Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

"КАЛИНИНГРАДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ"

Программирование

Учебно – методическое пособие по лабораторным работам

для студентов бакалавриата

по направлениям подготовки

" Информатика и вычислительная техника"

"Прикладная информатика"

(часть вторая)

Калининград

Издательство ФГБОУ ВПО "КГТУ"

2015

УДК 681.3.06

УТВЕРЖДЕНО

Ректором ФГБОУ ВПО

«Калининградский

государственный технический

университет»

АВТОРЫ: Гудков А.Л., к.т.н., доцент кафедры систем управления и вычислительной техники» ФГБОУ ВПО «Калининградский государственный технический университет»;

Калинина С.А., доцент той же кафедры.

Учебно – методическое пособие рассмотрено и одобрено кафедрой систем управления и вычислительной техники ФГБОУ ВПО «Калининградский государственный технический университет» 03 июля 2015 г., протокол №7.

Учебно – методическое пособие рекомендовано к печати методической комиссией факультета автоматизации производства и управления ФГБОУ ВПО «Калининградский государственный технический университет» \_\_ сентября 2015 г., протокол № \_\_.

**РЕЦЕНЗЕНТ –** кафедра систем управления и вычислительной техники ФГБОУ ВПО «Калининградский государственный технический университет»

**Введение**

Лабораторный практикум по дисциплине "Программирование" (с использованием системы Турбо-Паскаль) для студентов бакалавриата по направлениям подготовки в области информатики и вычислительной техники (прикладной информатики) предусматривает выполнение 11-ти работ. Задания по первым семи лабораторным работам, в числе которых:

1) Освоение системы программирования Турбо-Паскаль;

2) Разработка линейной программы;

3) Разработка разветвляющейся программы;

4) Разработка циклической программы;

5) Разработка и использование подпрограмм;

6) Разработка программы обработки массивов данных;

7) Разработка программы со сложным циклом,

и методические указания по выполнению их приведены в [1].

В настоящем пособии приводятся сведения по "второй части" лабораторного практикума из четырех работ:

8) Разработка программы обработки символьных данных;

9) Разработка программы диалоговой задачи;

10) Разработка программ создания и обработки типизированных файлов последовательным методом доступа;

11) Разработка программ создания и обработки типизированных файлов прямым методом доступа.

Приобретение умений и навыков по модульному программированию, разработке программ в соответствии с технологией структурного процедурно-ориентированного программирования (с использованием системы Турбо-Паскаль) осуществляется при выполнении курсовой работы по дисциплине [2] после лабораторного практикума.

**Лабораторная работа № 8**

**Разработка программы обработки символьных данных**

**Цель работы:** получение умений и навыков алгоритмизации и программирования процедур с символьными данными.

**Задание по лабораторной работе**

Разработать подпрограмму обработки исходного предложения на русском языке (из 50÷80 знаков, в котором слова отделяются друг от друга пробелами), с целью (результат обработки предложения определяется номером варианта):

1) подсчета в предложении знаков препинания;

2) замены в предложении всех экземпляров заданного знака на другой заданный знак;

3) подсчета в предложении количества слов;

4) замены в предложении всех знаков препинания знаком подчеркивания;

5) определения, какие гласные буквы в предложении содержатся;

6) подсчета в предложении количества слов, начинающихся заглавными буквами;

7) определения, содержится ли в предложении заданное слово;

8) подсчета в предложении заданного союза (например, "и", "или" и т.п.);

9) определения, какая буква в предложении встречается наибольшее число раз;

10) определения, все ли гласные буквы встречаются в предложении;

11) определения, есть ли в предложении слова, начинающиеся заглавными буквами;

12) подсчета числа слов в предложении, начинающихся согласными буквами;

13) определения, содержатся ли в предложении слова с заданным корнем;

14) подсчета числа слов в предложении, начинающихся заданной буквой;

15) удаления из предложения знаков препинания;

16) подсчета количества слов в предложении, которые не содержат заданную букву.

Включить текст подпрограммы в модуль Program и обеспечить многократные исполнения ее для разных вариантов исходных предложений и других (если они есть) аргументов процедуры.

**Методические указания по выполнению лабораторной работы**

1. Для выполнения задания следует усвоить форматы представления символьных данных (знаков и строк) в компьютере и в Паскаль-программе и изучить средства программирования операций с ними.

2. Система программирования Турбо-Паскаль располагает широким набором стандартных подпрограмм (процедур и функций), аргументами и результатами которых являются символьные данные. Поэтому необходимо ознакомиться с ними и выбрать те, которые уместно использовать для получения требуемого результата обработки исходного предложения.

3. При алгоритмизации обработки предложения на русском языке (исходной строки, состоящей из знаков) можно использовать алгоритмы "типовых процедур с одномерными массивами" (см. лабораторную работу №6 в [1]). При этом вместо операций с элементами массива данных (числами) следует предусматривать операции со знаками и (или) цепочками знаков, выделяемых из предложения–строки.

4. Для разработки программы с символьными параметрами необходимо уточнить особенности строковых формальных параметров (параметров-значений и параметров-переменных) и вызова подпрограмм с ними для исполнения.

**Теоретическая (справочная) часть**

К **символьным данным** относятся **знаки** и **строки**.

**Множество знаков** (их 256 – буквы, цифры, другие знаки) в Турбо-Паскале является стандартным типом данных с именем **char.** Изображения их констант заключаются в апострофы: ´5´, ´а´, ´+´ и т.п. Описание переменной, значением которой может быть знак, содержит имя соответствующего типа данных, например,

Var с: char;

и предусматривает выделение однобайтового поля памяти для представления в нем кода знака (каждый знак представляется уникальным кодом, который может интерпретироваться как знак или как десятичное число без знака из диапазона 0..255). Так, например, кодом знака ´\_\_´ (пробел) является число 32.

**Строка** – это конечная последовательность знаков (до 255-ти). Строковая константа заключается в апострофы (´Пример \_\_ строки´). При описании строковых переменных в Турбо-Паскале используют стандартное имя string, например:

Var s1: string [80];

s2: string;

В первом случае значением переменной s1 может быть строка не более чем из 80 знаков, во втором – максимум из 255-ти знаков. Однако, строковый тип (множество строк) не относится к стандартным типам и, хотя строка может использоваться в операциях как одно "одиночное" значение, она все-таки является структурным значением – из нее можно выбирать отдельные знаки (как элементы из одномерного массива). Например, по оператору

c: = s1 [1];

первый знак из строки, являющейся значением переменной s1, должен стать значением переменной **с** (типа char).

**Знаки и строки могут участвовать в операциях сравнения** (отношения). При этом сравниваются коды знаков (для строк - знаков, из которых они состоят) как десятичные числа. Например, ´А´ < ´а´, т.к. код прописной буквы больше кода заглавной. А ´Кран´ > ´Краб´, т.к. код буквы ´н´ больше кода буквы ´б´ (они возрастают в алфавитном порядке). Строки из разного числа знаков не "равны" ( ´строка´ < > ´строка\_´).

В Турбо-Паскале имеется широкий набор стандартных подпрограмм, аргументами и (или) результатами которых являются знаки и строки. Приведем наиболее часто используемые из них:

1) **Функция Length (s)** дает текущую длину строки s (количество знаков, из которых она в данный момент состоит);

2) **Функция Copy (s, n, i)** выделяет часть строки s ("подстроку", "цепочку знаков"), начиная со знака с номером n из i знаков;

3) **Функция Pos (s1, s2)** дает номер знака в строке s2, начиная с которого строка s1 первый раз в нее входит или значение 0, если s1 в s2 отсутствует (первый аргумент может быть знаком, тогда значение функции – номер этого знака в строке s2);

4) **Функция UpCase(Ch)** – преобразует строчную латинскую букву в прописную;

5) **Процедура Delete (s, n, i)** - удаляет из строки s i знаков, начиная со знака с номером n;

6) **Процедура Insert (s1, s2, n) -** вставляет строку (знак) s1 в строку s2, начиная со знака с номером n (в качестве знака с номером n);

7) **Процедура Val (S, V, Code)** – преобразует строковое значение S в числовое значение переменной V (вещественной или целой), Code - номер позиции в строке, в которой находится ошибочный для преобразования знак, если ошибочного знака нет, значение Code будет равно 0;

8) **Процедура Str (Х, S)** – преобразует числовое значение Х в строку знаков S, изображающих это число.

