РАБОТА С ИНФОРМАЦИОННЫМИ МАССИВАМИ

Методические указания к индивидуальным занятиям по курсу «Информатика» для студентов 1-го курса всех специальностей

Воронеж 2007

Составители В.П. Авдеев, В.И. Гильмутдинов, А.А. Кононов, А.Д. Кононов

УДК 69.003:658.5.012.22
ББК 32.973

Работа с информационными массивами [Текст]: метод указания к индивидуальным занятиям по курсу «Информатика» для студ. 1-го курса всех спец. / Воронеж. гос. арх.-строит. ун-т; сост.: В.П. Авдеев, В.И. Гильмутдинов, А..А.. Кононов, А.Д. Кононов. – Воронеж, 2007. – 20с.

Рассматриваются схемы типовых алгоритмов обработки информационных массивов. При водятся примеры, после подробного разбора которых предлагаются упражнения для проверки условия студентами изучаемого материала. Методические указания предназначены для использования на индивидуальных занятиях по программированию при изучении дисциплины «Информатика» студентами первого курса всех специальностей.

Ил. 26. Библиог.:4 назв.

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

Рецензент – В.К. Маршаков, к. ф. –м. н.. доц. Воронежского государственного университета

ВВЕДЕНИЕ

Настоящие методические указания предназначены для использования на индивидуальных занятиях по программированию в рамках дисциплины «Информатика» студентами всех специальностей.
Рассматриваются схемы типовых алгоритмов обработки массивов информации, оформленные как подпрограммы. Для проверки работы алгоритма на ЭВМ по схеме следует составить программу, дополнив ее операциями ввода и вывода. При разработке алгоритмов использованы только типовые процедуры (циклы «До» и «Пока», разветвление, обход) и их сочетания.

Приведенные в тексте методических указаний упражнения выполняются в следующем порядке. На индивидуальных занятиях происходит ознакомление обучающихся с теорией соответствующих разделов, в период самостоятельной работы конкретизируется блок-схема алгоритма и готовится рабочая программа по заданиям. На очередных индивидуальных занятиях осуществляется реализация подготовленных задач на ЭВМ и обсуждение полученных результатов.

ИСХОДНЫЕ ПОЛОЖЕНИЯ

Понятие «алгоритм» является значительно более общим, чем понятие «вычисление». Мы познакомимся с важным классом невычислительных алгоритмов, занимающих значительное место в практике работы современных ЭВМ. Заметим, что между вычислительными и невычислительными алгоритмами нет резкой грани. Особый интерес представляет то обстоятельство, что построение невычислительных алгоритмов, как правило, требует использования небольшого числа достаточно простых примеров /1, 2, 3, 4/.

Запись информационных массивов на бумаге мы будем оформлять в виде таблиц. Каждый массив имеет имя (буквенное обозначение), а номера его строк, столбцов и т.д. представляются значениями индексов его элементов. Правила обращения с таблицами соответствуют физическим принципам функционирования ячеек памяти ЭВМ: в любой клетке (ячейке) может помещаться только один элемент, при записи которого старое содержимое уничтожается.

ПЕРЕМЕЩЕНИЕ МАССИВОМ

Массив A длины n перепишем в массив B, т.е. все элементы массива A в том же порядке надо записать в соответствующие клетки (ячейки) массива B (рис.1). Решение этой задачи представлено на рис.2.

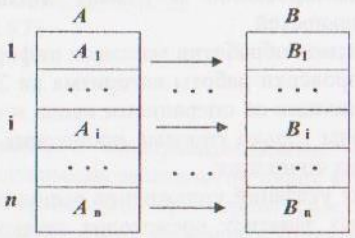

Рис.1

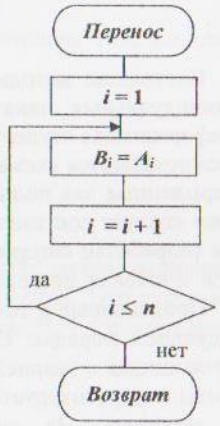

Рис. 2

Упражнения

1. Массив A длины n переносится в массив B так, чтобы первый элемент массива A стал k-м элементом массива B.
2. Совершите перенос массива A так, чтобы его последний элемент, т.е. n-й элемент, стал l –м элементом массива B.

