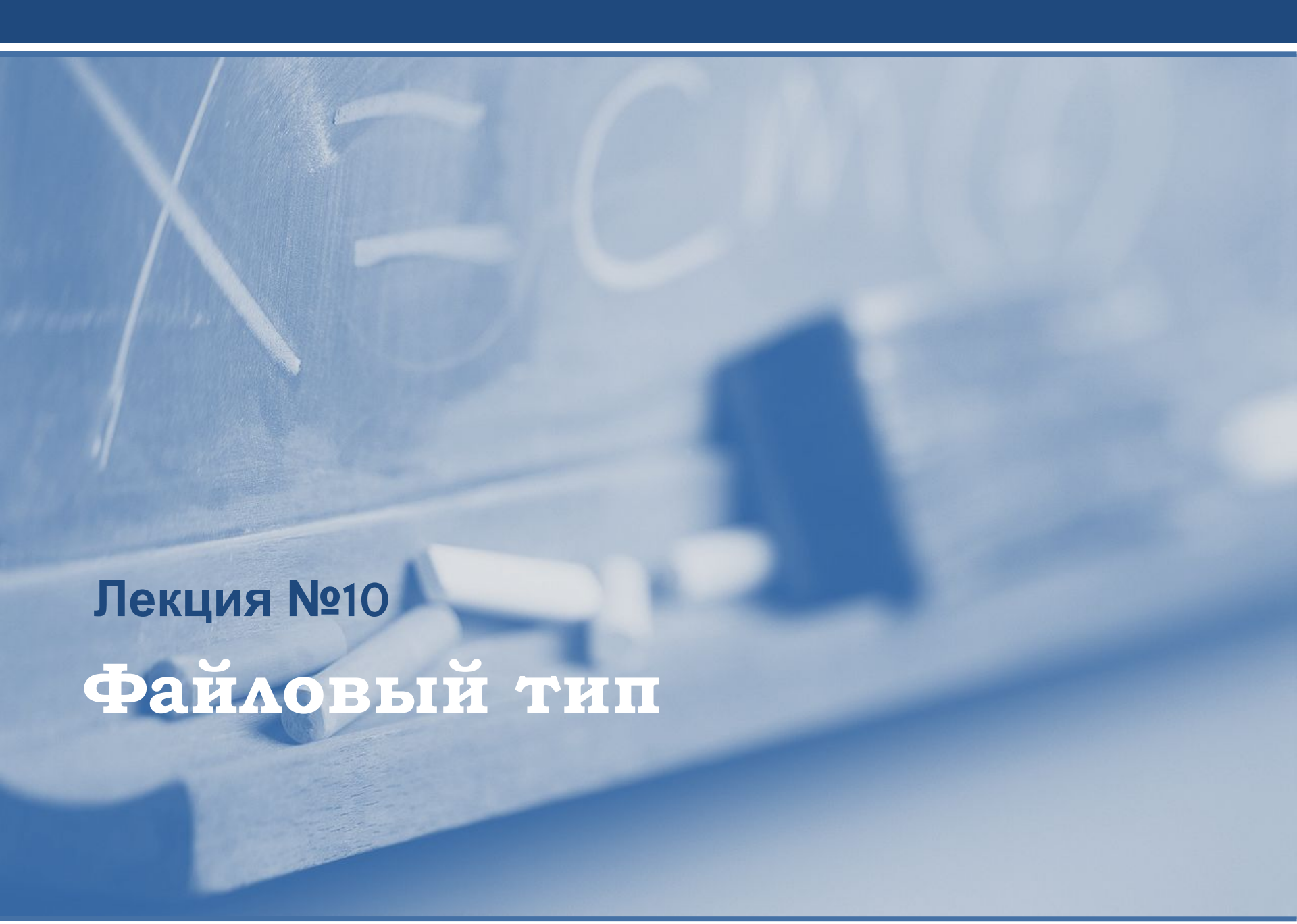

Лекция №10

Файловый тип

План лекции

1. Файлы. Виды файлов
2. Типизированные файлы
3. Связь файловой переменной с файлом
4. Закрытие файла
5. Открытие файла для чтения и записи
6. Чтение из файла и запись в файл
7. Признак конца файла
8. Изменение последовательного порядка доступа к файлу
9. Текстовые файлы
10. Открытие файла для добавления
11. Стандартные потоки ввода-вывода
12. Нетипизированные файлы

Файлы. Виды файлов

Файлом называется поименованная область памяти носителя информации.