Примеры использования этих стандартных подпрограмм приведены ниже.

…

Var s1, s2: string; r: boolean;

…

If Length (s1) > 0 then

If Length (s1) < = Length (s2) then

If Pos (s1, s2) > 0 then r: = true;

…

При s1 = ´программа´ и s2 = ´подпрограмма´ r = true, т.к. длина строки s1 не больше длины строки s2 (9 <12) и строка s1 входит в строку s2, начиная с 4-го знака (Pos (s1, s2) = 4). Первая проверка в этом примере позволяет определить, не является ли строка s1 "пустой", т.е. не содержащей ни одного знака (такой вариант возможен). Длина такой строки, изображаемой двумя апострофами (´´), равна нулю.

…

Var s: string; n: byte;

…

n := Pos (′\_′, s);

If n > 0 then Delete (s, n, l);

…

При s = ´пример \_\_ \_\_ строки´ n = 7 и тогда после процедуры Delete

s = ´пример \_ строки´ (из строки s будет удален 7-ой знак – первый из двух пробелов) и длина строки уменьшится на единицу.

…

Var s: string;

…

S: = ′Паскаль′;

Insert (′Турбо-′ , s, l);

…

После выполнения операторов значение строковой переменой s станет ′Турбо-Паскаль′ (его длина увеличится на 6 и составит 13 знаков).

**Знаки** и **строки** могут быть **параметрами подпрограмм**, а также значениями функций. Так, заголовок подпрограммы

Procedure P1 (c1, c2: char; Var s: string);

cодержит два аргумента, которыми должны быть знаки (с1, с2) и строковый параметр-переменную s (например, изменяемую по подпрограмме строку).

Отметим, что при описании строковых формальных параметров использование ограничителя (указателя) максимальной длины строки после имени типа string не допускается. Однако, соответствующий фактический параметр, при вызове подпрограммы может быть описан с таким указателем (т.е. иметь максимально возможную длину менее 255 знаков).

При этом следует учитывать, что если строковый формальный параметр является параметром-переменной, то перед оператором вызова подпрограммы следует указывать директиву компилятора {$V -}, отключающую контроль за совпадением максимальной длины фактического параметра с длиной формального параметра.

Например:

…

Var S80: string[80];

…

{$V -} P1 (′.′, ′;′, s80);

…

**Примеры программирования**

В качестве первого примера приведем **подпрограмму определения количества слогов в исходном русском слове** (подсчета числа гласных букв русского алфавита в исходной строке):

Function FCL (S: string): byte;

Const vl='АЕИОУЫЭЮЯаеиоуыэюя';

Var n, i: byte;

Begin

n:=0;

For i:= 1 to length (s) do

if Pos (S[i], vl) >0 then n:=n+1;

FCL:= n;

End;

В ней предусматривается поиск с помощью функции Pos каждого знака исходной строки s в строковой константе, состоящей из заглавных и прописных гласных букв русского алфавита. Если очередной знак является гласной буквой, значение Pos (номер этого знака в строке vl) больше нуля и число гласных букв (значение n) увеличится на единицу. Так, для строки ´Алгоритм´ значение функции (FCL (´Алгоритм´)) будет 3.

Вторым примером является **подпрограмма замены в исходной строке заглавных русских букв на прописные:**

Procedure PBL (Var S: string);

Const big='АБВГДЖЗИЙКЛМНОПРСТУФХЦШЩЪЬЫЭЮЯ';

lit: string = 'абвгджзийклмнопрстуфхцшщъьыэюя';

Var n,i: byte;

Begin

For i:= 1 to length (s) do

begin

n:= Pos (S[i], big);

If n>0 then S[i]:= lit[n];

end;

End;

В подпрограмме исходная строка s изменяется, если она содержит заглавные русские буквы. При этом предусматривается поиск в строковой константе из заглавных букв с именем big каждого знака исходной строки с помощью функции Pos. Если очередной знак является заглавной буквой, номер этой буквы в строке big становится ненулевым значением функции Pos и на место заглавной буквы помещается соответствующая ей прописная буква, выбираемая из другой строковой константы с именем lit. Например, строку ′Cидоров\_В.\_И.′ Подпрограмма преобразует в строку ′cидоров\_в.\_и.′.

Третий пример – **подпрограмма удаления из исходной строки всех пробелов:**

Procedure DDelB (Var S: string);

Var n, i: byte;

Begin

If S>'' then begin

n:= Pos(' ', S);

While (n>0) and (S>'') do begin

Delete (s, n , 1);

n:=Pos(' ', S);

end;

end;

End;

В ней, если строка s не "пустая", предусматривается поиск с помощью функции Pos знака "пробел" (′\_′), пока такие знаки еще в строке есть, и удаление с помощью процедуры Delete найденных пробелов. Здесь n – номер первого пробела в строке s.

На рис. 8.1 и рис. 8.2 приведены **схема алгоритма и** **подпрограмма формирования из исходного предложения на русском языке (строки) массива слов, из которых оно состоит**.Считается, что слова отделяются друг от друга пробелами (одним или несколькими) и предложение не содержит знаков препинания. В подпрограмме для каждого слова в предложении предусматривается удаление левых пробелов (которые предшествуют слову) и самого слова – после копирования его в качестве элемента формируемого массива. В итоге исходная строка, ставшая при вызове подпрограммы значением формального параметра s, станет "пустой" (s = ″). Однако фактический параметр не изменится. Так, для исходного предложения:

′\_ \_ \_ пример \_ \_ строк \_ из \_ \_ \_слов′

по этой подпрограмме будет получен массив из четырех слов. Однако после выбора в массив первого слова значение исходного параметра s станет

′\_ \_строк \_ из \_ \_ \_ слов′,

после выбора второго слова:

’\_ из \_ \_ \_ слов’,

после выбора третьего слова:

′\_ \_ \_ слов′.

После выбора четвертого (последнего) слова она станет "пустой" строкой (s = ″) и процедура завершится.

да

Есть текст?

Подготовка формирования массива

Н

Определение длины слова

Слово есть?

Выбор слова в массив

Удаление пробелов перед словом

Удаление слова из предложения

К

да

нет

нет

Рис. 8.1. Схема алгоритма формирования массива слов из предложения

|  |
| --- |
| Type taw=array [1..30] of string [20];  Procedure FormArw (S: string; Var Arw: taw; Var n: integer );  Var n1, lw: integer;  Begin  {Подготовка формирования массива}  n:=0; {n – количество слов в массиве}  {Цикл выбора слов из предложения с удалением слов в предложении}  While (S>'') do begin  {Удаление пробелов перед словом}  While (s[1]=' ') do Delete (s, 1, 1);  If S>'' then {слово есть}  begin  {определение позиции пробела после слова}  n1:=Pos(' ', s);  {определение длины слова lw }  If n1=0 then {последнее слово} lw:=length(s)  else lw:=n1-1;  {Формирование массива}  n:=n+1;  Arw[n]:= copy (s, 1,lw);  {Удаление слова из предложения}  Delete (s, 1, lw);  end;  end;  End; |

Рис. 8.2.Подпрограмма формирования массива слов из исходного предложения

**Контрольные вопросы и задания**

1. Что понимают под "символьными" данными при программировании? Как они представляются в Паскаль-программах и в компьютере?
2. Какие операции допустимы со знаками и строками? Как они выполняются?
3. Что понимается под длиной строки? При выполнении каких операций изменяется длина значения строковой переменной?
4. Какие стандартные подпрограммы используются для:

- определения длины строкового значения;

- выделения из строки ее части ("подстроки");

- поиска в строке определенного знака или определенной "цепочки знаков";

- удаления из строки определенной части;

- вставки в строку определенной "цепочки знаков"?

