений одновременно, и не исключено, что какое-нибудь из них попытается открыть уже открытый файл. Поэтому имеет смысл указать, может ли другое приложение открывать данный файл и в каком режиме. Допустимые значения аргумента `share` приведены в табл. 2.

**Таблица 2.** Перечисление `share`:

| Значение | Описание |
|---|---|
| `OpenShare.Shared` | Другие приложения могут работать с этим файлом |
| `OpenShare.LockRead` | Файл заблокирован для чтения |
| `OpenShare.LockWrite` | Файл заблокирован для записи |
| `OpenShare.LockReadWrite` | Файл не доступен для других приложений |

Открыв для работы файл с произвольным доступом, можно задать в последнем аргументе длину записи в байтах. В этом случае VB не будет сохраняет в файле информацию о длине и структуре записей. Но в приложении должна храниться информация о структуре записей открываемого файла.
Длина записи равна сумме байтов, занимаемых всеми ее полями. Вы можете вычислить ее либо самостоятельно, либо вызвав функцию `Len(record)`. В аргументе `record` задается имя структуры, используемой для создания записей файла с произвольным доступом.
Пример.

```
Dim n As Integer = FreeFile()
FileOpen(n, "c:\t\f.dat", _
OpenMode.Output, OpenAccess.ReadWrite)
```

Функция `FileClose()` закрывает файл, номер которого передан ей в качестве аргумента:

```
FileClose(file_number)
```

Так, следующая инструкция закрывает файл с номером n1:

```
FileClose(n1)
```

К оглавлению

## 4. Запись в файл и чтение из файла

Функция `PrintLine()` записывает данные в файл с последовательным доступом:

```
PrintLine(file_number, output_list)
```

В ее первом аргументе задается номер файла, в который производится запись, а в остальных аргументах – записываемые значения. Аргумент `output_list` представляет собой массив параметров, через который функции можно передать любое количество значений:

```
PrintLine(n, v1, v2, "Петров", 123.456)
```

Если в файл записывается несколько значений, они разделяются запятыми, и каждая такая запятая указывает, что очередной символ будет выводиться в следующей зоне вывода. Зоне вывода соответствуют 14 позиций.
Для чтения очередной записи данных из файла с последовательным доступом используется функция `LineInput()`:
**`LineInput`**`(file_number)`
В аргументе `file_number` ей передается номер уже открытого файла. При первом вызове она считывает все символы от начала файла до первого символа новой записи. Когда Вы вызываете ее во второй раз, она возвращает все последующие символы вплоть до следующего символа новой записи.
Символ новой строки не включается в результирующие данные – он используется только в качестве разделителя. Приведенный ниже код прочитает две очередные записи файла и присвоит их строковым переменным `st1` и `st2`:
```
st1 = LineInput(n)
st2 = LineInput(n)
```
Если требуется сохранить в файле на диске обычный текст, следует создать для этой цели файл с последовательным доступом. Для того чтобы прочитать этот текст из файла, нужно открыть файл снова и прочитать текст построчно с помощью функции `LineInput()`.
**`FilePut`**`(file_number, value[, record_number]),`
**`FileGet`**`(file_number, value[, record_number])`
Функции используются для записи и чтения записей из файла с произвольным доступом. Каждой из них может быть передан номер записи, с которой Вы хотите работать. В аргументе `record_number` задается номер записи, а в аргументе `value` – переменная, содержащая записываемую в файл запись или принимающая запись, которая считывается из файла. Аргумент `record_number` не обязателен; если он не задан, будет записываться или считываться текущая запись. После записи или чтения очередной записи текущей становится следующая запись. Таким образом, вызвав функцию `FilePut()` десять раз подряд без указания номера записи, Вы последовательно создадите или перезапишете первые десять записей файла с произвольным доступом. Точно так же, вызвав десять раз подряд функцию `FileGet()` без указания номера записи, Вы последовательно прочитаете первые десять записей файла.
Несколько слов о принципах обработки файлов с произвольным доступом. Предположим, Вы хотите создать файл с произвольным доступом для хранения списка товаров. Информация о каждом из товаров содержится в структуре `Товар`, определяемой следующим образом:
```
    Structure Товар
        Dim Код As String
        Dim Название As String
        Dim Цена As Decimal
    End Structure
```
Структура `Товар` будет использоваться для хранения информации о товаре перед ее записью в файл. Начнем с объявления переменой типа `Товар`:
```
Dim Тов As Товар
```
Теперь можно присвоить значения полям структуры `Тов`:
```
Тов.Код = "TV00180-A"
Тов.Название = "SONY Trinitron TV"
Тов.Цена = 799.99
```
Для записи данных, хранящихся в переменной `Тов`, в файл с произвольным доступом, мы воспользуемся функцией `FilePut()`. Конечно, файл сначала нужно создать с помощью инструкций:
```
fn = FreeFile()
FileOpen(fn, "c:\products.dat", OpenMode.Random)
```

