Лекция 5

ФАЙЛЫ

Любой обмен данными подразумевает наличие источника информации, канала передачи и ее приемника. В случае обмена данными между программой и периферийными устройствами одним концом канала обмена данными всегда является оперативная память ЭВМ. Другой конец этого канала определяют как файл.

Понятие файла достаточно широко. Это может быть обычный файл данных на диске, или коммуникационный порт, или устройство печати и т.д. Файл может быть источником информации-тогда мы читаем из файла (осуществляем ввод данных из файла), или приемником информации- в этом случае мы записываем в файл (осуществляется вывод данных в файл)

Операция вывода данных означает пересылку данных из ОЗУ в файл, а операция ввода-заполнение ячеек памяти данными, полученными из файла. Файловая система, реализуемая в Паскале, состоит из двух уровней: логических файлов и физических файлов.

Понятие логического файла

Логический файл описывается как переменная одного из файловых типов, определенных в Паскале.

Паскаль поддерживает три файловых типа:

текстовые файлы;

компонентные файлы;

бестиповые файлы.

Текстовые файлы состоят из ASCII кодов, включая расширенные и управляющие коды. Текстовые файлы организуются по строкам и обязательно содержат специальный код, называемый концом строки. Любую информацию (числовую, символьную, строчную) тестовый файл хранит в виде символов ее изображающих. Текстовый файл можно распечатать и прочитать. Например, текст программы на Паскале-это текстовый файл, но выполняемый EXE- файл, полученный после компиляции, не является текстовым, и даже если его содержимое удастся

Компонентные файлы в отличии от текстовых состоят из машинных представлений чисел, символов и структур, поэтому с их помощью можно осуществить обмен данными только между дисками и рабочей памятью программы, но нельзя напрямую вывести их на экран.
Бестиповые файлы тоже состоят из машинных представлений. Они работают с данными независимо от их типа, т.е. представляют собой канал ввода-вывода для доступа к любым файлам любого типа.
Файловые переменные не могут участвовать в операторах присваивания.

После того, как в программе в разделе описания переменных объявлена файловая переменная, она может быть использована как средство общения с любым физическим файлом, независимо от его природы. Само имя физического файла может появиться в программе только один раз, когда специальной процедурой устанавливается, что объявленный логический файл будет служить средством доступа именно к этому физическому файлу.

позволяет программисту не задумываться о технических проблемах организации доступа к данным. Различные физические файлы имеют разные механизмы ввода-вывода информации, но эти особенности спрятаны в механизме логических файлов, которые сами определяют как наладить обмен информацией со связанными физическими файлами. Другими словами, логические файлы унифицируют работу с файлами, позволяя работать не с ними, а с их

Описание файлов имеет следующий вид:

type имя типа=file of базовый тип;

В качестве базового типа элементов файла можно использовать любой тип данных, как простой, так и сложный, за исключением типа file.

Пример:

```
type mas=array[1..10] of real;
mno=set of 1..10;
zap=record
re,im:integer;
end;
var  f1: file of integer;
f2: file of real;
f3:file of char;
f4: file of  mas;
f5:file of  mno;
f6: file of  zap;
```

Структура физического файла

Представляет собой простую последовательность байт памяти носителей информации, например, жесткого магнитного диска

байт	байт	байт	байт	…	байт	байт	байт

Структура логического файла-это способ восприятия
файла в программе, другими словами- это шаблон, через который мы смотрим на физическую структуру файла. В языках программирования таким шаблонам соответствуют типы данных, допустимые в качестве компонент файлов.

Например: file of byte;

байт	байт	байт	…	байт	eof

file of char;

Код символа	Код символа	…	Код символа	eof		

file of integer;

Целое со знаком	Целое со знаком	…	Целое со знаком	eof

file of t;

здесь t=record

f:byte;

b:char;

c:integer; end;

байт	Код символа	Целое со знаком	…	байт	Код символа	Целое со знаком	eof

Логическая структура файла очень похожа на структуру массива. Различие между файлом и массивом в том, что количество элементов в массиве фиксируется в момент распределения памяти и он целиком располагается в оперативной памяти. Нумерация элементов массива выполняется соответственно верхней и нижней границам, указанным при его объявлении