5) Каковы особенности использования символьных данных в качестве параметров подпрограмм?

6) Какие стандартные подпрограммы и с какой целью используются в Вашей подпрограмме?

7) Прокомментируйте разработанную Вами подпрограмму, поясните логику и структуру предусмотренного в ней алгоритма обработки данных, назначение каждого оператора в ней.

**Лабораторная работа № 9.**

**Разработка программы диалоговой задачи**

**Цель работы:**  получение практических навыков программирования диалоговых процедур, программирования управления выводом сообщений на экран монитора в текстовом режиме.

**Задание по лабораторной работе**

Разработать программу, обеспечивающую прием и анализ (проверку на правильность, допустимость) ответа пользователя на вопрос, выведенный ему на экран монитора (текст вопроса определяется вариантом задания). В программе следует предусмотреть:

- подпрограмму вывода вопроса;

- подпрограмму проверки ответа пользователя;

- вывод пользователю сообщения о правильности/неправильности ответа;

- подпрограмму вывода пользователю справочной информации (по его желанию);

- возможность нескольких попыток ответа на вопрос, предоставляемых пользователю в случае неправильного ответа.

Варианты вопросов пользователю (задания по лабораторной работе):

1. "Приведите пример четного десятичного числа без знака ";
2. "Приведите пример любого года ХХI века";
3. "Укажите имя стандартной процедуры чтения данных в Турбо-Паскале";
4. "Приведите пример допустимого в Турбо-Паскале идентификатора";
5. "Укажите имя одного из стандартных целочисленных типов данных в Турбо-Паскале";
6. "Приведите пример указания типа строковой переменой при описании ее в Паскаль-программе";
7. "Укажите любую дату февраля текущего года (в формате ′чч.мм.гггг′);

8) "Укажите имя и фамилию (через пробел, на русском языке) автора стандартного Паскаля";

9) "Перечислите в произвольном порядке через запятую знаки логических операций, используемые в программах";

10) "Перечислите в произвольном порядке через запятую знаки операций отношения, используемые в программах";

11) "Приведите пример строковой константы со словом на русском языке";

12) "Укажите начальное слово оператора цикла на языке Турбо-Паскаль";

13) "Укажите строку из цифр десятичной системы счисления";

14) "Укажите через запятую положительные экзаменационные оценки, изображаемые цифрами";

15) "Укажите дату начала учебного года в ХХI веке".

16) "Укажите название одного из месяцев (в именительном падеже)".

**Методические указания по выполнению лабораторной работы**

1. Разрабатываемая программа должна обеспечивать определенный диалог компьютера с пользователем, т.е. целенаправленный обмен сообщениями между ними с помощью клавиатуры и монитора. Поэтому с учетом задания в программе необходимо предусмотреть:

- предъявление пользователю на экране монитора вопроса (задания);

- прием (чтение с клавиатуры) ответного сообщения (ответа) пользователя (как строкового значения);

- реакцию на ответ пользователя.

При этом следует учесть следующие варианты ответа пользователя:

- отказ от ответа (чтения "пустой" строки);

- запрос на получение справочной информации для ответа (чтения знака ′?′);

- неправильный ответ (ответ с ошибкой или ошибками);

- правильный ответ.

В программе рекомендуется предусмотреть несколько попыток ответа на вопрос (например, до 3-х).

На рис. 9.1 приведена схема возможного алгоритма приема и проверки ответа пользователя на вопрос компьютера.

2. Наиболее сложным и трудоемким при разработке диалоговых программ, как правило, является программирование процедур анализа (распознавания, проверки, контроля правильности, допустимости и т.п.) сообщений пользователя.

В лабораторной работе следует программировать **проверку правильности** **ответа** пользователя на предъявленный ему вопрос, **как соответствующую обработку строки – проверку соответствия ее определенному формату.**  Например, если в качестве ответа должен быть российский номер мобильного телефона, то он должен представлять собой строку, в которой десяти цифрам десятичной системы счисления предшествуют знаки ′+7′.

Для проверки соответствия строки заданному формату должны выполняться операции сравнения знаков и цепочек знаков, из которых состоит ответ-строка, с теми или иными "шаблонами", в т.ч. с использованием стандартной функции Pos (см. описание лабораторной работы № 8 и пример программирования).

3. В соответствии с заданием необходимо процедуры вывода вопроса, справочной информации, проверки ответа пользователю предусмотреть соответствующими подпрограммами. Первые две из них будут без параметров (обеспечивать вывод на экран монитора определенного текста), третья – с выходными параметрами, первый из которых – ответ пользователя (строка), второй – признак ошибки в ответе (логическое значение).

4. Предусматривая в программе диалог с пользователем, можно, используя средства стандартного модуля Турбо-Паскаля с именем Crt, обеспечить **управление** выводом на экран монитора текстовых сообщений – **текстовым видеорежимом**. А именно, определять на экране "окна" для сообщений, задавать цвет текста и его фона, очищать окна, управлять курсором и т.п. Поэтому рекомендуется в разрабатываемой программе использовать стандартные процедуры и функции модуля Crt (Window, ClrScr, TextBackGround, TextColor, GotoXY и т.п.) для более удобного для пользователя видеорежима при диалоге с компьютером. Использование же средств модуля Crt обеспечивается предложением

Uses Crt;

после заголовка программы (Program).

**Теоретическая (справочная) часть**

Сведения о средствах обработки строк, необходимых для программирования предлагаемого в задании диалога, приведены в описании лабораторной работы №8. Здесь же рассматриваются средства управления текстовым видеорежимом из стандартного модуля Турбо-Паскаля Crt.

Средства, предназначенные управлять выводом информации на экран в текстовом режиме, составляют значительную часть ресурсов модуля Crt. К ним относятся управление цветом символов, цветом фона, управление курсором, управление текстовыми окнами, очистка окна. При работе с экраном через модуль Crt весь экран разбивается на отдельные строки, а каждая строка на отдельные позиции, то есть весь экран разбит на отдельные неделимые прямоугольные элементы. На экране можно выделить прямоугольные окна, задавая координаты левого верхнего и нижнего правого углов.

Для управления цветом выводимых символов используется процедура **TextColor** (Color), а для управления цветом фона – **TextBackGround**(Color). Аргумент Color может принимать значения от 0 до 15, каждое из которых задает определенный цвет (от черного до белого).

Например: операторы TextColor (4); TextBackGround (7);

обеспечивают появление красных символов на светло-сером фоне.

Процедура **ClrScr** приводитк очистке активного окна, курсор при этом помещается в левый верхний угол экрана.

Процедура определения окна **Window** (x1, y1, x2, y2) использует координаты левого верхнего угла - x1 и y1 и координаты нижнего правого угла окна - x2 и y2 (целые числа типа byte). Если в качестве параметра задано неверное значение, процедура не выполняется.

Например: операторы Window (5, 5, 15, 10); ClrScr;

обеспечивают очистку квадратного окна размером 10 на 10 позиций.

Процедура управления курсором **GotoXY** (x, y) перемещает курсор в позицию, заданную координатами x, y (целые числа типа integer) в текущегм окне. Так, оператор GotoXY (1, 1) переместит курсор в верхний левый угол окна, но не экрана. Если в качестве параметра задано неверное значение, процедура не выполняется.