Язык Паскаль предлагает три вида представления файлов:

- Типизированные
- Текстовые
- Нетипизированные

Типизированные файлы

Типизированный файл – последовательность элементов одного типа.

Описание файлового типа имеет синтаксис:

file of < тип элементов>;

Примеры :

```
Type
    WordFile = file of word;
Var
    File1 : WordFile;
    File2 : file of word;
```

Типизированные файлы

```
Type
      Student = record
         Name, SerName : string;
         YearOld : byte;
         Sessia : array [1..10] of byte;
       end;
Var
      File1 : file of char;
      File2 : file of Student;
      File3 : file of string;
```

Файловые переменные имеют специфическое применение. Над ними нельзя выполнять никаких операций (присваивать значение, сравнивать и др.). Их можно использовать только для выполнения операций с файлами (чтения, записи, удаления файла и т.д.). кроме того, через файловую переменную можно получить информацию о конкретном файле (тип, параметры, имя файла и т.д.).

Связь файловой переменной с файлом

До начала работы с файлами устанавливается связь файловой переменной с именем файла. Это имя представляется в виде строки, содержащей полное имя файла.

```
Assign (MyFile, 'c:\MyDirectory\Result.dat');
```

здесь приведено полное (с указанием пути) имя пользовательского файла Result.dat.

Если путь не указан, программа будет искать файл только в своем рабочем каталоге.

Кроме указания имени файла на дисковом накопителе может быть указано стандартное имя одного из устройств ввода-вывода:

con - консоль, то есть дисплей и клавиатура,

prn или lpt1 - принтер.

Не разрешается связывать с одним физическим файлом различные файловые переменные в программе.

Закрытие файла

Все файлы, открытые в результате работы программы, **должны быть закрыты** при завершении программы процедурой

```
Close (MyFile);
```

При выполнении этого оператора закрывается физический файл на носителе и фиксируются изменения, связанные с использованием данного файла.

Обратите внимание на необходимость закрытия файлов во всех ветвях программы, в том числе в различных аварийных ситуациях.

Открытие файла для записи

Открытие нового файла для записи производится процедурой, единственный аргумент которой – переменная файлового типа

```
Rewrite (MyFile);
```

Эта процедура создает новый файл, имя которого связано с переменной MyFile процедурой Assign. Указатель работы с файлом помещается в начальную позицию.

Если файл с таким именем уже существует, он становится пустым, то есть его предыдущее содержание теряется.

После выполнения процедуры Rewrite файл доступен как для записи, так и для чтения.

Подготовка файла

Подготовку существующего файла для чтения или записи выполняет процедура

```
Reset (MyFile);
```

Эта процедура ищет уже существующий файл на носителе и открывает его для работы, помещая указатель в начальную позицию. Если файл с установленным в Assign именем не найден, возникает ошибка ввода/вывода.

Запись в файл

Запись в файл производится процедурой

```
Write (MyFile, var1, var2, ...., varN);
```

Первый аргумент этой процедуры – переменная файлового типа, далее следует список записываемых переменных, которые должны соответствовать объявленному типу файла.

При выполнении этой операции текущий указатель файла смещается на число позиций, равное числу переменных.

Чтение из файла

Чтение из файла производится аналогичной процедурой:

```
Read (MyFile, var1, var2, ...., varN);
```

Положение элементов в файле нумеруется, начиная с номера 0 для первого элемента.

После последнего элемента файла автоматически записывается признак конца файла.

Функция FileSize(MyFile) определяет число элементов в файле.

Признак конца файла

Функция логического типа

EOF(MyFile)

имеет значение True, если указатель указывает на маркер конца файла (End Of File).

Длина файла, то есть количество элементов в этой последовательности – величина произвольная, изменяемая в процессе работы.