СДВИГ МАССИВА

В предыдущем примере безразлично, в каком порядке обрабатываются элементы массива. Но так бывает далеко не всегда. Сместим массив A размерности (длины) n на один элемент вниз, т.е. 1-ый элемент массива A запишем на место 2-го, 2-й – на место 3-го и т.д. (рис.3).

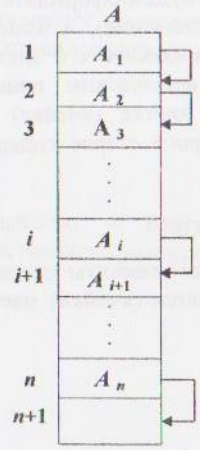

Рис. 3

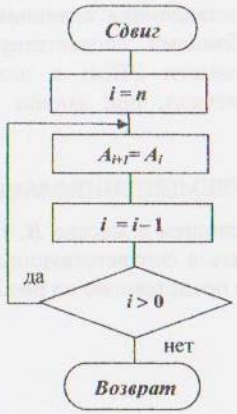

Рис. 4

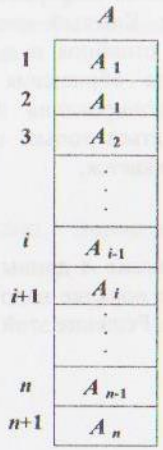

Рис. 5

Мы не можем начать решение этой задачи с первого элемента массива, потому что тогда безвозвратно сотрется значение второго элемента. Начинать приходится с конца (рис.4). В результате выполнения этого алгоритма массив A приобретает вид, показанный на рис.5. Обратим внимание на то обстоятельство, что первым элементом массива A по-прежнему является старое значение A_1: сам по себе он не уничтожается.

Упражнения

3. Совершите сдвиг массива A длины n на k элементов вниз (т.е. $A_1 \rightarrow A_{k+1}$ и т.д.)
4. Совершите сдвиг массива A длины n на k элементов вверх (т.е. $A_{k+1} \rightarrow A_1$ и т.д.)

ПРЕОБРАЗОВАНИЕ МАССИВОВ

В рассмотренных примерах массивы по существу не изменялись. Они совершали лишь параллельный перенос как одно целое. Переходим к задачам, связанным с изменениями внутренней структуры массивов. Начнем с простой задачи. Требуется поменять местами переменные a и b, т.е. содержимое некоторой ячейки, обозначенной буквой a, переписать в ячейку, обозначенную буквой b, и наоборот (рис.6).

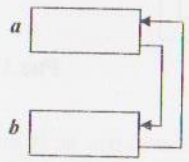

Рис. 6

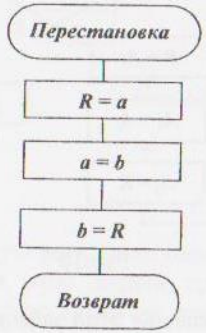

Рис. 7

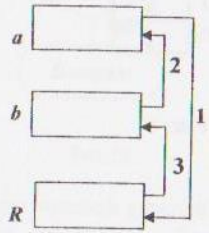

Рис. 8

Несмотря на чрезвычайную простоту задачи, решить ее напрямую не удается. Действительно, после переписывания a в b старое значение b стирается и не может быть переписано в a. Значит, содержимое a надо переписать предварительно в некоторую вспомогательную ячейку (такие ячейки обычно называют рабочими). Обозначим вспомогательную ячейку буквой R. Решение задачи дано на рис.7. Для большей наглядности схему потоков информации по этому алгоритму изобразим порядковыми номерами стрелок (рис.8).