Функции **WhereX** и **WhereY** определяют координаты текущего положения курсора (в виде целого числа).

Процедуры без параметров **DelLine**, **InsLine**, и **ClrEol** работают со строками.

**DelLine** – удаляет текущую строку (то есть строку, в которой находится курсор). При этом все строки, расположенные ниже удаляемой, сдвигаются вверх.

**InsLine** – вставляет пустую строку. При этом все нижние строки, а также строка, в которой размещался курсор, сдвигаются вниз. Текущее положение курсора не меняется.

**ClrEol** – стирает все символы в строке, начиная с текущей позиции до конца строки. Текущее положение курсора не меняется.

**Пример программирования**

В качестве примера **приведем алгоритм и программу, обеспечивающие** прием и проверку правильности ответа пользователя на вопрос – "Укажите номер мобильного телефона" (рис. 9.1 и рис. 9.2). В программе используются три внутренних (включенные в описательную часть модуля Program) подпрограммы:

- вывода вопроса на экран монитора (OutQ);

- вывода справки о формате правильного ответа (OutH);

- чтения и проверки правильности ответа пользователя (ReadA).

Первые две подпрограммы-процедуры не содержат параметров и предусматривают вывод на экран монитора определенных сообщений, представленных в виде строковых констант в операторах Writeln.

Выходными параметрами подпрограммы чтения и проверки ответа ReadA являются:

- ответ пользователя (строка);

- признак правильности ответа (логическое значение r - при r = true нет ошибки, при r = false ошибка есть).

Проверка ответа сводится к проверке его соответствия шаблону

′+7хххххххххх′, где х – цифра десятичной системы счисления.

В основной программе описан алгоритм, схема которого представлена на рис. 9.1. В ней уточнено условие выполнения цикла чтения и проверки ответа пользователя. Цикл повторяется, если в ответе пользователя есть ошибка (r = false) и число попыток ответа не более трех, или после вывода по запросу пользователя справочной информации.

нет

да

да

нет

нет

нет

Ответесть?

Н

Чтение и проверка ответа

К

Вывод вопроса

Ошибка?

Ответесть?

Справ-ка?

Вывод сооб-щения

Вывод сооб-щения

Вывод сооб-щения

Вывод сооб-щения

да

да

Рис. 9.1. Схема алгоритма приема и проверки ответа пользователя

|  |
| --- |
| Program Dialog;  **Uses Crt;**  **Procedure OutQ;**  {Подпрограмма вывода вопроса с определенной позиции экрана}  Begin  GotoXY (10,5); Writeln ('Укажите номер мобильного телефона');  GotoXY (10,5) ; Writeln ('Для справки нажмите клавишу " ?" ');  End;  **Procedure OutH;**  {Подпрограмма вывода справки с изменение цвета и фона шрифта}  Begin  Clrscr; TextColor (8); TextBackGround (7);  Writeln (' Номер телефона должен состоять из десяти цифр,  следующих за знаками " +7" ');  End;  **Procedure ReadA (Var a:string; Var r: boolean );**  {Подпрограмма чтения и проверки ответа}  Var l, i: byte;  Begin  Readln (a);  r:=false; l:=length(a);  If l=12 then  If (Copy(a, 1, 2) = '+7') then  Begin  r := true; i:=3;  While (r = true) and (i <= 12) do  If Pos(a[i], '0123456789') = 0 then  r:=false else i := i+1;  end;  End; |

Рис. 9.2. Программа приема и проверки правильности ответа на вопрос

|  |
| --- |
| {Основная программа}  Const  r1 = 'Ответ не получен'; r2 = 'Неправильный номер';  r3 = 'Возможный номер'; r4 = 'Все попытки ответа использованы';  Var a: string; r: boolean; i: integer;  Begin {Вывод вопроса на экран монитора}  OutQ;  r := true; i := 1;  While r and (i <= 3) do  Begin  {Определение цвета и фона для ввода ответа}  TextColor (15); TextBackGround (9);  ReadA (a, r); {Чтение и проверка ответа}  If a = '' then {отказ от ответа и выход из цикла} begin  Writeln(r1); r := false;  end  else begin  If a = '?' then {нужна справка}  begin OutH; {вывод справки}  OutQ; end  else begin  If r=false then { в ответе ошибка}  begin Writeln (r2); i := i+1;  If i>3 then {попытки исчерпаны}  Begin;  {Вывод сообщения в нижнем правом углу экрана}  Clrscr; Window (45, 22, 76, 24);  TextBackGround (4); Writeln (r4); end;  end  else {ответ правильный}Writeln (r3);  end;  end;  end;  End. |

Рис. 9.2. Программа приема и проверки правильности ответа на вопрос (продолжение)

**Контрольные вопросы и задания**

1) Что собой представляет диалог пользователя с компьютером?

2) Какие основные процедуры предусматриваются при программировании диалога пользователя с компьютером?

3) Какие приемы, средства использованы Вами для анализа ответных сообщений пользователя?

4) Каким образом предусматривается возможность многократных попыток ответа пользователя на вопрос (задание) и в каких случаях?

5) Как очистить экран перед выводом результата?

6) Какие средства Паскаля позволяют вывести сообщение с определенной позиции экрана?

7) Какие процедуры Паскаля позволяют изменять цвет выводимых сообщений?

**Лабораторная работа № 10.**

**Разработка программ создания и обработки типизированных файлов последовательным методом доступа**

**Цель работы:** получение практических навыков программирования процедур создания и обработки типизированных файлов на магнитных носителях информации последовательным методом доступа к их элементам с использованием записей.

**Задание по лабораторной работе**

Разработать три программы – создания файла, вывода файла на экран монитора и обработки файла на магнитном диске, состоящего из 10-20 записей заданной структуры (структура элемента файла, а также результат обработки определяются номером варианта задания в табл.10.1 или преподавателем).

**Варианты задания**

Таблица 10.1

| № | Структура элемента файла (записи) | Результат обработки файла |
| --- | --- | --- |
| 1 | шифр учебной группы (строка из 6 знаков);  фамилия и инициалы студента (строка из 25- знаков);  дата рождения (строка из 10 знаков). | Список студентов указанной группы |
| 2 | фамилия и инициалы автора книги (строка из 25 знаков);  название книги (строка из 40 знаков);  год издания (целое число). | Список книг, изданных в указанном году |
| 3 | наименование товара (строка из 25 знаков);  номер месяца (целое число);  объем продаж в стоимостном выражении (вещественное число). | Сведения об объемах продаж товаров в указанном месяце |
| 4 | наименование страны (строка из 20 знаков);  наименование города (строка из 20 знаков);  численность жителей (целое число – тысяч человек). | Список городов с населением не менее указанного значения |
| 5 | номер авиарейса (строка из 7 знаков);  наименование города назначения (строка из 20 знаков);  время вылета по расписанию (строка из 5 знаков); | Список авиарейсов в указанный город |
| 6 | фамилия и инициалы работника (строка из 25 знаков);  год рождения (целое число);  стаж работы на предприятии (целое число). | Список работников, стаж работы которых не менее указанного значения |
| 7 | фамилия и инициалы футболиста (строка из 25 знаков)  наименование команды (строка из 25 знаков);  количество забитых голов (целое число). | Список футболистов, забивших не менее указанного количества голов |
| 8 | марка автомобиля (строка из 20 знаков);  год выпуска автомобиля (целое число);  стоимость автомобиля (целое число). | Список автомобилей, стоимость которых не более указанной суммы |
| 9 | наименование судна (строка из 25 знаков);  дата прибытия в порт (строка из 10 знаков);  дата выхода в рейс из порта (строка из 10 знаков) | Список судов, которые в указанную дату будут находиться в порту |
| 10 | адрес квартиры (строка из 40 знаков);  количество комнат (целое число);  площадь квартиры (вещественное число);  стоимость квартиры (целое число). | Сведения о стоимости квартир, количество комнат в которых равно указанному значению |
| 11 | наименование учебной дисциплины (строка из 40 знаков);  форма аттестации по дисциплине (´зачет´/´экзамен´);  семестр аттестации (целое число). | Список дисциплин, аттестация по которым проводится в указанном семестре |
| 12 | фамилия, инициалы работников (строка из 25 знаков);  должность работника (строка из 20 знаков);  дата окончания трудового договора с работником (строка из 10 знаков). | Список работников, трудовые договора с которыми заканчиваются в указанном году |
| 13 | номер поезда (строка из 6 знаков);  город назначения (строка из 25 знаков);  время отправления (строка из 5 знаков);  время в пути (строка из 5 знаков). | Список поездов в указанный город |
| 14 | наименование страны турпоездки (строка из 25 знаков);  наименование месяца (строка из 8 знаков);  количество проданных туров (целое число). | Сведения о продаже туров в указанном месяце |
| 15 | наименование товара (строка из 25 знаков);  наименование магазина (строка из 20 знаков);  стоимость товара (вещественное число). | Сведения о стоимости указанного товара по магазинам |
| 16 | наименование вуза (строка из 10 знаков);  год выпуска специалистов (строка из 4 знаков);  количество выпускников (целое число). | Сведения о количестве выпускников вузов в указанном году |

