ФАЙЛЫ ПОСЛЕДОВАТЕЛЬНОГО ДОСТУПА

Архитектура 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. Файл состоит из записей. Запись характеризуется длиной, выраженной в байтах, и 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 последовательного доступа,
- файлы произвольного доступа,
- 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. 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.

Последовательный доступ к записям файла позволяют использовать все внешние устройства. 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 в 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.

1. 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()
```

2. Открытие файла

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

Чтобы иметь возможность использовать файл, его нужно открыть. Функция `FileOpen()`, выполняющая эту операцию, принимает множество аргументов, из которых обязательными являются только первые три.

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 чтения данных из файла. Текущей становится первая запись.
OpenMode.Output	Файл открывается только для записи данных в файл. Текущей становится первая запись.
OpenMode.Append	Файл открывается для добавления новых данных. Текущей становится следующая запись после последней записи файла.
OpenMode.Random	Файл открывается для произвольного доступа (чтения и записи, по одной записи за раз). Текущей становится первая запись.
OpenMode.Binary	Файл открывается как двоичный. Текущей становится первая запись.

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 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 открытия файла последовательного доступа для записи:

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

3. Закрытие файла

`FileClose(file_number)`

Функция `FileClose()` закрывает файл, номер которого передан ей в качестве аргумента. Так, следующая инструкция закрывает файл с номером `n1`:

```
FileClose(n1)
```

4. Последовательный доступ. Запись в файл и чтение из файла

`PrintLine(file_number, output_list)`

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

`LineInput(file_number)`

Для чтения очередной записи данных из файла с последовательным доступом используется функция `LineInput()`. 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()`.

5. Функции EOF и LOF

EOF(file_number)

LOF(file_number)

Эти две функции используются при работе с файлами постоянно.

Функция `EOF()` принимает в качестве аргумента номер открытого файла и возвращает `True`, если достигнут его конец. 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. Функция `EOF` (End Of File) возвращает значение `True` при достижении конца файла. На каждом шаге цикла считывается отдельная строка и к ней добавляется символ конца строки, который отбрасывает инструкция `LineInput`.

LOF(file_number)

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),
```

где `Len(record)` – это длина записи в байтах.

6. Проект «Файл последовательного доступа»

Проект, который Вы разработаете, позволит:

– создать файл последовательного доступа;

– открыть существующий файл последовательного доступа и отобразить его содержание в текстовом поле;

– редактировать информацию в текстовом поле;

– записать в файле содержание текстового поля.

Приступите к разработке проекта.

1. Создайте новый проект с именем ФайлПоследовательногоДоступа, следуя приложению 1.

2. Разместите на форме текстовое поле `TextBox1` и три кнопки `Button` так, чтобы вид формы соответствовала рис. 1.

Рис. 1. Элементы управления формы

3. Установите значения свойств элементов управления, чтобы интерфейс проекта соответствовал рис. 2.

