

Работа с файлами

Общие сведения о работе с файлами .	1
Основные операции при работе с файлами .	2
Обзор основных операций.	2
Объявление файла	2
Связь файловой переменной с конкретным файлом	3
Открытие файла	3
Чтение и запись в файл.....	4
Закрытие файла.	4
Ввод из файла .	5
Чтение чисел	5
Чтение строк	6
Конец файла.	7
Примеры программ.	7
Пример1: Запись и добавление данных в файл.	7
Пример2: Простая база данных «Погода»	9
Пример 3: чтение и редактирование файла	10
Контрольные вопросы.....	12

Литература:

1. Алексеев Е.Р., Чеснокова О.В., Кучер Т.В. Free Pascal и Lazarus : Учебник по программированию. – М.: Alt Linux, Изд. дом «ДМК-пресс», 2010. – 440 с.
2. Мансуров К.Т. Основы программирования в среде Lazarus, 2010. – 772 с.: ил. ISBN 978-9967-03-646-8

Общие сведения о работе с файлами

Программы, которые до настоящего момента рассматривались, выводили результат своей работы на экран. Вместе с тем, Lazarus позволяет со-хранить результаты работы программы на диске компьютера, в файле, что дает возможность использовать эти данные для дальнейшей обработки, ми-нуя процесс их ввода с клавиатуры.

Файл — это именованная структура данных, представляющая собой последовательность элементов данных одного типа,

причем количество элементов последовательности практически не ограничено. В первом приближении файл можно рассматривать как массив переменной длины неограниченного размера.

С точки зрения программиста, все файлы можно разделить на три класса:

- типизированные;
 - бестиповые;
-

- **текстовые.**

Файлы, состоящие из компонентов одного типа (целые, вещественные, массивы и т.д.), число которых заранее не определено и может быть любым, называются типизированными. Они заканчиваются специальным символом «конец файла», хранятся в двоичном виде, содержимое подобных файлов нельзя просмотреть обычным текстовым редактором, для просмотра подобных файлов нужно писать специальную программу.

В бестиповых файлах информация считывается и записывается блоками определенного размера. В подобных файлах хранятся данные любого вида и структуры.

Текстовые файлы состоят из любых символов. При записи информации в текстовый файл все данные преобразуются к символьному типу, в котором и хранятся. Просмотреть данные в подобном файле можно с помощью любого текстового редактора. Информация в текстовом файле хранится построчно. В конце каждой строки хранится специальный символ «конец строки». Конец самого файла обозначается символом «конец файла».

Основные операции при работе с файлами

Обзор основных операций

Для работы с файлами в программе необходимо выполнить следующие действия:

- ☐ объявить файловую переменную;
- ☐ установить связь между файловой переменной и конкретным файлом;
- ☐ открыть файл для чтения/записи/добавления в конец
- ☐ выполнить необходимые действия с файлом (прочитать данные, стереть предыдущие данные и записать новые, добавить данные в конец файла);
- ☐ закрыть файл.

Объявление файла

Как и любая структура данных (переменная, массив) программы, файл должен быть объявлен в разделе описания переменных. При объявлении файла указывается тип элементов файла.

В общем виде синтаксис объявления файла имеет вид:

Имя:`file of ТипЭлементов;`

Примеры:

```
res: file of char; // файл символов
koef: file of real; // файл вещественных чисел
f: file of integer; // файл целых чисел
```

Файл, компонентами которого являются данные символьного типа, называется *символьным*, или *текстовым*.

Описание текстового файла в общем виде выглядит так:

Имя: `TextFile`;

где: **имя** — имя файловой переменной; **TextFile** — обозначение-тип, показывающее, что **Имя** — это файловая переменная, представляющая текстовый файл.

Связь файловой переменной с конкретным файлом

Объявление файловой переменной задает только тип компонентов файла.

Для того чтобы программа могла выводить данные в файл или считывать данные из файла, необходимо указать конкретный файл.

Для работы с конкретным файлом необходимо связать файловую переменную с этим файлом.

Имя файла задается вызовом процедуры `AssignFile`, связывающей файловую переменную с конкретным файлом.

Описание процедуры `AssignFile` выглядит следующим образом:

`AssignFile (var f, ИмяФайла: string)`

Имя файла задается согласно принятым в Windows правилам. Оно может быть полным, т. е. состоять не только непосредственно из имени файла, но и включать путь к файлу (имя диска, каталогов и подкаталогов).

