ФАЙЛЫ В DELPHI

Понятие файла. Виды файлов

- **Файл** - это поименованная область памяти на внешнем носителе, предназначенная для хранения информации. В зависимости от типа хранящихся в файле данных в языке Delphi различают три вида файлов:
- - **типизированные файлы (FILE OF <ТИП>).** Предназна-чены для хранения однотипных данных;
- - **текстовые файлы (TEXTFILE).** Предназначены для хранения текстовой информации. Файл рассматривается как совокупность строк переменной длины;
- - **нетипизированные файлы (FILE).** Предназначены для хранения разнотипных данных. В данном случае работа с файлом осуществляется на физическом уровне в соответствии с внутренним форматом его данных. Это позволяет организовать высокоскоростной обмен данными между диском и памятью.

Объявление файловых переменных

а) через объявление типа в формате

TYPE
< имя типа > = FILE OF < тип компонентов >;

VAR
< идентификатор > : < имя типа >;

б) через объявление переменных в формате

VAR
< идентификатор > : FILE OF < тип компонентов >;

Примеры объявления файловых переменных

```
TYPE
        A = FILE  OF  REAL;
        M =  ARRAY [1..25] OF  INTEGER;
        Z = RECORD
                R, I : REAL
            END;

   VAR
        F : A;                  { переменная для вещественного файла}
        X : FILE  OF  M;    {  переменная для файла целочисленных
    массивов }
        Y : FILE  OF  Z;      { переменная для файла  записей }
        C : FILE  OF INTEGER;  { переменная для целочисленного файла
    }
```

Стандартные процедуры и функции для работы с типизированными файлами

ПРОЦЕДУРЫ

ASSIGNFILE (F,'NAME') - связывает файловую переменную F с внешним файлом с именем NAME. Имя файла записывается в соответствии с правилами операционной системы компьютера;

REWRITE (F) - открывает файл **для записи данных.** Текущий указатель устанавливается на нулевой компонент файла. Процедура чаще всего применяется для создания нового файла. Ее применение к уже существующему файлу приводит к стиранию содержимого файла без его удаления с диска;

RESET (F) - открывает существующий файл для **чтения из него данных**. Текущий указатель устанавливается на нулевой компонент файла. Предполагается, что открываемый файл уже существует, в противном случае возникает ошибка;

READ (F,C) - **считывает** компонент С из файла. За одно обращение можно считать несколько компонентов файла. В этом случае процедура будет иметь следующий вид: READ (F,C1,C2,...,CN), что эквивалентно последовательности операторов: READ(F,C1); READ(F,C2);...READ(F,CN);

WRITE (F,C) - записывает компонент С в файл. За одно обращение можно записать несколько компонентов файла. В этом случае процедура будет иметь следующий вид: WRITE (F,C1,C2,...,CN), что эквивалентно последовательности операторов: WRITE(F,C1); WRITE(F,C2);...WRITE(F,CN);

CLOSEFILE (F) - закрывает открытый файл;

Стандартные процедуры и функции для работы с типизированными файлами

SEEK (F,N) - перемещает указатель файла на компонент с номером N. Чтобы переместить указатель в конец типизированного файла можно написать SEEK (F , FILESIZE (F));

TRUNCATE (F) - уничтожает все компоненты файла, начиная с текущего положения указателя;

RENAME (F,'NEWNAME') - переименовывает внешний файл (с которым связана файловая переменная F процедурой ASSIGN). Новое имя файла - NEWNAME;

ERASE (F) - уничтожает существующий файл.

Примечания:

- первый компонент типизированного файла имеет порядковый номер 0;
- перед применением процедур CLOSE, RENAME, ERASE файл должен быть закрыт.
- к файлам, открытым процедурой RESET, допускается применение процедуры WRITE (для добавления данных в уже существующий файл).

Функции

EOF (F) - логическая функция, тестирующая конец файла. Возвращает TRUE, если файловый указатель находится в конце файла;

FILESIZE (F) - функция типа LONGINT, которая определяет текущий размер файла (в компонентах);

FILEPOS (F) - функция типа LONGINT, определяющая порядковый номер компонента файла, который будет обрабатываться следующей операцией ввода-вывода.