Как видите, последний аргумент, в котором задается длина записи, здесь опущен. Записи содержат строки, следовательно, имеют переменную длину. Информация о длине каждой строки хранится вместе с самой строкой, так что определять длину каждой записи не понадобится. Далее переменная `Тов` записывается в файл с помощью инструкции:

```
FilePut(fn, Тов)
```

Обратите внимание, что номер записи, в которой сохраняются данные, не указан. Вы можете изменить значения полей и сохранить в файле следующую запись с помощью точно такой же инструкции. После записи всех необходимых данных файл закрывается с помощью инструкции `FileClose(fn)`.

Для того чтобы прочитать данные из файла, нужно открыть его с применением той же функции `FileOpen()`, с помощью которой мы открывали его для записи данных:

```
fn = FreeFile()
FileOpen(fn, "c:\products.dat", OpenMode.Random)
```

Затем обычно выполняется цикл чтения записей.

Следующий фрагмент кода демонстрирует, как записать в файл с произвольным доступом записи разной длины с помощью функции `FilePut()` и как их прочитать с помощью функции `FileGet()`. Прежде всего, добавьте в форму до первой процедуры следующее объявление структуры:

```
    Structure Товар
        Dim Код As String
        Dim Название As String
        Dim Цена As Decimal
    End Structure
```

Затем поместите в форму кнопку и введите в обработчике ее события `Click` такие инструкции:

```
    Dim fn As Integer
    Dim Тов As Товар
    Тов.Код = "TV00180-A"
    Тов.Название = "SONY Trinitron TV"
    Тов.Цена = 799.99
    fn = FreeFile()
    FileOpen(fn, "c:\products.dat", OpenMode.Random)
    FilePut(fn, Тов)
    Тов.Код = "TV-RCA"
    Тов.Название = "This is an RCA Trinitron TV"
    Тов.Цена = 699.99
    FilePut(fn, Тов)
    Тов.Код = "TV810X"
    Тов.Название = "Real cheap BIG Trinitron TV"
    Тов.Цена = 399.99
    FilePut(fn, Тов)
    FileClose(fn)
    fn = FreeFile()
    FileOpen(fn, "c:\products.dat", OpenMode.Random)
    FileGet(fn, Тов, 2)
    FileClose(fn)
    Console.WriteLine(Тов.Код)
    Console.WriteLine(Тов.Название)
    Console.WriteLine(Тов.Цена)
```

Коды и описания различных товаров представляют собой строки разной длины. Первая часть приведенного кода записывает три записи в файл с произвольным доступом и закрывает его. Вторая часть кода считывает из файла вторую запись и выводит ее поля в

окне Output. Итак, функции позволяют создавать записи со строками, которые имеют переменную длину. Функции FilePut() и FileGet() берут на себя ответственность за работу с такими строками, предоставляя Вам доступ к данным файлов с произвольным доступом и используя запись в качестве базовой единицы длины. К оглавлению

## 5. Вспомогательные функции и методы

### 5.1. Функции Lock и Unlock

**Lock**(file_number[, fromRecord][, toRecord]),
**Unlock**(file_number[, fromRecord][, toRecord])

Функция Lock() позволяет заблокировать файл с произвольным доступом или его отдельные записи. Заблокированные записи не доступны другим приложениям. Однако Ваше приложение имеет доступ ко всему файлу. Если другое приложение попытается открыть заблокированный файл или получить доступ к заблокированной записи, VB сгенерирует для него ошибку времени выполнения.

Если необязательные аргументы не заданы, файл блокируется целиком. Но если задан аргумент fromRecord, блокируются все записи до конца файла, начиная с указанной записи. В случае же применения обоих необязательных аргументов будет заблокирован диапазон записей.

Функция Unlock() выполняет операцию, противоположную выполняемой функцией Lock(), то есть снимает установленную ею блокировку.

