ЛАБОРАТОРНАЯ РАБОТА N 9

Тема: " Работа с файлами"

Цель работы

1. Изучение файловых типов в языке Паскаль.
2. Получение навыков в организации файлов и использовании их для обработки информации.

Краткие сведения из теории

До сих пор данные вводились в программу через клавиатуру, т.е. с непременным участием человека. Такой способ ввода информации называется интерактивным режимом ввода. Возможен и иной подход, основывающийся на использовании набора данных, подготовленных заранее и хранящихся в виде файла на магнитном носителе. Иначе говоря, альтернативой интерактивному режиму является такой способ ввода, при котором информация поступает из источника, физически существующего вне программы. Этот процесс обычно называют считыванием данных из внешнего файла (или просто из файла). Указанный способ находит широкое применение при обработке информационных массивов весьма значительного объема, когда интерактивный режим становится слишком обременительным и малоэффективным. Вторым мотивом использования файлов является то, что он может быть создан какой-то другой программой. Таким образом, файл становится связующим звеном между разными задачами. Наконец, последнее немаловажное соображение: если входные данные поступают в программу из внешнего файла, то присутствие пользователя в момент фактического исполнения программы становится необязательным.

Файл - это именованная область внешней памяти компьютера (жесткого диска, гибкой дискеты,...), содержащая логически связанную совокупность данных.

На языке Паскаль можно создавать три типа файлов : текстовый, типизированный, нетипизированный, которые в программе (в разделе VAR) объявляются следующим образом:

< файловая переменная > : text;
< файловая переменная > : file of < тип компоненты >;
< файловая переменная > : file.
< файловая переменная > - это логическое имя файла, используемое в программе.
< тип компоненты > - компонентой файла может быть как переменная базового типа, так и структурного. Структурный тип определяет данные типа "запись" (record).

"Запись" - это логически связанная совокупность нескольких данных. К данным типа RECORD можно обращаться как ко всей совокупности данных, так и к ее отдельным данным (элементам). Для обращения к отдельным элементам "записи" используется уточненное имя.

Уточненное имя состоит из идентификатора "записи", десятичной точки и идентификатора элемента записи. В свою очередь каждый элемент "записи" может быть "записью". Тогда, для обращения к внутреннему элементу необходимо последовательно перечислить через десятичную точку все идентификаторы иерархически вложенных "записей", начиная от внешнего имени к внутреннему, последним в этой последовательности является идентификатор самого элемента.

При использовании структурного типа компоненты "запись" необходимо объявлять его в программе в разделе TYPE.

Пример:

```
TYPE
{ запись данных по студенту }
RecFile = RECORD { признак начала записи }
        Fam, Name, Otch       : string[15];
```

```
        GodR            : word;
        Ngrup           : string[10];
END;   { конец записи }
VAR
        F1  :  Text;
        F2  :  File of byte;
        F3  :  File of RecFile;
        F4  :  File;
        Buf :  RecFile; {_2буфер ввода-вывода_0, в который считываются  данные компоненты
        файла}
```

В этом фрагменте программы F1- файловая переменная текстового типа, F2 и F3 - файловые переменные типизированного типа, причем F2 может содержать только байтовые компоненты, а в файле F3 каждая компонента представляет из себя "запись" из трех элементов, F4- файловая переменная нетипизированного типа.

Уточненные имена элементов записи Buf: Buf.Fam, Buf.Name, Buf.Otch, Buf.GodR, Buf.NGrup.

12.1 Доступ к файлам

В первую очередь при работе с файлами необходимо привести в соответствие файловую переменную и имя файла, с которым он хранится на внешнем носителе. С этим именем работает операционная система(ОС) DOS. Соответствие устанавливается с помощью процедуры:

ASSIGN (< ф.п.>, < имя файла или л.у.>);

Здесь < ф.п.> - файловая переменная; < имя файла или л.у.> - это полное имя файла, которое включает в себя путь доступа, непосредственно имя файла и расширение. "л.у." - стандартное логическое устройство.

