Методические указания для выполнения контрольной работы по курсу «Введение в направление»

Обработка матриц. Блок-схемы.

Цель контрольной работы: получение навыков использования двумерных массивов.

Задание

Написать и отладить программу, выполняющую задание. Программа должна работать с двумерным числовым массивом (матрицей). В программе организовать меню. Все действия с матрицей выполнять в процедурах, которые вызываются по команде пользователя. Написать процедуру вывода, которая представляет матрицу так, как показано на рисунке 1. Построить блок-схемы для каждой процедуры и для основной программы.

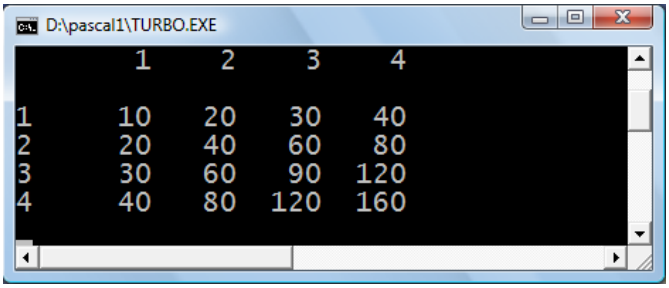


Рисунок 1. Вывод матрицы с нумерацией строк и столбцов.

Теоретические сведения.

Операторы цикла

Циклы предназначены для неоднократного выполнения одинаковых действий. У любого цикла имеются следующие конструктивные особенности: 1. Цикл содержит условие, которое определяет количество повторов; 2. Как правило, условие цикла содержит переменную, которая называется переменной цикла; 3. До начала цикла переменной цикла надо присвоить исходное значение, которое позволяет войти в цикл; 4. Внутри цикла переменная цикла должна изменяться таким образом, чтобы цикл мог через некоторое время завершиться. Операторы, которые выполняются несколько раз внутри цикла, называются «тело цикла». Иногда специально строят цикл таким образом, чтобы он никогда не завершался. Получается бесконечный цикл.

В зависимости от места расположения условия и количества возможных повторов выделяют три вида циклов: цикл с предусловием, цикл с постусловием, цикл с параметром.

1. Цикл с предусловием.

Условие проверяется в начале цикла, до выполнения тела цикла. Чтобы цикл выполнился, условие должно быть истинным. Количество выполнений цикла в общем случае неизвестно заранее.

Синтаксис на языке паскаль:

while <условие цикла> do <тело цикла>

2. Цикл с постусловием

Условие проверяется в конце цикла, после выполнения тела цикла. Таким образом, цикл обязательно выполнится хотя бы один раз. Чтобы цикл выполнился еще один раз, условие должно быть ложным.
Синтаксис на языке паскаль:
repeat <тело цикла> until <условие цикла>

3. Цикл с параметром

Здесь условие – это сравнение параметра цикла с конечным значением. Пока параметр цикла меньше или равен конечного значения, цикл выполняется. Условие проверяется в начале цикла, до выполнения тела цикла. Чтобы цикл выполнился, условие должно быть истинным. Переменная цикла должна иметь порядковый тип. Количество выполнений цикла известно заранее и равно разнице между начальным и конечным значениями. Переменная-параметр цикла изменяется автоматически, поэтому специальный оператор в теле цикла не требуется. Оператор for в паскале имеет два варианта: если используется to, то значения параметра цикла увеличиваются на единицу, а если используется downto, то значения параметра цикла уменьшается на единицу.
Синтаксис на языке паскаль:
for <параметр цикла> := <начальное значение> (to|downto) <конечное значение> do <тело цикла>

Массивы

Массивы – это структурный тип данных, который строится на основе какого-либо базового типа данных. Массив содержит упорядоченную совокупность однотипных данных. Каждый элемент массива имеет своеобразный адрес, называемый индексом. Так реализуется упорядоченность.

Массив ::= array [<тип индекса>{,<тип индекса> }] of <тип элемента>

Тип индекса определяет его допустимые значения. Таким типом могут быть следующие порядковые типы: boolean, char, integer, перечисляемый тип, а также диапазоны этих типов. В зависимости от количества индексов различают одномерные, двумерные, трехмерные и n-мерные массивы. Элемент массива может иметь любой допустимый тип, кроме файлового.