### 5.2. Функции EOF и LOF

**EOF**(file_number), **LOF**(file_number)

Эти две функции используются при работе с файлами постоянно.

Функция EOF()(End Of File) принимает в качестве аргумента номер открытого файла и возвращает True, если достигнут его конец (EOF). Функция EOF() часто используется в циклах для определения момента достижения конца файла:

```
    Dim s As String, T As String = ""
    Dim fn As Integer
fn = FreeFile()
FileOpen(fn, "c:\ReadMe.txt", OpenMode.Input)
Do Until EOF(fn)
    s = LineInput(fn)
    T = T & s & vbCrLf
Loop
```

В этом коде инструкция LineInput выполняется в цикле, пока не будет достигнут конец файла. При этом на каждом шаге цикла считывается отдельная строка и к ней добавляется символ конца строки, который отбрасывает инструкция LineInput. Функция EOF возвращает значение True при достижении конца файла и прекращает выполнение цикла.

Функция LOF() возвращает выраженную в байтах длину файла, номер которого передан ей в качестве аргумента. С помощью функции LOF() можно также вычислить количество записей в файле с произвольным доступом:

```
Количество_записей = LOF(file_number) / Len(record)
```

Здесь Len(record) – это длина записи в байтах. В аргументе record задается имя структуры, используемой для создания записей файла с произвольным доступом.

### 5.3. Функция Seek

**Seek**(file_number[, position])

Функция Seek() возвращает текущую позицию указателя ввода-вывода в заданном файле, если она вызвана без аргумента position. Задав в функции Seek() аргумент position, можно установить текущую позицию ввода/вывода. Например, для того

чтобы в файле с произвольным доступом перейти к началу третьей записи, нужно выполнить такую инструкцию:

```
Seek(fNum, 3)
```

### *5.4. Методы копирования, создания и удаления файлов*

В составе NET.FrameWork имеются разнообразные методы, позволяющие работать с файлами. Для работы с файлами применяются методы класса `File`. Рассмотрим некоторые из них.

Метод **`Copy`**

Копирует существующий файл в другое место, например:

```
File.Copy(источник, копия)
```

В аргументе `источник` задается путь к исходному файлу, а в аргументе `копия` – путь к результирующему файлу. Если результирующий файл уже существует, метод в такой форме приводит к ошибке времени выполнения. Для перезаписи существующего файла применяется другая форма метода:

```
File.Copy(источник, копия, перезапись)
```

Если последний аргумент имеет значение `True`, и файл, указанный во втором аргументе существует, то он перезаписывается.

Например:

```
File.Copy("c:\Temp\f.jpg", "d:\Image\f1.jpg")
```

Метод **`Delete`**

Удаляет заданный файл:

```
File.Delete(путь)
```

Перед удалением файл должен быть закрыт. К оглавлению

## 6. Пример 1. Файл последовательного доступа

Записать в файл последовательного доступа произвольного (по желанию пользователя) числа записей. Затем все записи прочитать из файла и отобразить в текстовом поле. Решение этой задачи содержится в листинге 7.1.

**Листинг 7.1. Работа с файлом последовательного доступа**

```
Private Sub btnПуск_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnПуск.Click
    Dim s, ss As String
    Dim k As Integer
    Dim b As Boolean
    k = FreeFile()
5:  FileOpen(k, "E:\MY Documents\F1.txt", OpenMode.Output)
    Do Until b
        s = InputBox("Содержание записи = ?")
        PrintLine(k, s)
    s = InputBox("Продолжать? Введите да или иное - (нет)")
10:     If s = "да" Then b = False Else b = True
    Loop
    FileClose(k)
    k = FreeFile()
    FileOpen(k, "E:\MY Documents\F1.txt", OpenMode.Input)
15: ss = ""
    Do Until EOF(k)
        s = LineInput(k)
        ss = ss & s & vbCrLf
    Loop
20: FileClose(k)
```

```
    txtЖурнал.AppendText(ss)
End Sub
```
К оглавлению

## 7. Пример 2. Файл произвольного доступа

Код, содержащийся в листинге 7.2, записывает в файл произвольного доступа произвольное (по желанию пользователя) число записей. Затем одна запись, номер которой задает пользователь, читается из файла и отображается в текстовом поле.

