МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТРАНСПОРТА

Кафедра "Информационные технологии"

РОГАЧЁВА Н.А., ЛИТВИНОВИЧ Т.Н., БОРИСЕНКО М.В.

ИНФОРМАТИКА

ЯЗЫК ПРОГРАММИРОВАНИЯ ПАСКАЛЬ

Методическое пособие для студентов
дневной формы обучения технических специальностей

Часть 3

Одобрено методической комиссией
факультета управления процессами перевозок
Белорусского государственного
университета транспорта

Гомель 2002

УДК 681.3.06
И 741

И 741 Информатика. Язык программирования Паскаль: Методическое пособие для студентов дневной формы обучения технических специальностей / *Рогачева Н.А., ЛитвинковичТ.Н., Борисенко М.В.* – Гомель: БелГУТ, 2002.- 15с.

Данное методическое пособие является продолжением по информатике охватывает следующие разделы: структура типов данных языка Паскаль, оператора цикла, формат описания и использования одномерных и двумерных массивов, описание и использование пользовательских процедур и функций.

Р е ц е н з е н т ы:
Доцент каф. “Прикладная математика”, к.т.н. **С.И. Жогаль**
ст. преподаватель каф. “Информационные технологии” **А.Б. Зборовская**

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Катализатором научно-технического прогресса на транспорте и в народном хозяйстве является широкое использование ЭВМ. Без компьютеров немыслимо развитие производства и науки. Для увеличения объемов выпускаемой продукции и повышения качества необходимо внедрять во все отрасли народного хозяйства автоматизированные и роботизированные комплексы.

Формированию будущего инженера способствует развитие его алгоритмического мышления. Основами алгоритмизации должны обладать инженер-технолог и инженер-конструктор. Даже если инженеру непосредственно не придется программировать при разработке и внедрении новых производственных задач с использованием ЭВМ, то может грамотно поставить задачу математику-программисту, обладая инженерно-алгоритмическим мышлением.

Данное методическое пособие по **информатике** охватывает следующие разделы:

- структура типов данных языка Паскаль, в котором приведена общая схема типов данных и примеры по каждому типу. Даны пояснения по их использованию;
- рассмотрены три основные оператора цикла для выполнения циклических операций и даны пояснения по их работе и особенностям использования;
- формат описания и использования одномерных и двумерных массивов;
- описание и использование пользовательских процедур и функций и на основе обработки массивов даны рекомендации по их использованию.

1 СТРОКОВЫЙ ТИП ДАННЫХ

Строка **(*string*)** - это один из дополнительных типов данных, введенных в системе программирования Турбо-Паскаль. Структура данных типа **string [*N*]** аналогична структуре данных типа а**rray [1..*N*] of char.** Количество символов в данных типа **string [*N*]** может быть любым от 1 до *N*. Максимальное возможное значение константы *N*=255. Это максимальное значение принимается по умолчанию, если в описании строки отсутствует конструкция **[*N*].** В отличие от массивов строки (а не только их отдельные элементы) могут быть параметрами в процедурах ввода и вывода. К любому символу в строке можно обратиться точно также, как к элементу одномерного массива, указав имя строки и индекс.

Например:

```
Const Adres='ул.Королева,2';
Type  FileName=string[150];
Var   St1: FileName;
      St2,St3:string[10];
```

В данном примере объявлены: константа строкового типа *Adres*, значение которой строка - 'ул.Королева,2'; пользовательский тип данных, максимальная длина строки которой 150, строковая переменная *Str1* типа *FileName*; строковые переменные *Str2*, *Str3* с максимальной длиной строки 10 символов.

В Паскале для работы со строками предусмотрен ряд процедур и функции:

К строкам можно применять операцию **"+"** – сцепление (конкатенация), например:

st:= 'a' + 'b';

st:= st + 'c'; {st содержит 'abc'}

- ***Copy*(*st, index, count*)** – функция, возвращающая значение типа *string*, копирует из строки *st* количество символов заданное параметром *count*, начиная с символа с номером *index*.
- ***Delete*(*st, index, count*)** – процедура, удаляющая *count* символов в строке *st*, начиная с символа с номером *index*.
- ***Insert*(*subst, st, index*)** – процедура, которая вставляет подстроку *subst* в строку *st*, начиная с символа с номером *index*.
- ***Length*(*st*)** – функция, возвращающая значение типа *integer*, определяет длину строки *st*. ()
- **Pos(subst,st)** - функция типа Integer; отыскивает в строке st первое вхождение подстроки subst и возвращает номер позиции, с которой она начинается; если подстрока не найдена, возвращается 0. **Str(x:[width[: decimals]], st)** - процедура; преобразует число x вещественного или целого типов в строку символов st так, как это делает процедура **Writeln** перед выводом. **Val(st,x,code)** - процедура; преобразует строку символов st во