У файла количество элементов в процессе работы программы может изменяться и он располагается на внешних носителях. Нумерация элементов файла, кроме текстовых файлов, начинается от нуля и количество элементов файла в каждый момент неизвестно. Зато известно, что в конце файла располагается специальный символ конца файла EOF,
в качестве которого используется символ ASCII 26.
Определить длину файла и произвести некоторые операции можно с помощью стандартных процедур и функций, предназначенных для работы с файлами

Общие процедуры работы с файлами

1)Assign(var F:file;Filename:string);

связывает имя файловой переменной F в программе с именем внешнего (физического) файла на диске, где Filename- выражение строкового типа вида ‘диск:\имя каталога\имя подкаталога\...\имя файла’

Если в параметре Filename имена диска и подкаталога не указаны, то выбирается текущий диск и текущий каталог. После выполнения процедуры Assing все действия над переменной F будут эквивалентны действиям над файлом, определенным спецификацией Filename.

Процедуру Assign необходимо использовать до начала работы с файлами.

2) Reset(var F:file); - открывает физический файл, связанный с файловой переменной F. Если F-текстовый файл, то он будет доступен только для чтения при последовательном доступе. Если F-типизированный файл, то он будет открыт как для чтения, так и для записи, как при прямом, так и при последовательном доступе. При этом указатель файла устанавливается на первый элемент файла и EOF(F)=false. Если файл пустой, то EOF(F)=true.

Если физический файл с указанным именем отсутствует, то возникает ошибка открытия файла. Ее можно подавить включением директивы компилятора {$I-}. Эта директива отменяет генерацию кода, проверяющего результат обращения к процедуре ввода-вывода. При такой установке директивы можно проанализировать результат завершения операции открытия файла с помощью функции IOResult, которая возвращает значение 0, если операция завершилась успешно и ненулевой код ошибки в противном случае.

3)Rewrite(var F:file); - создает новый физический файл, связанный с файловой переменной F для записи. Если такой физический файл уже существует, то он удаляется и на его месте создается новый пустой файл. При открытии указатель текущей позиции в файле устанавливается на его начало, т.е. функция EOF(f)=true.

4) Read(var F:file;var x1,x2,…xn); считывает переменные из файла F в x1,x2,…xn, начиная чтение с элемента, на который указывает текущий указатель. Если имя F опущено, то подразумевается стандартный файл Input. Read выполняется только в том случае, если EOF(f)=false

5) Write(var F:file; var x1,x2,…xn); - записывает одно, или несколько значений переменной в файл F , начиная с той позиции, на которую установлен указатель. Если имя F опущено, то подразумевается стандартный файл Output. Write выполним только в том случае, если Eof(F)=true.

6)Close(var F:file); закрывает физический файл с логическим именем F. Попытка закрыть уже закрытый, или неоткрытый файл приведет к сбою в программе. Особенно важно закрывать файлы, открытые для записи, это гарантирует сохранность информации.

7) Rename(var F;newname:string); - переименовывает физический файл, связанный с логическим именем F в имя newname. Эта функция применима только к закрытым файлам.

8)Erase(var F); - стирает физический файл, связанный с логическим именем F. Применим только к закрытым файлам и только если этот физический файл существует.

Тип типизированного файла представляет собой «шаблон», который накладывается на физический файл для доступа к его элементам. Можно создать файл, используя один шаблон, а прочитать, используя другой. Например, в программе сначала в файл типа Char записывается ряд символов, а затем этот же файл открывается как файл типа byte, и в результате на печать выводятся ASCII коды записанных в него символов.

```
program  CB;
var fc:file of char;
fb:file of byte;
ch:char;
b:byte;
```

```
program  CB;
var fc:file of char;
fb:file of byte;
ch:char;
b:byte;
    begin   assign(fc,'test.dat');
              rewrite(fc);
          for ch:='0' to '9' do write(fc,ch);
             for ch:='A' to 'J' do write(fc,ch);
            close(fc);
         assign(fb,'test.dat');
              reset(fb);
                 while not eof(fb) do
            begin   read(fb,b); write(b);  end;
                       close(fb)
                           end.
```

Пример 1

Вывести на экран элементы файла a:\nambers
и вычислить их количество.

```
var f:file of integer;
n, kol:integer;
begin writeln('элементы файла a:\nambers');
assign(f, 'a:\nambers');
reset(f);  kol:=0;
while not eof(f) do
begin readln(f,n); writeln(n); kol:=kol+1 end;
close(f); write('kol=', kol)
end.
```