**Листинг 7.2. Работа с файлом произвольного доступа**

```
Structure Чек
    Dim Номер As String
    Dim Дата As Date
    Dim Название As String
    Dim Цена As Decimal
End Structure
Private Sub btnПуск_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnПуск.Click
    Dim x As Чек, n, k As Integer, b As Boolean, s As String
        k = FreeFile()
    FileOpen(k, "E:\MY Documents\F2.dat", OpenMode.Random)
    Do Until b
5:      x.Номер = InputBox("Введите номер чека")
        x.Дата = InputBox("Введите дату в формате #дд/мм/гггг#")
        x.Название = InputBox("Введите название товара")
        x.Цена = InputBox("Введите цену товара")
        FilePut(k, x)
10:     s = InputBox("Продолжать? Введите да или иное - (нет)")
        If s = "да" Then b = False Else b = True
    Loop
    FileClose(k)
    k = FreeFile()
15: FileOpen(k, "E:\MY Documents\F2.dat", OpenMode.Random)
    n = InputBox("Введите номер считываемой записи")
    FileGet(k, x, n)
    FileClose(k)
    txtЖурнал.AppendText("Запись № " & n & vbCrLf)
20: txtЖурнал.AppendText("Номер" & vbTab & x.Номер & vbCrLf)
    txtЖурнал.AppendText("Дата" & vbTab & x.Дата & vbCrLf)
    txtЖурнал.AppendText("Назв." & vbTab & x.Название & vbCrLf)
    txtЖурнал.AppendText("Цена" & vbTab & x.Цена & vbCrLf)
End Sub
```
К оглавлению

## 8. Пример 3. Исследование функции Rnd

Для генерации случайных величин применяется функция Rnd(). Эта функция возвращает случайное число a типа Single, удовлетворяющее условию $0 \le a < 1$.

Проверим, насколько близка эта случайная величина, получаемая с помощью функции Rnd(), к известной в теории вероятностей случайной величине, имеющей равномерное распределение?

Важнейшими характеристиками случайной величины является ее математическое ожидание (среднее) и среднее квадратичное отклонение – СКО (мера разброса относительно среднего значения).

Для равномерно распределенной случайной величины известны значения ее математического ожидания и СКО:

```
Среднее = 0,5
СКО = Sqrt(1/12) = 0,288675134594813
```

Имея возможность с помощью функции Rnd() получить множество реализаций случайной величины $a_i$, можно вычислить экспериментальную оценку среднего и оценку СКО.
Для этого следует применить известные в теории вероятности формулы:

$$Среднее = \frac{1}{k}\sum_{i=1}^{k} a_i \qquad СКО = \sqrt{\frac{1}{(k-1)}\sum_{i=1}^{k}(a_i - Среднее)^2}$$

Для решения поставленной задачи следует предусмотреть выработку достаточного большого числа случайных величин и запись их в файл последовательного доступа. Эти действия будут предусмотрены в подпрограмме – обработчике события `Click` кнопки `btnПуск`. Затем будут вычислены оценка среднего значения и оценка СКО. Эти действия будут предусмотрены в подпрограмме – обработчике события `Click` кнопки `Button1`.
Поставленную задачу решает код, содержащийся в листинге 7.3.

**Листинг 7.3. Исследование функции Rnd**

```
Imports System.Math
Public Class Form1
    Private Sub Form1_Load(ByVal sender _
    As Object, ByVal e As  _
    System.EventArgs) Handles Me.Load
        Randomize()
    End Sub
    Private Sub btnПуск_Click(ByVal sender _
    As System.Object, ByVal e As  _
    System.EventArgs) Handles BtnПуск.Click
        'Генерация случайных чисел
        'и запись их в файл
        Dim k As Long
        k = InputBox("Сколько случайных" & _
        " чисел генерировать?")
        TxtЖурнал.AppendText("k = " & _
        k.ToString & vbCrLf)
        Dim fn As Integer
        fn = FreeFile()
        FileOpen(fn, "D:\РабочаяПапка\" & _
        "rndЧисла.txt", OpenMode.Output)
        Dim ai As Single
        For i = 1 To k
            ai = Rnd()
            PrintLine(fn, ai.ToString)
        Next
        FileClose()
        'Вычисление оценки среднего зачения
        'всех находящихся в файле случайных чисел
        Dim Сумма As Double
        fn = FreeFile()
        FileOpen(fn, "D:\РабочаяПапка\" & _
        "rndЧисла.txt", OpenMode.Input)
        Dim si As String, L As Long
        L = 0
```

```
            Do Until EOF(fn)
                si = LineInput(fn)
                Сумма = Сумма + CSng(si)
                L = L + 1
            Loop
            Dim Среднее As Double
            Среднее = Сумма / L
            TxtЖурнал.AppendText( _
            "Среднее значение = " & _
            Среднее.ToString & vbCrLf)
            FileClose()
            'Вычисление оценки СКО
            fn = FreeFile()
            FileOpen(fn, "D:\РабочаяПапка\" & _
            "rndЧисла.txt", OpenMode.Input)
            Сумма = 0
            L = 0
            Do Until EOF(fn)
                si = LineInput(fn)
                Сумма = Сумма + _
                (CSng(si) - Среднее) ^ 2
                L = L + 1
            Loop
            Dim СКО As Double
            СКО = Sqrt(Сумма / (L - 1))
            TxtЖурнал.AppendText( _
            "СКО = " & СКО.ToString _

            & vbCrLf)
            FileClose()
        End Sub
End Class
```