Ниже приведены примеры вызова процедуры `AssignFile`:

```
AssignFile(f, 'a:\result.txt');  
AssignFile(f, '\students\ivanov\korni.txt');  
fname:=('otchet.txt'); AssignFile (f,fname);
```

Если в проекте Lazarus используется компонент `OpenDialog`, то связь файловой переменной с конкретным файлом имеет следующий вид:

```
if OpenDialog1.Execute then begin  
    fname:=OpenDialog1.FileName;  
    AssignFile(f,fname);  
end;
```

В этом фрагменте вначале выполняется диалог открытия файла. Если диалог завершился успешно (файл выбран), то свойство **Execute** становится равным **True**, а свойство **Filename** содержит имя выбранного файла.

Открытие файла

Перед чтением/записью в файл его необходимо открыть.

Для чтения из файла используется процедура `Reset`:
`Reset (f) ;`

Перед вызовом процедуры **Reset** с помощью функции **AssignFile** файловая переменная должна быть связана с конкретным файлом.

Например, следующие инструкции открывают файл для ввода:

```
AssignFile(f, 'c:\data.txt'); Reset(f);
```

Если программа, формирующая выходной файл, уже использовалась, то возможно, что файл с результатами работы программы уже есть на диске. Поэтому программист должен решить, как поступить со старым файлом: заменить старые данные новыми или новые данные добавить к старым. Способ использования старого варианта определяется во время открытия файла.

Возможны следующие режимы открытия файла для записи в него данных:

перезапись (запись нового файла поверх существующего или создание нового файла):

```
Rewrite(f);
```

добавление в конец существующего файла (только **text**):

```
Append(f);
```

Чтение и запись в файл

Для чтения из текстового файла используются процедуры **read** и **readln**:

```
read(f, a); { чтение из файла f }
```

```
readln(f, s); { чтение из текстового файла }
```

Непосредственно вывод в текстовый файл осуществляется при помощи инструкции **write** или **writeln**.

В общем виде эти инструкции записываются следующим образом:

```
write (файловаяПеременная, СписокВывода) ;
```

```
writeln(файловаяПеременная, СписокВывода);
```

где: **файловаяПеременная** — переменная, идентифицирующая файл, в который выполняется вывод; **СписокВывода** -- разделенные запятыми имена переменных, значения которых надо вывести в файл. Помимо имен переменных в список вывода можно включать строковые константы.

Например, если переменная **f** является переменной типа **Text**, то инструкция вывода значений переменных **x1** и **x2** в файл может быть такой:

```
write(f, 'Корни уравнения: ', x1, x2);
```

Различие между инструкциями **write** и **writeln** состоит в том, что инструкция **writeln** после вывода всех значений, указанных в списке вывода, записывает в файл символ "новая строка".

Заккрытие файла

Перед завершением работы программа должна закрыть все открытые файлы. Это делается вызовом процедуры **Close**.

Процедура `Close` имеет один параметр — имя файловой переменной.
Пример использования процедуры:
`Close(f);`

Ввод из файла

Программа может вводить исходные данные не только с клавиатуры, но и из текстового файла. Для того чтобы воспользоваться этой возможностью, нужно объявить файловую переменную типа `TextFile`, назначить ей при помощи инструкции `AssignFile` имя файла, из которого будут считываться данные, открыть файл для чтения (ввода) и прочитать (ввести) данные, используя инструкцию `read` или `readln`.

Чтение чисел

Следует понимать, что в текстовом файле находятся не числа, а их изображения.

Действие, выполняемое инструкциями `read` или `readln`, фактически состоит из двух:
сначала из файла читаются символы до появления разделителя (пробела или конца строки), затем прочитанные символы, являющиеся изображением числа, преобразуются в число, и полученное значение присваивается переменной, имя которой указано в качестве параметра инструкции `read` ИЛИ `readln`.

Например, если текстовый файл `data.txt` содержит следующие строки:

```
23 15
45 28
56 71
```

то в результате выполнения инструкций:

```
AssignFile(f, 'data.txt'); Reset(f);
read(f, a); read(f, b, c); read(f, d);
```

значения переменных будут следующими: `a=23`, `b=15`, `c=45`, `d=28`.

Отличие инструкции `readln` от `read` состоит в том, что

При использовании процедуры `readln` после считывания из файла значения последней переменной в списке указатель чтения из файла автоматически перемещается в начало следующей строки файла, даже в том случае, если за прочитанным числом есть еще числа.

