Кафедра конструирования и производства радиоаппаратуры

УТВЕРЖДАЮ
Заведующий кафедрой КИПР
_____**В.Н. ТАТАРИНОВ**
“___” _____2012 г.

Программирование с использованием файлов

Лабораторная работа по дисциплинам ***«***Информатика***»*** для студентов специальностей 211000.62 «Конструирование и технология электронных средств» (бакалавриат) и 162107.65 «Информатика и информационные технологии» (специалитет)

Разработчик:
Доцент кафедры КИПР
_____**Ю.П. Кобрин**

Томск 2012

Оглавление

1 Цели работы

- Изучение возможностей использования файлов в ***Borland Pascal***.
- Закрепление практических навыков работы с записями в ***Borland Pascal***.
- Приобретение практических навыков в работе с файлами последовательного и прямого доступа в ***Borland Pascal***.

2 Порядок выполнения работы

В ходе выполнения этой работы следует:

1. Изучить описание лабораторной работы, обратив особое внимание на правила описания и использования файлов в ***Borland Pascal***. При необходимости использовать дополнительную литературу [1 - 9].
2. Ответить письменно на контрольные вопросы.
3. Войти в свой личный каталог, и настроить интегрированную среду ***Borland Pascal*** для последующей работы. Записать файл конфигурации в личный каталог.
4. В качестве индивидуального задания надлежит провести модернизацию разработанной на предыдущем занятии программы, прибавив возможность хранения данных в файле на диске.
5. Внимательно проштудировать учебный пример и по возможности максимально использовать его в своей задаче.
6. Ввести и отладить разработанную программу.
7. Продемонстрировать работоспособность программы для различных вариантов исходных данных.
8. Оформить отчет по лабораторной работе и защитить его у преподавателя.

3 Контрольные вопросы

Ответьте письменно на следующие контрольные вопросы:

1) Что называется файлом?
2) В чем различие между структурой логического и структурой физического файла?
3) В чем сходство и различие между массивом и файлом?
4) Что нужно выполнить для открытия файла?
5) Какие процедуры предназначены для открытия файлов и как они работают?
6) Для чего предназначена процедура ***Close***?
7) По каким признакам классифицируются файлы в ***Borland Pascal***?
8) Как нумеруются элементы типизированных файлов?
9) Какими встроенными подпрограммами располагает ***Borland Pascal*** для чтения и записи типизированных файлов?
10) Какую структуру имеет процедура ***Write*** для типизированных файлов?
11) Какие процедуры и функции предназначены для прямого доступа к элементам типизированных файлов?
12) Каково должно быть содержание программы по созданию файла и его корректировке (замена элементов, добавление новых элементов, удаление элементов)?

4 Отчетность

Отчет должен быть выполнен в соответствии с [10] и состоять из следующих разделов:

1) Тема и цель работы.
2) Индивидуальное задание.
3) Схема алгоритма решения задачи.
4) Текст программы и вводимые тестовые исходные данные.
5) Откомпилированный текст программы-заготовки (в электронном виде).
6) Результаты выполнения программы.
7) Ответы на контрольные вопросы.
8) Выводы.

При защите отчета по работе для получения зачета студент должен:

- *уметь* отвечать *на контрольные вопросы*;
- обосновать структуру выбранного алгоритма;
- уметь пояснять работу программы и доказать ее работоспособность;
- продемонстрировать *навыки работы в среде* ***Borland Pascal***.

5 Использование файлов при программировании

5.1 Понятие файла

Большинство современных информационных технологий (САПР, работа с текстовыми и графическими документами, ведение баз данных, справочников, картотек, бухгалтерского учета и т.д.) ориентировано на обработку больших объемов данных. Однако после выключения компьютера все данные, находящиеся в его оперативной памяти теряются. Для долговременного хранения, а также для ввода и вывода данных используются ***файлы*** на внешних носителях периферийных устройств компьютера.

Физический файл[1] – это именованная область на внешнем носителе, содержащая какие-либо данные. Структура физического файла представляет собой простую последовательность байт памяти носителя информации.

байт	**байт**	**байт**	**...**	**байт**	**байт**	**байт**

Имя файла - это любое выражение строкового типа, которое строится по правилам:

- имя содержит до восьми разрешенных символов;
- разрешенные символы - это прописные и строчные латинские буквы, цифры, и символы:

! @ # $ % ^ & ( ) ' ~ - _

- имя начинается с любого разрешенного символа;
- за именем может следовать расширение - последовательность до трех разрешенных символов;
- расширение, если оно есть, отделяется от имени точкой.

Перед именем может указываться так называемый ***путь к файлу***: имя диска и/или имя текущего каталога и имена каталогов вышестоящих уровней[2].

Имя диска - это один из символов A...Z, после которого ставится двоеточие. Имена A: и B: относятся к дисковым накопителям на гибких дискетах, имена C: - к жесткому диску. Остальные имена могут относиться как к магнитным дискам, так и

[1] Физически существует на некотором материальном носителе.
[2] В ***Borland Pascal*** максимальная длина имени вместе с путем - 79 символов.

носителям данных другого типа, в том числе и к одному или нескольким виртуальным дискам, созданным в оперативной памяти компьютера или к сетевым дискам.

Если имя диска не указано, подразумевается устройство по умолчанию - то, которое было установлено в операционной системе перед началом работы программы (или с помощью клавиши меню ***File\Change Dir*** в ***Borland Pascal***).

За именем диска может указываться ***имя каталога***, содержащего файл. Если имени каталога предшествует обратная косая черта, то путь к файлу начинается из корневого каталога, если черты нет - из текущего каталога, установленного в системе по умолчанию. За именем каталога может следовать одно или несколько имен каталогов нижнего уровня. Каждому из них должна предшествовать обратная косая черта. Весь путь к файлу отделяется от имени файла обратной косой чертой. Например, имя файла на диске может иметь вид:

'A:Baza.dat'	{файл Baza.dat на дискете}
'U:\236-1\Ivanov\lab3.pas'	{файл lab3.pas на сетевом диске U, каталоге 236-1, подкаталоге Ivanov}
'Primer3_1.pas'.	{файл Primer3_1.pas в текущем каталоге}

Понятие файла в ***Borland Pascal*** достаточно широко. Это может быть обычный файл на магнитном или лазерном диске, коммуникационный порт компьютера, устройство печати, клавиатура, сканер, флэш-память или другие устройства. В ***Borland Pascal*** зарезервированы стандартные имена устройств и портов, такие как:

'CON', 'LPT1', 'PRN', 'COM1', 'AUX', 'NUL'.

