ПАСКАЛЬ:
ПОДПРОГРАММЫ И СЛОЖНЫЕ ТИПЫ ДАННЫХ

Методические указания к выполнению лабораторных работ
по курсам «Информатика», «Информатика и программирование»
для студентов всех специальностей
очной формы обучения

Воронеж 2010

УДК 004.438-004.424
ББК 32.973-018.1п

Составители О.Е. Ефимова, А.В. Распопов, Д.В. Меркулов

Паскаль: подпрограммы и сложные типы данных [Текст]: метод. указания к выполнению лаб. работ по курсам «Информатика»; «Информатика и программирование» для студ. всех спец. / Воронеж. гос. арх.-строит. ун-т; сост.: О.Е. Ефимова, А.В. Распопов, Д.В. Меркулов. – Воронеж, 2010. – 38 с.

Изложен краткий теоретический материал по подпрограммам и сложным типам данных языка Паскаль. Все вопросы рассматриваются на большом количестве примеров программ. Представлены контрольные задания для самостоятельного выполнения по каждой теме. В приложениях представлен необходимый справочный материал по версии языка Паскаль, соответствующей интегрированной среде разработки Borland Pascal 7.0.

Предназначены для студентов очного обучения всех специальностей.

Библиогр.: 7 назв.

УДК 004.438-004.424
ББК 32.973-018.1п

Печатается по решению редакционно-издательского совета
Воронежского государственного архитектурно-строительного университета

Рецензент – Г.Т. Венгерова, к. ф.-м. н., доц. кафедры математического моделирования и вычислительной техники ВГАСУ.

Введение

В эпоху бурного развития информационных и наукоемких технологий изучение информатики студентами вузов является очень важной задачей, решение которой невозможно без приобретения практических навыков во время лабораторных занятий.

Настоящие методические указания ориентированы на развитие у студентов навыков практического выполнения широкого круга задач программирования. Дается целый комплекс контрольных заданий, после выполнения которых, каждый студент должен предоставить отчет о проделанной работе, включающий: формулировку задания, исходный текст (листинг) отлаженной программы и результат решения (с анализом).

Многие задачи имеют не единственный способ решения. При этом критериями качества программы служат следующие показатели:

– объем памяти, занимаемый программой (с учетом памяти, отводимой под переменные);

– трудоемкость вычислений, т.е. эффективность алгоритма;

– оригинальность решения;

– лаконичность и наглядность программы, включая наличие и качество комментариев;

– соответствие листинга программы признанным стандартам стилевого оформления исходного кода.

При написании программы также приветствуется обобщение постановки задачи, т.е. замена частного случая общим.

Лабораторная работа № 1

Подпрограммы: процедуры и функции

1.1. Цели работы

Научиться оформлять и пользоваться подпрограммами языка Паскаль: процедурами и функциями.

1.2. Краткие теоретические сведения

Подпрограммы представляют собой инструмент, с помощью которого любая программа может быть разбита на ряд независимых друг от друга частей. Такое разбиение необходимо по двум причинам:

– программы, состоящие из набора отдельных подпрограмм, легче пишутся, воспринимаются, тестируются и отлаживаются;

– это средство экономии памяти, так как каждая подпрограмма существует в программе в единственном экземпляре, в то время как обращаться к ней можно многократно из разных точек программы.

Описание подпрограммы состоит из заголовка и тела подпрограммы.

Заголовки процедуры и функции имеет следующий вид:

```
Procedure <Имя_процедуры> [(<список_формальных_параметров>)];

Function <Имя_функции> [(<список_формальных_параметров>)] :
        <тип_возвращаемого_функцией_результата>;
```

Список формальных параметров необязателен и может отсутствовать. Если же он есть, то в нем должны быть перечислены имена формальных параметров и их тип. Пример функции:

```
{ Определение максимума двух целых чисел }
Function Max (A, B: Integer): Integer;
begin
  if A > B then Max := A
    else Max := B;
end;
```

Отличие функции от процедуры заключается в том, что результатом исполнения операторов, образующих тело функции, всегда является некоторое единственное значение простого, строкового типа или указателя (чаще всего – имя функции).

Любой из формальных параметров процедуры может быть либо параметром-значением, либо параметром-переменной. Если параметры определяются как параметры-переменные, перед ними необходимо ставить зарезервированное слово **var**, например:

```
{ Вычисление суммы и разности квадратов двух чисел x и y }
Procedure SumSub (X, Y: Real; var S1, S2: Real);
begin
  S1 := Sqr(X) + Sqr(Y);
  S2 := Sqr(X) – Sqr(Y);
end;
```

Параметры-переменные используются для передачи результатов работы подпрограммы вызывающему блоку. Группа параметров, при описании которых используется ключевое слово **var**, но отсутствует тип, называется нетипизированными параметрами-переменными.

Для передачи подпрограмме в качестве параметра массива или строки необходимо сначала объявить их тип, например:

```
Type
  TMas = array [1..20] of Real;
  TStr = String [30];
...
Function St (X: TMas, Y: TStr): TStr;
```

Вызов подпрограммы осуществляется простым упоминанием имени процедуры в операторе вызова процедуры или имени функции в выражении:

```
SumSub (12.5, 10.3, A, B);
C := (A + B) / Max (3, 5);
```

1.3. Примеры решения типовых задач

1.3.1. Построить таблицу тригонометрических функций.

```
Program Example_1_3_1;
Uses
  WinCrt;
Const
  DegreesToRadian = Pi / 180;
Var
  Degrees: Byte;
  Radian: Real;
```

```
Function Tg (X: Real): Real;
begin
  Tg := Sin(X) / Cos(X);
end;

Function Ctg (X: Real): Real;
begin
  Ctg := Cos(X) / Sin(X);
end;

Begin
  WriteLn ('Degrees  Sin      Cos     Tg      Ctg     Degrees');
  for Degrees := 0 to 45 do
  begin
    Radian := DegreesToRadian * Degrees;
    Write (Degrees:5, Sin(Radian):9:4, Cos(Radian):9:4, Tg(Radian):9:4);
    if Degrees = 0 then
      WriteLn(90:15)
    else
      WriteLn (Ctg(Radian):9:4, (90-Degrees):6);
  end;
End.
```

1.3.2. Написать программу, вычисляющую суммарное количество секунд, соответствующее заданному числу часов, минут и секунд и, наоборот, определяющую, сколько часов, минут и секунд содержится в заданном числе секунд.

```
Program Example_1_3_2;
Uses
  WinCrt;
Var
  Choice: Integer;

Procedure Menu;
begin
  WriteLn ('1 - Преобразовать секунды в часы, минуты и секунды');
  WriteLn ('2 - Преобразовать часы, минуты и секунды в секунды');
  WriteLn ('3 - Завершить работу');
end;

Procedure SecondsToTime;
var
```



```
  TotalSeconds, Hours, Minutes, Seconds, Temp: LongInt;
begin
  WriteLn ('Введите суммарное количество секунд');
  ReadLn (TotalSeconds);
  Temp := TotalSeconds div 60;
  Seconds := TotalSeconds mod 60;
  Hours := Temp div 60;
  Minutes := Temp mod 60;
  WriteLn (TotalSeconds, ' секунд – это ', Hours,' часов ', Minutes, ' минут
',
    Seconds,' секунд');
  ReadLn;
end;

Procedure TimeToSeconds;
var
  TotalSeconds, Hours, Minutes, Seconds, Temp: LongInt;
begin
  WriteLn ('Введите часы, минуты, секунды');
  ReadLn (Hours, Minutes, Seconds);
  TotalSeconds := Hours * 3600 + Minutes * 60 + Seconds;
  WriteLn (Hours, ' часов ', Minutes, ' минут ', Seconds, ' секунд – это ',
    TotalSeconds, ' секунд');
  ReadLn;
end;

Begin
  Choice := 0;
  while Choice <> 3 do
  begin
    Menu;
    ReadLn (Choice);
    case Choice of
      1: SecondsToTime;
      2: TimeToSeconds;
    end;
  end;
End.
```

1.4. Контрольные задания

1.4.1. Написать функцию, подсчитывающую количество цифр числа. Используя ее, определить, в каком из десяти введенных с клавиатуры чисел больше цифр.