Объявить массив можно двумя способами:
А) в разделе объявления переменных var задать переменную-массив;

```
var
a: array [1..10] of integer; {одномерный массив из 10 целых чисел}
b: array [byte] of char; {массив из 256 символов, индекс изменяется от 0 до 255}
c: array [‘a’.. ‘c’, -5..-3] of byte; {матрица размером 3*3. Индекс строк меняется от ‘a’ до ‘c’, а
                                       индекс столбца от -5 до -3}
d: array [‘a’.. ‘c’] of array [-5..-3] of byte; {другое объявление матрицы c. Массив массива}
```

В) предварительно объявить новый пользовательский тип-массив, а потом объявить переменную этого типа.

```
type  mas = array [1..10] of integer;
var   a: mas;
```

Операции над массивами

Присваивание массивов целиком возможно, если массивы совпадают по типу.

```
var   a, b: array [1..10] of integer;
```

… a:= b;
Для доступа к элементу массива необходимо указать имя массива и далее в квадратных скобках значения индексов через запятую:
a [i] a[3] c[‘a’,-3]

Операции над матрицами.

Сложение матриц *A* + *B* есть операция нахождения матрицы *C*, все элементы которой равны попарной сумме всех соответствующих элементов матриц *A* и *B*, то есть каждый элемент матрицы *C* равен

$$c_{ij} = a_{ij} + b_{ij}$$

Умножение матриц (обозначение: *AB*, реже со знаком умножения $A \times B$) — есть операция вычисления матрицы *C*, элементы которой равны сумме произведений элементов в соответствующей строке первого множителя и столбце второго.

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

Количество столбцов в матрице *A* должно совпадать с количеством строк в матрице *B*. Если матрица *A* имеет размерность $m \times n$, *B* — $n \times k$, то размерность их произведения *AB* = *C* есть $m \times k$.

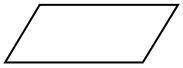
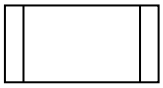
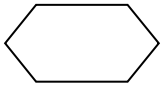
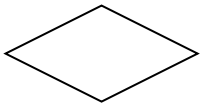
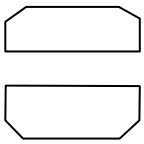
Транспонированная матрица — матрица A^T, полученная из исходной матрицы *A* заменой строк на столбцы. Формально, транспонированная матрица для матрицы *A* размеров n×m— матрица A^T размеров m×n, определённая как $A^T[i, j] = A[j, i]$.

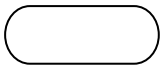
Исходная $A = \begin{pmatrix} 2 & 3 & 4 & 5 \\ 7 & 0 & 5 & 3 \\ 8 & 6 & 1 & 2 \end{pmatrix}$, транспонированная $A^T = \begin{pmatrix} 2 & 7 & 8 \\ 3 & 0 & 6 \\ 4 & 5 & 1 \\ 5 & 3 & 2 \end{pmatrix}$.

Блок-схемы алгоритмов

Для спецификации процессов используют различные средства, в том числе граф-схемы (блок-схемы или просто схемы) алгоритмов, каждая из которых представляет собой граф. Элементарный граф дополнен надстройкой, которая задает вычисления, каждому из которых соответствует путь в этом графе и последовательность меток узлов, лежащих на этом пути. Узлы графа, с помощью введения типизации, интерпретируются как действия, а дуги задают порядок передачи управления. Для каждого алгоритма существует одна начальная дуга и одна или множество конечных. ниже представлены типы узлов граф-схемы алгоритма в соответствии с ГОСТ 19.701-90 «Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения».

Таблица 1.

Символ	название	интерпретация
	данные	данные, носитель данных неопределен
	процесс	функция обработки данных любого вида (выполнение определенной операции или группы операций, приводящие к изменению значения, формы или размещения информации или к определению, по которому из нескольких направлений потока следует двигаться)
	предопределенный процесс	предопределенный процесс, состоящий из одной или нескольких операций или шагов программы, которые определены в другом месте (в подпрограмме, модуле)
	Подготовка	модификация команды или группы команд с целью воздействия на некоторую последующую функцию
	Решение	решение или функция переключательного типа, имеющая один вход и ряд альтернативных выходов, один и только один из которых может быть активизирован после вычисления условий, определенных внутри этого символа; соответствующие результаты вычисления могут быть записаны по соседству с линиями, отображающими эти пути
	граница цикла	начало и конец цикла; обе части имеют один и тот же идентификатор; условия инициализации, приращения, завершения, и т.д. помещаются внутри символа в начале или в конце в зависимости от расположения операции, проверяющей условие
	соединитель	выход в часть схемы и вход из другой части этой схемы; используется для обрыва линии и продолжения ее