**Методические указания по выполнению работы**

1. Для разработки трех требуемых программ необходимо усвоить основные понятия и средства обмена данными между оперативной и внешней (на магнитных носителях информации) памятью компьютера, используемых в Турбо-Паскале. К ним, в первую очередь относятся:

- понятие **файла** данных (типизированного файла, в частности) как структурного значения на магнитном носителе информации;

- понятие **файловой переменной**, являющейся "представителем" файла в исходной программе;

- **стандартные подпрограммы** (процедуры, функции) **для** **работы с файлами** (Assign, Reset, Rewrite, Read, Write, Close, Eof и т.д.).

Кроме того, следует изучить еще один тип структурных значений, называемых в Турбо-Паскале – **записями** (запись – это структура именуемых значений возможно различного типа), потому что элементами файла, который должен создаваться и обрабатываться по разрабатываемым программам, должны быть именно записи.

2. В программе **создания типизированного файла** следует описать:

- тип элемента файла (тип определенной записи);

- файловую переменную с использованием имени типа элемента файла;

- переменную, тип которой совпадает с типом элемента файла.

А также предусмотреть:

- подготовительные для создания файла процедуры Assign и Rewrite;

- цикл последовательной записи в файл элементов, значения которых должны предварительно (перед записью очередного элемента в файл) вводиться в оперативную память с помощью клавиатуры компьютера.

Условием выполнения цикла записи элементов в файл может быть ввод с клавиатуры в качестве первого элемента записи, которая должна стать очередным элементом файла, "непустой" строки.

3. В **программе вывода элементов файла на экран монитора** кроме описаний, имеющих место в программе создания файла, следует предусмотреть:

- подготовительные процедуры Assing и Reset;

- цикл последовательного чтения из файла элементов в оперативную память и вывода значений, из которых они состоят, на экран монитора.

В условии выполнения цикла следует использовать стандартную функцию **Eof** (ее аргументом в этом случае должна быть файловая переменная, связанная при выполнении процедуры Assing с созданным ранее файлом).

4. **Программа обработки созданного файла** должна предусматривать ввод с клавиатуры "запроса" пользователя на получение нужного списка (нужных сведений) и цикла последовательного чтения из файла элементов с выводом на экран монитора значений только тех элементов, которые соответствуют запросу (для управления циклом также можно использовать функцию Eof).

**Теоретическая (справочная) часть**

**Файлом** называют структуру данных на внешнем для компьютера носителе информации. На магнитных носителях (например, дисках) могут создаваться **типизированные файлы,** состоящие из элементов одного типа. Они именуются по принятым в операционной системе компьютера правилам (например, именем файла, расположенным на диске С, может быть строка 'C:\TP\Work\F1.dat'). Элементы файла последовательно нумеруются, начиная с нуля.

В Паскаль-программах создания и обработки файлов необходимо описывать, а затем использовать особую переменную – **файловую переменную**, являющуюся "представителем" файла в программе. При ее описании для типизированных файлов указывается тип элемента файла. Например, в описании

Var FV: file of string [80];

определена файловая переменная с именем FV для файла, элементами которого должны быть строки с максимально возможной длиной, равной 80-ти.

Как правило, элементами типизированного файла на магнитном носителе являются "записи". Под **записью** в Турбо-Паскале понимают именуемое структурное значение, состоящее из других именуемых значений возможно различного типа. Например, предложение

Var ab: record

fio: string [25];

adr: string [40];

nt: string [6];

end;

является описанием переменной, значением которой может быть запись из трех элементов – строк с разной максимально возможной длиной. При этом ab - имя записи, а fio, adr, nt - имена элементов записи (их значениями могут быть фамилия и инициалы, адрес и номер телефона определенного абонента). Поскольку имена есть у записи и ее элементов, в программе можно предусматривать операции с записями в целом и с отдельными элементами ее. При этом для элементов записи могут использоваться составные имена, в которых после имени записи через точку указывается имя элемента записи. Такие имена явно демонстрируют принадлежность значения к определенной записи.

**Тип** **записи** можно определять явно, предварительно дав ему имя в предложении Type, например, так:

Type tab = record

fio: string [25];

ard: string [40];

nt: string [6];

end;

Var ab: tab;

Для файла, состоящего из записей, тип которых описан выше, описание файловой переменной с именем TB будет таким:

Var TB: file of tab;

При создании и обработке файлов используются соответствующие стандартные подпрограммы (процедуры и функции). Первой из них должна выполняться процедура **Assing**, устанавливающая связь между конкретным файлом и файловой переменной, используемой в программе для него. Например, оператор

Assing (TB, 'c:\TP\Work\TB.dat');

предусматривает "назначение" дискового файла с указанным в виде строковой константы именем файловой переменной ТВ (именно она в дальнейшем должна использоваться в операторах "вместо" имени файла).

Перед записью данных во вновь создаваемый файл должна использоваться подготовительная процедура **Rewrite**. Так, если файл, связанный с файловой переменной ТВ, создается, вызов этой процедуры выглядит так

Rewrite (TB);

Если предусматривается обработка существующего файла, используется другая подготовительная процедура – **Reset**, например, так:

Reset (TB).

Для чтения элемента файла в качестве значения определенной переменной используется процедура **Read**. Так, по оператору

Read (TB, ab);

значение доступного для чтения элемента файла, связанного с файловой переменной ТВ, станет и значением переменной ab (ее тип такой же, как и тип элемента файла).

Для записи элемента в файл используется процедура **Write**. Например, по оператору

Write (TB, ab);

значение переменной ab станет и значением элемента файла, представителем которого в программе является файловая переменная ТВ.

Передача данных в типизированный файл или из него реализуется поэлементно. При этом после передачи очередного элемента (процедуры Read или Write) автоматически устанавливается доступ к следующему элементу в файле (заметим, что после процедур Rewrite, Reset доступ устанавливается к первому в файле элементу – "элементу с номером 0"). Поэтому для последовательной записи элементов в файл **("создания файла последовательным методом доступа"**) достаточно после процедуры Rewrite предусмотреть цикл выполнения процедуры Write. А для последовательной обработки файла ("**обработки файла последовательным методом доступа"**) после процедуры Reset в цикле должна выполняться процедура Read. Для организации цикла последовательной обработки файла удобно использовать стандартную логическую функцию **Eof.** Например, по операторам:

…

Reset (TB);

While not Eof (TB) do begin

Read (TB, ab);

…

end;

из файла, связанного с файловой переменной ТВ, будут последовательно (поочередно) читаться элементы, начиная с самого первого, пока не достигнут "конец файла" (при установке доступа к признаку окончания файла, который автоматически записывается после последнего элемента при создании файла, Eof = true, а до этого – Eof = false).

