Занятие 4

Операции ввода/вывода

Особенность файлов:

- Каждому файлу при создании указывается имя, по которому обрабатывающая его программа может отличить один файл от другого. Одна программа может работать одновременно с несколькими файлами.
- Файл содержит элементы только одного типа или тип его компонентов не оговаривается
- Длина файла – это число его элементов. При создании файла длина файла не задается заранее и ограничивается только емкостью устройств внешней памяти.

Виды файлов в Pascal

В Pascal имеются три вида файлов:

- текстовый файл (определяется типом text);
- типизированный файл (задается предложением file of Тип);
- не типизированный файл (определяется типом file).

Описание файловой переменной

- type
  ИмяТипа1=text;
  ИмяТипа2=file of Тип;
  ИмяТипа3= file;
- Var
  ИмяПеременной1: ИмяТипа1;
  ИмяПеременной2: ИмяТипа2;
  ИмяПеременной3: ИмяТипа3;

или

- Var

  ИмяПеременной1= text;

  ИмяПеременной2=file of;

  ИмяПеременной3=file;
- Например:

  type fak= text; {файловый тип}

  var a, b, x: fak, {файловые переменные}
- или

  var a, x: text; c: file;

Виды файлов

- **Файлом последовательного доступа** называется файл, к элементам которого доступ выполняется в той же последовательности, в какой они записались. Для таких файлов запрещено одновременно читать и записывать данные в файл.
- **Файл прямого доступа** – это файл, доступ, к элементам которого осуществляется по адресу (номеру). При поиске нужного элемента достаточно указать номер его позиции, что существенно ускоряет поиск. Для файлов прямого доступа разрешается одновременная запись и считывание данных.

Операции при работе с файлами

При работе с файлами программа должна провести следующие операции:

- открыть файл;
- чтение файла;
- закрыть файл.

Процедуры и функции, применимые для файлов любых типов

1. **AssignFile;**
2. **Reset**;
3. **Rewrite**;
4. **Close**;
5. **Rename**;
6. **Erase**;
7. **Ioresult**;
8. **Eof**.

Процедура AssignFile

- Для того, чтобы программа могла выводить данные в файл или считывать данные из файла необходимо указать конкретный файл, то есть связать файловую переменную с конкретным файлом.
- Описание процедуры AssignFile выглядит следующим образом:
- ***AssignFile (ФайловаяПеременная, ИмяФайла);***
- Имя файла должно представлять собой выражение строкового типа.

- *Примеры:*
- AssignFile (f, ‘a:\result.txt’);
- AssignFile (f, ‘c:\students\ivanov\korni.txt’);
- fname: =(‘otchet.txt’);
- AssignFile (f, fname);

Процедура **reset**

- Процедура **reset** открывает файл для ввода (чтения) и ставит указатель на начало первого элемента файла. Если при чтении файла возникает необходимость вернуть указатель в его начало, достаточно будет просто применить процедуру reset к этому файлу еще раз.
- **Reset (ФайловаяПеременная);**
- Например: AssignFile (f, ‘c:\data.txt’);

  Reset (f);
- Файловая переменная должна быть связана с конкретным файлом. При отсутствии внешнего файла с указанием имени возникает сообщение об ошибке.

Процедура **rewrite**

- Процедура **rewrite (ФайловаяПеременная)** создаёт и открывает новый файл для следующей записи данных. После её успешного выполнения файл готов к записи в него первого элемента.
- Обратите внимание! Если внешний файл с указанным именем уже существует, то он удаляется и на его месте создаётся новый пустой файл с тем же именем. Для предотвращения потери информации на практике необходимо создавать резервные копии файлов (обычно им назначают расширение bak).

Процедура **Close**

Процедура **Close (ФайловаяПеременная);**

Позволяет закрыть файл, после того как в программе будет завершена его обработка. В противном случае может произойти потеря данных. При закрытии внешний файл обновляется, его автоматически завершает символ конца файла.