Общие алгоритмы обработки типизированных файлов

1. Создание файла

а)
```
ASSIGN (F,' F.DAT ');
  REWRITE (F);
  FOR  I :=1  TO  N  DO
    BEGIN
       СОЗДАТЬ (C);
       WRITE (F,C)
    END;
```

б)
```
ASSIGN (F,' F.DAT ');
  REWRITE (F);
  WHILE  УСЛОВИЕ  DO
    BEGIN
       СОЗДАТЬ (C);
       WRITE (F,C);
    END;
```

в)
```
ASSIGN (F,' F.DAT ');
  REWRITE (F);
  REPEAT
       СОЗДАТЬ (F);
       WRITE (F,C);
  UNTIL  NOT  УСЛОВИЕ;
```

Общие алгоритмы обработки типизированных файлов

2. Обработка файла

а)
```
ASSIGN (F,' F.DAT ');
  RESET (F);
  FOR  I := 1  TO  FILESIZE (F)  DO
     BEGIN
        READ (F,C);
        ОБРАБОТАТЬ (C);
     END;
```

б)
```
ASSIGN (F,' F.DAT ');
  RESET (F);
  WHILE  NOT  EOF (F)  DO
     BEGIN
        READ (F,C);
        ОБРАБОТАТЬ (C);
     END;
```

в)
```
ASSIGN (F,' F,DAT ');
  RESET (F);
  REPEAT
        READ (F,C);
       ОБРАБОТАТЬ (C);
  UNTIL  EOF (F);
```

ПРИМЕРЫ

Сформировать файл, компонентами которого являются одномерные вещественные массивы. Вывести на экран количество отрицательных элементов каждого массива сформированного файла.

```
PROGRAM  A13;
TYPE
T = ARRAY [1..25] OF  REAL;
V = FILE  OF T;
VAR
A : T;
F : V;
K, I , J: INTEGER;
BEGIN
ASSIGN (F, 'F.DAT');
REWRITE (F);
FOR  I := 1  TO  10  DO
   BEGIN
    FOR  J := 1  TO  25  DO
          WRITE (A[J]);
   WRITE (F, A);
    END;
RESET (F); K := 0;
WHILE  NOT  EOF(F)  DO
   BEGIN
           READ (F, A);
FOR  I := 1  TO  25  DO
     IF  F[I] < 0 THEN  K := INC(K);
WRITELN (K);
      END;
END.
```

ПРИМЕРЫ

Создать файл A, компонентами которого являются целочисленные матрицы 5x5. Сформировать файл B, содержащий минимальные значения каждой матрицы файла A (с выводом их на экран).

```
PROGRAM NAME;
 TYPE
   M = ARRAY [1..5,1..5] OF INTEGER; { Целочисленная матрица 5x5 }
   F = FILE OF M; { Файл матриц }
VAR
    A, B : F;                    { Файловые переменные }
    K : M;                       { Переменная - матрица }
    C, I, J : INTEGER;           { Вспомогательные переменные }
 BEGIN
ASSIGN ( A,' A.DAT '); { Связываем файловую переменную A с файлом A.DAT }
 ASSIGN (B, 'B.DAT '); { Связываем файловую переменную B с файлом B.DAT }
 REWRITE (A);              { Открываем файл A на запись }
{ Цикл записи в файл A 10-ти целочисленных матриц 5x5 }
 FOR I := 1 TO 10 DO
   BEGIN
      FOR I := 1 TO 5 DO
          FOR J := 1 TO 5 DO
              READ (K[ I, J ]); { Вводим матрицу }
       READ (A, K);             { Записываем матрицу в файл A }
   END;
   RESET (A);                   { Открываем файл A на чтение }
   REWRITE (B);                 { Открываем файл B на запись }
 { Цикл просмотра файла A }
   WHILE NOT EOF (A) DO
  BEGIN
      READ (A, K);              { Считываем матрицу 5x5 }
{ Определяем минимальный элемент матрицы }
      C := K[1, 1];
     FOR I := 1 TO 5 DO
       FOR J := 1 TO 5 DO
          IF K[I, J] < C THEN C := K[I, J];
    WRITE (B, C);      { Заносим минимальный элемент в файл B }
     WRITE (C);        { Выводим минимальный элемент на экран}
   END;
    CLOSE (A);                  { Закрываем файл A }
    CLOSE (B);                  { Закрываем файл B }
END.
```