Поэтому в результате выполнения инструкций

```
AssignFile(f, 'data.txt'); Reset(f);
readln(f, a); readln(f, b, c); readln(f, d);
```

значения переменных будут следующими: `a=23`, `b=45`, `c=28`, `d=56`.

Если при чтении значения численной переменной в файле вместо изображения числа будет какая-то другая последовательность символов, то произойдет ошибка.

Чтение строк

В программе строковая переменная может быть объявлена с указанием длины или без нее.

Например:

```
stroka1:string[10]; stroka2:string;
```

При чтении из файла значения строковой переменной, длина которой явно задана в ее объявлении, считывается столько символов, сколько указано в объявлении, но не больше, чем в текущей строке.

При чтении из файла значения строковой переменной, длина которой явно не задана в объявлении переменной, значением переменной становится оставшаяся после последнего чтения часть текущей строки. Другими словами, если надо прочитать из файла всю строку, то объявите строковую переменную, длина которой заведомо больше самой длинной строки файла, и считывайте строки в эту переменную.

Если одной инструкцией **readln** осуществляется ввод нескольких, например, двух переменных, то первая переменная будет содержать столько символов, сколько указано в ее объявлении или, если длина не указана, всю строку файла. Вторая переменная будет содержать оставшиеся символы текущей строки или, если таких символов нет, не будет содержать ни одного символа (длина строки равна нулю).

Пусть, например, текстовый файл `friends.txt` содержит строки:

Косичкина Маша
Васильев Антон
Цой Лариса

В табл. 1 приведено несколько вариантов объявления переменных, инструкции чтения из файла `friends.txt` и значения переменных после выполнения инструкций чтения.

Таблица 1. Примеры чтения строк из файла

Объявления	Инструкция чтения переменных из файла	Значение переменных после чтения из файла
fam: string[15] name:string[10]	Readln(f,fam,name)	fam='Косичкина' name= 'Маша'
fam, name: string;	Readln(f,fam,name)	fam= 'Косичкина Маша' name= ''
drug: string[80]	Readin (f, drug)	drug = 'Косичкина Маша'

Конец файла

Пусть на диске есть некоторый текстовый файл. Нужно в диалоговое окно вывести содержимое этого файла. Решение задачи довольно очевидно: надо открыть файл, прочитать первую строку, затем вторую, третью и т. д. до тех пор, пока не будет достигнут конец файла. Но как определить, что прочитана последняя строка, достигнут конец файла?

Для определения конца файла можно воспользоваться функцией **EOF** (End of File — конец файла).

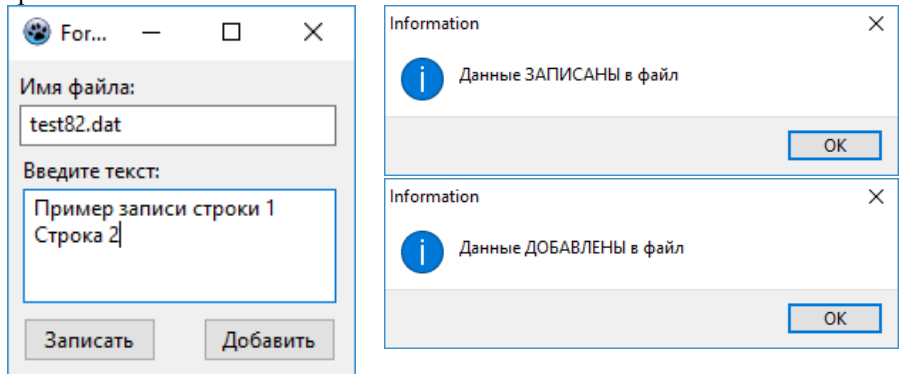
У функции **EOF** один параметр — файловая переменная. Значение функции **EOF** равно **False**, если прочитанный элемент данных не является последним в файле, т. е. возможно дальнейшее чтение. Если прочитанный элемент данных является последним, то значение **EOF** равно **True**.

Значение функции **EOF** можно проверить сразу после открытия файла. Если при этом оно окажется равным **True**, то это значит, что файл не содержит ни одного элемента данных, т. е. является пустым (размер такого файла равен нулю).

Примеры программ

Пример1: Запись и добавление данных в файл

В качестве примера рассмотрим пример программы, которая выполняет запись или добавление строки из поля Мемо в текстовый файл. Окно программы показано ниже.