Рассмотрим теперь более сложный пример. Даны число k и одномерный массив A длиной n. Требуется совершить такую перестановку элементов

массива A, чтобы все элементы со значением, большим чем число k, попали в верхнюю часть массива, а остальные элементы – в нижнюю, т.е. в построенном массиве все элементы $A_i > k$ должны иметь малые номера (индексы), а все $A_i \le k$ – большие номера.

На рис. 9 изображен массив с некоторым конкретным числовым наполнением и две вспомогательные ячейки с исходными данными.

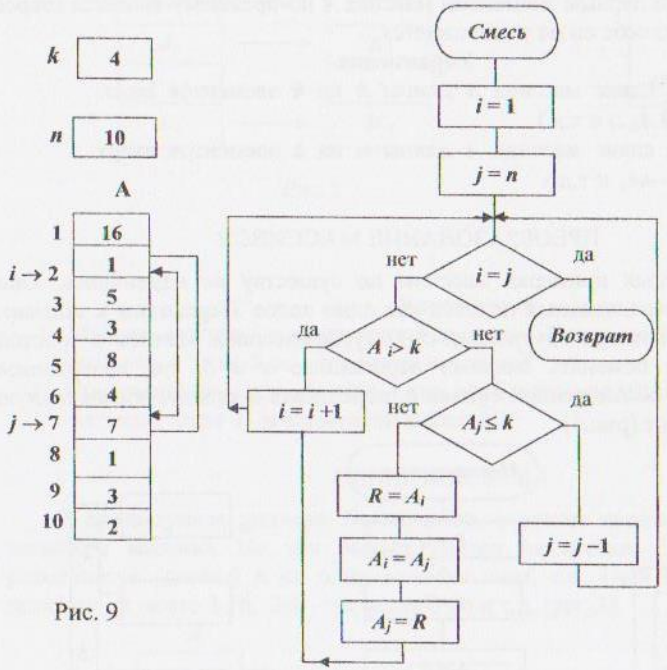

Рис. 9

Рис.10

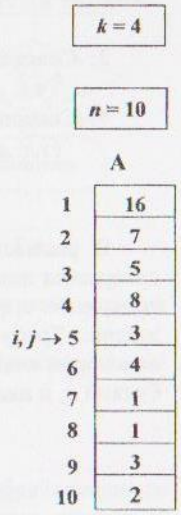

Рис.11

Покажем сначала идею решения этой задачи. Будем перебирать элементы массива сверху вниз до тех пор, пока впервые не встретится элемент со значением, меньшим или равным k (на рис.9 он помечен стрелкой i). Затем, наоборот, начнем перебирать элементы массива снизу вверх до тех пор, пока впервые не встретится элемент со значением, большим k (на рис.9 он помечен стрелкой j). Далее поменяем найденные элементы местами. Описанный процесс будем продолжать от указанных положений стрелок – перебор сверху вниз, перебор снизу вверх, перестановка. Процесс заканчивается, когда стрелки i и j укажут на один и тот же элемент результирующего массива A. На рис.10 изображена блок-схема алгоритма, реализующего указанную идею. Результирующий массив для примера, заданного на рис.9, приводится на рис.11.

В практике работы с массивами часто требуется отобрать его элементы с указанными номерами и образовать из них новый массив. Рассмотрим один из способов выполнения такого задания. Даны два массива A и Q длины n, причем массив Q состоит только из нулей и единиц. Надо построить массив B, состоящий из тех элементов массива A, которым соответствуют единичные элементы массива Q, и определить длину массива B (относительный порядок его элементов должен быть сохранен). Эти массивы представлены на рис.12.