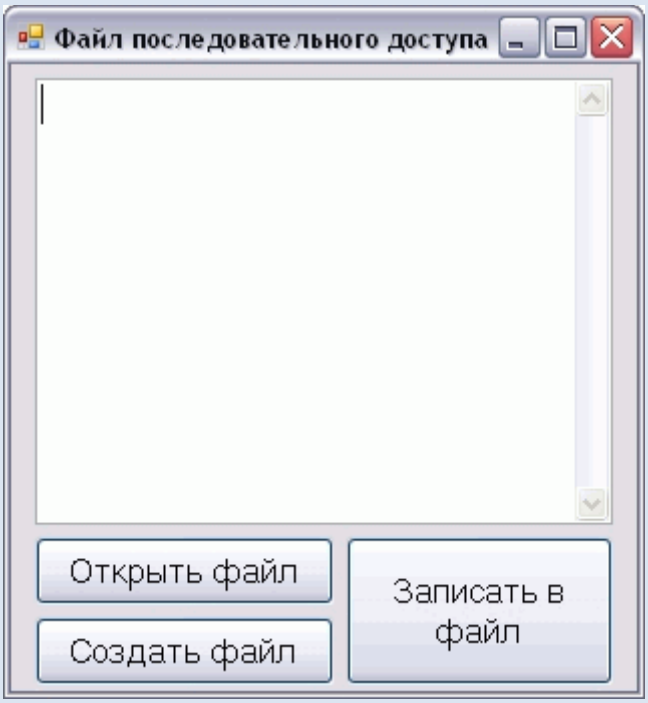

Рис. 2. Интерфейс проекта

4. До объявления первой процедуры в форме поместите объявление переменных, которые будут действовать во всех процедурах формы:

```
Dim ИмяФайла As String, k As Integer
```

5. Приступите к созданию кода подпрограммы `Button1_Click`. Нажатие на эту кнопку откроет существующий файл последовательного доступа и отобразит в текстовом поле `TextBox1` содержащиеся в нем записи. Дважды щелкните на кнопке `Button1` и в созданную системой заготовку подпрограммы `Button1_Click` введите код тела этой подпрограммы (конечно без начальной и конечной строки, которые были созданы системой):

```
Private Sub Button1_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles Button1.Click
    Dim s As String
    s = ""
```

```
        TextBox1.Clear()
        ИмяФайла = InputBox( _
        "Введите полное имя файла." & _
        "   Пример: d:\ИмяРабочейПапки\ИмяФайла")
5:      If ИмяФайла = "" Then
            MsgBox("Не задано имя файла! Повторите!")
            Exit Sub
        End If
        k = FreeFile()
10:     Try
            FileOpen(k, ИмяФайла, OpenMode.Input)
        Catch
            MsgBox("Ошибка при открытии файла! Повторите!")
            ИмяФайла = ""
15:         Exit Sub
        End Try
        Do Until EOF(k)
            s = LineInput(k)
            TextBox1.AppendText(s & vbCrLf)
20:     Loop
        FileClose(k)
    End Sub
```

Инструкция 4 запрашивает полное имя открываемого файла. Если при этом имя файла не будет задано, то будет выведено сообщение об ошибке и работа подпрограммы `Button1_Click` досрочно завершается (строки 5 – 8). Инструкция 9 определяет номер свободного канала ввода-вывода. Файл открывает для чтения инструкция 11. Если имя файла задано неправильно, то при открытии файла возникнет исключение, перехват которого предусматривают инструкции 10 – 16. При возникновении исключения выводится сообщение об ошибке при открытии файла (строка 13), отменяется введенное имя файла (строка 14) и подпрограмма досрочно завершается (строка 15). Если же исключение не возникло, то вся информация из файла считывается в текстовое поле `TextBox1` (строки 17 – 20), и файл закрывается (строка 21).

6. Дважды нажмите на кнопке `Button2`. Введите код тела подпрограммы `Button2_Click`, которая создает новый файл последовательного доступа и выполняет очистку текстового поля `TextBox1`:

```
    Private Sub Button2_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles Button2.Click
        TextBox1.Clear()
        Dim bc As Short
        bc = MsgBox("ОК - файл будет создан заново. " & _
        "Имеющееся содержание будет потеряно! " _
        & "Отмена - прекратить операцию.", _
        MsgBoxStyle.OkCancel, "Предупреждение !")
        If bc = 1 Then
5:          ИмяФайла = InputBox( _
            "Введите полное имя файла." & _
            "   Пример: d:\ИмяРабочейПапки\ИмяФайла")
            If ИмяФайла = "" Then
                MsgBox("Не задано имя файла! Повторите!")
                Exit Sub
            End If
10:         k = FreeFile()
```

```
            Try
                FileOpen(k, ИмяФайла, OpenMode.Output)
            Catch
                MsgBox("Ошибка при открытии файла! Повторите!")
15:             ИмяФайла = ""
                Exit Sub
            End Try
            FileClose(k)
        End If
    End Sub
```