В приложении объявлены переменные:

```
f: TextFile; // файл  
fName: String[80]; // имя файла
```

Сценарий работы приложения: после ввода текста в поле Мемо щелкаем на кнопке «Записать», данные будут записаны в текстовый файл (файл открывается процедурой **Rewrite**). Затем вводим новый текст и щелкаем на

кнопке «Добавить» - файл открывается процедурой Append, и данные будут дописаны в файл.

Исходный текст обработчика события –щелчок на кнопке Записать – приведен в листинге 1. Процедура открывает файл в режиме создания нового или замещения существующего файла и записывает текст, находящийся в поле компонента Memo1. Имя файла нужно ввести во время работы в поле Edit1. Можно задать predetermined имя файла во время разработки формы приложения. Для этого надо присвоить значение, например **test.txt**, свойству Edit1.Text.

Листинг 1. Создание нового или замещение существующего файла

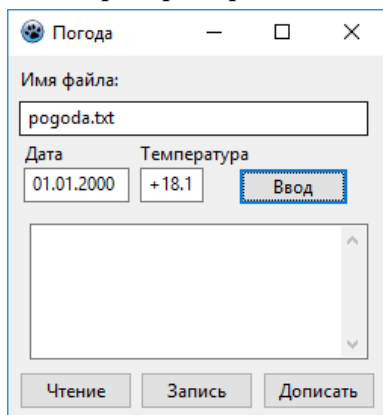
```
procedure TForm1.Button1Click(Sender: TObject);
var
  f: TextFile; // файл
  fName: String[80]; // имя файла
  i: integer;
begin
  fName := Edit1.Text;
  AssignFile(f, fName);
  Rewrite(f); // открыть для перезаписи
  // запись в файл (строки нумеруются с нуля)
  for i:=0 to Memo1.Lines.Count do
    writeln(f, Memo1.Lines[i]);
  CloseFile(f); // закрыть файл
  MessageDlg('Данные ЗАПИСАНЫ в файл', mtInformation,
    [mbOk], 0);
end;
```

В листинге 2 приведена процедура, которая запускается нажатием ко-мандной кнопки **Добавить**. Она открывает файл, имя которого указано в поле Edit1, и добавляет в него содержимое поля Memo1.

Листинг 2. Добавление в существующий файл

```
procedure TForm1.Button2Click(Sender: TObject);
var
  f: TextFile; // файл
  fName: String[80]; // имя файла
  i: integer;
begin
  fName := Edit1.Text;
  AssignFile(f, fName);
  Append(f); // открыть для добавления
  // запись в файл строки нумеруются с нуля
  for i:=0 to Memo1.Lines.Count do
    writeln(f, Memo1.Lines[i]);
  CloseFile(f); // закрыть файл
  MessageDlg('Данные ДОБАВЛЕНЫ в файл ', mtInformation,
    [mbOk], 0);
end;
```


Пример2: Простая база данных «Погода»



Следующая программа ведет простую базу данных.

В поля «Дата» и «Температура» пользователь вводит дату и среднесуточную температуру. После щелчка на кнопке «Ввод» данные записываются в поле Мемо.

При щелчке на кнопке «Чтение» в поле Мемо отображается содержимое файла. При щелчке на кнопке «Запись» все содержимое Мемо записывается в файл, а при щелчке на кнопке «Дописать» – добавляется в файл.

В приведенном ниже листинге отсутствует обработчик щелчка на кнопке «Дописать»: обучающимся предлагается сделать это самостоятельно.

Листинг 3 Простая база данных «Погода»

implementation

```
{ $R *.lfm }
var
  date: string[10]; temp: string[5];
  f: TextFile; // файл
  fName: String[80]; // имя файла

{ TForm1 }

procedure TForm1.Button2Click(Sender: TObject);
var i: integer;
begin
  fName := Edit1.Text; AssignFile(f, fName);
  Rewrite(f); // открыть для перезаписи
  // запись в файл (строки нумеруются с нуля)
  for i:=0 to Memo1.Lines.Count do
    writeln(f, Memo1.Lines[i]);
  CloseFile(f); // закрыть файл
  MessageDlg('Данные ЗАПИСАНЫ в файл', mtInformation,
    [mbOk], 0);
end;

procedure TForm1.Button1Click(Sender: TObject);
var buf: string;
begin
  fName := Edit1.Text; AssignFile(f, fName);
  {$I-} // отмена проверки результатов ввода-вывода
  Reset(f);
  {$I+} // включение проверки результатов ввода-вывода
```

```

        if IOResult<>0 then begin
            MessageDlg('Ошибка открытия файла#13'+Edit1.Text,
                mtInformation, [mbOk],0);
            exit;
        end;
        while not EOF(f) do begin
            readln(f, buf); // прочитать строку из файла
            if Length(buf)>0 then
                Memo1.Lines.Add(buf); // добавить строку в поле Memo1
        end;
        CloseFile(f);
    end;

    procedure TForm1.Button3Click(Sender:
    TObject); begin
        date:=Edit2.Text; temp:=Edit3.Text;
        Memo1.Lines.Add(date+' '+temp);
    end;

    procedure TForm1.Button4Click(Sender:
    TObject); var i: integer;
    begin
        fName := Edit1.Text; AssignFile(f, fName);
        {$I-}
        Append(f);
        {$I+}
        if IOResult<>0 then begin
            MessageDlg('Ошибка открытия файла#13'+Edit1.Text,
                mtInformation, [mbOk],0);
            exit;
        end;
        for i:=0 to Memo1.Lines.Count-1 do
            writeln(f, Memo1.Lines[i]);
        CloseFile(f);
    end;

end.