Результаты выполнения проекта показаны на рис. 7.1:

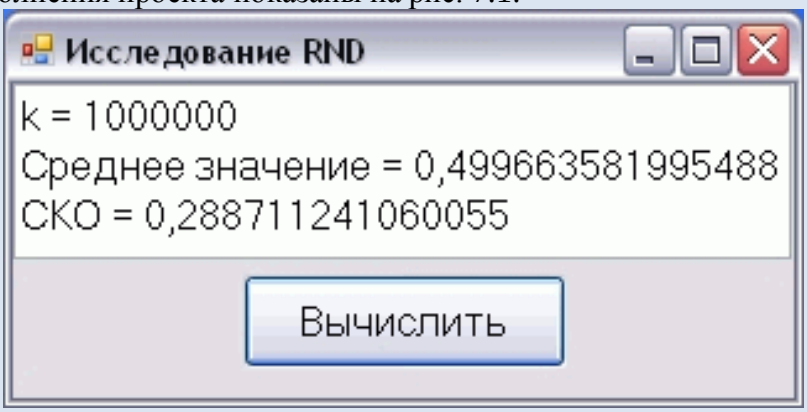


**Рис. 7.1. Полученные оценки математического ожидания и СКО функции Rnd**

Можно заметить, что экспериментальные оценки математического ожидания и СКО довольно близки к теоретическим значениям этих величин для случайной величины, имеющей равномерное распределение.

К оглавлению

## 9. Пример 4. Домашняя библиотека

Вам наверняка уже давно хотелось навести порядок в Ваших книгах. Записать каждую книгу, ее автора, где она хранится (шкаф, полка). Следующий проект, код которого содержит листинг 7.4, показывает, как это можно сделать, применив файл произвольного доступа. Этот проект можно также легко адаптировать для наведения порядка в аудиотеке, фильмотеке и пр.

В проекте применяются всего пять управляющих элементов:

текстовое поле `TextBox1`, для отображения текстовой информации;

управляющая кнопка `Button1` для создания записи в файле с данными о новой книге;

управляющая кнопка `Button2` для отображения значений всех полей записи с заданным номером и возможного выполнения изменения этих значений;

управляющая кнопка `Button3` для поиска записей с заданным названием книги;

управляющая кнопка `Button4` для поиска записей с заданной фамилией автора.

В главной секции General формы объявлены данные, которые действуют во всех процедурах формы:

структура `Библиотека`, в которой определены поля записей;

переменная `nf` – номер свободного канала ввода-вывода;

переменная `p` – номер текущей записи файла;

переменная `FileName` – полное имя файла произвольного доступа;

переменная `t` типа `Библиотека` применяемая как источник данных при выполнении записи в файл и как приемник данных при чтении из файла.

При запуске проекта сначала при загрузке формы выполняется подпрограмма `Form1_Load`. Если существует файл с заданным именем, то он будет открыт (строки 1 и 2). Затем эта подпрограмма, начиная с первой записи, поочередно в цикле (строки 5 – 9) считывает записи, пока не будет достигнут конец файла, а также в текстовом поле `TextBox1` отображает (строка 7) краткую информацию о каждой записи (номер записи, название книги и фамилию автора). При достижении конца файла в строке 5 функция `EOF(nf)` получит значение `True` и произойдет выход из цикла.