Если создаваемый файл уже существует, то при его открытии в режиме `OpenMode.Output` все содержимое файла удаляется. Это опасно, а система об этом не предупреждает. Инструкция 3 кода специально добавлена для предупреждения об опасности потери содержимого существующего файла. В инструкции 3, которая занимает четыре строки, второй аргумент `MsgBoxStyle.OKCancel` функции `MsgBox` задает стиль окна этой функции с двумя кнопками (`OK` и `Отменить`). Значение функции равно `1`, если окно этой функции закрыто кнопкой `OK` и равно `2` при его закрытии иным способом. И только при закрытии окна предупреждения функции `MsgBox` кнопкой `OK`, будет запрошено имя создаваемого файла (строка 5). В строках 6 – 9 выполняется проверка, чтобы заданное имя файла не являлось пустым. В строке 10 определяется номер свободного канала. При открытии файла (строка 12) предусмотрена обработка исключения (строки 11 – 17), которое может возникнуть при указании некорректного пути к файлу (например, указана папка, которая не существует). Если создаваемого с заданным именем файла не существует, то он будет создан. После успешного открытия файла он закрывается (строка 18).

7. Дважды нажмите на кнопке `Button3`. Введите код тела подпрограммы `Button3_Click`. Выполнение этой подпрограммы, приводит к открытию файла последовательного доступа и к записи в него всей информации, содержащейся в текстовом поле `TextBox1`:

```
    Private Sub Button3_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles Button3.Click
        Dim bc As Short
        bc = MsgBox("Сохранение изменений в файле! " & _
        "OK - Содержанием файла станет содержанием" & _
        " текстового поля. Отмена - прекратить операцию.", _
        MsgBoxStyle.OkCancel, "Предупреждение !")
        If bc = 1 Then
            If ИмяФайла = "" Then
5:              MsgBox( _
                "Сначала нужно создать или открыть файл!")
            Else
                k = FreeFile()
                FileOpen(k, ИмяФайла, OpenMode.Output)
                Try
10:                 PrintLine(k, TextBox1.Text)
                Catch
                    MsgBox("Ошибка при записи в файл!")
                End Try
                FileClose(k)
15:         End If
        End If
    End Sub
```

Запись в файл содержимого текстового поля выполняется инструкцией 10.

8. Проверьте работу проекта. Для этого запустите выполнение проекта. С помощью кнопки `Создать файл` создайте в своей рабочей папке файл с именем `f1.txt`. Запишите в текстовом поле несколько строк. С помощью кнопки `Записать в файл` запишите содержание текстового поля в файл `f1.txt`. Остановите выполнение проекта. Для проверки правильности работы проекта вновь запустите выполнение проекта. С помощью кнопки `Открыть файл` откройте только что созданный файл и убедитесь, что информация в нем сохранилась.

9. Добавьте на форму кнопку `Button4` и надпись `Label1`. Подпрограмма `Button4_Click`, код которой Вам необходимо составить самостоятельно, должна:

– открыть для записи существующий файл последовательного доступа или создать для записи новый файл последовательного доступа;

– поочередно генерировать `n` случайных чисел с помощью функции `Rnd` и записать каждое из них в открытый на предыдущем шаге файл последовательного доступа;

– закрыть файл последовательного доступа, содержащий n случайных чисел;

– открыть для чтения этот файл последовательного доступа с записанными в нем `n` случайными числами, прочитать их все из файла с одновременным вычислением (**не применяя массива**) их суммы и отображением их в текстовом поле `TextBox1`;

– вычислить среднее значение полученной последовательности `n` случайных величин и отобразить этот результат в надписи `Label1` в виде:

`Среднее = ПолученноеЗначение`.

10. Сохраните проект, проверьте его работу, задав n = 10000, и продемонстрируйте его работу преподавателю.

7. 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. Для чего применяется функция `EOF`?
9. Как можно изменить одну запись в файле последовательного доступа?