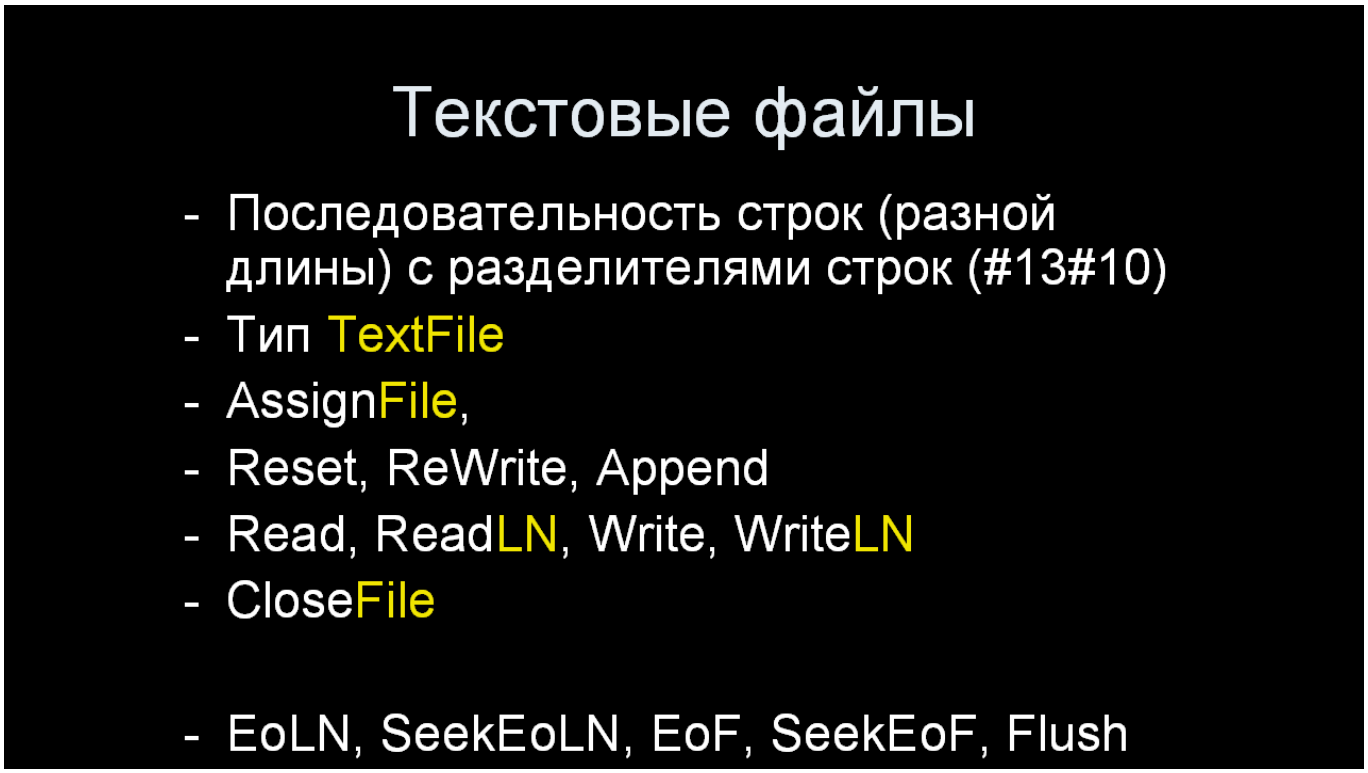
Лекция 2. Три вида файлов в Delphi.

С точки зрения языка Delphi существуют три вида файла: текстовый, типизированный, бестиповой. Delphi предоставляет *разные* возможности для работы с ними.



Текстовые файлы мы изучали в первом семестре. Они состоят из строк с разделителями. Окончание LN как раз связано с поиском и добавлением разделителя

Файловая переменная представляет собой запись, хранящую информацию о связанном с ней файле – Имя (поле Name); состояние файла (поле Mode): открыт для записи, чтения, и того и другого, закрыт; буфер (Buffer) из 128 символов для ускорения работы с внешней памятью, текущая позиция в буфере (BufPos) и т.д.