Пример записи в файл

```
Program W;
  Var
    FileName : string; {строка, содержащая имя файла}
    f1 : file of byte; {переменная файлового типа}
    i : byte;
  Begin
    Write ('Введите имя файла ');{предложение ввести имя файла}
    Readln (FileName);{ввод имени файла}
    Assign (f1, FileName);{связь имени файла и переменной}
    Rewrite (f1);{открытие файла для записи}
    for i := 1 to 100 do {цикл для расчетов и вывода данных в файл}
      Write (f1,i);{запись в файл f1 величины i}
    close (f1); {закрытие файла}
  End.
```

Пример чтения из файла

```
Program R1;
    Var
      FileName : string; {строка, содержащая имя файла}
      f1 : file of byte; {переменная файлового типа}
      i ,a : byte;
        n : integer;
    Begin
      Write ('Введите имя файла ');{предложение ввести имя файла}
      Readln (FileName);{ввод имени файла}
      Assign (f1, FileName);{связь имени файла и переменной}
      Reset (f1);{открытие файла для чтения}
        n:=FileSize(f1); {определение количества элементов в файле}
      for i := 0 to n-1 do  {цикл для чтения всех данных из файла}
        begin
            Read (f1,a);{чтение из файла f1 величины a}
            Write(a); {вывод величины a на экран}
        end;
      close (f1); {закрытие файла}
    End.
```

Пример чтения из файла

```
Program R2;
    Var
      FileName : string; {строка, содержащая имя файла}
      f1 : file of byte; {переменная файлового типа}
      i ,a : byte;
  Begin
     Write ('Введите имя файла ');{предложение ввести имя файла}
     Readln (FileName);{ввод имени файла}
     Assign (f1, FileName);{связь имени файла и переменной}
     Reset (f1);{открытие файла для чтения}
       while not EOF(f1) do {пока не встречен конец файла, выполняется
                             тело цикла}
       begin
            Read (f1,a);{чтение из файла f1 величины a}
            Write(a); {вывод величины a на экран}
         end;
     close (f1); {закрытие файла}
   End.
```

Пример

```
Program FileString;
      Var
        f, g : file of string;
        str1, str : string;
        i : integer;
      Begin
          assign(f,'f');
        rewrite(f);
        assign(g,'g');
        rewrite(g);
        repeat
          readln(str);
          write(f,str);
          for i:=length(str) downto 1 do
            str1:=str1+str[i];
          write(g,str1);
          str1:='';
        until str='';
        close(f);
        close(g);
```

```
assign(f,'f');
        reset(f);
        assign(g,'g');
        reset(g);
        while not eof(f) do
          begin
            read(f,str);
            writeln(str);
          end;
        while not eof(g) do
          begin
            read(g,str);
            writeln(str);
          end;
        close(f);
        close(g);

End.
```

Задача

Создать типизированный файл, содержащий информацию о работниках колледжа. Определить средний стажа работы в колледже. Вывести фамилий работников, у которых оклад больше заданного пользователем.

```
Program A;
      Type
       Dann=record
          stag : byte;
          Surname, WorkName : string;
          Oklad, Year : integer;
         End;
      Var
       Spisok : file of Dann; {файл типа записи Dann}
       Man : Dann; {переменная типа записи Dann для работы с файлом}
       Name : string[12]; {строка для хранения имени физического файла}

      count  : integer; {количество сотрудников}
Begin

      write('Введите имя файла данных:');
       readln(Name); {имя физического файла}
       assign(Spisok ,Name); {связываем файловую переменную с файлом}
       rewrite(Spisok ); {открываем файл для записи}
       write('Введите количество работников:');
       readln(count);
```

Задача

```
for i:=1 to count do
with Man do
        begin
          writeln('Введите данные ',i,'-го работника');
          write('Фамилия: ');
          readln(Surname);
          write('Год рождения:');
          readln(Year);
          write('Стаж работы:');
          readln(stag);
          writeln('Должность:');
          readln(WorkName);
          write('Оклад');
          readln(oklad);
          write(Spisok ,Man); {записать в файл созданный элемент Man}
        end;
close(Spisok ); {закрываем файл}
End.
```

Задача

```
Program B;
      Type
        Dann=record
            stag : byte;
            Surname, WorkName : string;
            Oklad, Year : integer;
          End;
      Var
        Spisok : file of Dann; {файл типа записи Dann}
        Man : Dann; {переменная типа записи Dann для работы с файлом}
        Name : string[12]; {строка для хранения имени физического файла}
       count, summ : integer;
       S : real;
Begin
       write('Введите имя файла данных:');
        readln(Name); {имя физического файла}
        assign(Spisok ,Name); {связываем файловую переменную с файлом}
        Reset(Spisok ); {открываем файл для чтения}
```