1.4.2. Составить программу, которая будет находить $\mathbf{a^n}$, то есть **n**-ю степень числа **a**, где **a** и **n** – это целые числа и **n > 0**, вводимые с клавиатуры. Для возведения числа в степень использовать процедуру.

1.4.3. Составить программу, подсчитывающую число сочетаний без повторений из N элементов по K элементов, используя формулу

$$C_n^k = \frac{n!}{k!(n-k)!}.$$

1.4.4. Вычислить значение выражения

$$D = f(a, b, c) + f(a / (c^2+1), a + b, c * a) - f(a - 2, b + c, 3 * c),$$

используя функцию $f(x,y,z) = \begin{cases} e^x + yz, \text{ если } z \le 0 \\ x - y/z, \text{ если } z > 0 \end{cases}$.

1.4.5. Дана матрица **A (m, n)**. Вычислить значение выражения

$$T = \sqrt{\left|\frac{Max_L}{Max_K^2 + 3{,}7}\right|} + Min_L \cdot Min_K,$$

где Max_L – максимальное значение **L**-й строки матрицы **A**, Max_K – максимальное значение **K**-й строки матрицы **A**;

Min_L – минимальное значение **L**-го столбца матрицы **A**, Min_K – минимальное значение **K**-го столбца матрицы **A**.

Нахождение максимального значения строки и минимального значения столбца оформить в виде процедур.

1.4.6. Даны натуральное число **n** и вещественная матрица размером **n x 9**. Найдите среднее арифметическое элементов каждого из столбцов, имеющих четные номера. Для вычисления среднего арифметического использовать подпрограмму-процедуру.

1.4.7. Дана вещественная матрица порядка **m x n**. Постройте последовательность $\mathbf{b_1,\ldots,b_m}$ из нулей и единиц, в которой $\mathbf{b_i = 1}$ тогда и только тогда, когда в **i**-й строке матрицы есть хотя бы один отрицательный элемент. Использовать логическую подпрограмму-функцию.

1.4.8. Определите, является ли заданная целая квадратная матрица магическим квадратом, то есть такой, в которой суммы элементов во всех строках и столбцах одинаковы.

1.4.9. Даны три квадратные символьные матрицы. Матрицу, где есть хотя бы одна гласная – транспонировать, то есть отобразить симметрично относительно главной диагонали.

1.4.10. Разработать программу поиска в двумерном массиве максимального элемента массива и удаления строки и столбца, содержащего этот элемент. Размеры массива задаются с клавиатуры.

1.4.11. Вычислить количество положительных элементов квадратной матрицы **n x n**, расположенных по ее периметру и на диагоналях.

1.4.12. Дана целочисленная квадратная матрица **A (n x n)**. Получите, используя подпрограммы, $\mathbf{b_1,\ldots,b_n}$, где элемент $\mathbf{b_i}$ равен:

a) $\sum_{j=1}^{n}(-1)^{i+j}a_{ij}$;

b) $\prod_{j=1}^{n}\left|a_{ji}\right|$;

c) $\max\limits_{1\le j\le n} a_{ij} \cdot \min\limits_{1\le j\le n} a_{ji}$.

1.4.13. Преобразовать матрицу **A (m x n)** так, чтобы строки с нечетными номерами были упорядочены по убыванию, а с четными – по возрастанию.

1.4.14. Дан одномерный массив **Y(n)**. Вставьте заданное с клавиатуры число после всех элементов массива, кратных 3.

1.4.15. В матрице целых чисел **A (m x n)** обнулить столбцы, в которых нет простых и совершенных чисел. Число называется *простым*, если оно делится без остатка только на единицу и на само себя. *Совершенное* число представляет собой сумму всех своих делителей, меньших его самого.

1.4.16. Для заданных массива **X (n)** и матриц **A (m x n)**, **B (n x m)** написать, используя подпрограммы, программу для вычисления:

a) $\mathbf{C_1 = A + B^T}$, где $c_{ij} = a_{ij} + b_{ji}$;

b) $\mathbf{C_2 = A * X}$, где $c_i = \sum_{j=1}^{n} a_{ij} \cdot b_j$;

c) $\mathbf{C_3 = A * B}$, где $c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj}$.

1.4.17. Найти все седловые точки заданной квадратной целочисленной матрицы. *Седловой* называется точка, максимальная в своей строке и минимальная в своем столбце и наоборот.

Лабораторная работа № 2

Рекурсивные алгоритмы

2.1. Цели работы

Научиться составлять программы с использований рекурсий.

2.2. Краткие теоретические сведения

Рекурсия – это ссылка на объект при описании и определении самого объекта.

Подпрограмма называется рекурсивной, если она вызывает сама себя. Рекурсивной также будет процедура, вызывающая другую процедуру, которая, в свою очередь, обращается к первой процедуре. Возможны и более сложные конструкции. В первом случае рекурсия называется прямой, а во втором – косвенной.

Рекурсия в языке Паскаль возможна благодаря тому, что при вызове процедуры динамически создаются новые локальные переменные.

Максимальное число рекурсивных вызовов процедуры без возвратов, которое происходит во время выполнения программы, называется глубиной рекурсии. Глубина рекурсии должна быть конечной. Число рекурсивных вызовов в каждый конкретный момент времени, называется текущим уровнем рекурсии.

2.3. Примеры решения типовых задач

2.3.1. Посчитать выражение **a · (a + 1) ·...· (a + n-1)**. Значения **a** и **n** заранее заданы.

```
Program Example_2_3_1;
Uses
  WinCrt;
Var
  n, a, k: Integer;
  Sum: Integer;

Function Rec (a, n: Integer): Integer;
begin
  if n =1 then
    Sum := a
  else
    Sum := (a + n-1) * Rec(a, n-1);
  Rec := Sum;
end;

Begin
  Write ('Введите n: ');
  ReadLn (n);
  Write ('Введите a: ');
  ReadLn (a);
  k := 1;
  Write ('Результат: ', Rec(a, n));
End.
```

2.3.2. С помощью рекурсии вычислить выражение

Sum = 1 / (sin(1)) + 1 / (sin(1) + sin(2)) + ... + 1 / (sin(1) + ... + sin(n)).

```
Program Example_2_3_2;
Uses
  WinCrt;
Var
  n, i: Integer;
  Znam, Sum: Real;

Function Recur (k: Integer): Real;
var
  Lastznam: Real;
begin
  if k > 0 then
    begin
      Lastznam := Znam;
      Znam := Znam – sin(k);
      Recur := (1 / Lastznam) + Recur(k - 1);
    end;
end;{Recur}

Begin
  {$I-}
  repeat
    Write ('Введите натуральное n : ');
    ReadLn (n);
  until (IOResult = 0) and (n>0);
  {$I+}
  {начальное формирование знаменателя}
  Znam := 0;
  for i := n downto 1 do
    Znam := Znam + sin(i);
  Sum := Recur(n);
  Write ('Результат: ', Sum:10:8);
End.
```

2.4. Контрольные задания

2.4.1. Составьте программу перевода любого натурального числа из десятичной системы счисления в двоичную, используя рекурсивную процедуру.