внутреннее представление целой или вещественной переменной x, которое определяется типом этой переменной; параметр code содержит ноль, если преобразование прошло успешно, и тогда в x помещается результат преобразования, в противном случае он содержит номер позиции в строке st, где обнаружен ошибочный символ, и в этом случае содержимое x не меняется; ведущие пробелы в строке st должны отсутствовать.

Операции отношения **=, <>, >, <, >=, <=** выполняются над двумя строками посимвольно, слева направо с учетом внутренней кодировки символов.

2 ТИП ДАННЫХ ФАЙЛЫ

2.1 Краткие теоретические сведения

При решении задач с большими объемами информации (бухучет, справочники, картотеки, базы данных и т. д.) используются файлы, которые позволяют работать с информацией, расположенной на внешних носителях. Файл – это поименованная совокупность данных, расположенных во внешней памяти. Компонентом файла может быть значение простого типа или структура, но не файл. Число элементов в файле (длина файла) не фиксирована и ограничивается только ёмкостью устройства, на которое файл записывается. Все элементы файла считаются пронумерованными; начальный элемент имеет номер 0. Доступ к компонентам файла осуществляется через указатель файла. При чтении или записи этот указатель перемещается к следующему компоненту и делает его доступным для обработки. В каждый момент доступен для записи (чтения) только тот компонент файла, на который установлен указатель.

Рисунок 2.1

Компонент 0	Компонент 1	Компонент 2	Компонент 3	. . . . .	Компонент i

▪ – указатель файла

Существует два способа доступа к компонентам файла: последовательный и произвольный (прямой). При последовательном доступе поиск начинается с начала файла и проверяется последовательно каждый элемент, пока не будет найден нужный. Прямой доступ позволяет обращаться к элементу файла по его порядковому номеру в файле.

Различают текстовые файлы, типизированные файлы (в их описании указывают тип компонент) и нетипизированные.

2.2 Понятие о текстовых и нетипизированных файлах

При работе с текстовыми файлами переменная файлового типа описывается как var <имя файла>: text. Компонентами текстового файла являются строки. Строка не должна быть длиннее 255 символов. В длине строки учитываются 2 байта, занятые под маркер конца строки EOLN().

Нетипизированные файлы описывают var <имя файла>: file. Они состоят из произвольных наборов данных, позволяют организовать высокоскоростной обмен данными между диском и оперативной памятью.

2.3 Типизированные файлы

Типизированные файлы допускают как последовательный, так и прямой доступ. Элементы таких файлов должны быть одного типа, а

следовательно, и одного размера. Это могут быть символы, числа, массивы или записи. Возможен произвольный доступ к любому элементу типизированного файла — по номеру однозначно определяется местоположение элемента файла.

Формат описания:

var <имя> : file of <базовый_тип>;

Пример описания:

```
var
   next : file of Char; {Компонентами файла являются символы}
   cok : file of String[20]; {Файл состоит из строк по 20 символов каждая}
```

С переменными файлового типа применимы следующие стандартные процедуры и функции:

Для того, чтобы начать работать с файлом, необходимо установить связь между файловой переменной и файлом на диске компьютера. Это делается с помощью процедуры:

Assign (<имя_файловой_переменной>, 'Name');, где Name – имя файла.

Затем файл открывается процедурой:

Reset (<имя_файловой_переменной>);

или:

ReWrite (<имя_файловой_переменной>);

Процедурой Reset открываются уже существующие файлы, ReWrite - новые. Если файл уже был открыт, то он сначала закрывается, а затем открывается вновь.

В отличие от текстовых, типизированные файлы допускают операции, как записи, так и чтения независимо от того, какой процедурой файл открыт.

Чтение из типизированного файла производится процедурой:

Read (<имя_файловой_переменной>, vl, v2 ... vN);,

а запись – процедурой:

Write (<имя_файловой_переменной>, vl, v2… vN);, где vl. v2, w. vN – переменные базового типа (того же, что и элементы файла).