Процедура **rename**

- Процедура **rename (ФайловаяПеременная, ИмяФайла)** используются для того, чтобы переименовать неоткрытый внешний файл любого типа. Новое имя задаётся строкой **ИмяФайла**.

Процедура **erase**

- Процедура **erase (ФайловаяПеременная)** удаляет неоткрытый внешний файл любого типа, задаваемый параметром Файловая Переменная.
- Обратите внимание! Процедуры rename и erase нельзя использовать для открытых файлов.

Функция **ioresult**

- Функция **ioresult** проверяет существование файла на диске. Как правило это делается автоматически, но иногда возникает необходимость использовать эту функцию.

Функция **eof**

- Логическая функция **eof (ФайловаяПеременная)** выполняет проверку, не достигнут ли конец файла (End Of File) при чтении из него данных. Функция возвращает true, если конец файла обнаружен, и указатель текущей позиции находится в конце файла за его последним символом. Это значит, что последний элемент в файле уже прочитан, или файл после открытия оказался пуст. В противном случае функция выполняет false.

Текстовый файл

- Текстовый файл – это последовательность символов char, сгруппированных в строки, заканчивающиеся специальным символом end. В конце любого файла, в том числе и текстового, ставится символ #26 (SUB) – конец файла eof.
- Объявление текстовых файлов в программе выглядит так:

```
Type ИмяТипа = text;
var  ФайловаяПеременная: Имя Типа;
     или
var ФайловаяПеременная: Text File;
```

- Файловая Переменная – имя файловой переменной.

Процедуры и функции для текстовых файлов.

- Процедура **Append (ФайловаяПеременная)** открывает существующий файл для дозаписи - применениима только для текстовых файлов. Указатель становится в конце файла, куда и будут дописываться новые компоненты. Файловая Переменная должна быть связана с внешним файлом с помощью процедуры assign.
- Если файл ранее уже был открыт с помощью reset или rewrite, использование append приведёт к закрытию этого файла и открытию его вновь для добавления.

Процедур **write, writeln**

- Вывод в текстовый файл осуществляется при помощи процедур write, writeln. Инструкция процедуры записи выглядит так:
- **write** (ФайловаяПеременная, y1, y2, …, yN);
- **writeln** (ФайловаяПеременная, y1, …, yN);
- **writeln** (ФайловаяПеременная);
- где y1, y2, …, yN – список вывода, то есть имена переменных, значения которых нужно вывести в файл, начиная с позиции текущего указателя. Список вывода содержит выводимые выражения разных типов (integer, real, char, string, boolean). Файл должен быть открыт для вывода.

Процедуры **read, readln**

- Чтение из файла выполняется при помощи read и readln.
- Процедура чтения
- **read** (ФайловаяПеременная x1, x2, …, xN);
- **readln** (ФайловаяПеременная x1, x2, …, xN);
- **readln** (ФайловаяПеременная);
- x1, x2, …, xN – список ввода, содержащий имена переменных разных типов (integer, real, char, string), значения которых процедура read считывает из текстового файла, начиная чтение с элемента, на который установлен текущий указатель. ФайловаяПеременная имеет тип text.

Пример

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

23 15

45 28

56 71

то в результате выполнения инструкций:

Assignfile (f, ‘a:\ data.txt’);

Reset (f); // открытие для чтения

read (f, a);

read (f, b, c.);

read (f, d);

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

- a в результате выполнения инструкций:
- Assignfile (f, ‘a:\ data.txt’);
- Reset (f); // открытие для чтения
- readln (f, a);
- readln (f, b. c);
- readln (f, d)
- значения переменных будут a=23; b=45; c=28; d=56.

Функция eoln

- Для контроля конца строки используется функция eoln (Файловая Переменная), принимающая true, если указатель текущей позиции находится на маркере конца строки (CR/LF), иначе - false. Если eof – true, то и eoln – true.