```

После нескольких запусков программы файл `pogoda.txt` может быть, например, таким:

```

22.12.2013 +5
03.03.2014 +12
04.03.2014 +15.4

```

Пример 3: чтение и редактирование файла

В листинге 4 приведена процедура, которая выполняет поставленную задачу. Она читает строки из файла, имя которого ввел пользователь во время работы программы, и выводит эти строки в поле Мемо. Окно программы имеет вид:

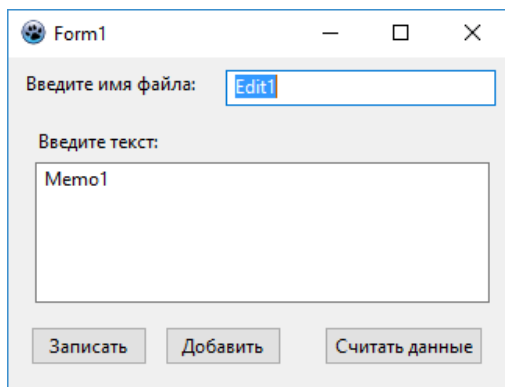


Рис. 6. Окно программы Чтение и изменение файла

Листинг 4. Чтение и изменение файла

```

procedure TForm1.Button1Click(Sender:
TObject); var
  f: TextFile; fName: String[80]; // имя файла
  i: integer;
begin
  fName := Edit1.Text; AssignFile(f, fName);
  Rewrite(f); // открыть для перезаписи
  // запись в файл (строки нумеруются с нуля)
  for i:=0 to Memo1.Lines.Count-1 do
    writeln(f, Memo1.Lines[i]);
  CloseFile(f); // закрыть файл
  MessageDlg('Данные ЗАПИСАНЫ в файл', mtInformation,
    [mbOk],0);
end;

procedure TForm1.Button2Click(Sender: TObject);
var
  f: TextFile; fName: String[80]; // имя файла
  i: integer;
begin
  fName := Edit1.Text; AssignFile(f, fName);
  Append(f); // открыть для добавления
  // запись в файл строки нумеруются с нуля
  for i:=0 to Memo1.Lines.Count-1 do
    writeln(f, Memo1.Lines[i]);
  CloseFile(f); // закрыть файл
  MessageDlg('Данные ДОБАВЛЕНЫ в файл ', mtInformation,
    [mbOk],0);
end;

procedure TForm1.Button3Click(Sender:
TObject); var
  f: TextFile; fName: String[80]; // имя файла

```

```

    s:string;
begin
    fName := Edit1.Text; AssignFile(f, fName);
    Reset(f); // открыть для чтения
    Memo1.Clear;
    Memo1.Lines.Add('файл '+fName+' открыт');
    // запись в файл строки нумеруются с нуля
    while not Eof(f) do begin
        readln(f, s); Memo1.Lines.Add(s);
    end;
    CloseFile(f);
    MessageDlg('Данные СЧИТАНЫ ', mtInformation,
        [mbOk], 0);
end;

```

Для организации обработки файла использована инструкция цикла **while**, которая обеспечивает проверку значения функции **EOF** перед каждым чтением, в том числе и перед первым.

Наличие кнопки Записать и соответствующей ей процедуры позволяет сохранить содержимое поля Мемо в файле, т. е. программа чтение из файла представляет собой примитивный редактор текста.

Добавление очередной прочитанной из файла строки в поле Мемо выполняется применением метода **Add** к свойству **Lines**.