2.4.2. Составить рекурсивную программу ввода с клавиатуры последовательности чисел (окончание ввода – число 0) и вывода ее на экран в обратном порядке.

2.4.3. Найти сумму первых **N** элементов арифметической (геометрической) прогрессии.

2.4.4. Составить программу вычисления значений функции Аккермана для неотрицательных чисел **n** и **m**, вводимых с клавиатуры.

$$A(n,m)=\begin{cases} m+1, & \text{если } n=0,\\ A(n-1,1), & \text{если } n\neq 0, m=0,\\ A(n-1,A(n,m-1)), & \text{если } n>0, m\geq 0. \end{cases}$$

2.4.5. Определить, является ли заданное натуральное число простым.

2.4.6. Функция **F(n)** определена для целых чисел следующим образом:

$$F(n)=\begin{cases} 1, & \text{если } n=1,\\ \sum_{i=2}^{n} F(n \text{ div } i), & \text{если } n\geq 2. \end{cases}$$

Вычислить значения этой функции для **n** = 5, 6, 7, …, 20.

2.4.7. Для заданного натурального числа $N \geq 1$ определить единственное натуральное число **a**, для которого выполняется неравенство $2^{a-1} \leq N < 2^a$.

Лабораторная работа № 3

Строки

3.1. Цели работы

Научиться работать со строковым типом данных языка Паскаль.

3.2. Краткие теоретические сведения

Тип **String** (строка) в Паскале широко используется для обработки текста. Он похож на одномерный массив символов **array** [0 .. N] **of Char**, однако, в отличие от массива, количество символов в строке-переменной не жестко задано, а может меняться от 0 до N, где N – максимальное количество символов в данной строке. Значение N определяется при объявлении типа **String** и может быть любой константой порядкового типа, но не больше 255. По умолчанию длина строки принимается максимально возможной (N = 255).

К любому символу в строке можно обратиться точно так же, как к элементу одномерного массива, например:

```
Var
  St : String [100];
Begin
  …
 if St [5] = 'A' then
   WriteLn ('Пятый символ строки – буква A');
  …
End.
```


Первый байт в строке имеет индекс 0 и содержит текущую длину строки. Первый значащий символ строки занимает второй байт и имеет индекс 1. Допускается формирование строк с использованием записи символов по десятичному коду (в виде комбинации # и кода символа) и управляющих символов (комбинации ^ и некоторых заглавных латинских букв), например:

```
Var
  S: String;
Begin
  …
  S := 'abc'#54#32#61^A^B^M'de';
  WriteLn (S);
  …
End.
```

Описание операций, а также стандартных процедур и функций для работы со строками, представлено в прил. 1.

3.3. Примеры решения типовых задач

3.3.1. Написать программу, которая выводит в противоположном порядке следования символов вводимую пользователем строку.

```
Program Example_3_3_1;
Uses
  WinCrt;
Var
  S: String;
  I: Integer;
Begin
  Write ('Введите строку текста: ');
  ReadLn (S);
  for i := Length(S) downto 1 do
    Write (S[i]);
End.
```

3.3.2. Дана строка, состоящая из нескольких слов, между словами стоит один пробел, в конце предложения – точка. Подсчитать число слов и вывести на экран только те из них, которые начинаются с буквы «а» (слов не более 30).

```
Program Example_3_3_2;
Uses
  WinCrt;
Const
  N = 30;
Type
  MyArray_Str = array [1..N] of String;
Var
  A: MyArray_Str;
  str: String [255];
  k: Byte;

Procedure Init (var b: MyArray_Str);
var
  i: Integer;
begin
  k :=1;
  {пока не встретится пробел, формируем очередное слово k, прибавляя}
  {по одной букве}
  for i := 1 to Length (str) - 1 do
    if str [i] <> ' ' then b [k] := b [k] + str [i]
      else
      {если это не последний символ, то увеличиваем счетчик слов и}
      {начинаем формировать соответствующий элемент массива }
        if i <> Length (str) - 1 then
          begin
            Inc (k);
            b [k] := ' ';
          end;
end;

Begin
  WriteLn ('Введите предложение (не более 30 слов) ');
  ReadLn (str);
  Init (A);
  WriteLn ('Всего слов: ', k);
  {просматриваем все слова, если первый символ очередного слова}
  {есть 'a', то выводим его}
  for i := 1 to k do
```



```
    if A [i][1] = 'a' then Write (A[i], ' ');
End.
```

3.4. Контрольные задания

3.4.1. Слово задано строкой символов. Получить новое слово, состоящее из символов исходного слова, записанных в обратном порядке.

3.4.2. Дано слово. Определить, является ли оно палиндромом. Палиндромы – это слова, которые читаются одинаково в обоих направлениях, то есть симметричны относительно своей середины (например, «потоп»).

3.4.3. Дана строка символов. Группа символов между пробелами считается словом. Определить самое длинное слово в строке и количество слов такой же длины.

3.4.4. Дана строка символов. Удалить из нее все пробелы.

3.4.5. С клавиатуры вводится строка символов и некоторое число. Вставить его после каждого пробела в исходный текст.

3.4.6. Вводится текст, состоящий из слов, отделенных друг от друга пробелом. Удалить из нее слово, имеющее наибольшую (наименьшую) длину.

3.4.7. Вводится произвольная строка символов. Удалить из строки среднюю букву, если длина строки нечетная, если четная – удалить две средние буквы.

3.4.8. Вводится произвольная строка символов. Заменить все вхождения подстроки **Str1** на подстроку **Str2** (подстроки вводятся с клавиатуры).

3.4.9. Дана последовательность слов, вводимых через запятую. Вывести на экран все слова в алфавитном порядке.

3.4.10. Вводится строка символов произвольной длины. Посчитать сумму чисел, входящих в заданную строку.

3.4.11. Вводится произвольная последовательность слов, разделенных пробелами (одним или несколькими). Удалить лишние пробелы. Вывести на экран те слова последовательности, которые отличны от последнего слова и удовлетворяют следующему свойству:

a) в слове нет повторяющихся букв;

b) буквы слова упорядочены по алфавиту.

Лабораторная работа № 4

Записи

4.1. Цели работы

Научиться использовать тип запись языка Паскаль.

4.2. Краткие теоретические сведения

Тип **Record** (запись) – это структура данных, представляющая собой совокупность ограниченного числа именованных, логически связанных, компонент, называемых полями записи. В отличие от массива, поля записи могут быть различного типа (любого, допустимого в языке Паскаль, за исключением файлового).

В языке Паскаль различают фиксированные (обычные) и вариантные записи.