Файловая запись (record)

```
TTextBuf = array[0..127] of Char;

TTextRec = packed record          (* 460 bytes *)
  Handle: Integer;
  Mode: Word;
  Flags: Word;
  BufSize: Cardinal;
  BufPos: Cardinal;
  BufEnd: Cardinal;
  BufPtr: PChar;
  OpenFunc: Pointer;
  InOutFunc: Pointer;
  FlushFunc: Pointer;
  CloseFunc: Pointer;
  UserData: array[1..32] of Byte;
  Name: array[0..259] of Char;
  Buffer: TTextBuf;
end;
```

- Напрямую не обращаться!

```
const
{ File mode magic numbers }

  fmClosed = $D7B0;
  fmInput  = $D7B1;
  fmOutput = $D7B2;
  fmInOut  = $D7B3;

{ Text file flags       }
  tfCRLF   = $1;    // Dos
compatibility flag, for CR+LF line
breaks and EOF checks
```

Гречкина П.В.,
Программирование, Файлы в Delphi

Два варианта процедур чтения – Read и ReadLN есть только для текстовых файлов

- Текстовые (с разделителями строк)

```
Var F: TextFile;  a,b: integer;
Begin
  AssignFile(F, 'Test1.txt');
  Reset(F);
  read(F, a);
  read(F, b);
  writeln(Format('Считано: a=%d b=%d', [a,b]));
  CloseFile(F);
End;
```

Test1.txt

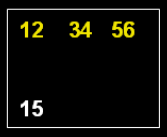
```
12  34

15
```

Например,

При чтении чисел a и b с помощью Read, из указанного считаются числа 12 и 34,. А при чтении с помощью ReadLN, из него же считаются числа 12 и 15, пропустив конец первой строки и пустую строку. Также будут пропущены пробелы, табуляции.

Файлы

Test1.txt



- Текстовые (с разделителями строк)

```
Var F: TextFile;  a,b: STRING[5];
Begin
   AssignFile(F, 'Test2.txt');
   Reset(F);
   read(F, a);
   read(F, b);
   writeln(Format('Считано: a=''%s'' b=''%s''', [a,b]));
   CloseFile(F);
End;
```

При чтении строк из 5 символов (первая строка файла целиком не уместится) тоже результат будет разным. При чтении с помощью Read считаются строки ’12 34’ и ‘ 56’. А при чтении с помощью ReadLN – строки ’12 34’ и ‘’ (пустая, из второй пустой строки файла).

При чтении символов также не будет пропусков пробелов, табуляций и символов разделителей строк (с кодами 13 и 10). Так, из первой же пустой строки считаются эти разделители:

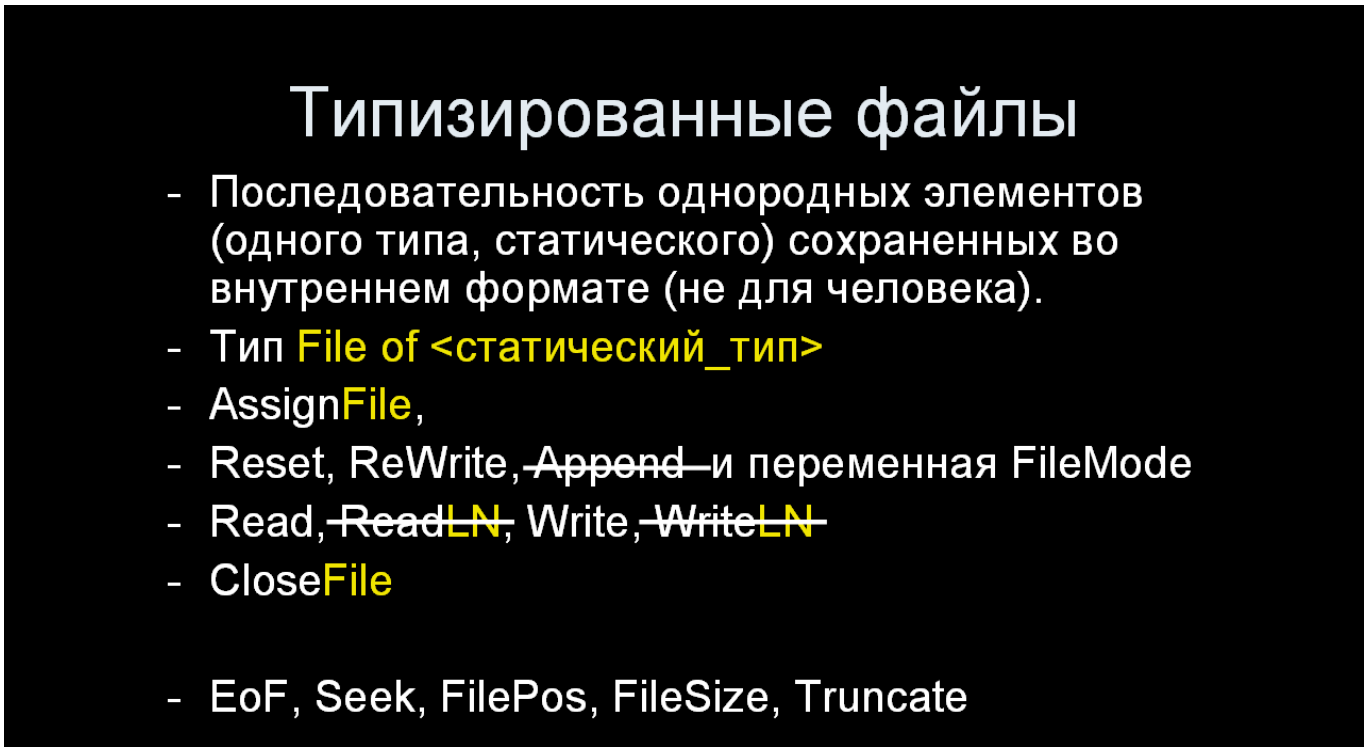
Файлы

Test1.txt

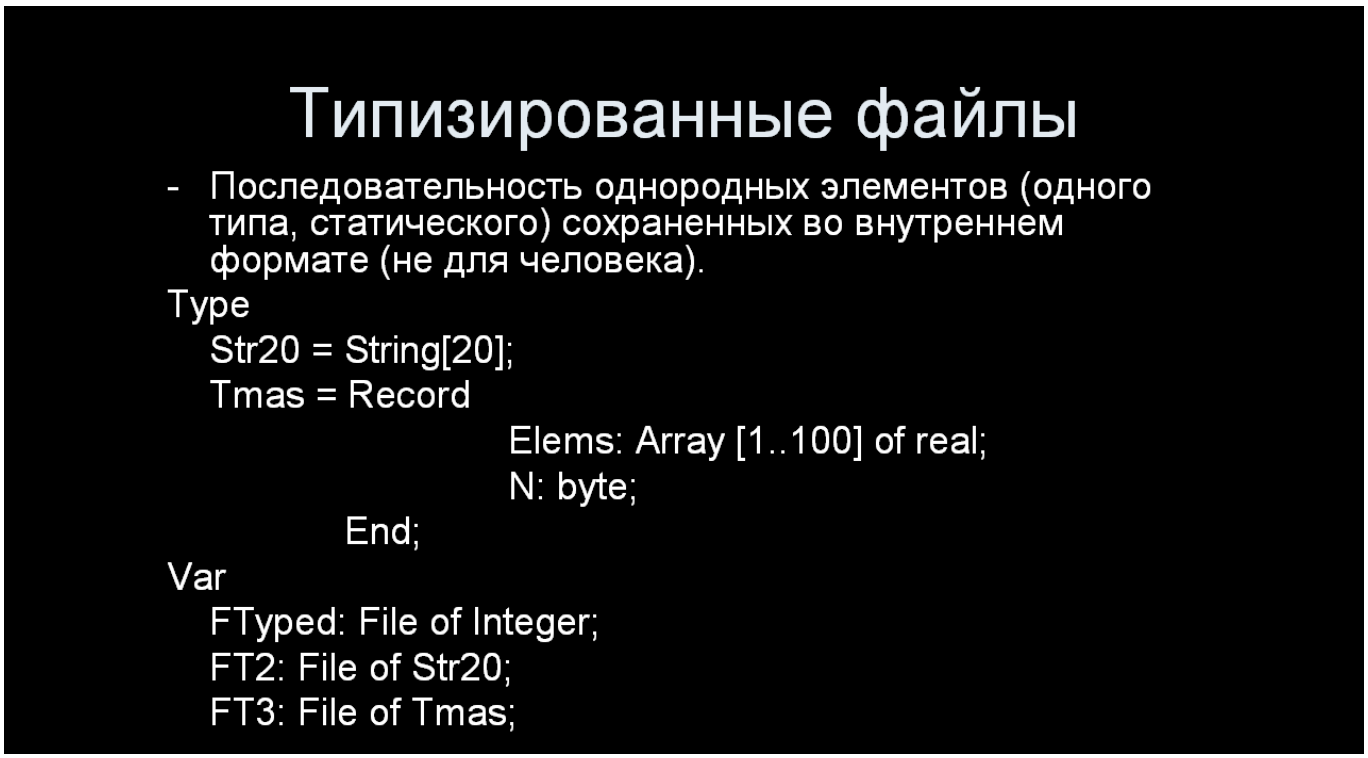
- Текстовые (с разделителями строк)

```
Var F: TextFile;  a,b: Char; //ANSIChar
Begin
   AssignFile(F, 'Test2.txt');
   Reset(F);
   read(F, a);
   read(F, b);
   writeln(Format('Считано: a=#%d b=#%d',
                                [Ord(a), Ord(b)] ) );
   CloseFile(F);
End;
```