Произвольный доступ элементам файла осуществляется процедурой:

Seek (<имя_файловой_переменной>, n); – устанавливает указатель файла на элемент с номером n. Именно в этот элемент будет записано значение переменной vl: Write (<имя_файловой_переменной>, v1); или считано значение:Read(<имя_файловой_переменнjй>, v1);

Close (<имя_файловой_переменной>) – закрывает файл, не следует выходить из программы не закрыв файл.

Erase(<имя_файловой_переменной>) – уничтожает файл. Перед уничтожением файл нужно закрыть.

Rename (<имя_файловой_переменной>, <фп1>) – переменование файла. Новое имя – фп1.
Truncate (<имя_файловой_переменной>) — отсечение от файла его хвостовой части, начинающейся с текущей позиции указателя включительно.

Текущее положение указателя файла определяется функцией:
FilePos(имя_файловой_переменной): LongInt;,

а общее количество записей — функцией:
FileSize(имя_файловой_переменной): LongInt;
EoF(<ф.п.>) — функция, которая принимает значение TRUE , если достигли конца файла.

Поскольку типизированные файлы не разбиты на строки, процедуры Readln и WriteLn для них не применяются.

2.4 Примеры программ

Так как обработка файлов является довольно трудоёмкой, целесообразно основные операции с файлами описать в виде словесных алгоритмов.

Для создания файла надо выполнить следующие шаги:

1. Присвоить файлу имя (процедура Assign).
2. Открыть новый файл для записи (процедура ReWrite).
3. Подготовить информацию для записи в файл.
4. Записать в файл компоненты (процедура Write).
5. Закрыть созданный файл (процедура Close).

Доступ к компонентам файла:

1. Присвоить файлу имя (процедура Assign).
2. Открыть уже существующий файл (процедура Reset).
3. Последовательно считать компоненты (процедура Read).
4. Обработать считанные компоненты.
5. Закрыть файл (процедура Close).

Для типизированного файла можно организовать прямой доступ к каждой компоненте: 2 способа организации произвольного доступа:

1. *Создать последовательный файл и обращаться к компонентам по их порядковому номеру.*
2. *Создать файл фиктивных записей, обращаться к компоненту по ключу с помощью процедуры Seek.*

Дополнение файла:

1. Присвоить файлу имя (процедура Assign).
2. Reset – открыть уже существующий файл.
3. Установить указатель за последним компонентом процедурой Seek ((FV, Filesize(Fv)).
4. Подготовить информацию для нового компонента.

5. Записать новый компонент (процедура Write)
6. Закрыть файл (процедура Close)

Корректирование записей:

1. Присвоить файлу имя (процедура Assign).
2. Открыть файл (процедура Reset).
3. Подвести указатель к корректируемому компоненту. (Seek)
4. Считать корректируемый компонент (процедура Read)
5. Откорректировать нужные поля.
6. Повторить процедуру подвода указателя (процедура Seek)
7. Записать откорректированный компонент (процедура Write).
8. Закрыть файл (процедура Close).

Выборка из файла по определённому признаку:

1. Присвоить файлу имя (процедура Assign).
2. Открыть уже существующий файл (процедура Reset).
3. Ввести данные для поиска.
4. Последовательно считывать компоненты (процедура Read), сравнивая нужные поля. Компоненты, подходящие по признаку сохранить или распечатать.
5. Закрыть файл (процедура Close).

Удаление записей из файла. Удалить запись с номером k:

1. Присвоить файлу имя (процедура Assign).
2. Открыть уже существующий файл (процедура Reset).
3. Установить указатель на удаляемую запись t:= k ; Seek(FV, t)).
4. Повторять пункты 4.a-4.c до окончания файла.
   a. Считать запись с номером t (процедура Read).
   b. Установить указатель на запись с номером t-1.
   c. Записать запись на место t–1 .
5. Установить указатель на последнюю переписанную запись и удалить все записи после неё.
   Seek(FV,t-1); Truncate(FV);
6. Закрыть файл (процедура Close).