Обычная фиксированная запись состоит из одного или нескольких полей, для каждого из которых при объявлении указывается имя (идентификатор) и тип. Например, отдельную строку телефонного справочника, содержащего фамилии и номера телефонов удобно представить в виде следующей записи:

```
Type
 TRec = Record
  FIO: String [20];
  Tel: String [6];
 end;
Var
 Rec: TRec;
```

Обращение ко всей записи в целом допускается только в операторах присваивания, где слева и справа от знака присваивания используются имена записей одинакового типа. Во всех остальных случаях оперируют отдельными полями записей, например:

```
Rec.FIO := 'Иванов А.А.';
Rec.Tel := '322223';
```

Обращение к компонентам записей можно упростить, если воспользоваться оператором присоединения **with**. Он позволяет заменить составные имена, характеризующие каждое поле, просто на имена полей, а имя записи определить в операторе присоединения:

```
with Rec do
begin
 FIO := 'Иванов А.А.';
 Tel := '322223';
end;
```


Вариантная запись включает две части: обычную фиксированную запись и вариантную, начинающуюся с зарезервированного слова **case**. Например, запись о некоторой геометрической фигуре (отрезок, треугольник, окружность):

```
var
 Figure: Record
  Type_of_figure: String [10];
  Color_of_figure: Byte;
  case Integer of
   1: (X1, Y1, X2, Y2: Integer);
   2: (A1, A2, B1, B2, C1, C2: Integer);
   3: (X, Y: Integer; Radius: Word);
 end;
```

4.3. Примеры решения типовых задач

4.3.1. Создать базу данных для библиотеки, в которой возможны два действия: добавлять книгу и показывать список книг. Для этого необходимо знать следующую информацию о книгах: имя автора, название книги, число страниц, цена, дата издания.

```
Program Example_4_3_1;
Uses
 WinCrt;
Const
 Max = 50; { максимальное количество книг }
Type
{ запись, отвечающая за дату}
 Tdate = record
          d, m, y : integer
         end;
{ запись, отвечающая за книгу }
Tbook = record
          author, name : string;
          page, cost : integer;
          date : Tdate
         end;
{ запись, отвечающая за библиотеку }
Tlib = record
        book : array [1 .. Max] of Tbook; { массив книг }
        num : byte { число книг в библиотеке }
       end;
```

```
Function Menu : char;
begin
 WriteLn (' БИБЛИОТЕКА.');
 WriteLn (' 1 - добавить книгу.');
 WriteLn (' 2 - показать книги.');
 WriteLn ('ESC - выход.');
 Menu := readkey
end;

Procedure Add (var new : Tlib);
begin
 if new.num = Max then
 begin
  WriteLn ('Библиотека переполнена.');
  ReadLn;
  Exit
 end;
 new.num := new.num + 1;
 with new.book[new.num] do
 begin
  Write ('Введите имя автора: ');
  ReadLn (author);
  Write ('Введите название: ');
  ReadLn (name);
  Write ('Введите число страниц: ');
  ReadLn (page);
  Write ('Введите цену: ');
  ReadLn (cost);
  Write ('Введите дату (ДД ММ ГГГГ): ');
  With date do
  begin
   ReadLn (d, m, y)
  end
 end
end;

Procedure Show (lib : Tlib);
var
 i : integer;
begin
 for i := 1 to lib.num do
 begin
```



```
  WriteLn ('=========================================');
  WriteLn ('Книга номер ', i);
  with lib.book[i] do
  begin
   WriteLn ('Автор ', author);
   WriteLn ('Название ', name);
   WriteLn ('Число страниц ', page);
   WriteLn ('Цена = ', cost);
   with date do
    WriteLn ('Дата ', d, '.', m,'.', y)
  end
 end;
 ReadLn
end;

var
 lib : Tlib;
 key : char;

Begin
 lib.num := 0;

 while key <> #27 do
 begin
  key := Menu;
  case key of
   '1': Add (lib);
   '2': Show (lib)
  end
 end
End.
```

4.3.2. Имеется список студенческой группы из 5 студентов, в котором записаны: фамилия студента, названия и результаты 3 зачетов и 4 экзаменов. Определить количество студентов сдавших сессию на «4» и «5», на «4» и «3» и не сдавших сессию.

```
Program Example_4_3_2;
Uses
 WinCrt;
Const
 ChisloStudentov = 5;
 ChisloZachetov = 3;
```

```
  ChisloEkzamenov = 4;
  NazvaniyaZachetov: array [1..ChisloZachetov] of String[11] = ('Физкультура',
   'Математика', 'Физика');
  NazvaniyaEkzamenov: array [1..ChisloEkzamenov] of String[22] =
('Информатика', 'Английский', 'Теоретическая механика',
'Материаловедение');
Type
  MassivFamiliiStudentov = array [1..ChisloStudentov] of String[20];
  MassivOcenokZachetov = array [1..ChisloStudentov, 1..ChisloZachetov] of
Byte;
  MassivOcenokEkzamenov = array [1..ChisloStudentov, 1..ChisloEkzamenov] of
                                                              Byte;
Var
  ChisloStudentov4_5, ChisloStudentov4_3, ChisloStudentov2, i, j: Integer;
  MFS: MassivFamiliiStudentov;
  MOZ: MassivOcenokZachetov;
  MOE: MassivOcenokEkzamenov;
  FlagSdachiZachetov, FlagSdachiEkzamenov, Flag3, Flag4, Flag5: Boolean;

Procedure VvodFamiliiStudentov (var A: MassivFamiliiStudentov);
var
  i: Integer;
begin
  WriteLn ('Введите фамилии студентов');
  for i := 1 to ChisloStudentov do
  begin
    Write (i:2, ' ');
    ReadLn (A[i]);
  end;
end;

Procedure VvodZachetov (var A: MassivOcenokZachetov);
var
  i, j: Integer;
begin
  WriteLn ('Введите результаты зачетов (0/1)');
  for i := 1 to ChisloZachetov do
  begin
    WriteLn ('Зачет: ', NazvaniyaZachetov[i]);
    for j :=1 to ChisloStudentov do
    begin
      Write (MFS[j], ' ');
      ReadLn (A[j, i]);
```



```
    end;
  end;
end;

Procedure VvodEkzamenov (var A: MassivOcenokEkzamenov);
var
  i, j: Integer;
begin
  WriteLn ('Введите результаты экзаменов (0..5)');
  for i := 1 to ChisloEkzamenov do
  begin
    WriteLn ('Экзамен: ', NazvaniyaEkzamenov[i]);
    for j :=1 to ChisloStudentov do
    begin
      Write (MFS[j], ' ');
      ReadLn (A[j, i]);
    end;
  end;
end;

Begin
  ChisloStudentov4_5 :=0;
  ChisloStudentov4_3 :=0;
  ChisloStudentov2 :=0;
  VvodFamiliiStudentov (MFS);
  VvodZachetov (MOZ);
  VvodEkzamenov (MOE);
  for i :=1 to ChisloStudentov do
  begin
    FlagSdachiZachetov := True;
    FlagSdachiEkzamenov := True;
    Flag3 := False;
    Flag4 := False;
    Flag5 := False;
    for j :=1 to ChisloZachetov do
      if MOZ[i, j] = 0 then
        begin
          ChisloStudentov2 := ChisloStudentov2 + 1;
          WriteLn (MFS[i]);
          FlagSdachiZachetov := False;
          Break;
        end;
    if FlagSdachiZachetov then
```

```
  for j:=1 to ChisloEkzamenov do
  begin
   if MOE[i, j] <= 2 then
    begin
     ChisloStudentov2 := ChisloStudentov2 + 1;
     WriteLn(MFS[i]);
     FlagSdachiEkzamenov := False;
     Break;
    end;
   if FlagSdachiEkzamenov then
    if MOE[i, j] = 3 then
     Flag3 := True
    else
     if MOE[i, J] = 4 then
      Flag4 := True
     else
      Flag5 := True;
  end;
 if (FlagSdachiZachetov) and (FlagSdachiEkzamenov) then
  begin
   if (Not(Flag3)) and (Flag4) and (Flag5) then
    ChisloStudentov4_5 := ChisloStudentov4_5 + 1;
   if (Not(Flag5)) and (Flag4) and (Flag3) then
    ChisloStudentov4_3 := ChisloStudentov4_3 + 1;
  end;
end;
 WriteLn ('Число не сдавших сессию: ', ChisloStudentov2);
 WriteLn ('Число сдавших на 4 и 5: ', ChisloStudentov4_5);
 WriteLn ('Число сдавших на 4 и 3: ', ChisloStudentov4_3);
End.
```