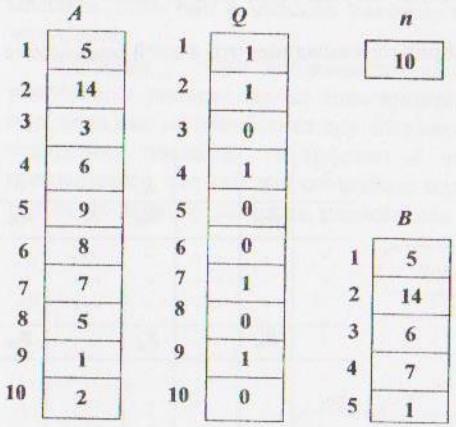

Рис. 12

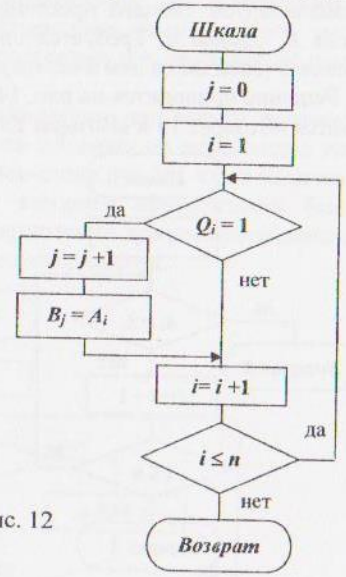

Рис.13

Массив Q за его специфическую роль часто называют логической шкалой или маской. Блок-схема решения этой задачи приведена на рис.13. Длиной результирующего массива B (рис.12) будет последнее по порядку значение индекса j.

Упражнения

5. Запишите в обратном порядке элементы массива A длины n.
6. В квадратной таблице размером $n \times n$ поменяйте местами строки и столбцы (транспонирование матрицы).
7. Из массива A длины n получите массив, в котором каждый элемент массива A повторяется k раз подряд. Рассмотрите два варианта:
   а) результирующий массив записывается отдельно (массив B);
   б) результирующий массив пишется на месте массива A, начиная с его первого элемента.
8. Осуществите сжатие массива A длины n, удалив из него все нулевые элементы.

9. Осуществите сжатие массива A длины n, удалив из него каждый k-й элемент.
10. В квадратной таблице размером $n \times n$ поменяйте местами i-ю и j-ю строки, а затем i-й и j-й столбцы.

МАССИВЫ И СПРАВОЧНАЯ СЛУЖБА

Рассмотрим теперь алгоритмы, обслуживающие массивы информации. Проанализируем сначала простейшую задачу о поиске. Даны число L и массив A длины n. Требуется определить порядковый номер этого числа в массиве A (если оно в нем имеется).

Решение приводится на рис. 14. Обратите внимание, что в этой блок-схеме имеются «Возврат 1» и «Возврат 2».

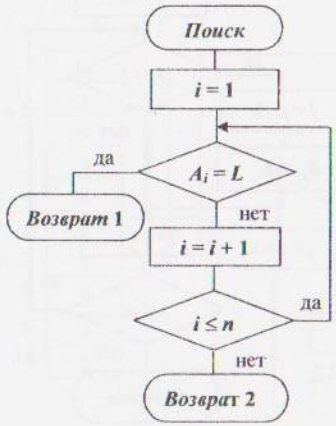

Рис.14

i \ j	1	...	j_0	...	n
1	B_{11}	...	B_{1j}	...	B_{1n}
.	.	...	.	...	.
.	.	...	.	...	.
.	.	...	.	...	.
i	B_{i1}	...	B_{ij}	...	B_{in}
.	.	...	.	...	.
.	.	...	.	...	.
.	.	...	.	...	.
m	B_{m1}	...	B_{mj}	...	B_{mn}

Рис.15

При решении задачи возможны две ситуации.

1). Число L в массиве A существует хотя бы в одном экземпляре. Тогда окончательное значение i указывает на первый по порядку номер заданного числа, т.к. перебор элементов прерывается до достижения конца массива.

2). В массиве A нет ни одного числа, равного L. В этом случае в конце работы алгоритма $i = n+1$, и может быть выведено сообщение об отсутствии объекта поиска.