Типизированные файлы



Элементами типизированного файла могут быть элементы простых и структурных, но обязательно ***статических*** типов. Например, рассмотренные ранее длинные динамические строки не могут лежать в файле, поскольку представляют собой указатель на Кучу в оперативной памяти задачи, которая существует во время работы этой программы, в отличие от файла, который останется во внешней памяти компьютера и при его отключении, и при завершении программы, когда указатели уже потеряют смысл – зачем же их хранить в файле?



Файловая переменная несколько отличается, хоть и тоже является записью:

Файловая запись (record)

```
TFileRec = packed record    // 332 байта
  Handle: Integer;
  Mode: Word;
  Flags: Word;
  case Byte of
    0: (RecSize: Cardinal;);
    1: (BufSize: Cardinal;
        BufPos: Cardinal;
        BufEnd: Cardinal;
        BufPtr: PChar;
        OpenFunc: Pointer;
        InOutFunc: Pointer;
        FlushFunc: Pointer;
        CloseFunc: Pointer;
        UserData: array[1..32] of Byte;
        Name: array[0..259] of Char;);
end;
```

- Напрямую не обращаться!

В типизированном файле числа хранятся во внутреннем представлении. Например, если одно и тоже число типа Integer (4 байта) записать в текстовый и типизированный файл, то открыв его в простейшем текстовом редакторе увидим разное. И даже размеры будут сильно отличаться: 10 (вместе с разделителями строки) и 4 байта. В текстовом файле число будет записано в виде символов в 10с/с, а в типизированном – каждый из 4 байт будет интерпретирован как символ.

Пример: Текстовый - Типизированный

```
var f: TextFile; i,n: integer;
begin
assignFile(f, 'txt.txt');
rewrite(f);
n:=2000000049;  write(f, n);
closeFile(f);

reset(f);
read(f, i);    writeln('i=',i);
closeFile(f);
readln
End.
```

2000000049 – 10 байт (+ #13#10)

```
var f: File of integer; i,n: integer;
begin
assignFile(f, 'txt.txt');
rewrite(f);
n:=2000000049;  write(f, n);
closeFile(f);

reset(f);
read(f, i);    writeln('i=',i);
closeFile(f);
readln
End.
```

1"5w – 4 байта

Гречкина П.В.,
Программирование, Файлы в Delphi

Типизированный файл

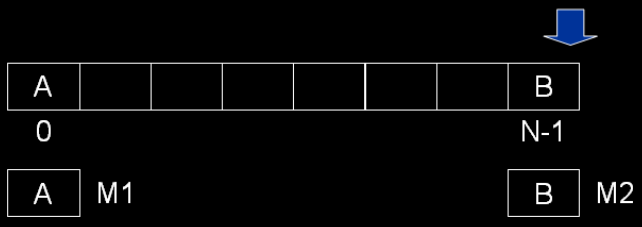
```
Var
   FTyped: File of Integer;          i: integer;
   FT2: File of Str20;          s: Str20;
   FT3: File of Tmas;           mas: Tmas;
Begin
   AssignFile(FT3, 'Test3.txt');
   Reset(FT3); // FileMode: Byte = 2(RW)  0(R)
                              // или ReWrite(FT3);
   Seek(FT3, FileSize(FT3)-1);  //  тек. позиция FilePos(F)
   Read(FT3, mas);            // или Write(FT3, mas);
   CloseFile(FT3);
End;
```

Гречкина П.В.,
Программирование, Файлы в Delphi

Чтобы поменять значениями первый и последний элемент, надо их сначала считать:

Поменять 1 и последний Elem

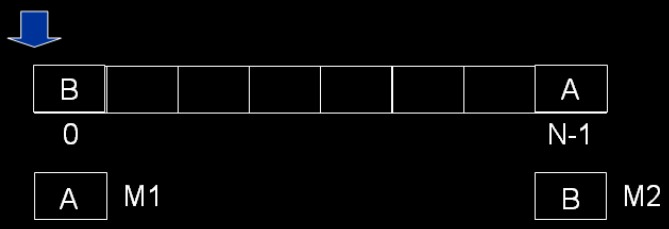
```
type Elem = …;
var
  F: File of Elem;
  M1,M2: Elem;
begin
   …


   Reset(F);

   Read(F, M1);                          { считать первый  (без перехода) }
   seek(F, FileSize(F)-1); Read(F, M2);  { считать последний с переходом  }


   …

end;
```



Гречкина П.В.,
Программирование, Файлы в Delphi

А затем записать в нужные позиции:

Поменять 1 и последний Elem




```
type Elem = …;
var
  F: File of Elem;
  M1,M2: Elem;
begin
  …


  Reset(F);

  Read(F, M1);                          { считать первый  (без перехода) }
  seek(F, FileSize(F)-1); Read(F, M2);  { считать последний с переходом  }
  seek(F, FileSize(F)-1); Write(F, M1); { переход + перезапись последнего}
  seek(F, 0); Write(F, M2);             { перезапись первого с переходом }

  …

end;
```

Гречкина П.В.,
Программирование, Файлы в Delphi

Лучше сначала проверить, что элементов больше одного:

Поменять 1 и последний Elem

```
type Elem = …;
var
  F: File of Elem;
  M1,M2: Elem;
  FileName: string;
begin
  write('введите имя файла'); readln(FileName);   { запрос имени файла }
    AssignFile(F, FileName);
    Reset(F);
     If FileSize(F)>1 then begin
    Read(F, M1);                          { считать первый  (без перехода) }
    seek(F, FileSize(F)-1); Read(F, M2);  { считать последний с переходом  }
    seek(F, FileSize(F)-1); Write(F, M1); { переход + перезапись последнего}
    seek(F, 0); Write(F, M2);             { перезапись первого с переходом }
     end;
    CloseFile(F);
  end;
end;
```

Гречкина П.В.,
Программирование, Файлы в Delphi

Да и проверки исключительных ситуаций при открытии, чтении и записи не помешают:

Поменять 1 и последний Elem

```
type Elem = …;
var
  F: File of Elem;
  M1,M2: Elem;
  FileName: string;
begin
  write('введите имя файла'); readln(FileName);   { запрос имени файла }
    AssignFile(F, FileName);
 try Reset(F); try try

    Read(F, M1);                          { считать первый  (без перехода) }
    seek(F, FileSize(F)-1); Read(F, M2);  { считать последний с переходом  }
    seek(F, FileSize(F)-1); Write(F, M1); { переход + перезапись последнего}
    seek(F, 0); Write(F, M2);             { перезапись первого с переходом }

    except writeln('Не удалось поменять!') end;
    finally CloseFile(F); end; except writeln('Не удалось открыть!') end;
  end;
end;
```

Гречкина П.В.,
Программирование, Файлы в Delphi

Бестиповые (нетипизированные, двоичные, бинарные) файлы удобны, например, для копирования файлов **большими порциями:**

Нетипизированные файлы

- Последовательность байтов

```
Var
    FUnTyped, FNew: File;
    Buffer: array [0..1023] of byte; Cr, Cw: integer;
Begin
    AssignFile(FUnTyped, 'Test1.txt');          AssignFile(FNew, 'TestCopy.txt');

    Reset(FUnTyped,1); ReWrite(FNew,1);  // FileMode = 2(RW) =0(R)

    While not EoF(FUnTyped) do
    begin
          BlockRead(FUnTyped, Buffer, 1024, Cr);  // быстрое копирование
          BlockWrite(FNew, Buffer, Cr, Cw);     // байтов без распознавания
    end;

    CloseFile(FUnTyped); CloseFile(Fnew);
End;
```

Seek, FileSize, FilePos как и для типизированного файла

При открытии таких файлов указывается в байтах размер элемента, например, 1. А при чтении указывается желаемое количество считываемых элементов указанного при открытии размера. А также буфер (куда читать, откуда брать) , и реально считанное /записанное количество можно узнать из последнего параметра (файл может оказаться короче 1024.