Для поддержки работы с файлами в язык ***Borland Pascal*** введен ***файловый тип***. Файловый тип обычно представляет собой *структурированный* тип данных, содержащий линейную последовательность компонентов любого типа[3] (чаще всего записей) одного типа и одной длины. Нумерация компонентов файла выполняется слева направо, начиная от нуля.

Логический файл[4] - это способ восприятия файла в программе. В программах логические файлы представляются ***файловыми переменными*** определенного типа. Например, файл, состоящий из целых чисел можно представить наглядно следующим образом:

file of Integer

Целое со знаком	**Целое со знаком**	**...**	**Целое со знаком**	***Eof***

В конце файла размещается специальный символ конца файла ***Eof*** (управляющий символ ***ASCII*** с кодом 26). Логическая структура файла очень похожа на структуру массива. Тем не менее вспомним, что в ***Borland Pascal*** количество компонент массива фиксируется на этапе написания программы и он полностью располагается в оперативной памяти.

Длина файла никак не оговаривается при его объявлении и ограничивается только емкостью устройств внешней памяти и это является основным отличием от массива.

[3] За исключением файлового типа или структурного типа, содержащего компонент с файловым типом.

[4] Существует только в нашем логическом представлении при написании программы.

Компоненты файла в отличие от массива *не имеют индексов*. Количество компонент файла в тексте программы не объявляется и может быть произвольным.

Файл называют ***пустым***, если он не содержит ни одного компонента и его длина равна нулю.

С файловой системой ***Borland Pascal*** связано понятие ***буфера ввода-вывода***, позволяющего более быстро и эффективно обмениваться информацией с внешними устройствами.

Буфер - это область памяти, которая выделяется для каждого файла.

При записи в файл вся информация сначала направляется в буфер и там накапливается до тех пор, пока весь объем буфера не будет заполнен. Только после этого или после специальной команды сброса происходит передача данных на внешнее устройство.

При чтении из файла данные вначале считываются в буфер, причем данных считывается не столько, сколько запрашивается, а сколько поместится в буфер.

5.2 Классификация файлов в Borland Pascal

Файлы в ***Borland Pascal*** классифицируются по двум признакам (*рис. 5.1*):

- по типу (логической структуре);
- по методу доступа к элементам файла.

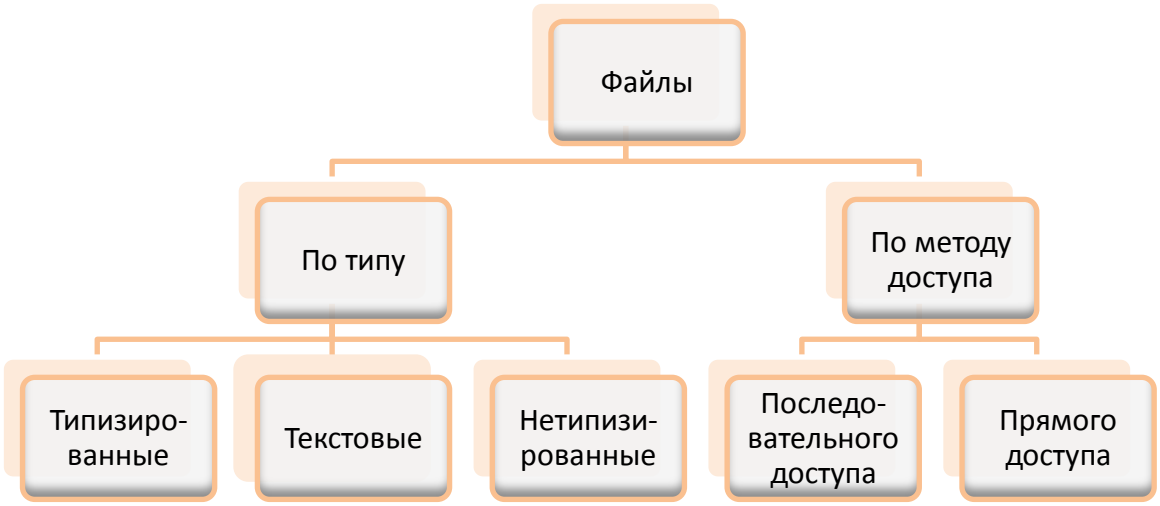


Рисунок 5.1 - Классификация файлов

5.3 Работа с файлами

Формат описания файла:

Type
<Имя типа файла> ***= file of*** <тип компонент>***;***

Var
<Идентификатор файловой переменной>***:*** <Имя типа файла>***;***

Или непосредственно в разделе ***Var***:

Var
<Идентификатор файловой переменной> ***: file of*** <тип компонент>***;***

Отметим некоторые особенности описания файлов:

- Если слово ***of*** и тип компонента опущены, то тип обозначает ***нетипизированный файл***. Нетипизированные файлы представляют собой каналы ввода-вывода нижнего уровня, в основном используемые для прямого доступа к любому файлу на диске, независимо от его внутреннего формата.

- Стандартный файловый тип ***Text*** определяет файл, содержащий символы, упорядоченные в строки. Текстовые файлы используют специальные процедуры ввода-вывода.

Пример. Описать файл для хранения информации об автомобилях.

```
Type  {раздел объявления нестандартных типов данных}
  tAuto = record   {заголовок типа данных: запись об автомобиле}
      ……..         {описания полей данных записи об автомобиле}
  end;             {конец определения  записи об автомобиле}

  tFile = file of tAuto; {тип данных – файл состоящий из записей об автомобилях}

Var     {раздел объявления переменных}
  Auto: tAuto;       {переменная для доступа к записи об автомобиле}
  FileAuto: tFile;   {переменная для доступа к файлу записей об автомобилях}
```

Файл удобно представить в виде именованной магнитной ленты, у которой есть фиксированное начало, затем идут ***компоненты***, а после самой последней компоненты записан ***признак конца файла*** *(Рис. 5.2)*.