Символ	название	интерпретация
		в другом месте; соответствующие символы-соединители должны содержать одно и то же уникальное обозначение
	терминатор	выход во внешнюю среду и вход из внешней среды (начало и конец программы)
	линия	поток данных или управления; для направлений справа-налево и снизу-вверх добавляются стрелки
	пунктирная линия	используется для обведения аннотируемого участка
	комментарий	пояснительные записи и примечания
	пропуск	пропуск символа или группы символов, в которых не определены ни тип, ни число символов; используется только в символах линии или между ними; служит для отображения общих решений с неизвестным числом повторений

Приведем основные свойства граф-схем.

1. Графическое представление.
2. Поддержка описания управляющей части алгоритма.
3. Возможность реализации синтаксического контроля.
4. Возможность проверки управляющей части алгоритма.
5. Отсутствие возможности верификации информационной части.

Необходимо отметить, что по граф-схеме алгоритма можно произвести оценку таких характеристик алгоритма, как временная сложность и сложность описания, однако, возможность оценки емкостной сложности отсутствует.

Стандарт не накладывает ограничений на символьные описания процессов внутри блоков. Можно использовать естественный язык, математическую нотацию или язык программирования.

Порядок выполнения

1) Получить вариант задания у преподавателя
2) Написать и отладить программу
3) Нарисовать блок-схемы для процедур и основной части программы
4) Подготовить отчет
5) Представить распечатку отчета преподавателю, защитить отчет (ответить на вопросы преподавателя по проделанной работе), показать выполнение программы по шагам и изменения значений переменных в окне просмотра.

Пример (неполный) выполнения контрольной работы

Задание

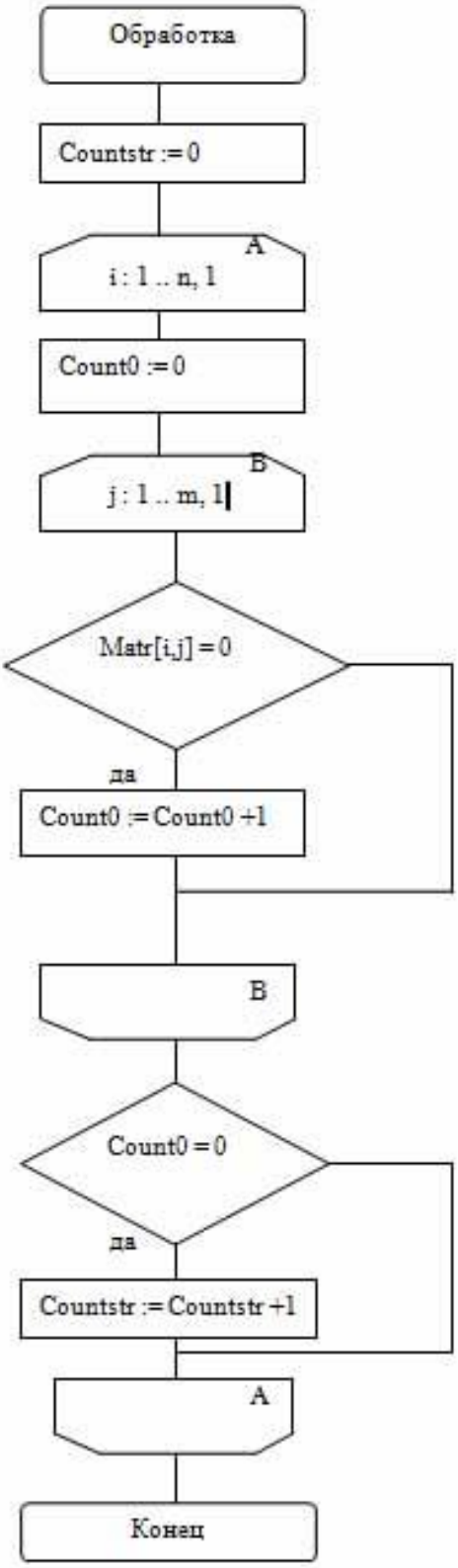
В матрице определить количество строк, не содержащих ни одного нулевого элемента.