После создания или обработки файла следует предусматривать процедуру "закрытия" его - **Close**, единственным параметром которой является соответствующая файловая переменная (например, Close (TB).

**Примеры программирования**

Для задачи создания и обработки файла "Телефонный справочник" (элементы которого представляют собой записи, состоящие из трех строк – фамилии и инициалов, адреса и номера телефона абонента) приведены примеры алгоритма и программы (рис.10.1, рис.10.2) создания файла, примеры алгоритма и программы (рис.10.3, рис.10.4) обработки файла.

В этих программах предусмотрено:

- чтение дискового имени файла с клавиатуры в качестве значения строковой переменной nf;

- последовательный метод доступа к элементам файла (при его создании и обработке соответственно).

нет

нет

За-пись?

Подготовка создания файла

Ввод имени файла

Н

Запись элемента в файл

Ввод значения элемента

Вве-дено?

Вывод сооб-щения

Закрытие файла

К

да

да

Рис. 10. 1. Схема алгоритма создания файла последовательным методом доступа

|  |
| --- |
| {Программа создания файла последовательным методом доступа}  Program PCreateF;  type tab=record  fio: string [25];  adr: string[40];  nt: string[6]; end;  Var  Tb: file of tab; nf: string; ab: tab; Begin{Получение имени файла} Writeln ('Имя файла ?'); Readln (nf);  {Подготовка создания файла}  Assign (TB, nf);  Rewrite (TB);  ab.fio :=' '; {Цикл чтения сведений об абоненте и записи в файл}While (ab.fio>'') do Begin Write ('Ф.И.О.? '); Readln (ab.fio);  If ab.fio>'' then begin  Write ('Адрес? '); Readln (ab.adr);  Write ('Номер телефона? '); Readln (ab.nt);  end;  If ab.fio > '' then {Сведения получены}  Write (TB, ab) {Запись элемента в файл }  else Writeln ('Запись элементов завершена');  End;  Close (TB);  End. |

Рис. 10. 2. Программа создания файла "Телефонный справочник"

последовательным методом доступа

нет

нет

да

да

n=0

Фамилия совпадает?

Подготовка обработки файла

n=0

Ввод имени файла

Н

Вывод сведе-ний

Чтение элемента файла

Элемент файла?

Вывод сооб-щения

К

да

Ввод фамилии

n=n+1

Рис. 10.3. Схема алгоритма получения из файла списка абонентов-однофамильцев (n – количество абонентов)

|  |
| --- |
| { Программа получения из файла списка сведений об абонентах с указанной фамилией}  Program GetList;  type  tab=record  fio: string [25];  adr: string[40];  nt: string[6];  end;  Var  Tb: file of tab;  nf, name: string; n: integer;  ab: tab; Begin Writeln (Имя файла? '); Readln (nf);  Writeln (Фамилия абонента ? '); Readln (name);  {Подготовка получения списка}  name := name + ' ';  n := 0;  Assign (TB, nf); Reset (TB);  {Цикл чтения элементов файла и вывода на экран списка }  While not eof (TB) do  Begin  Read (TB,ab);  If (Pos (name, ab.fio) =1) then {Фамилии совпадают}  begin  n := n+1;  {Вывод сведений об абоненте}  Writeln (ab.fio:20, ab.adr:40, ab.nt:8);  end;  End;  If n = 0 then Writeln (' Нет абонентов с фамилией ', name );  Close (TB);  End. |

Рис. 10. 4. Программа получения из файла "Телефонный справочник" списка абонентов-однофамильцев

Программа обработки файла обеспечивает получение списка абонентов с указанной пользователем фамилией (абонентов-однофамильцев). При этом предполагается, что инициалы в элементе файла отделяются от фамилии, как минимум, одним пробелом (например, ab.fio = 'Сидоров\_В\_П'). Поэтому для выбора абонента в список к введенной пользователем фамилии добавляется пробел (знак '\_') и используется функция Pos для определения, содержит ли первый элемент записи со сведениями об абоненте нужную фамилию (в этом случае значение функции Pos равно единице).

Программу вывода на экран всех элементов файла можно получить из второй программы, удаляя в ней условные операторы, размещая вызов процедур Write и Writeln после вызова процедуры чтения Read элемента из файла.

**Контрольные вопросы**

1) Что собой представляют "типизированные файлы данных"?

2) Что собой представляют используемые при программировании на языке Турбо-Паскаль "записи"? Как они и их типы описываются и используются в программе?

3) Зачем нужны "файловые переменные"? Как они описываются и используются?

4) Что собой представляют и с какой целью используются подготовительные процедуры создания и обработки файлов?

5) Что понимают под "последовательным доступом к элементам файла"?

6) Каким образом предусматриваются и реализуются последовательная запись элементов в файл и последовательная обработка его?

**Лабораторная работа № 11.**

**Разработка программ создания и обработки типизированных файлов прямым методом доступа**

**Цель работы:**  получение практических навыков программирования процедур создания и обработки типизированных файлов на магнитных носителях информации с прямым методом доступа к их элементам.

**Задание по лабораторной работе**

Разработать программы создания и обработки файла на магнитном диске, состоящего из 10÷20 элементов заданной структуры (структура элемента файла, а также результат обработки его определяются номером варианта задания из табл.11.1 или преподавателем). При этом значение элемента файла должно определять его номер.