Пример 1: Программа phones создаёт телефонный справочник. Телефоны после запуска программы вводятся пользователем и записываются в типизированный файл tel_book, затем содержимое файла выводится на экран:

```
program phones;
type
   Zapis = record {файл состоит из элементов типа запись}
```

```
                fam: String[20];
                tel: String[6]
          end;
    var
          out: file of Zapis;
          nam: Zapis;
          kon: Char;
    begin
          Assign(out,'tel_book');
          Rewrite (out);
      repeat
          Write ('Введите фамилию абонента:');
          ReadLn(nam.fam);
          Write(' Введите номер телефона:');
          ReadLn(nam.tel);
          Write(out, nam);{запись помещается в файл}
          WriteLn ('Будете продолжать? Y/N');
          ReadLn(kon);
      until kon='N';
      {Выведем содержимое файла на экран}
      reset(out);
      {Без закрытия файла не будут видны изменения,
    reset закроет файл, а потом откроет его заново}
      while not Eof(out) do begin
          Read(out, nam);
          WriteLn(nam.fam:14, '-', nam.tel); end;
      Close(out);
    end.
```

Программа 2: Программа создания, записи в файл и чтения из файла с форматированным выводом на экран базы данных, в которой будут отражены данные о человеке: фамилия, имя, отчество, дата рождения, месяц рождения, год рождения:

```
Procedure bazad;
  type {создаем тип}
  dan=record {тип записи с именем dan}
    fam,ima,otc : string[20];{фамилия, имя, отчество}
    data_rog : 1..31;        {день рождения}
    mes_rog  : 1..12;        {месяц рождения}
    god_rog  : 1800..1999;   {год рождения}
    end; {конец описания типа записи dan}
    var
       baza_dan: dan;
      fbaza: file of dan; {типизированный файл,куда и
```

```
                              будем заносить все данные}
      kn: char; {переменная для контроля окончания ввода
данных}
      i: byte;
  Procedure oform;{процедура вывода на экран названий полей
для качественного оформления вывода базы данных}
    begin
      Clrscr;
      gotoxy(1,1);{оформление названий столбцов}
      writeln('N');
      gotoxy(3,1);
      writeln('Фамилии');
      gotoxy(24,1);
      writeln('Имена');
      gotoxy(45,1);
      writeln('Отчества');
      gotoxy(66,1);
      writeln('День');
      gotoxy(69,1);
      writeln('Мес');
      gotoxy(74,1);
      writeln('Год');
    end;
     {.......................}
  begin
    assign(fbaza,'c:\uceb\bd1.baz');
    rewrite(fbaza);
    kn:=' ';
    while ord(kn)<>27 do {пока код переменной kn не равен 27
(Esc) повторяем}
      begin
        Clrscr;
        writeln('Задайте фамилию');
        readln(baza_dan.fam);
        writeln('Задайте имя');
        readln(baza_dan.ima);
        writeln('Задайте отчество');
        readln(baza_dan.otc);
        writeln('Задайте день рождения');
        readln(baza_dan.data_rog);
        writeln('Задайте месяц рождения');
        readln(baza_dan.mes_rog);
        writeln('Задайте год рождения');
        readln(baza_dan.god_rog);

        write(fbaza,baza_dan);{запись в файл
                               очередной порции данных}
```

```
        writeln;
        writeln('Нажмите Esc, если ввод данных закончен,
иначе пробел.');
        kn:=readkey;
      end;
      close(fbaza);
{прочитаем данные из типизированного файла с выводом на
экран}
      oform;{обращаемся к процедуре оформления экрана}
      reset(fbaza);{открываем файл с базой данных}
      i:=0; {i - указатель на строку, которую читаем}
      while i<filesize(fbaza) do
        begin
          seek(fbaza,i);{устанавливаем указатель на компонент
с номером i;счет для типиз. файлов нач. с 0}
          read(fbaza,baza_dan);{читаем из файла в переменную
baza_dan очередную запись}
          gotoxy(1,i+2);{далее выводим поля записи в нужных
местах экрана}
          write(i+1);
          gotoxy(3,i+2);
          write(baza_dan.fam);
          gotoxy(24,i+2);
          write(baza_dan.ima);
          gotoxy(45,i+2);
          write(baza_dan.otc);
          gotoxy(66,i+2);
          write(baza_dan.data_rog);
          gotoxy(69,i+2);
          write(baza_dan.mes_rog);
          gotoxy(74,i+2);
          write(baza_dan.god_rog);
          inc(i);{к следующей записи в файле}
      end;
     {.......................}
      Readkey;{сделаем так, чтобы строки из файла выводились
те, которые мы зададим}
      oform;{вновь обращаемся к процедуре оформления экрана}
      i:=0;
      while i<filesize(fbaza) do
        begin
          gotoxy(1,21);
          writeln('Задайте номер записи, которую вы хотите
видеть.');
          writeln('Если номер будет больше последнего номера
записи в файле');
```

```
          writeln('(т.е.>',filesize(fbaza)-1,') то вывод
закончится.');
          readln(i);{задаем номер той записи, которую хотим
видеть}
          if i<filesize(fbaza) then begin
             seek(fbaza,i);{устанавливаем указатель на
компонент с номером i}
             read(fbaza,baza_dan);{читаем из файла в
переменную baza_dan указанную нами запись}
             gotoxy(1,2);{далее выводим поля записи в нужных
местах экрана}
             ClrEol;{очистка строки от курсора до конца
экрана, чтобы следующая строка не накладывалась на
предыдущую}
             write(i);
             gotoxy(3,2);
             write(baza_dan.fam);
             gotoxy(24,2);
             write(baza_dan.ima);
             gotoxy(45,2);
             write(baza_dan.otc);
             gotoxy(66,2);
             write(baza_dan.data_rog);
             gotoxy(69,2);
             write(baza_dan.mes_rog);
             gotoxy(74,2);
             write(baza_dan.god_rog);
          end;
      end;
      close(fbaza);
      readkey;
    end;
{---Основная программа-----------------------}
begin
  bazad;
end.
```