4.4. Контрольные задания

4.4.1. Записать информацию о сотрудниках некоторого предприятия: фамилия, домашний адрес, телефон, образование, оклад. Вывести на экран список сотрудников, имеющих высшее образование.

4.4.2. Создать программу, в которой осуществляется проверка, принадлежат ли два адреса одному городу и одной улице. Использовать переменную типа запись с 4 полями: город, улица, дом, квартира.

4.4.3. Пусть имеются сведения о многих студентах (Ф.И.О., дата рождения, адрес, курс и группа), например, одного факультета. Вывести на экран из общего списка фамилии студентов 2-го курса.

4.4.4. Имеется список сотрудников, в котором записаны: фамилия сотрудника, год рождения, месяц и день рождения, домашний адрес (отдельно улица, номер дома, номер квартиры) и телефон. Определить:

a) самого молодого и самого старого сотрудника;

b) кто из сотрудников родился зимой, а кто осенью;

c) сотрудников, которые родились в год быка (1901, 1913, 1925 и т.д.);

d) фамилии сотрудников, которые являются абонентами указанной телефонной станции и проживают на указанной улице в домах с четными номерами (номер станции и название улицы вводятся с клавиатуры).

4.4.5. Имеется список студенческой группы, в котором записаны: фамилия, имя, отчество студента, номер группы, названия и результаты 5 экзаменов. Определить:

a) для каждого студента средний балл по экзаменам и перераспределить список студентов в соответствии с набранными баллами;

b) кто из студентов будет получать стипендию (те, кто сдал без троек);

c) кто из студентов имеет наибольший и наименьший средний балл по экзаменам;

d) списки студентов, которые не будут получать стипендию (у кого есть тройки), которые будут получать обычную стипендию (менее 50% отличных оценок) и повышенную стипендию (более 50% отличных оценок);

e) средний балл по заданному предмету в заданной группе;

f) какой экзамен и в каких группах был сдан успешнее остальных.

4.4.6. Выяснить, сколько дней прошло между указанными датами. Даты заданы записью

```
Type
 Tdate = record
          Day: 1 .. 31;
          Month: 1 .. 12;
          Year: word;
         end;
```

Лабораторная работа № 5

Множества

5.1. Цели работы

Научиться использовать тип-множество языка Паскаль.

5.2. Краткие теоретические сведения

Множества – это наборы однотипных логически связанных друг с другом объектов. Характер связей между объектами лишь подразумевается про-

граммистом и никак не контролируется компилятором. Количество элементов множества может меняться в пределах от 0 до 255 (множество, не содержащее элементов, называется пустым). Именно непостоянством количества своих элементов множества отличаются от массивов и записей.

Два множества считаются эквивалентными тогда и только тогда, когда все их элементы одинаковы, причем порядок следования элементов множества безразличен. Если все элементы одного множества входят также и в другое, говорят о включении первого множества во второе. Пустое множество включается в любое другое.

Пример определения и задания множеств:

```
Type
  DigitChar = set of '0'..'9';
  Digit = set of 0..9;
Var
  S1, S2, S3: DigitChar;
  S4, S5, S6: Digit;
Begin
  S1 := ['1', '2', '3'];
  S2 := ['3', '2', '1'];
  S3 := ['2', '3'];
  S4 := [0..3, 6];
  S5 := [4, 5];
  S6 := [3..9];
  ...
End.
```

Здесь множества S1 и S2 эквивалентны, а множество S3 включено в S2.

К переменным и константам множественного типа применимы операции присваивания (:=), объединения (+), пересечения (*) и разности(–):

```
S4*S6 содержит [3];
S4*S5 содержит [] – пустое множество;
S4+S5 содержит [0, 1, 2, 3, 4, 5, 6];
S5=S6 содержит [3, 4, 5, 6, 7, 8, 9];
S6–S5 содержит [3, 6, 7, 8, 9];
S4–S5 содержит [0, 1, 2, 3, 6].
```

К множественным величинам применимы операции: тождественность (=), нетождественность (<>), содержится в (<=), содержит (>=). Результат выполнения этих операций имеет логический тип, например:

```
['A', 'B'] = ['A', 'C'] даст False;
```



```
['A', 'B'] <> ['A', 'C'] даст True;
['B'] <= ['B', 'C'] даст True
['C', 'D'] >= ['A'] даст False.
```

Кроме этих операций для работы с величинами множественного типа в языке Паскаль используется операция **in**, проверяющая принадлежность элемента базового типа, стоящего слева от знака операции, множеству, стоящему справа от знака операции, например:

```
'A' in ['A', 'B'] даст True;
2 in [1, 3, 6] даст False.
```

Дополнительно к этим операциям можно использовать две процедуры. **Include** (S, I) – включает новый элемент I в множество S. **Exclude** (S, I) – исключает элемент I из множества S.

5.3. Примеры решения типовых задач

5.3.1. Дано 100 целых чисел от 1 до 50. Определить, сколько среди них чисел Фибоначчи и сколько чисел, первая значащая цифра в десятичной записи которых – 1 или 7.

```
Program Example_5_3_2;
Uses
  WinCrt;
Const
  MinDig = 1;   { минимально возможное число }
  MaxDig = 50;  { максимально возможное число }
Type
  TRange = MinDig .. MaxDig; { множество возможных чисел }
  TSetRange = set of TRange;
Var
  i, n, Last, Current: Byte;
  Fib: TSetRange; { множество чисел Фибоначчи }
  Digit: TRange; { обрабатываемые числа }
  CountFib, Count: Byte;

Function FirstDig( k: TRange): Byte;
{ функция определения первой значащей цифры числа k }
var
  i : Byte;
begin
  repeat
```

```
    i := k mod 10;
    k := k div 10;
  until k = 0;
  FirstDig := i;
end; { FirstDig }

Begin
{ формирования множества чисел Фибоначчи, }
{ принадлежащих заданной границе чисел }
 Fib := [];
 Last := 1;
 Current := 1; { последнее и текущее число }
 repeat
  Include (Fib, Current);
  Current := Last + Current;
  Last := Current – Last;
 until Current > MaxDig;
  Write (' Введите количество чисел ( > 0 ): ');
  repeat
   ReadLn (n);
   if n <= 0 then
    Write ('Данные введены неправильно. Повторите ввод: ');
  until n > 0;
  Write ('Вводите числа в границах от ');
  WriteLn (MinDig,' до ', MaxDig);
  for i :=1 to n do
  begin
   ReadLn (digit);
   if digit in Fib then
    CountFib := CountFib + 1;
   if FirstDig(digit) in [1,7] then
    Count := Count + 1;
  end;
  WriteLn ('Кол-во чисел Фибоначчи = ', CountFib);
  WriteLn ('Кол. чисел с первой знач. цифрой 1 или 7:', Count);
End.
```

5.4. Контрольные задания

5.4.1. Пусть дан текст. Посчитайте общее число вхождений английских букв в текст и найдите наибольшее количество цифр, идущих подряд.