**Варианты задания**

Таблица 11.1

| № | Структура элемента файла | Результат обработки файла |
| --- | --- | --- |
| 1 | Регистрационный номер абитуриента факультета (строка из 5-ти знаков формата ’5хххх’, в котором 5 - номер факультета, а последние четыре знака – цифры);  Фамилия и инициалы абитуриента (строка из 25 знаков);  Сумма баллов егэ (целое число). | Сведения об абитуриенте с указанным регистрационным номером. |
| 2 | Табельный номер работника предприятия (строка из 5-ти знаков формата ’Тхххх’, в котором последние четыре знака – цифры);  Фамилия и инициалы работника (строка из 25 знаков);  Стаж работы на предприятии (целое число). | Сведения о работнике с указанным табельным номером. |
| 3 | Номер избирательного участка округа (строка из 5-ти цифр формата ’39ххх’, в котором 39 – номер округа);  Число избирателей, зарегистрированных на участке (целое число);  Число проголосовавших избирателей (целое число). | Сведения о численности избирателей и проголосовавших на участке с указанным номером. |
| 4 | Инвентарный номер единицы имущества организации (строка из 5-ти знаков в формате ’Ихххх’, в котором последние четыре знака – цифры;  Наименование единицы имущества (строка из 30 знаков);  Балансовая стоимость единицы имущества (вещественное число). | Сведения о единице имущества с указанным инвентарным номером. |
| 5 | Учетный номер судна у судовладельца (строка из 3-х символов в формате ’Схх’, в котором последние два знака – цифры);  Название судна (строка из 20 знаков);  Фамилия и инициалы капитана (строка из 25 знаков). | Сведения о судне с указанным учетным номером. |
| 6 | Лицевой счет владельца квартиры (строка из 6-ти знаков в формате ’лсхххх’, в котором последние четыре знака – цифры);  Фамилия и инициалы владельца квартиры (строка из 25 знаков);  Адрес квартиры (строка из 40 знаков). | Сведения об адресе и владельце квартиры по указанному лицевому счету. |
| 7 | Номер автобусного маршрута (строка из 4-х знаков в формате ’Аххх’, в котором последние три знака – цифры);  Названия конечных остановок (две строки по 20 знаков). | Сведения об автобусном маршруте по указанному номеру. |
| 8 | Номер личного дела сотрудника организации (строка из 6-ти знаков в формате ’лдхххх’, в котором последние четыре знака – цифры);  Фамилия и инициалы сотрудника (строка из 25 знаков);  Должность сотрудника (строка из 15-ти знаков). | Сведения о сотруднике по указанному номеру личного дела. |
| 9 | Номер поезда (строка из четырех знаков в формате ’хххч’, в котором первые три знака – цифры);  Пункт отправления (строка из 15 знаков);  Пункт назначения (строка из 15 знаков). | Сведения о маршруте поезда с указанным номером. |
| 10 | Номер авиарейса (строка из 6-ти знаков в формате ’suхххх’, в котором последние четыре знака – цифры);  Город вылета (строка из 15 знаков);  Город прилета (строка из 15 знаков);  Время вылета (строка в формате ’чч.мм’. | Сведения об авиарейсе по указанному номеру. |
| 11 | Артикул товара (строка из 5-ти знаков в формате ‘Ахххх‘, в котором последние четыре знака – цифры);  Наименование товара (строка из 20-ти знаков);  Стоимость единицы товара (вещественное число). | Сведения о товаре по указанному артикулу. |
| 12 | Учетный номер отделения банка (строка из 6-ти знаков в формате ‘39хххх‘, в котором 39 - номер региона, а последние четыре знака – цифры);  Адрес отделения банка (строка из 50-ти знаков);  Номер телефона отделения (строка из 6-ти цифр). | Сведения об отделении банка по указанному учетному номеру. |
| 13 | Учетный номер магазина (строка из четырех знаков в формате ‘унхх‘, в котором два последних знака – цифры);  Адрес магазина (строка из 40 знаков);  Номер телефона магазина (строка из 6-ти цифр). | Сведения о магазине по указанному учетному номеру. |
| 14 | Учетный номер экспоната музея (строка из 6-ти знаков в формате ‘эмхххх‘, в котором последние четыре знака – цифры);  Наименование экспоната (строка из 30 знаков);  Номер зала (или запасника) музея (целое число). | Сведения об экспонате музея по указанному учетному номеру. |
| 15 | Регистрационный номер автомобиля (строка из 7-ми знаков в формате ‘кххххкк‘, в котором четыре знака, начиная со второго – цифры);  Марка автомобиля (строка из 16 знаков);  Год выпуска автомобиля (целое число). | Сведения об автомобиле по указанному регистрационному номеру. |
| 16 | Код базовой учебной дисциплины (строка из трех знаков в формате ‘Бхх‘, в котором последние два знака – цифры);  Трудоемкость дисциплины в зачетных единицах (целое число);  Форма аттестации по дисциплине (строка ‘зачет‘ или строка ‘экзамен‘). | Сведения об учебной дисциплине по указанному коду. |

**Методические указания по выполнению лабораторной работы**

1. Задание предусматривает программирование создания и обработки такого типизированного файла, в котором значение элемента определяет его номер, т.е. между значением элемента и его номером в файле должно быть установлено взаимооднозначное соответствие. Структура элемента файла, определенная в задании, представляет собой запись, в которой первым элементом является строка, представляющая собой тот или иной "номер" (регистрационный номер, табельный номер, учетный номер и т.п.). Поэтому, прежде всего, следует определить арифметическую зависимость между этим "номером" (его частью, преобразованной в целое число) и номером элемента файла, его содержащим (см. пример программирования).

Для преобразования цифровой строки в целое число, его изображаемое, может использоваться стандартная процедура **Val.**

2. Для того, чтобы элемент файла размещался в нем в качестве элемента с "нужным" номером, следует предусматривать "прямой доступ к элементу файла" при записи его в файл, т.е. перед выполнением процедуры **Write** устанавливать процедурой **Seek** доступ к элементу файла с указанным номером (этот номер должен быть аргументом процедуры Seek). Для чтения из файла элемента с определенным номером до процедуры **Read**  также следует предусматривать выполнение процедуры **Seek**.

3. Создание файла, к элементам которого возможен целенаправленный прямой доступ (например, для получения сведений по запросу, который может сопоставляться с номером элемента файла, их содержащего) следует выполнять в два этапа. Сначала файл должен создаваться из одинаковых элементов нужного типа, но имеющих значения, явно отличающихся от реально возможных значений (т.н. "фиктивных" элементов). А затем можно записывать в файл "реальные" значения в качестве элементов с соответствующими этим значениям номерами (заменять "фиктивные" элементы "реальными"). Это нужно для того, чтобы при обработке файла после чтения элемента из него можно было определить, содержит ли элемент искомые сведения или нет (является "фиктивным").

**Теоретическая (справочная) часть**

**Прямой доступ к элементу типизированного файла** на магнитном носителе информации устанавливается при выполнении стандартной подпрограммы-процедуры **Seek**, аргументами которой являются файловая переменная и номер доступного для передачи элемента файла. Например, по оператору

Seek (FV, n);

в котором FV - файловая переменная, связанная с типизированным файлом, n – целочисленная переменная (типа longint), будет установлен доступ к элементу номером которого является значение переменной n. Поэтому при выполнении процедуры Write будет осуществляться передача в файл, а при выполнении процедуры Read - из файла, этого элемента.

Для **преобразования цифровой строки, изображающей целое число, в** **число** в Паскаль-программе может использоваться стандартная подпрограмма-процедура **Val (s, n , c**), параметрами которой являются: s – исходная строка, n - целочисленная переменная, значением которой должно стать число (результат преобразования строки s), с – признак успешности преобразования (если строка правильно изображает число, с = 0; если в строке есть недопустимый для изображения числа знак, значением **с** становится номер этого знака в строке). Например, при s = ‘2015‘ результатами процедуры Val будут n = 2015, c = 0, а при s = ‘2О15‘ (второй знак не цифра ‘0‘, а буква ‘О‘) - n = 0, c = 2.

**Примеры программирования**

В качестве примеров приведем:

- программу создания файла "Телефонный справочник" из "фиктивных записей" (рис. 11.1);

- алгоритм и программу записи элементов в файл "Телефонный справочник" прямым методом доступа (рис. 11.2 и рис.11.3);

- алгоритм и программу получения из файла "Телефонный справочник" сведений об абоненте по номеру телефона с прямым доступом к элементу файла (рис. 11.4 и рис. 11.5).

Предполагается, что файл "Телефонный справочник" содержит сведения об абонентах (их фамилии с инициалами и адреса) с номерами телефонов из диапазона ‘930000‘ - ‘939999‘. Поэтому между номером телефона (его числовым эквивалентом) и номером элемента файла, содержащим сведения об абоненте, установлена простая арифметическая зависимость:

nэл = nт – 930000

(здесь nэл – номер элемента файла "Телефонный справочник", nт – номер телефона в виде целого числа).

Первая программа обеспечивает создание файла из 10000 одинаковых "фиктивных" элементов (вместо фамилии и инициалов, а также адреса абонента в них определены знаки ‘**\***‘, а вместо номера телефона – строка ‘000000‘).

|  |
| --- |
| {Программа создания файла "Телефонный справочник" из фиктивных элементов}  Program PInputF;  Type tab=record  fio: string [25];  adr: string[40];  nt: string[6]; end;  Var TB: file of tab; nf: string; ab: tab; n: longint; Begin {Определение значения "фиктивного" элемента}  ab.fio := '\*'; ab.adr := '\*'; ab.nt := '000000';  {Получение имени файла}  Writeln (Имя файла? '); Readln (nf);  {Подготовка к созданию файла}  Assign (TB, nf); Rewrite (TB);  {Запись в файл "фиктивных" элементов}  For n := 0 to 9999 do Write (TB, ab);  {Закрытие файла}  Close (TB);  Writeln ('Файл создан');  End. |

Рис. 11.1. Программа создания файла "Телефонный справочник" из фиктивных элементов

нет

нет

да

да

Номер получен

Номер прочитан???