Пример 2: Вычислить сумму элементов файла.

```
program fsumma;
var f: file of integer;
x,summa:integer;
begin {$I-}
assign(f,’MyFile.dat’);
reset(f);
if IoResult<>0 then
begin writeln(‘ошибка открытия файла’); halt;
end; summa:=0;
while not eof(f) do
   begin read(f,x); summa:=summa+x; end;
write(‘summa=‘, summa);
close(f); end.
```

Понятие буфера ввода-вывода

С файловой системой связано понятие буфера ввода-вывода. Ввод и вывод данных осуществляется через буфер. Буфер-это область памяти, отводимая при открытии файла. При записи в файл вся информация сначала направляется в
буфер и там накапливается до тех пор, пока весь его объем не будет заполнен.
Только после этого, или после специальной команды сброса буфера происходит передача данных по их

Аналогично, при чтении из файла считывается не столько, сколько запрашивается, а сколько уместится в буфер. Если, например, запрашивается 4 числа, а в буфер умещается 64, то следующие 60 чисел будут считываться уже из буфера. Механизм буферизации позволяет более быстро и эффективно обмениваться информацией. Вывод текстовой информации на экран реализован так, что эффект буферизации исчезает автоматически

Последовательный и прямой доступ к элементам файла

Файл-это последовательная структура данных и естественным способом доступа к файлам является последовательный доступ. После открытия файла он ждет чтения или записи начиная с первого компонента. После каждого следующего обращения к файлу он готов выдать или принять следующий компонент. В таком случае можно только косвенно управлять последовательностью чтения или записи.

Например, пусть нужно прочитать 5 элемент файла f и записать его в переменную v.

```
reset(f);
for i:=1 to 5 do read(f,v);
close(f);
```

В современных версиях Паскаля предусмотрена процедура прямого доступа

Filesize(var f):longint;-возвращает число записей в открытом файле (счет от 1);

filepos(var f):longint;-возвращает номер записи в открытом файле, предшествующей той, которая будет следующей записана или считана (позиции нумеруются от 0)

seek(f,n) – устанавливает в открытом файле номер n компонента, который будет считан или записан при следующей операции ввода-вывода.
trunkate(f)-отсекает часть открытого файла, начиная с той компоненты, которая была бы считана следующим оператором ввода и подтягивает конец файла.

Пример 3

Пусть задан файл f, каждый элемент которого-запись, содержащая фамилию студента и его возраст. Обновить данные этого файла путем увеличения возраста всех студентов на 1 .

```
Program cor;
type zap=record
        fio:string[30];
         voz:0..99
             end;
var f:file of zap;
      z:zap,i:integer;
     begin  assign(f,'a:\file');
reset(f);
      i:=0;
    while not eof(f) do
     begin read(f,z);
         z.voz:=z.voz+1;
         seek(f,i);write(f,z);
           i:=i+1; end;
            close(f)
               end.
```

Добавление данных к файлу

Пусть необходимо к уже существующему файлу добавить ряд элементов. Для этого необходимо выполнить следующие действия:

1)Открыть уже существующий файл процедурой reset;

2)установить указатель за последним его компонентом процедурой

seek(f,filesize(f));

3)записать дополнительные данные процедурой write

4)закрыть файл процедурой close.

Пример 4Пусть уже создан файл f, состоящий из целых чисел 1,2,3,4,5. Необходимо добавить к этому файлу числа 10,20,30

```
program pr2;
var f: file of integer;
      i,x:integer;
 begin assign(f,'f'); reset(f);
    seek(f,filesize(f));
     for i:=1 to 3 do
         begin x:=10*I;
               write(f,x); end;
                  close(f)
                        end.
```

Прямая выборка из элементов файла

Для этого используется процедура seek(f,n), где f- имя файла, который уже открыт процедурой reset, n-целое положительное число или выражение, соответствующее порядковому номеру элемента в файле. После выполнения процедуры seek первое обращение к операторам read или write будет связано n-ым элементом файла. Каждое повторное использование этих операторов будет осуществлять последовательное чтение или запись элементов, начиная с n+1.