Если же файла с заданным именем нет, то будет создн пустой файл.

Подпрограмма `Button1_Click` предназначена для добавления в файл новой записи о книге. Сначала файл тоткрывается (строки 1 и 2).Затем определяется номер `p` последней записи (строки 3 – 7). После этого задаются конкретные значения всех полей новой записи (строки 10 – 17) и выполняется сама операция записи в файл (строка 18).

Подпрограмма `Button2_Click` позволяет увидеть значения всех полей записи с заданным номером, а также изменить значения всех полей записи и обновить содержание записи. Эта подпрограмма дает возможность пользователю задать номер редактируемой записи (строка 1), открывает файл (строки 2 и 3) и считывает запись с заданным номером (строка 4). Затем выполняется отображение в текстовом поле `TextBox1` значений всех полей записи (строка 6). После этого пльзователь может решить: следует ли ему изменять значения полей записи и перезаписывать запись в файл (строка 7). Функция `MsgBox`, когда ее второй аргумент равен 4 открывает окно с текстом, заданным ее первым аргументом и кнопками «Да» и «Нет».

Если пользователь закроет окно функциии `MsgBox`, нажав на кнопку «Да», то функция возвращает значение 6. Это приведет к выполнению строк 10 – 19. Значения всех полей записи будут обновлены и запись перезаписана в файл (строка 18).

Если же пользователь закроет окно функциии `MsgBox`, нажав на кнопку «Нет», то файл будет закрыт без обновления записи (строка 21).

Подпрограмма `Button3_Click` обеспечивает поиск книг по названию. Для выполнения поиска не обязательно задавать название книги полностью. Достаточно задать фрагментназвание книги. В текстовом поле `TextBox1` выводится список всех книг с одинаковым названием.

В подпрограмме `Button3_Click` применяется функция обработки строк `InStr(s1,s2)`. Оба аргумента этой функции – строки. Функция `InStr` возвращает номер позиции в строке `s1`, начиная с которой строка `s2` входит в строку `s1` или 0, если вхождения нет.

В подпрограмме `Button3_Click` также применяется функция обработки строк `StrConv(s,VbStrConv.Uppercase)`, которая обеспечивает нечуствительность при сравнении строк к регистру символов. Эта функция возвращает строку `s`, преобразованную к верхнему регистру.

Подпрограмма `Button4_Click` выполняет поиск книг по фамилии автора. Ее работа происходит аналогично работе подпрограммы `Button3_Click`. В текстовом поле `TextBox1` выводится список всех книг одного и того же автора.

К оглавлению

**Листинг 7.4. Домашняя библиотека**

```
Public Class Form1
    Structure Библиотека
        Dim Серия As String
        Dim Автор As String
        Dim Название As String
        Dim Том As String
        Dim Год As String
        Dim Цена As Decimal
        Dim Количество As Byte
        Dim Расположение As String
    End Structure
    Dim nf As Byte
    Dim p As Long
    Dim FileName As String = _
        "D:\РабочаяПапка\Домашняя_Библиотека"
    Dim t As Библиотека

    Private Sub Form1_Load(ByVal sender As  _
    System.Object, ByVal e As System.EventArgs) _
    Handles MyBase.Load
        nf = FreeFile()
        FileOpen(nf, FileName, OpenMode.Random, _
        OpenAccess.ReadWrite)
        p = 1
        TextBox1.Clear()
5:      Do Until EOF(nf)
            FileGet(nf, t, p)
            TextBox1.AppendText("Запись № " _
                & p & vbTab & t.Автор & _
                vbTab & t.Название & vbCrLf)
            p = p + 1
        Loop
10:     FileClose(nf)
    End Sub

    Private Sub Button1_Click(ByVal sender As  _
    System.Object, ByVal e As System.EventArgs) _
    Handles Button1.Click
        nf = FreeFile()
        FileOpen(nf, FileName, OpenMode.Random, _
        OpenAccess.ReadWrite)
        p = 1
        Do Until EOF(nf)
5:          FileGet(nf, t, p)
            p = p + 1
        Loop
        TextBox1.Clear()
        TextBox1.AppendText("Запись № " & p & vbCrLf)
10:     t.Серия = InputBox("Серия ?")
        t.Автор = InputBox("Автор ?")
        t.Название = InputBox("Название ?")
        t.Том = InputBox("Том ?")
        t.Год = InputBox("Год ?")
15:     t.Цена = InputBox("Цена ?")
        t.Количество = InputBox("Количество ?")
        t.Расположение = InputBox("Расположение ?")
        FilePut(nf, t, p)
        FileClose(nf)
20:     TextBox1.Text = "Запись добавлена!"
    End Sub
```