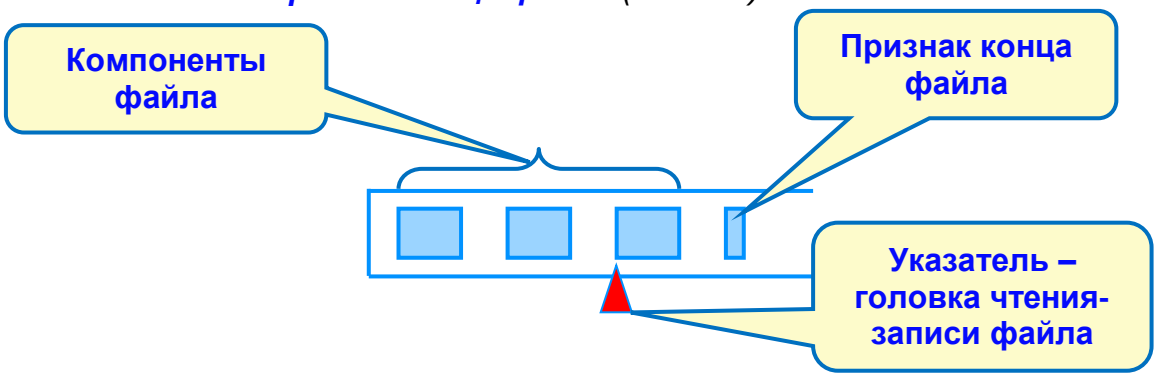


Рисунок 5.2 - Условное представление файла в виде магнитной ленты

Доступ к компонентам файла осуществляется через указатель файла (буферную переменную). При чтении или записи этот указатель перемещается к следующему компоненту, делая его доступным для обработки. В любой момент для чтения и записи доступен только тот компонент файла, на который установлен указатель.

Для работы с файлами в Паскале предусмотрен ряд процедур и функций. Они входят в модуль ***System*** и доступны для вызова всегда. При их описании будем использовать следующие обозначения:

FV - файловая переменная – системная запись, хранящая информацию о файле (имя, размер, время создания и другие атрибуты файлы);

FileName - строковая переменная, содержащая информацию о имени и нахождении файла вида:

'<диск>:\<имя каталога>\<имя подкаталога>\…\<имя файла> '.

Если ***FileName = ''*** (пустая строка), то файл ***FV*** считается связанным со стандартными файлами ввода с клавиатуры ***Input*** (если он открыт для чтения) или вывода на экран ***Output*** (если он открыт для записи).

P - переменные того же типа, что и компоненты файла ***FV***.

N - переменная целого типа, номер (указатель) записи в файле.

Таблица 5.1 – Важнейшие встроеннные процедуры для работы с файлами

Описание	Функция
Assign(FV, FileName)	Присваивает имя внешнего файла ***FileName*** файловой переменной ***FV.*** Обязательно применяется *до начала* работы с файлом!
Rewrite(FV)	Создает и открывает новый *пустой* файл ***FV.*** Если файл с таким именем *уже* был, вся информация в нем *уничтожается*!
Reset(FV)	Открывает *существующий* (иначе – ошибка ввода-вывода!) файл ***FV.*** Указатель файла устанавливается на первый компонент.
Read(FV, P)	Считывает одно или более значений из файла ***FV*** в одну или более переменных ***P***. После считывания компоненты указатель перемещается к следующему компоненту файла. Если ***FV*** опущена, то подразумевается ввод из стандартного файла ***Input*** (клавиатура).
Write(FV, P)	Записывает в файл ***FV*** одно или более значений переменных ***P***. Указатель перемещается к следующему компоненту файла. Если ***FV*** опущена, то подразумевается вывод в стандартный файл ***Output*** (экран).
Seek(FV, N)	Позволяет осуществить прямой доступ к компонентам файла. Перемещает указатель в файле на элемент ***N*** (счет компонентов файла с нуля!). Для текстовых файлов не используется.
Close(FV)	Закрывает открытый файл. Если не закрыть, файл может быть испорчен.
Erase(FV)	Стирает внешний файл ***FV*** (навсегда!).
Rename(FV, FileName)	Переименовывает внешний файл ***FV***. Новое имя - ***FileName***

Таблица 5.2 – Важнейшие встроенные функции для работы с файлами

Описание	Функция
Eof(FV)	Возвращает для файла состояние ***end-of-file*** (конец файла). Значение функции ***True***, если указатель находится сразу за последним компонентом файла, иначе – ***False***.
FilePos(FV)	Возвращает текущее значение указателя в файле ***FV***. Для текстовых файлов не используется
FileSize(FV)	Возвращает текущий размер файла (количество компонент). Для текстовых файлов не используется

Описание	Функция
GetDir	Возвращает строку символов с текущим каталогом на заданном диске
IOResult(FV)	Возвращает целое значение кода ошибки, являющееся состоянием последней выполненной операции ввода-вывода. Если ошибок нет, то возвращает значение 0.

Для работы с каким-либо физическим файлом, размещенном на каком-либо внешнем носителе, необходимо первоначально связать его с файловой переменной (логическим файлом), с помощью которой будет осуществляться доступ к этому физическому файлу.

Связывание логического и физического файлов выполняется процедурой ***Assign***, которая может использоваться только для закрытого файла.

Первым параметром этой процедуры является файловая переменная, а вторым параметром - строковая константа или идентификатор строковой переменной, значением которых должно быть имя физического файла, указанное согласно правилам записи идентификаторов в ***MS-DOS***.

```
Assign(FV,  'MyFile.dat')
```

В приведенном примере выполняется связывание логического файла ***FV*** с физическим файлом ***MyFile.dat***, при условии, что он находится в текущем каталоге активного диска.