5.4.2. Пусть дан текст. Верно ли, что в нем имеются буквы, входящие:

a) в слово «шина»;

b) в слово, задаваемое пользователем?

5.4.3. Дан текст, состоящий из строчных латинских букв и цифр. Определите, каких букв, гласных (a, e, i, o, u) или согласных, больше в этом тексте.

5.4.4. С клавиатуры вводится текст. Выведите на экран символы, входящие в текст не менее двух раз.

5.4.5. Дана непустая последовательность слов из строчных русских букв. Между соседними словами присутствует запятая, за последним – точка. Выведите на экран в алфавитном порядке:

a) гласные буквы, которые входят в каждое слово;

b) согласные буквы, которые не входят ни в одно слово;

c) все глухие согласные буквы, которые входят в каждое нечетное слово и не входят хотя бы в одно четное слово.

Примечание: Буквы а, е, и, о, у, ы, э, ю, я – гласные; все остальные – согласные, кроме й, ь, ъ. Звонкие согласные: б, в, г, д, ж, з, л, м, н, р; глухие согласные: к, п, с, т, ф, х, ц, ч, ш, щ.

5.4.6. Не используя дополнительных переменных, поменяйте местами значения *A* и *B* множественного типа.

5.4.7. Написать программу для определения пересечения, объединения и разности двух множеств *A* и *B*. Определить, принадлежит ли переменная *k* одному из полученных множеств.

5.4.8. Пусть дан фрагмент программы, приведенный ниже. Опишите функцию *Sum*(*A*, *S1*, *S2*), вычисляющую сумму тех элементов матрицы *A*, номера строк и номера столбцов которых принадлежат, соответственно, непустым множествам *S1* и *S2* типа *num*.

```
const
  n = 10;
 type
  number = 1..n;
  matrix = array [number, number] of real;
  num = set of number;
```

5.4.9. Дан текст, заканчивающийся точкой. Текст состоит из слов, разделенных пробелами. Напечатайте все слова, которые состоят из тех же букв, что и последнее слово текста, и вывести на экран те буквы, которые встречаются в каждом слове текста только один раз.

5.4.10. Дан фрагмент программы. Опишите процедуру *Наличие* (*Маг, A, B, C*), которая по информации из массива *Маг* типа *магазины* присваивает параметрам *A*, *B*, *C* типа *ассортимент* значения, перечисленные ниже.

```
type
  продукт = (хлеб, масло, молоко, мясо, рыба, соль, сыр, колбаса, сахар, чай, кофе);
```

ассортимент = **set of** продукт;
магазины = **array** [1..20] **of** ассортимент;
a) *A* – продукты, которые есть во всех магазинах;
b) *B* – продукты, которые есть хотя бы в одном магазине;
c) *C* – продукты, которых нет ни в одном магазине.

5.4.11. Дан фрагмент программы. Опишите логическую функцию *Везде* (*ГР*), определяющую, есть ли в группе хотя бы один человек, побывавший в гостях у всех остальных из группы (*ГР*[*x*] – множество людей, побывавших в гостях у человека с именем *x*; $x \notin ГР[x]$).

type
имя = (Вася, Володя, Ира, Марина, Миша, Наташа, Олег, Оля, Света, Юля, Саша);
гости = **set of** имя;
группа = **array** [имя] **of** гости;

5.4.12. В столовой имеются отдельные меню на завтрак, обед и ужин. Известно, что в каждом из этих меню не более 10 видов блюд. Определите, какие виды блюд имеются и на завтрак, и на обед, и на ужин, если такие есть. Определите виды блюд, которые есть только на завтрак, только на обед, только на ужин (виды блюд рассматривайте как данные перечисляемого типа).

Лабораторная работа № 6

Файлы

6.1. Цели работы

Научиться создавать и использовать файлы при помощи языка Паскаль.

6.2. Краткие теоретические сведения

Турбо Паскаль поддерживает три файловых типа:

- типизированные файлы (последовательность компонент любого заданного типа, кроме типа «файл»);
- нетипизированные файлы (последовательность компонент произвольного типа);
- текстовые файлы (совокупность строк, разделенных метками конца строки; сам файл заканчивается меткой конца файла).

Доступ к файлу в программе происходит с помощью переменных файлового типа, которые описываются одним из трех способов:

- **File** of *тип* – типизированный файл (указывается тип компоненты);

- **File** – нетипизированный файл;
- **Text** – текстовый файл.

Примеры описания файловых переменных:

```
Var
  f1: File of Integer;
  f2: File;
  t: Text;
```

Все операции с файлом в программе производятся с помощью связанной с ним файловой переменной. Функция **Assign** (f, FileName) связывает файловую переменную f с физическим файлом, полное имя которого задано в строке FileName. Установленная связь будет действовать до конца работы программы, или до тех пор, пока не будет сделано переназначение.

После связи файловой переменной с дисковым именем файла в программе нужно указать направление передачи данных (открыть файл). В зависимости от этого направления говорят о чтении из файла или записи в файл.

Функция **Reset** (f, BufSize) открывает для чтения файл, с которым связана файловая переменная f. После успешного (без ошибок) выполнения процедуры **Reset** файл готов к чтению из него первого элемента. Процедура завершается сообщением об ошибке, если указанный файл не найден. Если f – типизированный файл, то процедурой **Reset** он открывается для чтения и записи одновременно.

Параметр BufSize используется только для нетипизированных файлов. Он задает число байтов, считываемых из файла или записываемых в него за одно обращение. Минимальное значение BufSize – 1 байт, максимальное – 64K байт. Если BufSize не указан, то по умолчанию он принимается равным 128.

Функция **Rewrite** (f, BufSize) открывает для записи файл, с которым связана файловая переменная f. После успешного выполнения этой процедуры файл готов к записи в него первого элемента. Если указанный файл уже существовал, то все данные из него уничтожаются.

Функция **Close** (f) закрывает открытый до этого файл, связанный с файловой переменной f. Вызов процедуры **Close** необходим при завершении работы с файлом. Если по какой-то причине процедура **Close** не будет выполнена, файл все же будет создан на внешнем устройстве, но содержимое последнего буфера в него не будет перенесено.

Доступ к компонентам файла осуществляется по их порядковым номерам. Компоненты нумеруются, начиная с 0. После открытия файла указатель (номер текущей компоненты) стоит в его начале на нулевом компоненте. После каждого чтения или записи указатель сдвигается к следующему компоненту.