Например:

```
ASSIGN (F1, 'a:\Tp5\DAT\St629.DAT');
ASSIGN (F2, 'Dannye.DAT').
```

Если не указан путь к файлу, то запись или считывание осуществляется с текущего директория или в текущий директорий.

В качестве имени файла в процедуре ASSIGN можно указывать логическое устройство из следующего списка: CON, PRN, AUX.

CON - это имя консоли. На персональном компьютере (ПК) под консолью понимается совокупность двух устройств: клавиатуры и дисплея. Клавиатура используется для ввода информации, а дисплей для вывода.

PRN - это стандартное имя принтера. В ОС PRN стандартно назначается LPT1. В модуле Printer Паскаля объявлена файловая переменная Lst. Поэтому при отображении данных на принтер, достаточно подключить модуль Printer к программе, а в предложениях Write и Writeln первым аргументом записать имя файловой переменной Lst:

Write (Lst, < список выводимых данных >);

Пример вывода информации без использования модуля Printer :

```
...
VAR F : File;
BEGIN
        Assign(F, PRN);
        ReWrite(F);
        Writeln(F, 'Пример использования Л.У. - PRN');
        Close(F);
END;
```

AUX - это имя коммуникационного порта. Обычно их бывает два у ПК: COM1 и COM2. Стандартно AUX назначается COM1. Этот порт обычно используется для подключения нестандартных устройств. Например, "мыши", дигитайзера, графопостроителя и т.п.

12.2 Инициация файла и завершение работы с ним

Прежде чем начать обработку файла необходимо выполнить некоторые операции по работе с устройством, на котором хранится или будет храниться файл. Так например, при создании файла необходимо:

выделить область памяти на внешнем устройстве, в которую будут записываться данные файла;

запомнить имя файла и адрес этой области.

Если предстоит работа с файлом, уже существующим на внешнем носителе, то необходимы следующие действия:

по указанному имени файла найти адрес, с которого записаны данные этого файла;

установить головку устройства на начало файла.

Эта совокупность операций называется инициацией файла или "открытием" файла.

Инициируется файл с помощью процедур Reset и ReWrite. С помощью процедуры Reset инициируется, т.е. открывается ранее созданный файл. С помощью процедуры ReWrite инициируется файл для записи, то есть вновь создаваемый файл.

Синтаксис:

Reset(< ф.п.> [,< размер записи в байтах >]);

ReWrite(< ф.п.> [,< размер записи в байтах >]);

Второй аргумент указывается только для нетипизированных файлов. Текстовые файлы можно инициировать также и процедурой Append:

Append(< ф.п.>);

В этом случае ранее созданный файл открывается для добавления данных в конец файла.

Завершив работу с файлом, необходимо его закрыть. При закрытии файла ОС подсчитывает размер файла в байтах и запоминает его. Кроме того, запоминается также информация о дате и времени создания файла или его последней модификации (корректировки).

Закрытие файла данных осуществляется процедурой Close:

Close(< ф.п.>);

При считывании данных из ранее созданного файла конец файла можно определить с помощью функции EOF:

EOF(< ф.п.>);

Эта функция имеет значение TRUE при считывании маркера конца файла. В противном случае она будет иметь значение FALSE. Данная функция обычно используется для организации цикла по чтению всех компонент файла:

```
...
while not EOF(F1) do begin
        ...
        < считывание и обработка компонент файла >
        ...
end;
```

12.3. Считывание данных из файла и запись их в файл

Непосредственный ввод информации осуществляется предложениями READ и READLN, а вывод (запись) информации - WRITE и WRITELN. Особенностью их применения к файлу является обязательность указания файловой переменной в качестве первого параметра в списке элементов ввода или вывода:

Read(< файловая переменная >, < список ввода >);

ReadLn(< файловая переменная >, < список ввода >);

Write(< файловая переменная >, < список вывода >);

WriteLn(< файловая переменная >,< список вывода >).