Задание:

1. Пусть имеется типизированный файл. Создайте такую процедуру, чтобы можно было ранее созданный файл дополнять новыми записями.
Примечание: установите указатель на число, равное filesize(fbaza).
2. Пусть имеется типизированный файл. Написать процедуру, отсекающую все записи, кроме записей с номерами 0 и 1.

3. Пусть имеется типизированный файл, описанный как file of char. Написать программу, подсчитывающую сколько раз символ ‘R’ встречается среди элементов файла.
4. Пусть имеется типизированный файл, описанный как file of char. Написать программу, определяющую имеются ли в нём все символы от ‘A’ до ‘Z’
5. Пусть имеется типизированный файл. Написать программу, корректирующую записи по указанному номеру записи.
6. Пусть имеется типизированный файл целых чисел. Написать программу сортирующие элементы файла по неубыванию.
7. Пусть имеется типизированный файл компонентом которого являются целые числа. Написать программу, вычисляющую сумму простых чисел, содержащихся в файле.
8. Пусть имеется типизированный файл компонентом которого являются одномерные массивы. Написать программу, вычисляющую среднее арифметическое элементов каждой компоненты файла.
9. Пусть имеется типизированный файл, описанный в программе 1. Осуществить выборку данных об абонентах, номера телефонов которых начинаются с цифр ‘53’.
10. Пусть имеется типизированный файл, описанный как file of char. Написать программу, вычисляющую среднее арифметическое кодов символов, содержащихся в файле.
11. Написать программу, которая считывает записи из одного файла и записывает их во второй файл в обратном порядке.

ЛИТЕРАТУРА

1. *Абрамов С. А., Зима В. С.* Начало программирования на языке ПАСКАЛЬ. – М.: Наука, 1987. – 112 с.
2. *Вальвачев А. Н., Крисевич В. С.* Программирование на языке Паскаль для персональных ЭВМ ЕС: Справочное пособие. – Мн.: Вышэйшая школа, 1989. – 223 с.
3. *А.П. Кейзер, Т.Е. Кабакова, З.Н.Рогачева, С.Г. Халамов* Решение задач контрольной работы средствами математического пакета MathCad и табличного процессора Excel. – Гомель, 2001. – 14 с.
4. Информатика. Программирование на языке Паскаль: Практикум по лабораторным работам. Ч.1 /*.А.П. Кейзер, Ю.А. Пшеничнов, М.В. Борисенко, О.И. Еськова;* Под ред. Ю.А. Пшеничнова – Гомель:БелГУТ, 2001.- 46с.
5. *Фаронов В.В.* Программирование на персональных ЭВМ в среде Турбо-Паскаль. – М.: Изд-во МГТУ, 1990. – 590 с.
6. *Фаронов В.В. Турбо Паскаль 7.0.* Начальный курс: Учеб. пособие. – М.: "Нолидж", 1998. – 616 с.