Решение этой задачи имеет естественное расширение. Рассмотрим таблицу B размером $m \times n$, т.е. двумерный массив B, содержащий m строк и n столбцов (рис. 15). Пусть каждая строка таблицы содержит информацию об одном объекте из некоторой совокупности однородных объектов (например, таблица **B** - это табель ученика, а ее строка – это оценки за четыре четверти по одному предмету). Тогда первым элементом строки должно быть ее ключевое слово (ключ), т.е. условное число, отличающее ее от других строк (например, ключ строки табеля ученика – это шифр изучаемой дисциплины).

Для того, чтобы прочитать содержание строки с заданным ключом, ее сначала надо найти в таблице **B** или убедиться, что там ее нет. Алгоритм поиска подобен изображенному на рис.14, только проверка условия $A_i = L$ заменяется проверкой $B_{i1} = L$. После найденного номера строки i можно сразу обращаться к любому элементу B_{ij}. Поиск можно сделать намного быстрее, если предварительно элементы массива будут упорядочены.

Пусть дан массив A длины n, упорядоченный по возрастанию его элементов. Дано произвольное число L. Определим наличие этого числа в массиве. Если оно в массиве имеется, то определим его порядковый номер в массиве A.

Решение 1. Оно основывается на соображении, что вместо проверки на равенство в данном случае надо производить проверку на неравенство до тех пор, пока оно не приобретет другой знак (блок-схема рис.16). Факт наличия или отсутствия числа L в массиве A этот алгоритм обнаруживает быстрее предыдущего, так как при его работе перебираются не все элементы массива, а только до первого элемента, равного или превосходящего L.

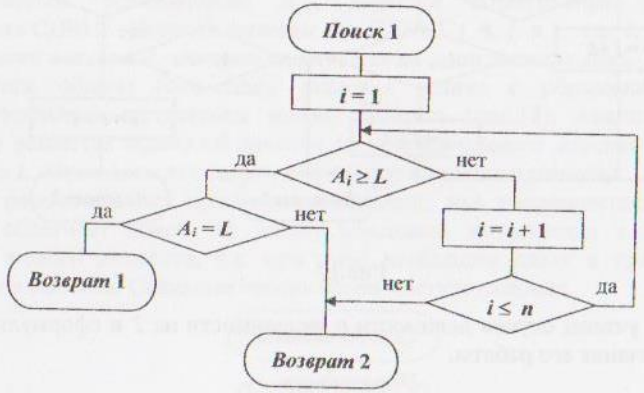

Рис 16.

Однако существует значительно более быстрый алгоритм поиска по упорядоченному массиву.

Решение 2. Идея быстрого поиска состоит в следующем. Прежде всего выясним, больше ли число L числа, состоящего в середине массива. Если да, то повторим этот же вопрос для середины нижней половины массива, а если нет, то верхней. Так последовательно сужая область поиска делением пополам, мы в конце концов найдем искомое число (а значит, и его номер) или убедимся в его отсутствии.

Для массива длины n этот способ потребует не более чем $\log_2 n+1$ сравнений, а значит, и повторов цикла, тогда как прежнее решение в среднем требует $n/2$ сравнений, а в самом худшем случае (когда L больше самого большого числа в массиве) – даже n сравнений. Функция $\log_2 n$, конечно, является неограниченно возрастающей функцией от n, но растет она настолько медленно, что на практике этим ростом можно пренебречь (сравните $n = 1000$ $\log_2 n < 10$,

$n = 1000000$ $\log_2 n < 20$). Это оправдывает название « быстрый поиск ». Алгоритм, оснований на идее быстрого поиска (рис.17), сложнее предыдущего. Основным объектом его обработки является суживающийся фрагмент массива A. Введем следующие обозначения: i - начало фрагмента, j – конец фрагмента, k – полудлина фрагмента.