Используя процедуру seek можно корректировать отдельные элементы файла непосредственно.
Для этого нужно выполнить следующие действия
1)открыть корректируемый файл (reset)
2)подвести указатели файла к корректируемому элементу (с помощью seek)
3)считать корректируемый элемент;
4) откорректировать элемент файла;
5)повторить процедуру подвода указателя файла (seek);
6)записать откорректированный элемент(write);
7) закрыть файл(close)

Пример 5

Пусть был создан файл f, содержащий числа 1,2,3,4,5. Необходимо вместо последних трех вставить 30,40,50.

```
program pr3;
var f: file of integer;
       x,i:integer;
begin assign(f,'f'); reset(f);
      for i:=3 to 5 do begin
       seek(f,i-1); read(f,x); x:=x*10;
          seek(f,i-1); write(f,x); end;
close(f);
      reset(f);
        while not eof(f) do
            begin  read(f,x);  writeln(x)  end;
                   close(f)  end.
```

Пример 5

Отсортировать элементы файла.

```
program sort;
var f:file of integer;
        x,y,i,j: integer;
begin assign(f,'sort.dat'); reset(f);
writeln('исходный файл');
for i:=1 to filesize(f) do
begin  read(f,x); write(x); end; close(f);
```

```
{сортировка}
reset(f);
for i:=filesize(f)-1 downto 1 do
    for j:=0 to i-1  do
begin    seek(f,j); read(f,x,y);
       if x>y then
            begin   seek(f,j);
                    write(f,y,x);end;
                                    end; close(f);
reset(f);
writeln('отсортированный файл');
for i:=1 to filesize(f) do
begin  read(f,x); write(x); end; close(f); end.
```

Структура текстового файла

Особый вид файла представляют собой текстовые файлы, которые являются разновидностью файлов file of char.

Для их описания используется тип text:

file Textfile: text;

В текстовых файлах помимо признака конца файла eof используется признак конца строки eoln. Текстовый файл можно представить как страницу книги:

Код символа	Код символа	eoln		
Код символа	Код символа	Код символа	eoln	
Код символа	Код символа	eof		

Текстовые файлы являются файлами с последовательным доступом. В любой момент времени доступна только одна запись файла. Другие записи становятся доступными лишь в результате последовательного продвижения по файлу. Текстовые файлы внутренне разделены на строки, длины которых различны. Обрабатывать их можно только последовательно с помощью процедур и функций:

Readln (f , st)- чтение строки st из файла f и переход на начало следующей ;

Writeln (f, st)- запись строки st в файл f и маркера конца строки ;

Можно использовать и процедуры Read (f , st)- если не надо переходить на начало следующей строки и Writeln (f,st)- если не надо переводить строку.

Открытие текстового файла можно произвести двумя стандартными способами:

- поставить в соответствие файловой переменной имя файла (процедура Assign), открыть новый текстовый файл (процедура Rewrite);

- поставить в соответствие файловой переменной имя файла (процедура Assign), открыть уже существующий файл (процедура Reset).

Текстовый файл в силу своей специфики во время работы допускает только один вид операции: чтение или запись. В связи с этим для работы с текстовыми файлами используется еще одна процедура открытия файла:
Append (f) - Эта процедура открывает уже существующий файл и позиционирует указатель обработки на конец файла. После этого в текстовый файл можно только добавлять информацию, причем только в конец файла.
Eoln (st)- логическая функция, результат выполнения которой равен TRUE, если достигнут маркер конца строки st.
Процедура SetTextBuf(f; Buf[;Bufsize]) устанавливает размер буфера для файла равным Bufsize байт. Должна выполняться перед открытием файла f.
Процедура Flush(f) выводит содержимое буфера в физический файл не дожидаясь его заполнения. Имеет смысл только для записи в файл.