В общем случае, чтобы действие процедуры ***Assign*** не зависело от текущих настроек, записывается полное имя файла с указанием диска, пути каталогов и имени файла, например

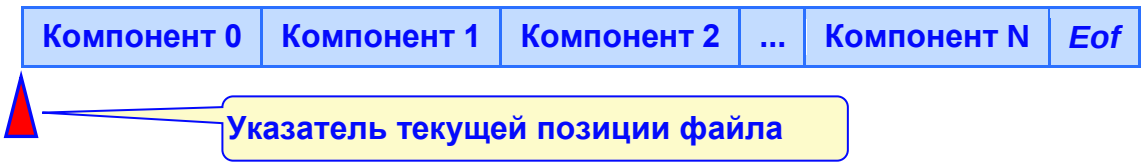
```
FileName := 'u:\236-1\Ivanov\MyFile.dat';
Assign(FV, FileName);
```

Перед выполнением каких-либо операций чтения и записи в файлах, эти файлы должны быть ***открыты***. Открытие файлов выполняется процедурами ***Reset(FV)*** и ***Rewrite(FV)***, а закрытие - процедурой ***Close(FV)***.

Процедура ***Reset*** открывает существующий физический файл, который был связан с файловой переменной ***FV***.

Если ***FV*** - текстовый файл, то он будет доступен только для чтения при последовательном доступе к элементам, если ***FV*** -типизированный файл, то он будет открыт и для чтения, и для записи, как при последовательном доступе, так и при прямом.

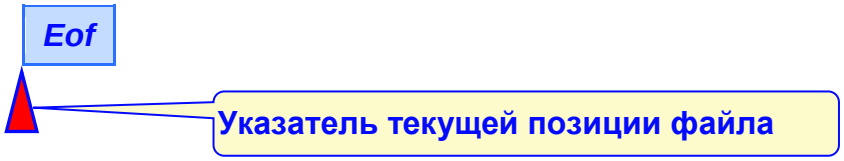
При открытии файла указатель текущей позиции файла устанавливается в его начало:



Если физический файл с указанным именем отсутствует, то возникает ошибка времени исполнения, которую можно подавить выключением директивы компилятора ***{$I-}***. 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.

Процедура ***Rewrite*** создает новый (пустой) физический файл, имя которого связано с файловой переменной ***FV***.

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 в его начало:



Еще одной функцией, используемой практически во всех программах, является функция ***Eof(FV)***. Функция ***Eof(FV)*** возвращает значение ***True***, если указатель текущей позиции в файле ***FV*** находится за последним элементом файла, или, если файл пуст. В противном случае она возвращает значение ***False***.

Чтение из типизированных файлов выполняется только процедурой ***Read***, а запись - только процедурой ***Write***. При этом единицей чтения/записи может быть только переменная того же типа, что и тип компонентов файла.

Процедура ***Read*** для типизированных файлов имеет следующий формат:

Read(<имя файловой переменной>, список переменных)

Процедура ***Write*** для типизированных файлов имеет такой формат:

Write(<имя файловой переменной>, список переменных)

При записи каждой переменной в файл ***указатель текущей позиции в файле, также как и при чтении, перемещается на следующий элемент***.

Если указатель текущей позиции файла находится за последним элементом, т.е. в конце файла (***Eof(f) = True***), то при выполнении процедуры ***Write*** файл ***расширяется***.

Для работы с прямым доступом предназначены следующие процедуры и функции:

- ***FilePos(FV)*** - возвращает номер текущей позиции указателя в файле (позиции нумеруются от нуля !);
- ***FileSize(FV)*** - возвращает текущий размер файла (число компонентов файла при счете от единицы !);
- ***Seek(FV, N)*** - перемещает указатель текущей позиции в файле на компонент с заданным номером ***N*** (при счете от нуля !).

6 Учебный пример работы с файлами

6.1 Условия задания

Пример 11.1. Разработать программу[5], работающую с файлом записей (картотекой) книг вида:

Номер книги	Фамилия автора	Название книги

[5] Эта программа представляет собой дальнейшее развитие учебного примера, проанализированного в лабораторной работе «Программирование с использованием записей» [1]

Предусмотреть многократный ввод данных о книгах с помощью простейшего диалога типа «запрос-ответ»; признак окончания ввода данных – ввод пустой строки. Программа должна работать в режимах:

- ввода данных о книгах,
- коррекции введенных ранее данных о книге,
- формирование запроса на поиск и выбор нужных книг;
- удаления записи о книге с заданным номером.
- Подпрограммы следует оформить в виде пользовательского модуля.

6.2 Основная программа

Основная программа ***Primer11_1*** использует подпрограммы и описания данных из специально разработанного для нее пользовательского модуля ***Unit11_1*** . Она представляет собой бесконечный цикл, последовательно реализующий решение следующих подзадач:

- вывод меню выбора режима работы,
- ввод номера режима работы,
- работа в выбранном режиме.

```
Program Primer11_1; {Учебная база данных регистрации и поиска книг}

Uses {Подключение библиотечных модулей}
  Crt,       {Системный модуль процедур работы с экраном}
  Unit11_1; {Пользовательский модуль с подпрограммами и описаниями данных файла
          записей о книгах}
begin {Основная программа Primer11_1}

        {Устанавливаем цвет фона и текста: желтые символы на голубом фоне }
  TextColor(Yellow);
  TextBackGround(Blue);
  Repeat     {Начало диалогового выбора и выполнения режимов работы}
    ClrScr; {очистка экрана}

     {Вывод меню выбора  режимов работы}
    GoToXY(1, 5);
    Writeln('  Учебная база данных регистрации и поиска книг');
    Writeln;
    Writeln('            Главное меню');
    Writeln;
    Writeln('  1: Создать базу данных книг с новым именем');
    Writeln('  2: Ввод информации о книгах');
    Writeln('  3: Коррекция записи о книге');
    Writeln('  4: Удаление записи о книге');
    Writeln('  5: Выбор и вывод информации о книгах');
    Writeln('  6: Переименовать базу данных');
    Writeln('  7: Закончить работу');
    Writeln;
    Write('    Введите номер режима работы: ');
    Readln(Otvet);
    Writeln;
```

```
    {Устанавливаем параметры - светло-серые символы на голубом фоне }
  TextColor(LightGray);
    {Выполнение выбранного режима работы}
  Case Otvet of
    1: NewFile;        {Создание нового файла с записями о книгах}
    2: Vvod;           {Ввод и запись информации о книгах в файл}
    3: Korr;           {Коррекция записи в файле о книге Number}
    4: Udal;           {Удаление записи в файле о книге с номером Number}
    5: Out;            {Вывод информации о всех книгах на экран}
    6: RenFileName;    {Изменить имя файла}
    7: Halt;           {Закончить работу}
  else begin
         Writeln('Неверно выбран режим.'#7);
         Pause{Временная остановка}
       end
  end;

  TextColor(Yellow);  {Восстанавливаем - желтые символы на голубом фоне}

until false; {Конец "бесконечного" цикла выбора и выполнения режимов работы}

end. {Конец основной программы Primer11_1}
```

