Файловый тип

Файл - именованная область внешней памяти ПК. Файлом может называться логическое устройство, потенциальный источник или приемник информации.

Структура задания файлового типа

```
Type
<имя>=FILE OF <тип>;                    {типизированный}
<имя>=TEXT;                             {текстовый}
<имя>=FILE;                             {не типизированный}
```

Пример.

```
Type
  text60=file of string [60];
  File1 = text;
  Filef = file;
Var
  f1: FILE OF CHAR;   {типизированный файл символов}
  f2: TEXT;    f4:File1;       {текстовый файл}
  f3: FILE;      f5:Filef;              {не типизированный}
  f6: TEXT60;             {ФАЙЛ СТРОК}
```

Т.е. возможно объявление переменных, минуя объявление типов,

```
Var
<переменная1>: TEXT;
<переменная2>: FILE OF <тип>;
<переменная3>: FILE;
```

Доступ к файлам

Любой программе доступны два стандартных файла:
INPUT - для чтения данных с клавиатуры,
OUTPUT- для вывода на экран.

```
Program Name (input, output);
```

Связывание выполняется стандартной процедурой ASSIGN:

ASSIGN (<файловая_переменная>, <имя_файла_или_л.у.>);

<файловая_переменная> - правильный идентификатор, объявленный как переменная файлового типа;

<имя_файла_или_л.у.> - текстовое выражение, содержащее имя файла или логического устройства (заключается в апострофы).

Пример.

```
VAR   f, f1, f2: text;        {объявление файловых переменных  f, f1, f2}
  Begin
    ASSIGN(f, 'nameF.txt');      {связывание с файлом ‘nameF.txt’}
    ASSIGN(f1, 'PRN');            {связывание с принтером}
    ASSIGN(f2, ' ');    {связывание со стандартным файлом (в\в)}
```

переменная связывается со стандартным файлом **INPUT или OUTPUT.**

<Имя_файла> - это любое выражение строкового типа, которое строится по правилам определения имен в MS DOS:

- содержит до 8 разрешенных символов: прописные и строчные латинские буквы, цифры и следующие символы **! @ # $ % ^ & ( ) ' ~ - _**
- начинается с любого разрешенного символа,
- за именем может следовать расширение - последовательность до трех разрешенных символов.

Например: Lab11.pas
date.txt

Нельзя вставлять символы шаблонов **'*' и '?'** в имя файла

Перед именем путь к файлу или имя диска e:\

Максимальная длина имени вместе с путем - 79 символов

```
VAR
 F1: text;
 F2: file of string;
Const  name= ‘d: \IVT- 134\ lr11.pas’ ;
...
Assign(f1, 'date.txt');          {текущий каталог}
Assign(f2, name);          {имя переменной, заданное в разделе CONST}
```

Логические устройства в Турбо Паскале

Assign (f, ‘<л.у.>’);

Пример: **Assign (f, ‘CON‘);**

CON - (консоль) клавиатура (для чтения) или экран дисплея (для записи);

PRN - логическое имя принтера; **LPT1** (синоним PRN), **LPT2, LPT3;**

AUX - логическое имя коммуникационного канала, два канала **COM1 (AUX) и COM2;**

NUL - логическое имя пустого устройства, используется в отладочном режиме и трактуется как устройство-приемник информации неограниченной емкости. При обращении к NUL как к источнику информации выдается признак конца файла **(EOF);**

Модуль Printer

объявляет имя файлов переменной LST и связывает с LPT1

```
Uses Printer;
Begin
  Writeln (LST, ‘печатаемый текст’);
            …
End.
```

Общие процедуры для работы с файлами

Процедуры и функции	Выполняемые действия
Assign(Var f, Name:String);	Связывание ф.п. f с файлом Name
RESET(Var f);	Открыть файл с логическим именем f для чтения
REWRITE(Var f) ;	Открыть файл с логическим именем f для записи
CLOSE(f);	Закрыть файл с логическим именем f
Erase(Var f);	Удаление файла (стирание) с диска
ReName(f, NewName:String);	Переименование физического файла в файл с именем NewName.
EOF(f): boolean	Функция тестирования конца файла
FLUSH(<ф.п.>);	Очищает внутренний буфер файла. (Игнорируется, если файл инициирован процедурой RESET)

Открытие файлов (инициализация)

RESET(<ф.п.>); - для чтения,

REWRITE(<ф.п.>); - для записи данных

Только после открытия файла становится возможным чтение и запись повторное обращение в REWRITE **сотрет** текущее содержимое файла

