

Работа с файлами в паскале

Виды файлов в зависимости от их описания и режимом работы

1. текстовые (тип **text**) файлы со строками неопределенной длины;
2. файлы с типом записей (двоичные или типизированные (**file of**));
3. файлы без типа для передачи данных блоками записей нетипизированные (**file**).

Описание файловых переменных:

var

f1: file of char; {типизированный файл}
f2: file of integer; {типизированный файл}
f3: file; {нетипизированный файл}
f: text; {текстовый файл}

Для связи файла в коде программы и действительного файла на внешнем носителе используется процедура **ASSIGN**:

assign(myfile,'c:\text.txt');

где **myfile** — имя переменной (объявленной ранее в области **var**), ассоциированной с файлом

c:\text.txt — путь к реальному файлу

Первый аргумент процедуры **assign** в паскаль — переменная, второй – путь к файлу на диске.

Для считывания из файла достаточно связать поток ввода с файлом:

Считывание строки	Считывание массива из N целых
begin Assign(input,'24.txt'); var s := ReadString; ... end.	begin Assign(input,'26.txt'); var N := ReadInteger; var a := ReadArrInteger(N); ... end.

Текстовые файлы в паскале: процедуры работы

Текстовый файл в Паскале — это совокупность строк произвольной длины, которые разделены между собой метками конца строки, а весь файл заканчивается меткой конца файла.

Важно: Если быть точными, то каждая строка текстового файла завершается специальной комбинацией, называемой «конец строки».

Комбинация «конец строки» состоит из двух символов: перевод каретки (ASCII-код #13) и перевод строки (#10). Завершается текстовый файл символом конца файла (#26).

Возможные расширения файлов: *.txt, *.log, *.htm, *.html

Метод работы с текстовым файлом в Паскале предусматривает лишь последовательный доступ к каждой строке файла. Это означает, что начинать всегда возможно только с первой строки, затем проходя по каждой строке, дойти постепенно до необходимой. Т.е. можно сказать, что чтение (или запись) из файла (в файл) ведутся байт за байтом от начала к концу.

Предусмотрены два режима работы: режим для записи в файл информации и для чтения ее из файла. Одновременная запись и чтение запрещены.

ОТКРЫТИЕ ФАЙЛА (КЛАССИЧЕСКИЙ PASCAL)

Допустим, мы в программе описали переменную для работы с текстовым файлом:

```
var f: text;
```

Рассмотрим дальнейшую последовательность работы с ним, и рассмотрим процедуры, необходимые для работы с текстовым файлом в Паскале:

процедура открытия существующего файла для чтения при последовательном доступе:

```
Reset (f);
```

процедура открытия создаваемого файла для записи в него информации; если файл с таким именем уже существует, то информация в нем стирается:

```
Rewrite (f);
```

процедура добавления в конец:

```
Append (f);
```

- При открытии курсор устанавливается в начало файла.

ЧТЕНИЕ ИЗ ФАЙЛА (КЛАССИЧЕСКИЙ PASCAL)

```
Read (f, список переменных);
```

```
ReadLn (f, список переменных);
```

Отличие ReadLn от Read в том, что при использовании readln после прочтения данных пропускаются все оставшиеся символы в данной строке, включая метку конца строки.

- чтение осуществляется с той позиции, где в данный момент стоит курсор;
- после чтения курсор сдвигается к первому непрочитанному символу.
- Чтение до конца файла: оператор EOF (end of file).

- Чтение до конца строки: оператор EOL (end of line).
- Для возврата в начало файла:

close (f);

reset (f); { начинаем с начала }

ЗАПИСЬ В ТЕКСТОВЫЙ ФАЙЛ (КЛАССИЧЕСКИЙ PASCAL)

Write (f, список переменных);

WriteLn (f, список переменных);

где f — файловая переменная, а второй параметр – выводимые из программы и вводимые в файл данные (в виде значений переменных или просто данные)

ПРОЦЕДУРЫ РАБОТЫ С ФАЙЛОМ И ЗАКРЫТИЕ ФАЙЛА

Нахождение конца файла:

Eof(f);

Логическая функция, возвращающая True, если достигнут конец файла.

Нахождение конца строки:

eoln(f);

Логическая функция, возвращающая True, если достигнут конец строки.

Удалить файл в Паскале

erase(переменная_файла);

Переименование файла в Паскале

rename(переменная_файла, 'новое имя файла');

Закрытие:

Close (f); { закрытие файла }

Важно: Таким образом, работа с файлом осуществляется через три основных шага:

1. Процедура assign.
2. Процедура reset или rewrite.
3. Процедура close.

Рассмотрим пример работы с файлами в паскале:

Пример 1:

В файле text.txt записаны строки. Вывести первую и третью из них на экран. (предварительно создать text.txt с тремя строками)

Решение:

```
var
```

```
    filetext: text;
```

```
    a,b,c:string;
```

```
begin
```

```
assign(filetext,'c:\text.txt');
```

```
reset(filetext);
```

```
readln(filetext,a);
```

```
readln(filetext,b);
```

```
readln(filetext,c);
```

```
close(filetext);
```

```
writeln(a);
```

```
writeln(c);
```

end.

Пример 2:

Считать из файла input.txt числа (числа записаны в столбик). Затем записать их произведение в файл output.txt

Решение:

```
var p, x: integer;  
    f: text;  
begin  
  assign(f, 'input.txt');  
  reset(f);  
  p := 1;  
  while not eof(f) do begin  
    readln(f, x);  
    p := p * x;  
  end;  
  close(f);  
  assign(f, 'output.txt');  
  rewrite(f);  
  writeln(f, 'Произведение чисел ', p);  
  close(f);  
end.
```

РАБОТА С ДАННЫМИ ИЗ ФАЙЛА КАК С МАССИВОМ

Пример 3: В файле input.txt записаны числа (каждое — с новой строки), их количество не превышает 100. Необходимо отсортировать их по возрастанию и записать в файл output.txt.

Трудности:

- для сортировки необходим массив, для того чтобы одновременно работать со всеми числами;
- неизвестно общее количество чисел.

Алгоритм решения:

- объявляем массив для 100 элементов;
- открываем файл на чтение, просчитываем количество чисел, заполняя массив, сохраняем количество в N;
- сортируем N элементов массива;
- записываем результат в файл.

Фрагмент решения:

```
{ Определяем глобальные переменные: }  
var A: array[1..100] of integer;  
    f: text;  
    N, i: integer;  
{ Определяем функцию, считывающую числа из файла, и записывающую их  
в массив. Функция возвращает кол-во элементов массива: }  
function ReadFromFile: integer;
```

```
var i: integer;
begin
  assign(f, 'input.txt');
  ...; { открытие файла в режиме чтения }
  i := 0;
  while (...) and (...) do begin
    i := i + 1;
    readln(...,...);
  end;
  close(f);
  ReadFromFile := i;
end;
{ Основная программа }
Begin
  N := ReadFromFile ;
  { сортировка N элементов по возрастанию }
  { ... }
  { запись отсортированного массива в файл: }
  assign(..., ...);
  ...; { открытие файла в режиме записи }
  for i:=1 to N do
    writeln(..., ...);
  close(f);
end.
```