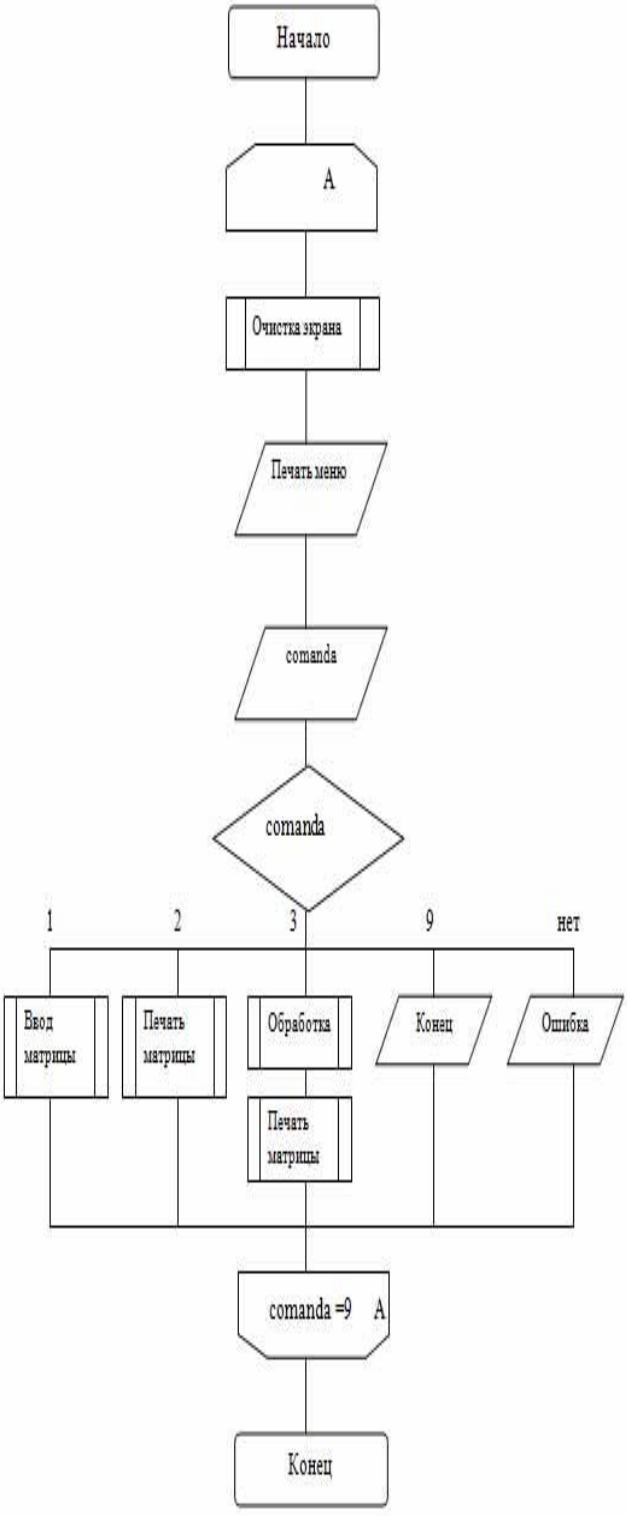
Программная реализация

```
program kr;
uses crt;   {подключение модуля, в котором находится функция очистки экрана clrscr }
const
 n =4;
 m = 4;   {определение констант для размера матрицы}
type matriza = array [1..n, 1..m] of byte;
{объявление пользовательского типа – целочисленного двумерного массива размером n на m}

procedure Vvod(var matr:matriza);
{процедура ввода элементов матрицы  matr – формальный параметр процедуры. Через эту переменную процедура получает матрицу при вызове в меню. Пометка var перед именем переменной означает, что изменения, совершенные с переменной внутри процедуры, будут переданы в вызывающую точку программы}
var
  i,j : byte;
{локальные переменные процедуры, которые используются в качестве переменных циклов}

begin
    for i:= 1 to n do    {первый (внешний) цикл перебора элементов матрицы – переход по строкам}
      for j := 1 to m do
         {второй (внутренний) цикл перебора элементов матрицы – переход по столбцам внутри одной строки }
      begin
        writeln ('Vvedite element ', i,' ',j);  {вызов стандартной процедуры печати на экран}
        readln(matr[i,j]);  {вызов стандартной процедуры чтения с клавиатуры}
      end;
end;

procedure print(matr:matriza);
{процедура печати матрицы}
var
  i,j : byte;

begin
     for i:= 1 to n do
     begin
        for j := 1 to m do
         write ( matr[i,j]:5);
         writeln;   {перевод строки}
     end;
end;
```

```
procedure obrabotka(matr:matriza; var countstr : byte);
{процедура, которая выполняет задание по варианту. Она имеет два формальных параметра.