Варианты выбора буферной переменной:

- Массив или связанный список строк, в который будет считан сразу весь файл;
- Одна переменная строкового типа, в которую будут считываться по очереди строки файла;
- Одна переменная символьного типа, в которую по очереди будут считываться символы.

Типизированные файлы

- Типизированный файл состоит из последовательности элементов одного типа и длины. Их число и, следовательно, размер файла не ограничивается при его задании. Каждый элемент файла имеет номер. Первый элемент считается нулевым. В каждый момент времени программе доступен только один – текущий элемент, на который установлен указатель файла.
- Так как все элементы файла имеют одинаковую длину, позиция каждого элемента легко вычисляется. Поэтому указатель может быть перемещен на любой элемент файла, обеспечивая к нему прямой доступ.

Объявление типизированных файлов

Type ИмяТипа = file of Тип

Var ФайловаяПеременная: ИмяТипа;

Или

Var ФайловаяПеременная =file of Тип;

- При обработке таких файлов могут использоваться некоторые дополнительные процедуры и функции, ряд известных нам общих имеют свои особенности.
- Открытия типизированного файла можно произвести стандартными способами: reset и rewrite.
- Следует знать:
- типизированные и нетипизированные файлы всегда допускают одновременно как чтение, так и запись, независимо от того, были ли они открыты с помощью reset или rewrite;
- для чтения и записи типизированного файла применяются только процедуры reset или write. Использование readln и writeln – запрещено.

Процедуры и функции для типизированных файлов

- Функция **filepos (ФайловаяПеременная)** возвращает целое число – текущую позицию в файле. Функцию нельзя использовать для текстовых файлов. Файл должен быть открыт. Если текущей позицией является начало файла – его первый компонент, например, после выполнения reset, то функция возвращает значение ноль. При переходе от одного элемента к другому его значение увеличивается на единицу. Но номер физической записи будет по-прежнему на единицу меньше номера логической записи, хотя их общее число совпадает. Для случая конца файла, когда eof возвращает true, filepos возвращает номер последнего элемента файла (совпадает со значением функции filesize). Результат – longint.

Функция **filesize**

- Функция **filesize (ФайловаяПеременная)** возвращает число элементов файла, но не текущий размер файла в байтах. Если файл пуст, возвращает 0. функцию нельзя использовать для текстовых файлов. Файл должен быть открыт. Результат – longint.

Процедура **seek**

- Процедура **seek (ФайловаяПеременная, НомерПозиции)** перемещает указатель файла из текущей позиции к позиции с указанным номером, не выполняя чтение или запись. Поскольку номер первого элемента файла равен 0, то оператор seek (ФайловаяПеременная, filesize (ФайловаяПеременная)) ставит указатель файла за его конец, что используется при добавлении данных к файлу. Функцию не используют для текстовых файлов. Файл должен быть открыт. НомерПозиции – longint.

Процедура **truncate**

- Процедура **truncate (ФайловаяПеременная)** усекает размер файла до его текущей позиции. Функцию не используют для текстовых файлов. Файл должен быть открыт. Все элементы файла после текущей позиции удаляются, после чего текущая позиция становится концом файла.

Нетипизированные файлы

При выполнении копирования или обработке баз данных приходится иметь дело с файлами, состоящими из компонентов одинакового размера, структура которых не известна или не имеет значения. На практике это приводит к тому, что любой файл, подготовленный как текстовый или типизированный, можно открыть и начать работу с ним как с нетипизированным набором данных прямого доступа.

Объявление нетипизированных файлов

```
type ИмяТипа = file;
 var ФайловаяПеременная: ИмяТипа;
 или
 var ФайловаяПеременная = file ;
```

- За исключением процедуры read и rewrite для всех нетипизированных файлов допускается использование любой стандартной процедуры, которая пригодна для типизированных файлов. Вместо процедур read и write здесь используются соответственно процедуры blockread (считывает из файла в переменную одну или более записей) и block write (записывает одну или более запись из переменной в файл), которые пересылать данные с высокой скоростью.