6.3 Модуль для программы работы с файлом записей о книгах

```
{$F+,O+,R+}   {Директивы трансляции модуля}
Unit Unit11_1; {Модуль с подпрограммами и описаниями данных файла записей о
                книгах}

Interface {начало интерфейсной секции}
{==================== Раздел объявлений констант ======================}
Const
{============================ Ограничения ============================}
  MaxLenFile = 25;      {Максимальная длина имени файла (с маршрутом)}
  LenAutor  = 15;       {Максимальная длина ФИО автора книги}
  LenNazv = 15;         {Максимальная длина названия книги}

{============ Строка с именем и маршрутом файла по умолчанию ==========}
  FileName: string[MaxLenFile] = 'd:\bibl.dat';
{=========== Раздел объявлений типов нестандартных данных =============}
Type
  tBook = record {запись о книге}
    Number: integer;          {поле с порядковым номером книги}
    Autor: string[LenAutor];  {поле с ФИО автора книги}
    Nazv:  string[LenNazv];   {поле с названием книги}
  end; { tBook }
```

{========== Раздел объявлений переменных ==========}

```
Var
  BooksFile: file of tBook;      {файловая переменная библиотечного файла}
  Book, FindBook: tBook;         {экземпляры записи о книге}
  NBook: integer;                {указатель на запись о книге}
  Error: integer;                {код ошибки при работе с файлами. 0 - нет ошибки}
  Otvet: integer;                {код ответа при диалоге}
```

{========== Раздел объявлений функций и процедур ==========}

```
Procedure Pause; {Временная остановка, до нажатия Enter}
{Функция YesNo -возвращает True если ответ Y, д, Д или Enter, иначе False}
Function YesNo: Boolean;
{Функция FileExists - возвращает True, если файл с именем FileName
                      существует,или False, если файла с именем FileName нет;
                      закрывает существующий файл.}
Function FileExists(FileName: String): Boolean;
Function VvodFileName: string;      {Возвращает новое имя полное файла}
Procedure NewFile;                  {Создать новый файл}
Procedure OpenFile;                 {Открыть существующий файл}
Procedure  RenFileName;             {Изменить имя файла}
{Функция VvodBook - Ввод информации о книге, возвращает True, если ввод
                    успешен,False, если во введенном поле Avtor - пустая
                    строка}
Function VvodBook: Boolean;
Procedure VvodFindBook; {Ввод информации о параметрах поиска книги}
Procedure Vvod;             {Ввод и запись информации о книгах в файл}
{Функция PoiskNumb - Поиск в файле книги с номером NBook. PoiskNumb = True,
                     если поиск удачен, PoiskNumb = False, если иначе. NRec -
                     номер записи в файле о книге с порядковым номером NBook}
Function PoiskNumb(var NBook, NRec: integer): Boolean;
Procedure Korr;          {Коррекция записи в файле о книге с номером Number}
Procedure  Udal;         {Удаление записи в файле о книге с номером Number}
Procedure  OutBook;      {Вывод информации о книге}
Procedure Out;           {Вывод информации о всех книгах на экран}
```