Чтение и запись для разных видов файлов осуществляется по-разному.

Для типизированных файлов запись осуществляется с помощью процедуры **Write** (f, *список переменных*), которая записывает в файл f всю информа-

цию из списка переменных. Чтение из файла осуществляется с помощью процедуры **Read** (f, *список переменных*), которая читает из файла f компоненты в указанные переменные. Тип файловых компонент и тип переменных должны совпадать.

Для нетипизированных файлов запись осуществляется с помощью процедуры **BlockWrite** (f, X, Count, QuantBlock), которая производит запись из переменной X количества блоков, заданное параметром Count, при этом длина блока равна длине буфера. Необязательный параметр QuantBlock возвращает число блоков, успешно записанных текущей операцией **BlockWrite**. Чтение из файла осуществляется процедурой **BlockRead** (f, X, Count, QuantBlock), которая производит чтение в переменную X количества блоков, заданного параметром Count, при этом длина блока равна длине буфера.

Чтение из текстового файла осуществляется с помощью процедур **Read** (f, *список переменных*) или **ReadLn** (f, *список переменных*). Способ чтения зависит от типа переменных, стоящих в списке. Отличие **ReadLn** от **Read** в том, что в нем после прочтения данных пропускаются все оставшиеся символы в данной строке, включая метку конца строки. Запись в текстовый файл осуществляется с помощью процедур **Write** (f, *список переменных*) или **WriteLn** (f, *список переменных*). Способ записи зависит от типа переменных в списке. **WriteLn** от **Write** отличается тем, что после записи всех значений из переменных записывает еще и метку конца строки (формируется законченная строка файла).

Остальные процедуры и функции работы с файлами приведены в прил. 2.

6.3. Примеры решения типовых задач

6.3.1. Создать файл целых чисел *abc.int*. Необходимо поменять местами его максимальный и минимальный элементы.

```
Program Example_6_3_1;
Uses
 WinCrt;
Var
 f: File of Integer;
 a: array [0..200] of Integer;
 i, n, c, max, imax, min, imin: Integer;
Begin
 Assign (f, 'abc.int');
 Rewrite(f); {открытие файла на запись}
 Write ('Количество элементов в файле =');
 ReadLn (n);
 for i :=1 to n do
 begin
```



```
  Write ('c=');
  ReadLn (c);
  Write (f, c);
 end;
 Close (f);
 Reset (f); {открытие файла на чтение}
 for i := 0 to FileSize(f) - 1 do
  Read (f, a[i]);
 max := a[0]; imax := 0;
 min := a[0]; imin := 0;
 {Поиск макс. и мин. элементов в файле и их индексов}
 for i:=1 to FileSize(f) - 1 do
 begin
  if a[i] > max then
  begin
   max := a[i];
   imax := i;
  end;
  if a[i] < min then
  begin
   min := a[i];
   imin := i
  end;
 end;
 {перезапись макс. и мин. значений в файл}
 Seek (f, imax);
 Write (f, min);
 Seek (f, imin);
 Write (f, max);
 {вывод результата на экран}
 Seek (f, 0);
 for i := 0 to FileSize(f) - 1 do
 begin
  Read (f, c);
  WriteLn (c);
 end;
 Close (f);
End.
```

6.3.2. Создать файл *Note.txt*, состоящий из строк различной длины. Определить длину самой длинной строки.

```
Program Example_6_3_2;
```

```
Uses
  WinCrt;
Var
  F: Text; { переменная текстового файла }
  G: File; { переменная нетипизированного файла }
  C: Char;
  Max, Count: Integer;
Begin
  Assign (G, 'Note.txt');
  { открываем файл, и устанавливаем размер блока в 1 байт }
  Rewrite (G, 1);
  WriteLn ('Введите текст, заканчивающийся точкой:');
  while c <> '.' do
  begin
   Read (c);
{записываем в файл G символ c, а точнее Sizeof(char) блоков по одному байту} {из переменной c}
    BlockWrite (G, c, Sizeof(char));
  end;
  Close (G);
  Assign (F, 'Note.txt');
{проверка существования файла и возможности чтения данных из него}
  {$I-}
  Reset (F);
  {$I+}
  if IOResult <> 0 then
   WriteLn ('Error!')
  else
  begin
   Max := 0;
   while not Eof (F) do
   begin
    count := 0;
    {подсчет количества символов в строке}
    while not Eoln (F) do
    begin
     Read (F, C);
     count := count + 1;
    end;
    if count > Max then
     Max := count;
    ReadLn (F);
  end;
```



```
   WriteLn ('Макс. стр. состоит из ', Max,' знаков');
   Close (F);
  end;
End. .
```

6.4. Контрольные задания

6.4.1. Даны три упорядоченных по возрастанию файла целых чисел. Найти наименьшее из чисел, встречающихся во всех трех файлах. Файлы должны быть прочитаны не более одного раза.

6.4.2. Создать файл, содержащий информацию об авиапассажирах: фамилия пассажира, количество вещей багажа, общий вес багажа, номер рейса (входной файл *reis.txt*). Определить:

a) фамилии пассажиров, летящих указанным рейсом, и суммарный вес багажа этих пассажиров (номер рейса вводится с клавиатуры, результат записать в файл);

b) кто из пассажиров имеет багаж весом более 35 кг и какими рейсами они летят (результат записать в файл);

c) средний вес одной вещи в общем багаже и фамилии пассажиров, у которых средний вес одной вещи в их багаже отличается от среднего веса одной вещи в общем багаже не более чем на 1 кг (результат записать в файл).

6.4.3. Текст из 5 строк хранится в файле. Определить, сколько в тексте знаков препинания.

6.4.4. Текст из 10 строк хранится в файле. Выполнить следующие операции:

a) определить количество слов в нечетных строках текста;

b) определить количество знаков препинания в четных строках текста;

c) определить количество предложений, учитывая, что предложение заканчивается точкой, вопросительным или восклицательным знаками;

d) вычислить, сколько в тексте слов-палиндромов;

e) вывести на экран самое длинное предложение.

6.4.5. Сформировать файл F, компонентами которого являются целые числа $c_i = 2i + 3i + 1$. Количество чисел равно n, где n-натуральное число. Переписать компоненты файла F в файл G с отрицательным знаком. Последним числом в файл G записать сумму полученных компонентов.

6.4.6. В файл записать информацию об абитуриентах, поступающих в университет: фамилия, имя, отчество, домашний адрес, номер телефона, изучаемый язык, сумма баллов, полученных на вступительных экзаменах. Напечатать список абитуриентов, набравших на экзаменах N и более баллов.

6.4.7. Сформировать файл F, компонентами которого являются целые числа. Записать в файл C все четные числа файла F, а в файл N – все нечетные. Порядок следования чисел сохраняется.

6.4.8. Дано натуральное n. Сформировать файл G, компонентами которого являются целые числа b_1, b_2, ..., b_n, определяемые по формуле $b_i = i$. Последним числом в файл G записать сумму четных чисел.

6.4.9. В файл записать информацию о книгах университетской библиотеки: автор, название книги, год издания, цена. Напечатать список книг, изданных начиная с N года.

6.4.10. Сформировать файл T, компонентами которого являются целые числа. Записать в файл R все компоненты файла T, которые делятся на 3 и не делятся на 7.