12.4. Текстовые файлы

Текстовый файл - это совокупность строк переменной длины. Переменная длина строк определяет наличие маркеров, которые отмечают конец строки. В качестве маркеров используются два управляющих символа "Перевод строки" и "Возврат каретки", их десятичные коды: #10, #13. Названия управляющих символов "Перевод строки"(LF - Line Feed) и "Возврат каретки"(CR - Carriage Return) взяты по аналогии работы с пишущей машинкой.

Конец строки можно определить с помощью функции EOLn:

EOLn(< ф.п.>);

Для записи данных в файл используются процедуры WRITE и WRITELN:

Write(< ф.п.>, < список вывода стрингов >);

Writeln(< ф.п.>, < список вывода стрингов >).

По предложению WRITE значения данных из списка запишутся в файл подряд без всяких разделителей. Поэтому программист, используя предложение WRITE, должен позаботиться о разделителях между данными, если они нужны.

По предложению WRITELN в файле после каждого выведенного стрингового значения будут записаны признаки конца строки.

Для чтения данных из файла используются процедуры READ и READLN:

Read(< ф.п.>, < список вводимых стрингов >);

Readln(< ф.п.>, < список вводимых стрингов >);

По предложению READ из файла выбирается столько символов, сколько указано в описании текущего стринга, принадлежащего списку ввода. Выбранная последовательность символов присваивается текущему стрингу. Эта совокупность операций повторяется для всех элементов списка ввода. По предложению READLN из файла последовательно считываются строки и присваиваются стрингам из списков. Если выбранная строка имеет большее количество символов, чем указано в описании текущего стринга, то она обрезается до указанной длины, при этом часть информации теряется. Поэтому необходимо следить за соответствием длин стрингов, записываемых в файл и считываемых из файла.

Пример:

```
...
Var
        Fio, Otch   : string[15];
        Name        : string[10];
        i           : integer;
        F           : text;
 Begin
        Assign(F, 'St629.DAT');  { файл будет создаваться  в текущем каталоге }
        { создание файла или первичная запись данных в файл}
        ReWrite(F);  { открытие файла для записи }
        for i:=1 to 5 do  { ограничимся вводом пяти студентов }
        begin
                {Ввод данных: F, Fam, Name,
                Otch}
        end;
        close(F);
        { чтение данных из файла и вывод их на экран }
        WriteLn('  Фамилия      Имя      Отчество');
        Reset(F);  { открытие существующего файла }
        for i:=1 to 5 do
        begin
```

```
            Read(F, Fam, Name, Otch);
            {Вывод данных: Fam: 16, Name:11, Otch:15}
    end; close(F);

End.
```

12.5. Типизированные файлы

Компоненты этого файла могут быть следующих типов:
базового: byte, word, longint, integer, real, запись, char, string;
структурного;
регулярного.
При этом все компоненты файла имеет один и тот же тип. Это означает, что длина компоненты фиксирована.
Объявляется такой файл в программе следующим образом :

```
Var
    F1 : File of byte;
    F2 : File of string[80];
    F3 : File of real;
    . . .
    F  : File of RecFile;
```

Здесь F1, F2, F3, F - это файловые переменные, которые указывают на файлы, компонеты которых соответственно являются типа byte, string, real и record.
Чтение компонент файла выполняется процедурой:

```
    Read(< ф.п.>, < список ввода >);
```

Запись компонент в файл выполняется процедурой:

```
    Write(< ф.п.>, < список вывода >);
```

Пример:

```
Var
    X, Y : array[1..100] of integer; { массивы координат }
    F    : file of real;
    i    : byte;
Begin
    . . .
    < операторы по вводу 100 значений координат X, Y >
    . . .
    Assign(F, 'Coor.dat');   { файл будет создаваться в текущем каталоге }
    ReWrite(F);   { открытие файла для записи }
    For i:= 1 to 100 do
    Write(F, X[i], Y[i]);   { запись координат в файл }
    Close(F);
End.
```