Переменная matr (матрица) не меняется внутри процедуры, поэтому перед ней нет слова var. Переменная countstr будет содержать ответ на задание – количество строк, в которых нет элементов, равных нулю. Пометка var необходима, чтобы вернуть полученное значение в точку вызова процедуры.}

var
  i,j, count0 : byte;
{count0 – переменная для подсчета количества нулей внутри одной строки}
begin
    countstr := 0;
    for i:= 1 to n do
      begin
      count0 := 0;
      for j := 1 to m do
        if matr[i,j] = 0 then count0:= count0 +1;
        {здесь завершается внутренний цикл, в котором идет счет нулей в строке}
      if count0 = 0 then countstr := countstr +1;
      end;
end;

var
  comanda, countstrok : byte;
  mm : matriza;
{раздел переменных основной части программы.
mm – переменная, в которой хранится матрица }

begin
  repeat  {цикл для работы меню}
  clrscr;   {очистка экрана}
  writeln ('1 - vvod matrizy,  2 - vyvod, 3 - obrabotka,  9 - vyhod');
  writeln ('Vvedite comandu');    {печать пунктов меню}
  readln(comanda);          {чтение введенной команды}
  case comanda of           {выбор пункта по значению переменной comanda}
  1 :    Vvod(mm);                {вызов процедуры ввода матрицы с фактическим параметром mm}
  2 :    print(mm);       {вызов процедуры печати матрицы с фактическим параметром mm}

  3 : begin
        obrabotka(mm, countstrok);
          {вызов процедуры обработки матрицы с фактическими параметрами mm и countstrok}
        writeln ('kolichestvo strok bez 0  = ',countstrok);
                  {  и печать ответа}
      end;
  9 :  writeln('End')
  else writeln('Oshibka');  {если введена другая команда}
  end;     { конец оператора case}
  readln;
  until comanda = 9;   {завершение цикла, если введена команда 9}
end.   {конец программы}
```

