

**Файлы**

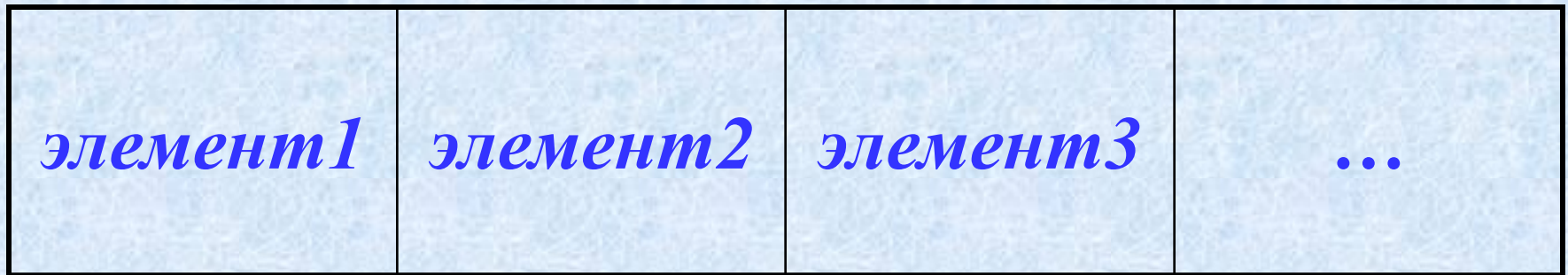
# Файл

*- именованная область  
внешней памяти ЭВМ,  
хранящая  
совокупность данных.*

# Особенности файлов

- **Файл имеет имя** (до 8 лат. букв, цифр или символов !, @, #, \$, %, ^, &, (, ), ` , ~, -, \_ и после точки – расширение до 3 символов. Перед именем можно указать путь к файлу)
- **Файл содержит компоненты одного типа**
- **Длина создаваемого файла не оговаривается при его объявлении, а ограничивается только ёмкостью устройств внешней памяти ЭВМ**

***Файл можно представить как потенциально бесконечный список значений одного типа.***



 *Текущий указатель*

***В любой момент времени программе доступен только один элемент файла, на который ссылается текущий указатель.***

# По способу доступа к элементам различают

- **Файлы последовательного доступа** (к элементам обеспечивается доступ в той же последовательности, в которой они записывались)
- **Файлы прямого доступа** (доступ к элементам осуществляется по адресу элемента)

*Обмен данными с файлом  
осуществляется с помощью  
переменных файлового типа.*

## *Типы файлов (файловых переменных)*

- Текстовые*
- Типизированные*
- Нетипизированные*

# Формат описания файлов

**var** <имя ф. пер.>: **text** ;  
{текстовый файл}

**var** <имя ф. пер.>: **file of** <тип>;  
{типизированный файл}

**var** <имя ф. пер.>: **file** ;  
{нетипизированный файл}

# Примеры

**var**

*f1* : **text;**

*f2* : **file of integer;**

*f3* : **file;**



# Процедуры и функции для работы с файлами

1. Процедура, связывающая файловую переменную с конкретным файлом.

*assign*(*f*, <имя файла>);

**Пример**

**assign**(*f1*, 'c:\tp\10b\file.dat');

2. Процедура, открывающая существующий файл для чтения.

*reset(f);*

3. Процедура, создающая и открывающая новый файл для записи. (*Если файл ранее содержал данные, то они уничтожаются*).

*rewrite(f);*

4. Процедура, открывающая существующий файл для добавления. (*Только для текстовых файлов*).

*append(f);*

**5. Процедура, закрывающая  
файл с сохранением в нем  
данных.**

*close(f);*

## 6. Процедура, переименовывающая неоткрытый файл.

*rename*(*f*, <новое имя файла>);

# 7. Процедура, удаляющая неоткрытый файл.

*erase(f);*



8. Функция, возвращающая **True**, если получен признак конца файла, иначе – **False**.

*eof(f)*

9. Функция, возвращающая **0**, если открытый файл существует, иначе – **др. целое число**.

*(Работает с директивой отключения стандартной проверки {\$I-}).*

***ioresult***

# Текстовые файлы

*Могут содержать строки,  
символы и числа любого типа.*

*Перед записью в файл  
внутреннее представление  
переменных преобразуется в  
последовательность символов,  
т. е. текст.*

**Процедуры и  
функции для  
работы с  
ТЕКСТОВЫМИ  
файлами**

# 1. Процедуры ввода значений переменных из файла.

**read**(*f*, <список переменных>);

**readln**(*f*, <список переменных>);

## 2. Процедуры вывода значений выражений в файл.

**write**(*f*, <список выражений>);

**writeln**(*f*, <список выражений>);

3. Функция, возвращающая **True**, если получен признак конца строки, иначе – **False**.

*eoln(f);*

# Типизированные файлы

*Позволяют организовать*

*прямой доступ к компоненту по  
его порядковому номеру.*

*Перед первым обращением к  
процедурам ввода-вывода  
указатель файла стоит в его  
начале и указывает на первый  
компонент с номером **нуль**.*



*Типизированные файлы также используют процедуры **read**(*f*, <список пер.>) и **write**(*f*, <список выр.>).*

*Переменные и выражения в списках ввода-вывода должны иметь тот же тип, что и компоненты файла.*

**Процедуры и  
функции для  
работы с  
типизированными  
файлами**

1. Процедура, смещающая указатель на компонент с номером  $N$  (выражение типа *longint*).

*seek(f, N);*

2. Процедура, удаляющая часть файла с текущей позиции до его конца.

*truncate(f);*

3. Функция, возвращающая количество компонентов файла (*типа longint*).

*filesize(f);*

4. Функция, возвращающая номер текущего элемента  
(типа *longint*).

*filepos(f);*

# Нетипизированные файлы

*Позволяют организовать высокоскоростной обмен данными.*

*Вместо процедур **read** и **write***

*используются процедуры **blockread** и **blockwrite**,*

*позволяющие определять*

*параметры буферов,*

*использующихся при обмене*

*данными.*