Подготовка записи элементов

Ввод имени файла

Н

Ввод сведений об абоненте

Ввод номера телефона

Запись нужна?

Запись элемента в файл

К

да

Определе-ние номера элемента

Установка прямого доступа

Вывод сообщения

Вывод сообщения

нет

Рис. 11.2. Схема алгоритма записи элементов в файл прямым методом доступа

|  |
| --- |
| {Программа записи элементов в файл "Телефонный справочник" прямым методом доступа}  Program PWriteF;  {Подпрограмма определения номера элемента файла по номеру телефона}  Procedure PDefNE ( nt: string; Var n: longint);  Var c: integer;  Begin  {Преобразование строкового номера в число}  Val (nt, n, c);  If (c=0) then {преобразование успешно}  If (930000 <= n) and (n <= 939999) then  {допустимый номер телефона}  {Определение номера элемента файла}  n:= n – 930000  else n:=-1;  {Признак недопустимости номера телефона};  end;  Type tab=record  fio: string [25];  adr: string[40];  nt: string[6]; end;  Var Tb: file of tab; ab: tab; n: longint; nf, nt: string;  Begin  {Получение имени файла}  Writeln (Имя файла? '); Readln (nf);  {Подготовка к заполнению файла записями}  Assign (TB, nf); Reset (TB);  nt := '000000';  {Цикл чтения сведений об абонентах и записи в файл}  While ( nt > '') do  Begin  {Чтение номера телефона}  Write ('Номер телефона? '); Readln ( nt);  If ( nt > '') then {Номер телефона прочитан}  begin  {Определение номера элемента файла}  PDefNE ( nt, n);  If n > 0 then {Номер элемента получен}  begin  ab.nt := nt;  {Чтение сведений об абоненте}  Write (' Ф.И.О.? '); Readln (ab.fio);  Write (' Адрес? '); Readln (ab.adr);  {Установка прямого доступа к элементу файла}  Seek (TB, n);  {Запись элемента в файл}  Write (TB,ab);  end  else Writeln ('Недопустимый номер телефона');  end  else Writeln (' Запись в файл завершена');  end;  Close (TB);  End. |

Рис. 11.3. Программа записи элементов в файл прямым методом доступа

да

да

нет

Номер получен

Подготовка к обработке файла

Ввод имени файла

Н

Вывод сведений

Номер прочитан?

Чтение элемента файла

К

да

Опреде-ление номера элемента

Установка прямого доступа

Вывод сооб-щения

Вывод сооб-щения

нет

нет

Ввод номера телефона

Есть данные?

Ввод сообще-ния

Рис. 11.4. Схема алгоритма получения сведений об абоненте по номеру телефона прямым доступом к элементу файла

|  |
| --- |
| {Программа получения из файла "Телефонный справочник" сведений об абоненте по номеру телефона}  Program PGetab;  Procedure PDefNE ( nt: string; Var n: longint);  …  end;  type tab=record  fio: string [25];  adr: string[40];  nt: string[6]; end;  Var  Tb: file of tab; ab: tab; n: longint; nf: string; nt:string; Begin Writeln (Имя файла? '); Readln (nf); {Получение имени файла}  Write ('Номер телефона ? '); Readln (nt); {Чтение номера телефона}  If nt > '' then {Номер телефона прочитан}  Begin {Определение номера элемента файла}  PDefNE (nt, n);  If n>=0 then {Номер элемента получен}  begin {Подготовка к обработке файла}  Assign (TB, nf); Reset (TB);  {Установка прямого доступа к элементу файла }  Seek (TB, n);  {Чтение элемента из файла} Read (TB, ab);  If ab.tb = '000000' then  Writeln (' Сведений об абоненте нет' );  else begin  Write (' Ф.И.О. – ', ab.fio:25);  Write (' Адрес - ', ab.adr:40); end;  else Writeln (' Недопустимый номер телефона' );  Close (TB);  end  else Writeln (' Запрос на поиск отсутствует' );  End. |

Рис. 11.5. Схема алгоритма получения сведений об абоненте по номеру телефона прямым доступом к элементу файла

Во второй и третьей программах предусматривается определение номера элемента файла, который должен содержать сведения об абоненте с указанным пользователем номером телефона. Эта процедура предусматривается в обеих программах подпрограммой PDefNE (рис. 11.3). Если номер телефона указан пользователем правильно и он допустим для данного "Телефонного справочника", то устанавливается прямой доступ к элементу файла с определенным по номеру телефона номером для передачи данного элемента (записи элемента в файл или чтения элемента из файла). В программе получения сведений об абоненте (рис. 11.5) предусматривается проверка – содержит ли прочитанный элемент искомые сведения или является "фиктивным". Во втором случае вместо сведений об абоненте выводится сообщение об их отсутствии.

В приведенных программах в качестве типа номера элемента файла используется стандартный целочисленный тип с именем **longint** (множество целых чисел в диапазоне от – 2147483648 до 2147483647).

**Контрольные вопросы и задания**

1) Что понимают под "прямым методом доступа к элементам файла"? Каким образом он предусматривается в Паскаль-программах?

2) Для каких файлов прямой метод доступа к элементам возможен и целесообразен?

3) Почему по Вашей программе получение требуемых сведений из файла по запросу пользователя возможно с использованием прямого доступа к элементу, их содержащему?

4) Каким образом предусматривается преобразование цифровой строки в число, его изображаемое? Зачем в Ваших программах такое преобразование предусмотрено?

5) Почему создание файла по заданию данной работы предусматривается в два этапа? Что собой они представляют?

6) Прокомментируйте разработанные Вами программы.

**Список учебной литературы**

1. Немнюгин С.А. Turbo Pascal: учеб. – СПб.: Питер, 2000, - 491 с.
2. Павловская Т.А. Паскаль. Программирование на языке высокого уровня: Учебник для вузов – СПб.: Питер, 2007. – 393 с.: ил.
3. Аляев Ю.А., Гладков В.П., Козлов О.А. Практикум по алгоритмизации и программированию на языке Паскаль: учебное пособие – М.: Финансы и статистика, 2004. – 528 с.
4. Аляев Ю.А., Козлов О.А. Алгоритмизация и языки программирования Pascal, C ++, Visual Basic: Учеб.-справ. пособие для курс. воен. учеб. завед. и студ. техн. вузов – М.: Финансы и статистика, 2002. – 319 с.
5. Сухарев М. Turbo Pascal 7.0. Теория и практика программирования. – 2-е изд., перераб. и доп. – СПб: Наука и техника, 2004. 640 с.
6. Фаронов В.В. Турбо-Паскаль: Учеб. пособие. – М: Питер, 2007. – 367 с.
7. Тарануха Н.А., Гринкруг Л.С., Бурменский А.Д., Ильина С.В. Обучение программированию: Язык Pascal. – М.: СОЛОН-ПРЕСС, 2009. – 384 с.
8. Учебно – методическое пособие по лабораторным работам для студентов бакалавриата по направлениям подготовки "Информатика и вычислительная техника" "Прикладная информатика" (часть первая) – Калининград, ФГБОУ ВПО КГТУ, 2015. – с.

Интернет-ресурсы (ссылки на учебники и учебно-методические пособия):

http://progbook.net/pascal

http://sprinter.ru/10038

Содержание

Введение 3

Лабораторная работа № 8. Разработка программы обработки символьных данных 4

Лабораторная работа № 9. Разработка программы диалоговой  
 задачи 16

Лабораторная работа № 10. Разработка программ создания  
 и обработки типизированных файлов последовательным методом  
 доступа 26

Лабораторная работа № 11. Разработка программ создания   
и обработки типизированных файлов прямым методом доступа 40

Список учебной литературы 55