В приведенном фрагменте программы координаты записаны последовательными парами X, Y. При такой организации файла происходит частое обращение к внешнему носителю, это приводит к замедлению работы программы, что особенно заметно при работе с большими объемами данных. Поэтому рекомендуется данные записывать в файл и считывать из файла большими блоками, примерно кратными 512 байтам.
Согласно этому модифицируем программу следующим образом:

```
Type
    Coord = array[1..100] of integer; { массивы координат }
    . . .
Var
```

```
        X, Y : Coord; { массивы координат }
        F    : file of Coord;  { файл регулярного типа }
        i    : byte;
Begin
        . . .
        < операторы по вводу 100 значений координат X, Y >
        . . .
        Assign(F, 'Coor.dat');   { файл будет создаваться в текущем каталоге }
        ReWrite(F);   { открытие файла для записи }
        Write(F, X);   { запись массива координат X в файл }
        Write(F, Y);   { запись массива координат Y в файл }
        Close(F);
End.
```

Теперь в файле сначала записаны 100 координат X, а затем 100 координат Y. Данные файла мы записали двумя большими блоками по 600 байтов каждый. Следует помнить, что это не самый лучший способ организации файла для данных примера, но достаточно понятный.

Считывание координат из файла выполняется аналогично, в программе нужно вместо процедуры REWRITE использовать процедуру RESET, а вместо предложения WRITE использовать предложение READ.

Хорошо структурируемые данные, например, данные о каком-либо объекте, удобно описывать типом "запись". В этом случае компонента файла будет структурного типа.

Пример:

```
Type
        RecFile = record    { запись данных по студенту }
                Fam, Name, Otch : string[15];
                GodR            : word;
                NGrup           : string[10];
        end;
Var
        i       : integer;
        Buf     : RecFile;
        FilStud : file of RecFile;
Begin
        Assign(FilStud, 'Stud.dat');
        ReWrite(FilStud);
        {Введите данные по студентам:}
        For i:= 1 to 10 do { ограничимся 10 записями }
        begin
        { интерактивный ввод данных }
                Фамилия      : Buf.Fam
                Имя          : Buf.Name
                Отчество     : Buf.Otch
                Год рождения : Buf.GodR
                N группы     : Buf.NGrup
        { запись данных в файл }
                Write(FilStud, Buf); { Buf - обращение ко всей записи }
        end;
        Close(FilStud);
End.
```

Чтение компонент типизированного файла можно осуществлять как последовательным, так и прямым методом доступа.

Последовательный доступ - это есть доступ к компоненте файла только после перебора всех предыдущих.
Прямой доступ - это есть доступ сразу к указанной компоненте.
Так как типизированные файлы обладают компонентами фиксированной длины, существует возможность организовать прямой доступ. Для организации прямого доступа к компонентам файла существуют стандартные процедуры Seek, FilePos, FileSize :

Seek(< файловая переменная >,< номер компоненты >);
FilePos(< файловая переменная >);
FileSize(< файловая переменная >).

Процедура Seek осуществляет прямой доступ к любой компоненте файла.
Здесь < номер компоненты > - позиция указателя компонент файла.
Она может принимать следующие значения:

+1 - установить указатель на следующую компоненту;
-1 - установить указатель на предыдущую компоненту;
i - установить указатель на i-ую компоненту.

Процедура FilePos определяет номер текущей позиции в файле, а точнее номер текущей компоненты.
Процедура FileSize определяет размер указанного файла - количество компонент.
Нумерация компонент начинается с нуля.

Пример.

```
Type
  RecFile = record    { запись данных по студенту }
              Fam, Name, Otch : string[15];
              GodR            : word;
              NGrup           : string[10];
            end;
Var
  i       : integer;
  Buf     : RecFile;
  FilStud : file of RecFile;
Begin
  Assign(FilStud, 'Stud.dat');
  Reset(FilStud);
  i:= FileSize;
  WriteLn('В файле ', i, ' компонент');
  Seek(FilStud, i-1);  { встали перед последней записью }
  Read(FilStud, Buf);  { прочитали ее }
  { можно ее скорректировать и записать вновь в файл    }
  Buf.GodR:= '1973';
  { перед записью нужно вновь установить указатель перед этой
  записью   }
  Seek(FilStud,   -1);
  Write(FilStud,   Buf);
  Close(FilStud);
End.
```

Примечание: Открывая типизированный файл процедурой RESET, можно этот файл не только читать, но и записывать в него новую информацию. При этом файл должен уже существовать на диске.