Пример 1 Создать на диске а текстовый файл и записать в него 5 чисел.

```
  var f:text;
n:integer;   {числа}
 i:integer;   {счетчик чисел}
begin
   writeln('Создание файла. Введите 5 чисел');
    Assign(f,'a:\numbers.txt');
      rewrite(f);
       for i:=1 to 5 do
          begin  readln(n);
                    writeln(f,n);  end; close(f);
                write('числа записаны в файл 'a:\numbers.txt'')
                          end.
```

Пример 2 Дописать в файл a:\phone.txt ФИО и № телефона абонента. Если файла нет на диске, то создать его.

```
program spr;
var f:text;
fam: string[15];
name: string[15];
tel:string[10];
    begin
    writeln('добавить в телефонный справочник');
    assign(f,' a:\phone.txt');
     {$I-}
     Append(f);
      if IOResult<>0 then {вероятно файла на диске нет}
      begin   rewrite(f);
   if IOResult<>0 then  begin writeln('ошибка обращения к файлу'); halt end;
end;
write('фамилия');readln(fam); {вводим данные}
write('имя');readln(name);
write('телефон');readln(tel);
writeln(f,fam); writeln(f,name); writeln(f,tel);   {записываем данные в файл}
        writeln('информация добавлена')
                    end.
```

```
Program primer;
Var f1,f2:text;
   I,n: integer;
   S: string;
Begin   {формируем первый файл}
   Assign(f1, 'file1.txt');
   Rewrite(f1);  {открываем файл для записи}
   Readln(n) {определим количество вводимых строк}
   for i:=1 to n do
   begin
      readln(s); {вводим с клавиатуры строки}
      writeln(f1,s); {записываем последовательно
строки в файл}
   end;
   close(f1);
```

```
{часть вторая: чтение из первого файла и формирование второго}
   Reset(f1); {открываем первый файл для чтения}
  Assign(f2, ‘file2.txt’);
   Rewrite(f2); {открываем второй файл для записи}
   While not eof(f1) do
   Begin Readln(f1,s);{считываем очередную строку из первого файла}
       If (s[1]=’A’) or (s[1]=’a’) then
       Writeln(f2,s); {записываем во второй файл строки, удовлетворяющие условию}
   End;
   Close(f1,f2); {заканчиваем работу с файлами}
   Writeln(‘Второй файл содержит строки:’);
   Reset(f2); {открываем второй файл для чтения}
   While not eof(f2) do {пока не конец второго файла}
   Begin  Readln(f2,s);{считываем очередную строку из второго файла}
       Writeln(s); {выводим строку на экран}
   End;
End.
```

Бестиповые (нетипизированные) файлы

Бестиповые **файлы предназначены для записи участков памяти компьютера на внешний носитель и считывания их в память. В отличие от типизированных файлов, нам не нужно знать информация какого типа хранится в них. А все потому, что данные из файлов, не имеющих типа, считываются в виде байт, после чего они «подстраиваются» под переменную, в которую происходит считывание.**

Общая форма записи нетипизированных файлов

Var <идентификатор>: File;

отличается от типизированных отсутствием части of <тип данных>. Кроме того, немного изменяется принцип действия процедур Reset и Rewrite. К ним прибавляется второй параметр типа Word:

reset(<имя файловой переменной>, <значение>)

rewrite(<имя файловой переменной>, <значение>)

Здесь «*значение*» — это новый размер буфера, который по умолчанию равен 128 байтам. В качестве минимального значения можно указать 1 байт, а максимального — 64 кбайт (число в байтах).

Также в бестиповых файлах для записи и чтения информации используются не стандартные процедуры Read и Write, а две новые: BlockRead и BlockWrite. Рассмотрим каждую из них.

Процедура BlockRead

Данная процедура считывает из файла заданное число записей, которые помещаются в память.

Общая форма записи:

BlockRead(<имя файловой переменной>, <x>, <количество байт>);

x, y – обычные переменные, в первую помещается прочитанные данные, во вторую – количество считанных байт.

Процедура BlockWrite

Для записи информации в бестиповый файл предназначена процедура BlockWrite:

BlockWrite(<имя файловой переменной>, <x>, <количество байт>);

Параметры процедуры BlockWrite точно такие же, как и у BlockRead. Да и принцип их одинаков, за исключением того, что первая считывает данные из файла, а вторая записывает их в него.

ПРИМЕР

```
program no_type_fail;
var
f: file;
x, i ,n: byte;
begin
assign(f, 'f');
rewrite(f, 1);
write('n = '); readln(n);
for i:=1 to n do
begin
x:=n-i;
blockwrite(f, x, 1); {запись в файл}
end; close(f);
reset(f, 1);
while not eof(f) do
begin
blockread(f, x, 1); {чтение из файла}
write(x,' ');
end;
close(f);
end.
```