Задача

```
Writeln('Введите параметр – оклад');
Read (summ);
Writeln( ' Список сотрудников, имеющих оклад более',summ,' руб.');
count:=0;
S:=0;
While not EOF(Spisok) do
   begin
              Reed(Spisok ,Man); {чтение из файла элемента Man}
                inc (count);
                S:=S+Man.stag;
                if Man.Oklad>summ then
                     writeln (Man.Surname);
   end;
Writeln ('Средний стаж – ',S/count,' лет.');
close(Spisok ); {закрываем файл}
End.
```

Изменение последовательного порядка доступа к типизированному файлу

Если есть необходимость нарушения последовательной записи или чтения из файла, текущий указатель, может быть изменен процедурой

```
Seek (MyFile, n);
```

где n – требуемое положение указателя.

Нумерация элементов типизированного файла начинается с нуля.

Поэтому, чтобы обратиться к третьму элементу, нужно записать Seek (File, 2).

Seek (MyFile, 0) – устанавливает указатель в начальной позиции (на первый элемент).

Seek (MyFile, FileSize(MyFile)) – устанавливает указатель после последнего элемента, то есть на признак конца файла.

Текущую позицию указателя дает функция

```
FilePos (MyFile);
```

Изменение последовательного порядка доступа к типизированному файлу

Если есть необходимость нарушения последовательной записи или чтения из файла, текущий указатель, может быть изменен процедурой

```
Seek (MyFile, n);
```

где n – требуемое положение указателя.

Нумерация элементов типизированного файла начинается с нуля.

Поэтому, чтобы обратиться к третьму элементу, нужно записать Seek (File, 2).

Seek (MyFile, 0) – устанавливает указатель в начальной позиции (на первый элемент).

Seek (MyFile, FileSize(MyFile)) – устанавливает указатель после последнего элемента, то есть на признак конца файла.

Текущую позицию указателя дает функция

```
FilePos (MyFile);
```

Пример

Составить программу, которая переписывает существующий файл, заменяя все строчные латинские буквы на заглавные.

```
Program RW;
      Var
        FileName : string; {строка, содержащая имя файла}
        FVar : file of char; {переменная файлового типа}
        Index : integer;
        Letter : char; {читаемый из файла символ}
      Begin
        write('Enter filename: '); {предложение ввести имя файла}
        readln (FileName); {ввод имени файла}
        assign (FVar,FileName); {связь имени файла и переменной}
        {$I-} {отключен контроль ввода/вывода}
        reset (FVar); {открытие файла для чтения и записи}
        {$I+} {включен контроль ввода/вывода}
        if IOResult <> 0 {выход, если файл не открыт}
          then
            begin
              writeln ('Не открыт файл ', FileName);
              Halt
            end;
        while not EOF (FVar) do {цикл до конца файла}
          begin
            read (FVar, Letter); {чтение символа из файла}
            Letter:=Upcase(Letter); (преобразование букв)
            Seek(FVar,FilePos(FVar)-1); {перемещение указателя назад на 1 позицию}
            write(FVar,Letter); {запись преобразованной буквы}
          end; {конец цикла}
        close(FVar) {закрыть файл}
      End.
```

IOResult

Функция **IOResult** предназначена для поиска ошибок, возникающих при работе с файлами.

Эта функция возвращает результат последней операции ввода/вывода, если автоматический контроль за ошибками, возникающими при выполнении операций ввода/вывода, отключен с помощью директивы компилятора {$I-}.

При безошибочном выполнении операций ввода/вывода функция IOResult всегда возвращает результат равный нулю. Поэтому, как правило, ее используют в операции сравнения с нулем.

При использовании функции IOResult нужно помнить о том, что она возвращает величину, которую можно интерпретировать как флаг ошибки лишь в том случае, когда эта функция вызывается следом за операцией ввода/вывода. А если Вы хотите провести анализ ошибки позже, Вам придется сохранить возвращаемое значение в некоторой промежуточной переменной.

Выполнение команд работы с файлами

Изменение имени файла производится процедурой

```
Rename(MyFile, FileName);
```

первый аргумент которой – переменная файлового типа, а второй аргумент – строкового типа – новое имя файла, которое может быть сокращенным или полным (с указанием пути). Действие этой процедуры эквивалентно действию аналогичной процедуры операционной системы.