{========== Секция реализации функций и процедур ==========}

```
Implementation
Uses    {Подключение библиотечных модулей}
  Crt;  {Системный модуль процедур работы с экраном}
```

```
Procedure  Pause;  {Временная остановка}
begin
   writeln;
   write('Для продолжения нажмите <Enter>', #7); {#7 - звуковой сигнал}
   ReadIn
end;  {--- Pause ---}

{--- Функция YesNo ---}
function YesNo: Boolean;  {Возвращает True - если ответ Y, д, Д, Enter,или
                           False, если иначе}
var
   Otv: char;   {Символ с начальной буквой ответа}
begin
   Write(' (y/n): ');  Otv := ReadKey;   {Ввод символа ответа}
   Otv := UpCase(Otv);                   {Преобразование к верхнему регистру}
   if (Otv = 'Y') or (Otv = 'Д') or (Otv = 'д') or (Otv = #13)
      then YesNo := True     {если положительный ответ}
      else YesNo := False;  {если отрицательный ответ}
   WriteIn
end;  {---YesNo---}

{--- Функция FileExists ---}
function FileExists(FileName: String): Boolean;
var
   F: file;
begin
   Assign(F, FileName);  {Присоединяем файл с именем FileName}
   {$I-}  {Блокировка системной обработки ошибок ввода-вывода}
   Reset(F);  {Попытка открыть файл}
  Close(F);  {Закрыть файл, если он есть}
   {$I+}      {Отмена блокировки системной обработки ошибок ввода-вывода}
  {Функция IOResult возвращает код ошибки: 0 - ошибок при открытии файла нет.
  {FileExists = True, если файл с именем FileName существует, или
   FileExists = False, если файла с именем FileName нет}
   FileExists := (IOResult = 0) and (FileName <> '')
end;  {--- FileExists ---}
```

```
{--- процедура VvodFileName ---}
function VvodFileName: string; {Ввод имени файла}
var
   St: string; {Вспомогательная строка для хранения имени файла}
begin
   ClrScr;
   VvodFileName := FileName; {Запоминаем прежнее имя файла}
   Write('Прежнее имя файла ', FileName, '. Изменить?');
   if YesNo
      then begin {имя файла изменять нужно}
              Write('Введите новое полное имя файла: '); Readln(St);
              VvodFileName := St
           end
end;  {---VvodFileName---}

{--- процедура NewFile ---}
procedure  NewFile; {создать новый файл}
var
   St: string[4]; {Вспомогательная строка}
begin
   ClrScr;
   FileName := VvodFileName; {Ввод нового имени файла}
   if FileExists(FileName)
      then St := ' уже'   {Файл "уже" существует}
      else  St := ' не';   {Файл "не"  существует}
   Write('Файл ', FileName, St, ' существует. Создать новый?');
   if not YesNo
      then  Exit; {Выход, если файл создавать не нужно}
   Assign(BooksFile, FileName);   {Присоединяем файл FileName}
   {$I-} {Блокировка системной обработки ошибок ввода}
  ReWrite(BooksFile); {Попытка создать файл}
  {$I+} {Отмена блокировки системной обработки ошибок ввода}

  {Функция IOResult возвращает код ошибки: 0 - ошибок при создании файла нет}
   Error := IOResult; {Определяем код ошибки файловых операций}
```

```
  If Error = 0
     then begin
            WriteLn( 'Файл ', FileName, ' успешно создан.');
            Close(BooksFile) {Закрыть файл}
          end
     else begin
            WriteLn('Ошибка (Error = ', Error, ') при создании файла ', FileName);
            Pause
          end
end; {--- NewFile ---}

{--- процедура OpenFile ---}
procedure OpenFile; {Открыть существующий файл}
begin
  Assign(BooksFile, FileName); {Присоединяем файл с именем FileName}
  {$I-} {Блокировка системной обработки ошибок ввода-вывода}
  Reset(BooksFile);            {Попытка открыть файл}
  {$I+} {Отмена блокировки системной обработки ошибок ввода-вывода}

  {Функция IOResult Возвращает код ошибки: 0 - ошибок при открытии файла нет}
  Error := IOResult; {Определяем код ошибки файловых операций}
  If Error = 0
    then WriteLn('Файл ', FileName, ' успешно открыт.')
    else begin
           WriteLn('Ошибка (Error = ', Error, ') при открытии файла ',
                   FileName, #7);
           Pause;      {Временная остановка}
           NewFile     {создать новый файл}
         end
end; {---OpenFile---}

{--- процедура RenFileName ---}
procedure RenFileName; {Изменить имя файла}
var
  St: string;{Вспомогательная строка для имени файла}
begin
  ClrScr;
```

```
  St := VvodFileName; {Запомнили старое имя файла}
  if FileExists(St)
    then begin
           Write('Файл ', St, ' существует. Переименовать?');
           if not YesNo
             then Exit; {Выход, если файл переименовывать не нужно}
          {$I-} {Блокировка системной обработки ошибок ввода}
           ReName(BooksFile, St); {Попытка переименовать файл}
           {$I+} {отмена блокировки системной обработки ошибок ввода}

           Error := IOResult; {Определяем код ошибки файловых операций}
           If Error = 0
            then begin
                   FileName := St;
                  Writeln('Файл ', FileName, ' успешно переименован.')
                end
            else begin
                   {Присоединяем файл со старым именем FileName}
                   Assign(BooksFile, FileName);
                   Writeln('Ошибка (Error = ', Error,
                           ') при переименовании файла ', St, #7);
                   Pause
                end
        end
end; {---RenFileName---}

{--- процедура VvodBook ---}
{ Ввод информации о книге. Возвращает True, если ввод успешен, или False, если
во введенном поле Autor - пустая строка}
function VvodBook: Boolean;
begin
  with Book do
    begin
      Number := FilePos(BooksFile);
      Writeln;
      Writeln('Порядковый номер книги = ', Number);
      Writeln;
```

```
        Write('Автор книги   : '); Readln(Autor);
        if Autor = ''
          then begin
                 VvodBook := False; {информация о книге не введена}
                 Writeln;  Writeln('Ввод информации завершен.');
                 Pause; {Временная остановка}
                 Exit    {Выход}
               end;
        VvodBook := True; {информация о книге введена}
        Write('Название книги: '); Readln(Nazv);
        Writeln
      end
end; {---VvodBook---}

{--- процедура VvodFindBook ---}
procedure VvodFindBook; {Ввод информации о параметрах поиска книги}
var
  St: string; {вспомогательная строка для ввода номера книги}
begin
  with FindBook do
    begin
      WriteLn('Найти книги с параметрами: ');
      repeat
        Error := 0; {Флаг Error = 0, если порядковый номер книги в допустимом
                     диапазоне, Error > 0, если нет}
        Writeln;
        Write('Порядковый номер книги (Нажмите Enter, если искать книги
              с любым номером): ');
        ReadLn(St);
        if St = ''
          then Number := -1
          else begin
                 Val(St, Number, Error); {Преобразуем текст St в число Number}
                 if (Error > 0) or (Number < 0) or (Number > FileSize(BooksFile))
                   then begin
                          GoToXY(1,5);
```

```
                writeln(' Недопустимый порядковый номер
                    книги!!', #7);
                Error := 1;
                Pause
              end
            end
        until Error = 0;
        Writeln('Автор книги (Нажмите Enter, если искать книги любого