Закрытие файлов **CLOSE(<ф.п.>);** **Пр.** CLOSE(f);

Переименование файлов

RENAME (<Ф.п.>, <новое имя>);

Пример. **RENAME** (f, NewName);

Уничтожение файла

ERASE(<ф.п.>);

Функция тестирования конца файла

EOF(<ф.п.>)

Если достигнут признак конца файла – признак TRUE, иначе FALSE

Текстовые файлы

Файлы последовательного доступа

Var <ф.п.>: TEXT;

В конце строки - **EOLN (End Of LiNe** - конец строки)

(код 13)+(код 10) – **Enter {13(ВК)+10(ПС)}**

В конце файла - **EOF (End Of File - конец файла) с кодом 26 (Ctrl+Z)**

Доступ к записям текстового файла

READ, READLN, WRITE, WRITELN

ASSIGN

Процедура READ обеспечивает ввод чисел, символов, одной строки

READ (<ф.п.>,<список_ввода>);

или **READ (**<список_ввода>**)**;

где <список_ввода> - последовательность из одной или более переменных, типа CHAR, STRING или любого целого или вещественного типов,

например:

READ (X); READ (f,x1,x2,x3); READ (f,x);

CHAR чтение одного символа

CR (13) **EOF** (код 26)

Для **STRING** в строке читается **только одна строка** числовых данных

При чтение числовых данных выделяется подстрока.
Пример: **_ _ _ 2.42_ _ _**

Процедура **READLN**

READLN ([f], x1[, x2, … xN]); обеспечивает ввод чисел, символов, строк
READLN (f, x); **READLN** (f, x1,x2,...,xN); **READLN (f);**
READLN;

READ

Процедура **WRITE**

WRITE (<ф.п.>,<список_вывода>]); **WRITE (<список_вывода>]);**
WRITE ([f], x1[, x2, … xN]);

Пример: **WRITE(f,x);** **WRITE(f,x1,x2,xN);** **WRITE(x1,x2,xN);**

TEXT **ASSIGN** **OUTPUT**

Элемент списка вывода - **X[:w[:d]]**
X - имя переменной или выражение;
w, d - выражения целого типа WORD (0..65535);
w - если параметр w присутствует, указывает минимальную ширину поля (**по умолчанию w=23**);

Пр. Write (X:6:2, Y:6:2) – 3.22 5.56

Write (X:2:2, Y:2:2) – 3.225.56

d - количество десятичных знаков в дробной части.
Если d=0, выводится только целая часть числа

При выводе логических выражений
TRUE и FALSE
Ввод логических констант не предусмотрен.

Процедура **WRITELN**

WRITELN([f], x1[, x2, … xN]);

Пример: WRITELN (f, x); WRITELN(f,x1,x2,x3);

WRITELN (f); WRITELN;

Пример1:

```
RPOGRAM FileFD;
    VAR y,x1,x2,x3:real;
    Ft:TEXT;
 BEGIN
  ASSIGN (Ft,'FXD.DAT');
     READ(x1,x2,x3);                          {ввод трех чисел с клавиатуры}
     REWRITE (Ft);                                {открыть файл для записи}
   y:=x1+x2+x3;
 WRITELN (Ft,'x1=',x1,' x2=',x2,' x3=',x3);   {вывод в файл x1,x2,x3}
 WRITELN (Ft,'сумма =',y);                     {вывод в файл суммы}
     CLOSE (Ft);                                {закрыть файл}
END.
```

1 2 3 - вводим с клавиатуры

Результат:

X1=1 X2=2 X3 =3

Сумма=6

Пример2. Разработать программу, считывающую числовые данные из исходного файла и записывающего четные числа в один файл, нечетные – в другой.

```
• Program File11;              {Заранее (до компиляции прг.)}
uses crt;                    {необходимо создать файл date.txt }
Var f1, f2, f3: text;          {например, в другом окне }
          a: array[1..10] of byte;
     i: byte;
BEGIN   clrscr;
    ASSIGN(f1, 'date.txt');
    ASSIGN(f2, 'res1.txt');
    ASSIGN(f3, 'res2.txt');
    RESET(f1);
    for i:=1 to 10 do          READ(f1, a[i]);
    close(f1);
    Rewrite(f3);   writeln (f3, 'нечетные');
    Rewrite(f2);   writeln (f2, 'четные');
    for i:=1 to 10 do
     if ODD(a[i]) then   write (f3, a[i]:4)       {нечетные}
          else   write (f2, a[i]:4);      {четные}
    close(f2);
    close(f3);
    writeln('Просмотри файлы RES1.txt, RES2.txt ');
    READKEY
END.
```