12.6. Нетипизированные файлы

Нетипизированные файлы могут содержать в своем составе любые типы компонент. При этом правильность записи и считывание этих компонент полностью возлагается на

программиста. Длина компонент может быть различной. Для открытия нетипизированных файлов используются процедуры Reset, ReWrite:
Reset(< файловая переменная >, < max размер буфера >);
ReWrite(< файловая переменная >, < max размер буфера >).
Так как за одно обращение к нетипизированному файлу можно считывать не одну компоненту, а несколько, и так как длины компонент могут быть различны, то в процедурах Reset и ReWrite указывается максимальный размер буфера ввода-вывода в байтах.
Чтение компонент из файла и запись их в файл выполняется процедурами BlockRead и BlockWrite:
BlockRead(< файловая переменная >, < буфер >,< кол-во компонент, считываемых за один раз >, [,< кол-во считанных компонент >]);
BlockWrite(< файловая переменная >,< буфер >,< кол-во записываемых компонент >, [,< кол-во записанных компонент >]).
Четвертый параметр необязателен. Он формируется системой и используется для проверки правильности завершения операций чтения или записи.
Нетипизированные файлы рекомендуется использовать для организации эффективной работы с файлами, так как они позволяют работать, во-первых, с компонентами различной длины, и, во-вторых, с переменным числом обрабатываемых компонент.

2.7. Процедуры и функции для работы с файлами

Все рассматриваемые функции и процедуры принадлежат стандартному модулю DOS, поэтому его необходимо подключить к программе с помощью предложения USES.
1. ReName(< файловая переменная >,< новое имя файла >) - переименование файла.
2. Erase(< файловая переменная >) - удаление файла.
3. ChDir(< путь >) - изменение директория, где <путь> - путь к новому директорию.
4. GetDir(< устройство >, < директорий >) - определение текущего каталога, где <устройство> задается следующим образом:
0 - текущее устройство;
1 - устройство A;
2 - устройство B и т.д.
5. MkDir(< директорий >) - создание нового каталога. В аргументе <директорий> указывается полный путь до того каталога, который создается.
6. PmDir(< директорий >) - удаление каталога. В качестве аргумента указывается полный путь до удаляемого каталога. При этом удаляемый каталог должен быть обязательно пустым.
7. IOResult - проверка правильности завершения работы той или иной операции ввода-вывода. Эта функция имеет тип WORD и возвращает значение 0, если операция ввода-вывода выполнилась успешно, и в противном случае следующие значения:
1 - файл не найден,
2 - путь не найден,
3 - слишком много открытых файлов,
5 - запрет доступа к файлу,
12 - некорректный код доступа к файлу
и так далее.
При применении этой функции в программе необходимо с помощью директивы компилятора отключить стандартную проверку - {$I-}, а после выполнения операций ввода-вывода включить - {$I+}.
Данная функция записана в стандартном модуле SYSTEM.
8. DiskFree(< устройство >) - определение числа свободных байтов на заданном диске. Эта функция типа LONGINT.