6.4.11. Составить программу, которая создает файл, состоящий из N значений типа integer. Прочитать файл и вывести только четные элементы.

6.4.12. Создать программу "Телефонный справочник". Справочник содержит следующую информацию: Ф.И.О. абонента, домашний адрес, номер телефона. Программа осуществляет поиск номера телефона по фамилии, имени и отчеству абонента.

ПРИЛОЖЕНИЕ 1

ОПЕРАЦИИ НАД СТРОКАМИ. СТАНДАРТНЫЕ ПРОЦЕДУРЫ И ФУНКЦИИ

Таблица П.1.1

Операции над строками

Знак	*Операция*
S1 + S2	*конкатенация (сцепление)*
S1 = S2	*сравнение на равенство*
S1 < S2	*сравнение на меньше*
S1 > S2	*сравнение на больше*
S1 <> S2	*сравнение на неравенство*

Например: St := 'a' + 'bc'; (после конкатенации в строке St содержится 'abc'). При этом, если длина сцепленной строки превысит свою максимально допустимую длину N, то «лишние» символы будут отброшены.

Операции отношения выполняются над двумя строками посимвольно, слева направо с учетом ASCII кодировки символов. Если коды символов с индексом 1 сравниваемых строк равны, то сравниваются коды следующих символов. Если одна строка меньше другой по длине, то недостающие символы короткой строки заменяются символом с кодом 0.

Таблица П.1.2

Стандартные процедуры и функции для работы со строками

Заголовок стандартной процедуры или функции	*Назначение*
function Concat(S1, [S2, ..., Sn]: String): String;	*Выполняет сцепление строк-параметров, которых может быть произвольное количество. Если длина результирующей строки превышает 255 символов, то она усекается до 255 символов. Функция эквивалентна операции конкатенации.*
procedure Copy(S: String; Indx, Count: Integer);	*Возвращает подстроку, выделенную из строки* S, *длиной* Count *символов, начиная с символа под номером* Indx.
procedure Delete(var S: String; Indx, Count: Integer);	*Удаляет из строки* S *подстроку длиной* Count *символов, начиная с символа под номером* Indx.
procedure Insert(String1: String; var S: String; Indx: Integer);	*Вставляет строку* String1 *в строку* S, *начиная с символа под номером* Indx. *Если длина результирующей строки превышает 255 символов, то она усекается до 255 символов.*
function Length(S: String): Integer;	*Возвращает значение текущей длины строки* S.
function Pos(Sub, S: String): Byte;	*Производит поиск подстроки* Sub *в строке* S. *Возвращает номер первого символа подстроки* Sub *в строке* S. *Если подстрока не найдена, то функция возвращает 0.*
procedure Str(X [:Size[:Dec]], var S: String);	*Преобразует численное выражение* X *(целого или вещественного типа) в его строковое представление, в соответствии с необязательными параметрами форматирования* Size *и* Dec *(общий размер и размер дробной части; аналогичны необязательным параметрам численных выражений в стандартной процедуре* Write*). Сохраняет результат в строке* S.

Окончание таблицы П.1.2

Заголовок стандартной процедуры или функции	*Назначение*
procedure Val(S: String; var V; Code: Integer);	*Преобразует строковое представление числа, содержащееся в S, в его численное представление и сохраняет результат (целого или вещественного типа) в* V. *Если в* S *встречается недопустимый символ (с точки зрения правил записи чисел), то преобразование не происходит, а в* Code *сохраняется его номер. В противном (успешном) случае* Code *равно 0.*

Например:

```
S := Concat ('Hello,', ' World'); { S = 'Hello, World' }
S2 := Copy(S, 8, 5); { S2 = 'World' }
Delete(S, 6, 7); { S = 'Hello' }
Insert(', Student', S, 6); { S = 'Hello, Student' }
N := Length(S); { N = 14 }
X := Pos('lo', S); { X = 4 }
Str(Pi:4:2, S2); { S2 = '3.14' }
Val('7', X, Y); { X = 7; Y = 0 }
```

Отметим, что кроме использования типа **String**, для работы со строками применяют так называемые строки с завершающим нулем (ASCIIZ – строки), которые могут содержать более 255 символов. Для этого задействуются процедуры и функции стандартного модуля **Strings**.

ПРИЛОЖЕНИЕ 2

ПРОЦЕДУРЫ И ФУНКЦИИ РАБОТЫ С ФАЙЛАМИ

Таблица П.2.1

Процедуры и функции работы с файлами

Функция	*Описание*
Append(f)	*Открывает существующий текстовый файл для добавления информации в конец (используется вместо функции* Reset*).*
Erase(f)	*Стирает внешний файл.*
EoF(f)	*Возвращает* true, *если текущая позиция файла находится в его конце, иначе –* false.
FilePos(f)	*Возвращает текущую позицию в файле. Для текстовых файлов не используется.*
FileSize(f)	*Возвращает текущий размер файла. Для текстовых файлов не используется.*
IOResult(f)	*Возвращает целое значение, являющееся состоянием последней выполненной операции ввода-вывода.*
Seek(f, n)	*Перемещает текущую позицию в файле на элемент с номером* n. *Для текстовых файлов не используется.*

БИБЛИОГРАФИЧЕСКИЙ СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1. **Алексеев, Е.Р.** Турбо Паскаль 7.0 / Е.Р. Алексеев, О.В. Чеснокова – М.: НТ Пресс, 2006. – 320 с.

2. **Марченко, А.И.** Программирование в среде Turbo Pascal 7.0. Базовый курс / А.И. Марченко, Л.А. Марченко. – К.: ВЕК+, 2003.

3. **Немнюгин, С.А.** Изучаем Turbo Pascal / С.А. Немнюгин, Л.В. Перколаб. – СПб.: Питер, 2003. – 320 с.

4. **Лабораторный практикум по информатике**: учеб. пособие / Под ред. Острейковского В.А. – М.: Высш. шк., 2003. – 275 с.

5. **Могилев, А.В.** Практикум по информатике: учеб. пособие для студ. высш. учеб. заведений / А.В. Могилев, Н.И. Пак, Е.К. Хеннер; под ред. Е.К. Хеннера. – 2-е изд. – М.: Издательский центр «Академия», 2005. – 608 с.

6. **Программирование на языке Паскаль**: задачник / Под ред. Усковой О.Ф. – СПб.: Питер, 2002. – 336 с.

7. **Юркин, А.Г.** Задачник по программированию. – СПб.: Питер, 2002. – 192 с.

ОГЛАВЛЕНИЕ

Паскаль: подпрограммы и сложные типы данных

Методические указания к выполнению лабораторных работ
по курсам «Информатика»; «Информатика и программирование»
для студентов всех специальностей
очной формы обучения

Составители: к. т. н., ст. преп. Ольга Евгеньевна Ефимова,
к. т. н., доц. Андрей Вячеславович Распопов,
к. т. н., доц. Дмитрий Васильевич Меркулов

Подписано в печать 17.05.2010. Формат 60Х48 1/16. Уч.-изд. л. 2,4.
Усл.-печ. л. 2,5. Бумага писчая. Тираж 500 экз. Заказ № 313

Отпечатано: отдел оперативной полиграфии Воронежского
государственного архитектурно-строительного университета
394006 Воронеж, ул. 20-летия Октября, 84