Уничтожение части файла от текущего положения до конца производится процедурой

```
Truncate(MyFile);
```

Уничтожение всего файла производится процедурой

```
Erase(MyFile);
```

действие которой эквивалентно удалению файла в операционной системе.

Текстовые файлы

Текстовый файл структурно несколько похож на "файл из байтов" (file of byte) с той разницей, что в нем, помимо содержательной информации, встречаются символы специального назначения.

Его можно схематически представить в следующем виде:

. . . . . .#13#10

. . . . . .#13#10

. . . . . .#13#10

. . . . . .#13#10

. . . . . .#13#10

#26

Описанная структура текстовых файлов хорошо согласуется с интуитивно понимаемым построением текстовой информации и полностью совпадает со стандартной структурой текстов.

Текстовые файлы

Описание текстовых файлов

```
Var
      TextFile : text;
```

Обращение к файлу в дальнейшем идёт через файловую переменную.

Далее доступ к файлу требуется открыть. Открыть любой файл можно на чтение и на запись. Для этого существуют процедуры Reset, Rewrite.

Пример:

```
Var
     f: text;
    Begin
     assign(f, 'd:\tp7\bin\text.txt'); {Полный путь до файла }
     reset(f);{Открыть на чтение}
     . . .
    End.
```

Текстовые файлы

Рассмотрите простую программу, выполняющую чтение из текстового файла целых чисел и вывод на печать только четных чисел.

```
Program TextFile;
      Var
        f : text;
        Put : string;
        a : integer;
      Begin
        Put := 'D:\Primer.txt'; {Полный путь до файла }
        assign(f, Put); {Связываем файл с переменной f}
        reset(f); {Открываем файл на чтение.}
        while not Eof(f) do{Пока нет конца файла делай...}
          begin
            readln(f, a);{Считываем число в переменную a}
             if not odd(a) {Если число нечетное,}
               then
                 writeln(a);{То выводим его на экран}
          end;
        close(f);
        readln
      End.
```

Eoln

Часто для обработки текстовых файлов используется специфичная для них функция Eoln, позволяющая определить достигнут ли конец строки.

Если достигнут – значение функции равно True, а если нет – False.

Таким образом, для анализа конкретных символов строк файла можно применить вложенный цикл типа:

```
while not Eof(NameFale) do {пока нет конца файла NameFale делай}
    while not Eoln(NameFale) do {пока нет конца строки файла NameFale делай}
    begin
    {группа операторов обработки символов очередной строки}
    end;
```

Открытие файла для добавления

Append(f : Text) – процедура открывает существующий файл для присоединения. Если файл уже открыт, то он сначала закрывается, а затем открывается заново. Текущая позиция устанавливается на конец файла.

Если в последнем блоке файла размером 128 байтов присутствует символ Ctrl+Z (26 в коде ASCII), то текущая позиция устанавливается в файле таким образом, что при записи первым в блоке будет "затираться" символ Ctrl+Z.

После обращения к append файл f становится доступным только по записи и Eof(f) принимает всегда значение True(истина).

Стандартные текстовые файлы Input и Output

В Паскале существуют два стандартных текстовых файла Input и Output. Эти файлы считаются известными в любой Pascal-программе. Они обозначают соответственно стандартный файл ввода и стандартный файл вывода. Обычно эти стандартные файлы связаны с конкретными физическими устройствами компьютера. Так, файловая переменная Input связана с клавиатурой, файловая переменная Output - с экраном дисплея. эти файлы считаются заранее открытыми, а соответствующие идентификаторы можно использовать в операциях ввода-вывода.

Для стандартных файлов Input и Output допускается сокращенная форма записи операций ввода-вывода. Так, если в процедурах read и readln первый параметр опущен, то по умолчанию подразумевается файл Input. Аналогично, отсутствие в процедурах write и writeln первого параметра означает вывод в стандартный файл Output. Вывод в стандартный файл Output используется очень часто - всегда, когда необходимо выдать некоторую информацию из программы на экран.

Стандартные текстовые файлы Input и Output

Стандартные файлы ввода-вывода могут быть "переназначены", то есть связаны с другими физическими устройствами или файлами.

Простейшим способом переназначения является использование для этой цели процедуры assign, например,

```
Assign (Output,'MyFile.out');
```

После выполнения такого оператора стандартный файл вывода будет переназначен, то есть файловая переменная Output будет связана с файлом MyFile.out из текущего каталога.