В качестве аргумента указывается номер устройства. Если указано несуществующее устройство, то вместо объема свободной памяти на диске эта функция возвращает значение -1. Функцию рекомендуется применять перед созданием файла, чтобы выяснить, достаточно ли места для создаваемого файла на указанном накопителе.
9. DiskSize(< устройство >) - определение числа свободных байтов на диске. Тип функции LONGINT. Аргумент задается так же, как и в предыдущей функции.
10. FindFirst(< уточненное имя файла>, < атрибуты >, < доп.инф-я >) - поиск указанного файла.
В процедуре входным параметром является только первый. Два последних параметра являются выходными. Параметр < атрибуты > имеет тип BYTE, параметр < дополнительная информация > должен быть объявлен как SearchRec. Этот тип описан в стандартном модуле Dos.
11. FindNext(< следующий файл >) - поиск указанного файла. Процедуры FindFirst и FindNext зачастую используются для просмотра всех файлов, находящихся в каталоге.
12. FSearch(< имя файла>,<список каталогов >) - поиск файла в списке каталогов. Функция имеет тип PathStr (описана в стандартном модуле Dos).
13. FSplit(< уточненное имя файла >, < путь >,< имя >,< расширение >) - выделение из уточненного имени файла трех переменных:
< путь >, < имя файла >, < расширение >.
14. FExpand(< имя файла >) - добавление к имени файла, находящегося в текущем каталоге, полного пути доступа к нему.

Примечание: перед использованием первых четырех процедур файл должен быть обязательно закрыт.

Задание к работе

Задание А. Разработать программу в соответствии с вариантом задания, которая должна выполнять следующие функции:

создание файла;

чтение данных из файла;

вывод считанных данных на экран дисплея.

Задание Б. В программу, разработанную по заданию А, добавить блок обработки данных, инцидентных файлу, в соответствии с индивидуальным заданием. Все полученные результаты отобразить на экране.

Методические указания

1. При разработке процедуры создания файла необходимо придерживаться следующей схемы действий:
   a) проверить с помощью процедуры DISKSIZE, есть ли место на диске;
   b) проверить, нет ли файла с таким же DOS - им именем на диске (процедура FINDFIRST, FINDNEXT);
   c) привести в соответствие DOS - ое имя файла с файловой переменной, используемой в программе (процедура ASSIGN);

d) открыть файл (процедура REWRITE);
e) ввести данные, предназначенные для записи в файл;
f) записать данные в файл (предложения WRITE / WRITELN);
g) закрыть файл (процедура CLOSE).

2. При создании процедуры чтения необходимо:
   a) проверить, существует ли такой файл на диске (процедура FINDFIRST, FINDNEXT);
   b) если файл не существует, то необходимо уточнить имя в интерактивном режиме и снова перейти к пункту a);
   c) если файл существует, привести в соответствие DOS - ое имя файла с файловой переменной, используемой в программе (процедура ASSIGN);
   d) открыть файл (процедура RESET);
   e) считать данные из файла (предложения READ / READLN);
   f) отобразить считанные данные на экране дисплея;
   g) закрыть файл (процедура CLOSE).
3. В начале каждой процедуры необходимо:
   a) отключить стандартную проверку выполнения операций ввода-вывода, используя директиву компилятора {$I-};
   b) после выполнения каждой операции ввода-вывода самостоятельно проверять код ее завершения с помощью функции IORESULT;
   c) при неуспешном завершении операции ввода-вывода устранить причину , приведшую к этой ситуации.

Варианты индивидуальных заданий

Вариант 1

А. Создать файл, содержащий сведения о месячной заработной плате рабочих завода. Каждая запись содержит поля - фамилия рабочего, наименование цеха, размер заработной платы за месяц. Количество записей - произвольное.

Б. Вычислить общую сумму выплат за месяц по цеху X, а также среднемесячный заработок рабочего этого цеха. Вывести ведомость для начисления заработной платы рабочим этого цеха.

Вариант 2

А. Создать файл, содержащий сведения о количестве изделий, собранных сборщиками цеха за неделю. Каждая запись содержит поля - фамилия сборщика, количество изделий, собранных им ежедневно в течение шестидневной недели (в понедельник, вторник и т.д.). Количество записей - произвольное.

Б. По каждому сборщику просуммировать количество деталей, собранное им за неделю. Определить сборщика, собравшего наибольшее число изделий, и день, когда он достиг наивысшей производительности труда.