            автора): ');
        Readln(Autor);
        Writeln('Название книги   (Нажмите Enter, если искать книги с любым
            названием): ');
        Readln(Nazv); Writeln
      end
end; {---VvodFindBook---}

{--- процедура Vvod ---}
procedure Vvod; {Ввод и запись информации о книгах в файл}
begin
  OpenFile; {Открыть файл}
  if Error <> 0
  then Exit; {Выход, если файл не был открыт}
  ClrScr;
  {Указатель номера книги перемещаем на очередную книгу,которую добавляем в
    конец файла!}
  Seek(BooksFile, FileSize(BooksFile));
  Repeat {цикл ввода информации по книгам}
    if VvodBook
      then write(BooksFile, Book) {если запись Book о книге сформирована,
                                   сохраняем ее в файл BooksFile}
      else begin
             Close(BooksFile); {Закрыть файл, если запись Book о книге не
                                сформирована}
             Exit {Выход}
           end
  until False;
  Close(BooksFile) {Закрыть файл}
end; {---Vvod ---}
```

```
{--- PoiskNumb ---}
{Поиск в файле книги с номером NBook. PoiskNumb = True, если поиск удачен,
PoiskNumb = False, если иначе. NRec - номер записи в файле о книге с порядковым
номером NBook}
function PoiskNumb(var NBook, NRec: integer): Boolean;
var
  NumbBook: integer;   {количество книг в файле}
  flag: boolean;       {флаг ошибки задания номера книги}
begin
  PoiskNumb := false; {Книга не найдена}
  OpenFile; {Открыть файл}
  if Error <> 0
    then Exit; {Выход, если файл не был открыт}
  NumbBook := FileSize(BooksFile); {Определяем количество книг в файле}
  if NumbBook = 0
    then begin
           GoToXY(1,5); Writeln('Список книг пуст..', #7);
           Pause;
           Exit
         end;
  repeat
    Writeln;
    Write('Введите порядковый номер книги: ');
    Readln(NBook);
    flag := (NBook >= 0); {flag = true, если порядковый номер книги в
                           допустимых пределах,  flag = false, если нет}
   if not flag
    then begin
           GoToXY(1,5); writeln(' Недопустимый порядковый номер книги!!');
           Pause
         end
  until flag;
  flag := false;
  {Поиск книги с порядковым номером NBook}
  NRec := 0;
```

```
  while(not Eof(BooksFile)) do
    begin
      Read(BooksFile, Book);  {Читаем запись о книге с номером NBook}
      if Book.Number = NBook
        then begin  {Книга найдена}
                flag := true;
                PoiskNumb := true;
                Break
             end;
      NRec := NRec + 1
    end;

  if not flag
    then begin
           GoToXY(1,5);
           writeln('Книги с порядковым номером ', NBook, ' в картотеке
                нет!!', #7);
           Pause;
           Exit
         end;
  writeln;
   OutBook {Выводим на экран информацию о книге с номером NBook}
end; {--- PoiskNumb ---}

{--- процедура Korr ---}
procedure Korr; {Коррекция записи о книге введенным номером NBook в файле}
var
  NBook: integer;  {порядковый номер книги в файле}
   NRec: integer;   {номер записи в файле о книге с порядковым номером NBook}
begin
  ClrScr;
  Writeln('Коррекция данных о книге с заданным номером');
  writeln;
{NRec - номер записи в файле о книге с порядковым номером NBook}
  if PoiskNumb(NBook, NRec)
    then begin  {книга найдена}
```

```
            writeln;
            Write('Корректировать данные о книге? ');
            if not YesNo
              then begin
                     Close(BooksFile); {Закрыть файл}
                     Exit    {Выход, если не корректировать}
                   end;
{Перемещаемся по файлу к записи NRec книге с порядковым номером NBook}
            Seek(BooksFile, NRec);
            if VvodBook
              then write(BooksFile, Book) {если запись о книге скорректирована,
                                           то сохраняем ее в файле вместо ста-
                                           рой записи}
               else begin
                     Close(BooksFile); {Закрыть файл}
            Exit
          end
      end;
  Close(BooksFile) {Закрыть файл}
end; {---Korr---}

{--- процедура Udal ---}
procedure Udal; {Удаление записи в файле о книге с номером Number}
var
  NBook: integer;  {Порядковый номер книги в файле}
  NRec: integer;   {номер записи в файле о книге с порядковым номером NBook}
begin
  ClrScr;
  Writeln('Удаление книги с заданным номером');
  writeln;
  {NRec - номер записи в файле о книге с порядковым номером NBook}
  if PoiskNumb(NBook, NRec)
    then begin  {книга найдена}
            writeln;
            Write('Удалять запись об этой книге? ');
            if not YesNo
              then begin
```

```
                    Close(BooksFile); {Закрыть файл}
                    Exit    {Выход, если не удалять}
                  end;

{номер записи о книге устанавливаем максимально-допустимым,вне диапазона
номеров записей}
                  Book.Number := MaxInt;
{Перемещаемся по файлу к записи NRec книге с порядковым номером NBook}
                  Seek(BooksFile, NRec);
{сохраняем скорректированную запись в файле вместо старой записи}
                  Write(BooksFile, Book)
                end;
  Close(BooksFile) {Закрыть файл}
end; {--- Udal ---}

{--- процедура OutBook ---}
procedure OutBook; {Вывод на экран информации о книге Book}
begin
  with Book do
    writeln(Number:5, '  ', Autor: LenAutor, '  ', Nazv: LenNazv)
end; {---OutBook---}

{--- процедура Out ---}
procedure Out; {Вывод информации о всех книгах на экран}
begin
  OpenFile; {Открыть файл}
  if Error <> 0
    then Exit; {Выход, если файл не был открыт}

  ClrScr; {Очистка экрана}
 if FileSize(BooksFile) = 0
    then begin
                  GoToXY(1,5); Writeln('Список книг пуст..'#7);
                  Pause;
                  Exit
                end;
  VvodFindBook; {Ввод информации о параметрах поиска книги}
```

```
  {Вывод заголовка списка книг}
  Writeln;
  Writeln('Номер', 'Автор': LenAutor+2, 'Название книги': LenNazv+2);
  Writeln;
  while not Eof(BooksFile) do
    with Book do
      begin
        Read(BooksFile, Book); {чтение информации о книге Book из файла
                                BooksFile}
        if (Number = FindBook.Number) or (FindBook.Number < 0)
          and (Number <> MaxInt)
          then begin
        if (FindBook.Autor = '') and (FindBook.Nazv = '')
         then OutBook {Вывод на экран информации о книге Book}
         else if FindBook.Autor = ''
                then begin
                       if Pos(FindBook.Nazv, Nazv) <> 0
                         then OutBook {Вывод на экран информации о книге
                                       Book}
                     end
                else if FindBook.Nazv = ''
                       then if Pos(FindBook.Autor, Autor) <> 0
                              then OutBook {Вывод на экран информации о книге
                                            Book}
      end
    end;
  Writeln;
  writeln('Всего в списке книг: ', FileSize(BooksFile));
  Pause; {Временная остановка}
  Close(BooksFile); {Закрыть файл}
end; {---Out---}

begin     {Секция инициализации модуля (пустая)}
end.      {Конец модуля}
```