Текст-ориентированные процедуры и функции

Процедуры и функции **Выполняемые действия**

Append(<ф.п.>) ; Процедура открывает уже существующий файл f для до записи в его конец (для расширения);

пр. Append(f) ;

EOLN(<ф.п.>) Функция возвращает значение TRUE, если во входном файле f достигнуты маркеры конца строки EOLN или конца файла EOF, и FALSE – в противном случае;

пр.if EOLN(f)

SeekEOLN(<ф.п.>) Функция пропускает все пробелы и знаки табуляции до первого признака EOLN или первого значащего символа. Возвращает значение TRUE, если обнаружен маркер конца файла или конца строки. Если <ф.п.> опущена, функция проверяет стандартный файл INPUT;

пр. If SeekEOLN(f)

SeekEOF(<ф.п.>) Функция пропускает все пробелы, знаки табуляции и маркеры конца строк до маркера конца файла или первого значащего символа. Если маркер обнаружен, возвращает значение TRUE.

пр. If SeekEOF(f)

Типизированные файлы

Прямой доступ **Var <ф.п.>: file of <тип компонет>;**

Процедуры и функции для работы с типизированными файлами

Процедуры и функции **Выполняемые действия**

READ(<ф.п.>, <список ввода>); Процедура обеспечивает чтение очередных компонентов файла;

WRITE(<ф.п.>,<список вывода>); Процедура используется для записи новых компонентов в файл, в качестве элементов вывода может быть выражение;

SEEK(<ф.п.>, <номер_компоненты>); Процедура смещает указатель файла к компоненте с указанным номером; <номер_компоненты> - выражение типа LONGINT. (К текстовым файлам применять нельзя);

FILESIZE(<ф.п.>); Функция возвращает количество компонент, содержащихся в файле. (Типа LongInt);

FILEPOS(<ф.п.>); Функция возвращает порядковый номер компоненты файла, доступной для чтения или записи;

CLOSE(<ф.п.>); Функция закрытия файла.

<ф.п.> - файловая переменная д.б. объявленная предложением FILE OF связана с именем файла процедурой ASSIGN

Открыть процедурой **RESET** для чтения, **REWRITE** для записи.

Переместить указатель в конец файла

Пр.: **Seek (f, FileSize(f));** **где f – файловая переменная**

Пример. Сформировать файл (Files.dat) из символов. Вывести текст на экран

- по 20 символов в строке (каждое предложение - с новой строки)

```
PROGRAM file3;
Var s:char;   k: byte;
 fs: file of char;                {строки не читаются}
       Begin
 Assign(fs,'Files.dat');
 writeln('Введи 0 для записи текста; 1 - для чтения'); READ(k);
 IF k=0 then                      Begin
Rewrite(fs); {нельзя другим способом создать типизированный файл}
  writeln('Введи текст');
 While not EOF do                        {CTRL+Z}
              Begin
       READ(s);   write (fs, s);
             end;
 close(fs);             end
       else      Begin          Reset(fs);
            Repeat
            READ(fs, s); write(s);
                      {if s='.' then    write(#10#13) для предложения }
 if FILePOS(fs) mod 20 =1 then write(#10#13);   {по 20 симв }
            Until EOF(fs);
       close(fs);       end
END.
```

Процедуры и функции для работы с каталогами

MkDir(<каталог>) Создает новый каталог с именем k на диске
RmDir(<каталог>) Удаляет пустой каталог

Не типизированные файлы

VAR fn: FILE; **TYPE FF = FILE;**

RESET и REWRITE

Например:

```
VAR fn: file;  ...
  ASSIGN (fn, 'a.dat');
  RESET (fn, 512);
```

(512 байт) длина записи WORD (65535)

кластер

READ и WRITE не применимы
Надо BlockREAD и BlockWRITE

BlockREAD(<ф.п.>,<буфер>,<N>,[,Var<NN>]);
BlockWRITE(<ф.п.>,<буфер>,<N>,[,Var<NN>]);
одно обращение N*r байт

Процедурами SEEK, FilePos и FileSize можно обеспечить доступ к любой записи нетипизированного файла.