Блок- схемы процедуры обработки (слева) и основной части программы (справа)





Требования к отчету

Отчет должен содержать:

1) Титульный лист
2) Номер варианта и текст задания
3) Блок-схемы для процедур и основной части программы
4) Листинг программы
5) Копию экрана с матрицей до и после обработки

Замечания

1. Программа должна быть написана на языке паскаль.
2. При желании кроме процедур, можно использовать функции.
3. Если в варианте сказано, что надо сформировать матрицу, то ввод с клавиатуры не требуется.
4. При желании для заполнения матрицы можно использовать генератор случайных чисел.
5. В уже готовую процедуру печати матрицы необходимо обязательно добавить команды, выводящие на экран номера строк и столбцов (см. рис.1). Их также надо отобразить в соответствующей блок-схеме.
6. В блок-схемах при оформлении циклов используйте символы границ циклов (см. образец блок-схем).

Варианты заданий

1) Из матрицы размером n на m удалить строки, максимальный элемент которых равен y. Удаление реализовать как «затирание» текущей строки нижележащими строками, при этом фактический размер матрицы должен уменьшиться, т.е. при выводе матрицы, из которой «удалена» одна строка, необходимо печатать матрицу размером n-1 на m.

2) Заполнить матрицу произвольного размера n на m по образцу (приведен пример для матрицы 3 на 5):

1	2	3	4	5
10	9	8	7	6
11	12	13	14	15

3) Заполнить матрицу произвольного размера n на m по образцу (приведен пример для матрицы 5 на 7):

0	0	0	0	0	0	0
7	0	0	0	0	0	7
7	7	0	0	0	7	7
7	7	7	0	7	7	7
7	7	7	7	7	7	7

4) Заполнить матрицу произвольного размера n на m по образцу (приведен пример для матрицы 5 на 5):

1	0	0	0	0
2	1	0	0	0
3	2	1	0	0
4	3	2	1	0
5	4	3	2	1

5) Из матрицы размером n на m удалить строки, содержащие более трех отрицательных элементов. Удаление реализовать как «затирание» текущей строки нижележащими строками.
6) На основе матрицы размером n на m сформировать одномерный массив, элементами которого будет количество элементов каждой строки матрицы, превышающих среднее арифметическое значение целой матрицы. Если в некоторой строке таких элементов нет, то в соответствующий элемент массива записать ноль.
7) В матрице размером n на m в каждой строке поменять местами максимальный и минимальный элементы, считая, что все числа матрицы различны.
8) Поменять местами две строки матрицы с заданными номерами
9) Удалить столбец матрицы с заданным номером. Удаление реализовать как «затирание» текущего столбца столбцам, расположенными правее (см. объяснение к варианту 1).
10) Все элементы матрицы, которые отличаются от заданного f, заменить нулями.
11) Найти в матрице сумму элементов, меньших Z.
12) В матрице найти в каждой строке наибольший элемент, а потом среди них найти наименьший (задача минимакса).
13) Сформировать матрицу, каждый элемент которой равен произведению номера строки на номер столбца.
14) На основе исходной матрицы сформировать новую, в которой строка матрицы с номером n, умноженная на число z, прибавлена к строке с номером m.
15) Транспонировать матрицу.
16) Найти сумму двух матриц одинакового размера.
17) Найти произведение двух матриц согласованного размера.
18) В матрице найти сумму элементов в строках, в которых нет отрицательных чисел.
19) Найти количество строк матрицы, среднее арифметическое элементов которых меньше заданной величины.
20) Определить номер первого столбца матрицы, в котором есть хотя бы один отрицательный элемент.

Основная литература

1. [004.4(075) – И 21] Иванова Г.С. Основы программирования: учебник / Г. С. Иванова. - 4-е изд., стер. - М.: Изд-во МГТУ им. Н.Э. Баумана, 2007. - 415 с. Количество экз. в библ**. – 5.**
2.[004.42(075)-М82] Москвитина О.А. Сборник примеров и задач по программированию: учебное пособие/ О. А. Москвитина, В. С. Новичков, А. Н. Пылькин. - М.: Горячая линия - Телеком, 2007. - 244 с.Количество экз. в библ**. – 20**
3. [004.432(075)-Н50]НемнюгинС.А.TurboPascal : программирование на языке высокого уровня: учебник/ С. А.Немнюгин. - 2-е изд.. - СПб.: ПИТЕР, 2007. - 544 с. Количество экз. в библ**. –49**

Дополнительная литература

1.[519.682(075)-П32] Пильщиков В.Н. Сборник упражнений по языку Паскаль: Учебное пособие для студентов вузов/ В. Н.Пильщиков. - М.: Наука, 1989. - 154 с.Количество экз. в библ**.** –69
2.[519.682(075)-А16]Абрамов, В. Г.. Введение в язык Паскаль: учебное пособие/ В. Г. Абрамов, Н. П. Трифонов, Г. Н. Трифонова. - М.: Наука, 1988. - 319 с. Количество экз. в библ. – 52
3. [004.43-А50] Алкок, Д. Язык Паскаль в иллюстрациях = ILLUSTRATING PASCAL: монография/ Д. Алкок; Пер. А. Ю. Медников ; Ред. А. Б. Ходулев. - М.: Мир, 1991. - 192 с. Количество экз. в библ. – 23
4. [004.42(075)-Н50]Немнюгин, С. А.TurboPascal: Учебник/ С. А.Немнюгин. - СПб.: Питер, 2000. - 491 с. Количество экз. в библ. – 7
5. [004.43-П12]Павловская, Т. А.. Паскаль. Программирование на языке высокого уровня: учебник/ Т. А. Павловская. - СПб.: ПИТЕР, 2006. - 392 с.: Количество экз. в библ. – 101