Вариант 3

А. Создать файл, содержащий сведения о количестве изделий категорий A, B, C, собранных рабочим за месяц. Структура записи имеет поля - фамилия сборщика, наименование цеха, количество изделий по категориям, собранных рабочим за месяц. Количество записей - произвольное.
Б. Считая заданными значения расценок Sa, Sb, Sc за выполненную работу по сборке единицы изделия категорий A, B, C соответственно, подсчитать:
- общее количество изделий категорий A, B, C, собранных рабочим цеха X;
- ведомость заработной платы рабочих цеха X;
- средний размер заработной платы работников этого цеха.

Вариант 4
А. Создать файл, содержащий сведения о телефонах абонентов. Каждая запись имеет поля - фамилия абонента, год установки телефона, номер телефона. Количество записей - произвольное.
Б. По вводимой фамилии абонента выдать номер телефона. Определить количество установленных телефонов с XXXX года. Номер года вводится с терминала.
Вариант 5
А. Создать файл, содержащий сведения об ассортименте игрушек в магазине. Структура записи - название игрушки, цена, количество, возрастные границы, например 2 ÷ 5, т.е. от двух до пяти лет. Количество записей - произвольное.
Б. Найти игрушки, которые подходят детям от 1 до 3 лет. Определить стоимость самой дорогой игрушки и ее наименование. Определить игрушку, которая по стоимости не превышает X руб. и подходит ребенку в возрасте от A до B лет. Значения X, A, B ввести с терминала.
Вариант 6
А. Создать файл, содержащий сведения о сдаче студентами первого курса сессии. Структура записи - индекс группы, фамилия студента, оценки по пяти экзаменам, признак участия в общественной работе: "1" - активное участие, "0" - неучастие. Количество записей - 30.
Б. Зачислить студентов группы X на стипендию. Студент, получивший все оценки "5" и активно участвующий в общественной работе, зачисляется на повышенную стипендию (доплата 50 %), не активно участвует - доплата 25 %. Студенты, получившие "4" и "5" , зачисляются на обычную стипендию. Студент, получивший одну оценку "3", но активно занимающийся общественной работой, также зачисляется на стипендию, в противном случае зачисление не производится. Индекс группы вводится с терминала.
Вариант 7
А. Создать файл, содержащий сведения о сдаче студентами сессии. Структура записи - индекс группы, фамилия студента, оценки по пяти экзаменам и пяти зачетам ("З" означает зачет, "Н" - незачет). Количество записей - 25.
Б. Определить фамилии неуспевающих студентов с указанием индексов групп и количества задолженностей. Найти средний балл, полученный каждым студентом группы X, и всей группой в целом.
Вариант 8
А. Создать файл, содержащий сведения о личной коллекции книголюба. Структура записи - шифр книги, автор, название, год издания, местоположение (номер стеллажа, шкафа и т.д.). Количество записей - произвольное.
Б. Найти:
1) местонахождение книги автора X названия Y;
2) список книг автора Z, находящихся в коллекции;
3) число книг издания XX года, имеющееся в библиотеке.

Значения X, Y, Z, XX ввести с терминала;
Вариант 9

А. Создать файл, содержащий сведения о наличии билетов и рейсах Аэрофлота. Структура записи - номер рейса, пункт назначения, время вылета, время прибытия, количество свободных мест в салоне. Количество записей - произвольное.
Б. Найти время отправления самолетов в город X; наличие свободных мест на рейс в город X с временем отправления Y.
Значения X, Y вводятся по запросу с терминала.
Вариант 10
А. Создать файл, содержащий сведения об ассортименте обуви в магазине фирмы. Структура записи - артикул, наименование, количество, стоимость одной пары. Количество записей - произвольное. Артикул начинается с буквы Д для дамской обуви, М для мужской, Р для детской.
Б. Определить наличие в файле обуви артикула X, узнать ее стоимость; ассортиментный список дамской обуви с указанием наименования и имеющегося в наличии числа пар каждой модели.
Значение X вводится по запросу с терминала.