Все операции вывода, явно или неявно работающие с файлом Output, будут выводить информацию в указанный файл.

Нетипизированные файлы

Нетипизированные файлы – это файлы, поддержка которых осуществляется с максимально возможной скоростью. Введение таких файлов в Паскаль было вызвано стремлением повысить эффективность программ, участвующих в интенсивном обмене с внешними наборами данных.

Эти файлы в отличие от уже рассмотренных не имеют строго определенного типа.

Нетипизированный файл рассматривается в Паскале как совокупность символов или байтов. Представление char или byte не играет никакой роли, важен лишь объем занимаемых данных.

Такое представление стирает различия между файлами независимо от типа их объявления. 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.

Определения в программе нетипизированного файла:

```
Var
    MyFile : file;
```

Нетипизированные файлы

Нетипизированный файл является файлом прямого доступа, что говорит о возможности одновременного использования операций чтения и записи.

Для таких файлов самым важным параметром является длина записи в байтах. Открытие нетипизированного файла с длиной записи в 1 байт можно выполнить следующим образом:

rewrite(MyFile, 1) или **reset(MyFile, 1)**

Второй параметр, предназначенный только для использования с нетипизированными файлами, задает длину записи файла на сеанс работы.

При работе с нетипизированными файлами могут применяться все процедуры и функции, доступные типизированным файлам.

BlockRead

Blockread(Var F : file; Var Buf; Kolblocks : word; result : word);

Процедура считывает из файла F определенное число блоков в память, начиная с первого байта переменной Buf.

Параметр Buf представляет любую переменную, которая будет участвовать в обмене данными с дисками. Эту переменную нужно описать в программе так, чтобы ее размер не был меньше размера записи, установленного в параметрах rewrite или reset (как правило, для этих целей используется некоторый массив).

Параметр Kolblocks задает число считываемых блоков, которые должны быть прочитаны за одно обращение к диску.

Параметр result является необязательным и содержит после вызова процедуры число действительно считанных записей.

Использование параметра result подсказывает, что число считанных блоков может быть меньше, чем задано параметром Kolblocks. Если result указан при вызове, то ошибки ввода-вывода в такой ситуации не произойдет. Для отслеживания этой и других ошибок чтения можно использовать опции {$I-}, {$I+} и функцию IOresult.

Кроме того, что переменная F должна быть описана как нетипизированный файл, она должна быть связана с конкретным физическим диском процедурой assign. Файл должен быть открыт процедурой reset.

BlockWrite

blockwrite(Var F : file; Var Buf; Kolblocks : word; result : word);

Процедура предназначена для быстрой передачи в файл F определенного числа записей из переменной Buf. Все параметры процедуры blockwrite аналогичны параметрам процедуры blockread. Разница лишь в том, что файл должне быть подготовлен для записи процедурой rewrite. Содержимое переменной Buf целиком помещается в файл, начиная с текущей записи.

Дело в том, что чтение информации из файла в буфер, равно как и запись из буфера в файл, производится без типового контроля. Поэтому несоблюдение указанного условия может привести к порче соседних с буфером данных или к помещению на файл посторонней информации.

Если при чтении указана переменная Buf недостаточной длины или если в процессе записи на диск не окажется нужного свободного пространства, то произойдет следующее. Если последний параметр result в этих вызовах не задан, то возникает ошибка ввода-вывода; если параметр result задан, то ошибка не будет зафиксирована, а после выполнения процедуры его значение не будет совпадать с значением параметра Kolblocks. Последнее обстоятельство можно проверить, сравнив два указанных значения.

После завершения процедуры указатель смещается на result записей.

Пример

Составить программу, которая создает массив целых чисел и записывает его в нетипизированный файл, а также вычисляет среднее арифметическое элементов файла.

```
Program N;
      Var
       f : file;
       i, k, s : integer;
       Mas : Array [1..10] of byte;
      Begin
         Randomize;
       for i := 1 to 10 do
         Mas[i] := Random(10);
       assign(f, 'file.dat');
       rewrite(f,1);
       blockwrite(f, Mas, 10);
       close(f);
       reset(f,1);
       while not Eof(f) do
         begin
           blockread(f, k, 1);
           s:= s+k;
           Inc(i);
         end;
       close(f);
       write(s/i:5:2);
       readln;
      End.
```