К оглавлению

```
    Private Sub Button2_Click(ByVal sender As  _
    System.Object, ByVal e As System.EventArgs) _
    Handles Button2.Click
        p = InputBox("Номер записи = ?")
        nf = FreeFile()
        FileOpen(nf, FileName, OpenMode.Random)
        FileGet(nf, t, p)
5:      TextBox1.Clear()
        TextBox1.AppendText( _
         "Запись № " & p & vbCrLf & _
         "Серия: " & t.Серия & vbCrLf & _
         "Автор: " & t.Автор & vbCrLf & _
         "Название:" & t.Название & vbCrLf & _
         "Том: " & t.Том & vbCrLf & _
         "Год: " & t.Год & vbCrLf & _
         "Цена: " & t.Цена & vbCrLf & _
         "Количество: " & t.Количество & vbCrLf & _
         "Расположение: " & t.Расположение & vbCrLf)
        Dim b As Integer
        b = MsgBox("Нужно редактировать?", 4)
        If b = 6 Then
10:         t.Серия = InputBox("Серия ?")
            t.Автор = InputBox("Автор ?")
            t.Название = InputBox("Название ?")
            t.Том = InputBox("Том ?")
            t.Год = InputBox("Год ?")
15:         t.Цена = InputBox("Цена ?")
            t.Количество = InputBox("Количество ?")
            t.Расположение = InputBox("Расположение ?")
            FilePut(nf, t, p)
            TextBox1.Text = "Запись исправлена!"
20:     End If
        FileClose(nf)
    End Sub

    Private Sub Button3_Click(ByVal sender As  _
    System.Object, ByVal e As System.EventArgs) _
    Handles Button3.Click
        Dim nazv As String, n As Integer
        nazv = InputBox( _
        "Фрагмент названия искомой книги?")
        nf = FreeFile()
        FileOpen(nf, FileName, OpenMode.Random, _
        OpenAccess.ReadWrite)
5:      p = 1
        TextBox1.Clear()
        Do Until EOF(nf)
            FileGet(nf, t, p)
            n = InStr(StrConv(t.Название, _
            VbStrConv.Uppercase), StrConv(nazv, _
            VbStrConv.Uppercase))
10:         If n > 0 Then
                TextBox1.AppendText("Запись № " _
                & p & vbCrLf & t.Автор & _
                vbTab & t.Название & vbCrLf)
            End If
            p = p + 1
        Loop
15:     FileClose(nf)
    End Sub
```

К оглавлению

```
    Private Sub Button4_Click(ByVal sender As  _
    System.Object, ByVal e As System.EventArgs) _
    Handles Button4.Click
        Dim avt As String, n As Integer
        avt = InputBox( _
        "Фрагмент ФИО автора искомой книги?")
        nf = FreeFile()
        FileOpen(nf, FileName, OpenMode.Random, _
        OpenAccess.ReadWrite)
5:      p = 1
        TextBox1.Clear()
        Do Until EOF(nf)
            FileGet(nf, t, p)
            n = InStr(StrConv(t.Автор, _
            VbStrConv.Uppercase), StrConv(avt, _
            VbStrConv.Uppercase))
10:         If n > 0 Then
                TextBox1.AppendText("Запись № " _
                & p & vbCrLf & t.Автор & _
                vbTab & t.Название & vbCrLf)
            End If
            p = p + 1
        Loop
15:     FileClose(nf)
    End Sub
End Class
```

К оглавлению

## Вопросы для контроля

1. Какие типы доступа к файлам поддерживаются в VB.NET?
2. Какие типы доступа к файлу делают возможным замену отдельной записи без перезаписи всего файла?
3. Для чего применяется функция `FreeFile`?
4. Каково назначение функции `FileOpen`?
5. Каково назначение функции `FileClose`?
6. Каково назначение функции `PrintLine`?
7. Каково назначение функции `LineInput`?
8. Каково назначение функции `FilePut`?
9. Каково назначение функции `FileGet`?
10. Для чего применяется функция `EOF`?
11. Для чего применяется функция `Seek`?

К оглавлению