7 Индивидуальные задания

7.1 Требования к программе и рекомендации по ее разработке

При выполнении задания в качестве заготовки программы разумно максимально использовать рассмотренный выше учебный пример. Для этого, в первую очередь, следует скопировать файлы ***Prim10_1.pas*** и ***Unit10_1.pas*** в свой каталог, переименовать их в соответствии со своим вариантом задания и взять затем за основу разрабатываемой программы.

Вслед за тем, используя клавишу ***Replace*** (***Замена***) замените все слова ***Book*** (***Книга***) в тексте своей программы на выбранное имя записи своего варианта задания. Аналогично замените все слова ***книг*** (используются в комментариях) на русский эквивалент имени записи своего варианта задания. После этого внесите исправления в те участки программы, которые связаны с названиями новых полей записи (объявления новых полей, ввод и вывод их значений и т.п.). Внимательно просмотрите текст комментариев в программе и уточните окончания слов.

Тестовый пример формируется самостоятельно. Он должен позволять оценить работоспособность программы.

При составлении программы по заданию предусмотреть:

- использование структурного и модульного подхода;
- простейший диалог типа «запрос-ответ» при вводе данных;
- многократный ввод данных при исполнении программы, т.е. возможность повторной обработки при иных исходных данных. Признак окончания ввода данных – ввод пустой строки;
- вывод результатов в удобном для пользователя виде;
- освоить методику поиска причин и исправления ошибок, а также трассировки программы «по шагам»[6] по тестовому примеру.
- подпрограммы, составленные при выполнении задания, должны быть оформлены в виде пользовательского модуля.
- программа должна работать в режимах:
- создание файла записей на диске;
- ввода исходных данных;
- хранение данных в файле;
- формирование запроса на поиск и выбор требуемых записей в файле;
- коррекции введенных ранее записей;
- удаления записи с заданным номером.

7.2 Варианты заданий

Создать программу, работающую с файлами записей:

1. Самолеты

Наименование типа	Фамилия конструктора	Год выпуска	Количество кресел	Грузоподъемность, т

2. Расчет движения

Наименование воздушной линии	Тип самолета	Количество рейсов	Налет, тыс. км	Пассажирооборот человеко-км

3. Перевозки

Тип самолета	Номер борта	Количество рейсов	Налет в часах	Налет, тыс. км

[6] Используйте функциональные клавиши F7 или F8.

4. Расписание

Номер рейса	Наименование рейса	Тип самолета	Стоимость билета	Протяженность линии

5. Сооружения аэропорта

Наименование	Площадь	Этажность	Год сооружения	Стоимость

6. Ремонт аэродромных сооружений

Наименование	Шифр	Вид ремонта	Стоимость	Наименование подрядчика

7. Кассы авиабилетов

Номер кассы	Ф.И.О. кассира	Количество проданных билетов	Суммарная выручка	Дата продаж

8. Характеристики персональных компьютеров

Тип процессора	Тактовая частота	Емкость ОП, Мбайт	Емкость ЖМД, Мбайт	Тип монитора

9. Города

Наименование	Количество жителей	Площадь, кв. км	Год основания	Количество школ

10. Московские мосты

Наименование	Высота	Ширина	Количество опор	Протяженность

11. Линии московского метро

Наименование	Район линии	Год пуска	Протяженность, км	Количество поездов

12. Легковые автомобили

Марка	Цвет	Стоимость	Изготовитель	Максимальная скорость

13. Продажа программных продуктов

Наименование	Фирма изготовитель	Стоимость, тыс. руб	Объем, Мбайт	Количество на складе

14. Абонентская плата за телефон

Ф.И.О. абонента	Телефон	Год установки	Количество абонентов	Плата за телефон

15. Детские сады

Наименование детского сада	Номер сада	Количество детей	Район города	Плата за месяц

16. Сотрудники

Ф., И., О.	Табельный номер	Дата рождения	Оклад, тыс. руб.	Стаж

17. Ведомость зарплаты за текущий месяц

Ф., И., О.	Номер отдела	Табельный номер	Количество рабочих часов	Размер зарплаты

18. Музеи

Наименование	Назначение	Адрес	Время работы	Стоимость билета

19. Экскурсии

Наименование	Страна	Стоимость	Продолжительность	Транспорт

20. Кинофильмы

Наименование кинотеатра	Стоимость билета	Время сеансов	Адрес	Количество мест

21. Книга - почтой

Наименование книги	Ф.И.О. автора	Номер по каталогу	Издательство	Стоимость книги

22. Квартиры

Адрес	Площадь, кв. м	Сторона света	Стоимость 1 кв. м	Этаж	Количество комнат

23. Склад товаров

Номер магазина	Наименование товара	Артикул товара	Цена единицы товара	Количество товара

24. Телевизоры на складе магазина

Наименование	Фирма изготовитель	Стоимость	Размер экрана	Количество на складе

25. Холодильники на складе магазина

Наименование	Фирма изготовитель	Стоимость	Емкость камеры	Количество на складе

Список рекомендуемой литературы

1. Кобрин Ю.П. Программирование с использованием записей. Лабораторная работа по дисциплине «Информатика» для студентов специальностей 210201 (200800) и 201300. – Томск: ТУСУР, 2007. – 17 с.
2. Д. Ван Тассел. Стиль, разработка, эффективность, отладка и испытание программ: Пер. с англ. – М.: Мир, 1985. – 332 с.
3. Н. Вирт. Алгоритмы и структуры данных. : Пер. с англ. – М.: Мир, 1989. – 360 с.
4. Основы информатики. Учеб. Пособие / Аладьев В.З., Хунт Ю.Я., Шишаков М.Л. - М.: Информационно-издательский дом "Филинъ", 1998. - 496 с.
5. Марченко А.И., Марченко Л.А. Программирование в среде Borland Pascal 7.0. – К.: ЮНИОР, 1998. – 480 с.
6. Зуев Е.А. Программирование на языке Турбо Паскаль 6.0, 7.0. - М: Веста, Радио и связь, 1993. - 384 с.
7. Епанешников А., Епанешников В. Программирование в среде Turbo Pascal 7.0. - М.: "ДИАЛОГ-МИФИ", 1993. - 288 с.
8. Фаронов В.В. Turbo Pascal 7.0. Начальный курс. Учебное пособие.- М.: "НОЛИДЖ", 2001. - 576 с.
9. Фаронов В.В. Turbo Pascal 7.0. Практика программирования. Учебное пособие.- М.: "НОЛИДЖ", 1998. - 432 с.
10. ОС ТУСУР 6.1-97. Работы студенческие учебные и выпускные квалификационные. - Томск: ТУСУР, 1999.- 10 с.