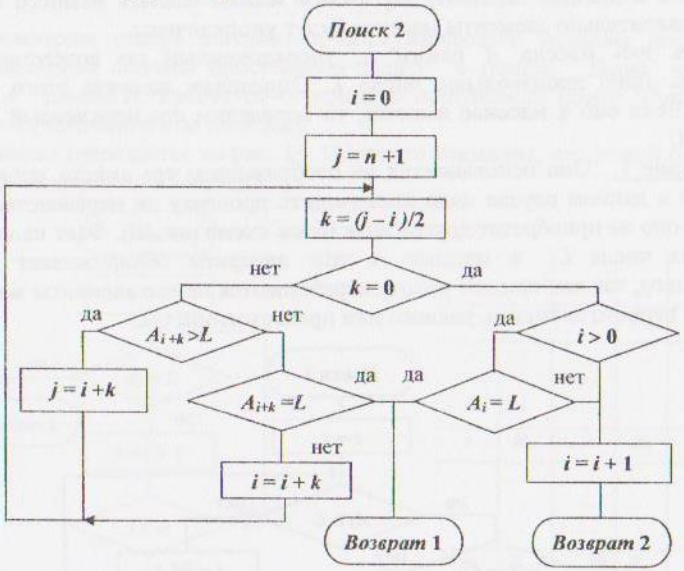

Рис.17

В алгоритме учтены случаи делимости и неделимости на 2 и сформулированы условия окончания его работы.

Упражнения

11. Выберите все простые числа из ряда n первых натуральных чисел. Для этого последовательно замените на нуль все числа, делящиеся на 2, потом – на 3, на 4 и т.д. Сообразите, до какого именно делителя такие действия надо производить. Затем сожмите массив с удалением из него всех нулей. Этот способ был известен еще в древней Греции и носит название решето Эратосфена.
12. В массиве A длины n из всех одинаковых элементов оставьте только по одному представителю. Остальные удалите, сжав при этом массив, а в места, оставшиеся за его новым концом, внесите нули.
13. Выполните предыдущее упражнение в предположении, что массив упорядочен по возрастанию своих элементов.

ВНЕСЕНИЕ ИЗМЕНЕНИЙ В МАССИВ

Наряду с поиском при работе с массивами часто возникают задачи, связанные с внесением изменений в массив. Пусть массив A длины n упорядочен по возрастанию своих элементов. Пусть дано число L. Требуется вставить число L в массив так, чтобы упорядоченность последнего не нарушалась.

Ясно, что для решения этой задачи надо последовательно выполнить следующие этапы:

1) найти в массиве A первое по порядку число, большее или равное L;
2) начиная с этого числа, сдвинуть конец массива вниз на один элемент;
3) записать число L на освободившееся место;
4) скорректировать длину нового массива, т.е. увеличить ее на единицу.

Для реализации п.1 достаточно воспользоваться алгоритмом поиска. Назовем его коротко ПОИСК (A, n, L, i), где n – заданная длина массива A, i – искомый номер первого по порядку элемента массива A, большего или равного L.

Алгоритм, реализующий п.2, является естественным обобщением алгоритма СДВИГ. Коротко назовем его СДВИГ (A, i, n), где i, n – начало и конец части массива A, которые сдвигаются на один элемент вниз.

Теперь общую блок-схему решения задачи с обращением к двум вспомогательным алгоритмам можно записать (рис.18). Аналогично этому примеру решается задача об изъятии из упорядоченного массива A заданного элемента L. Оставляем эту задачу для самостоятельного решения.

Анализ рассмотренных примеров показывает, что упорядоченность массива сильно облегчает поиск, но ставит серьезные затруднения в организацию вставки нового элемента, т.к. при этом необходим сдвиг в среднем на n/2 элементов массива. Создадим теперь более жесткие условия.

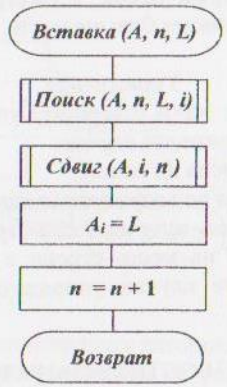

Рис. 18

Предположим, что элементы массива - это редкие книги, которые не рекомендуется трогать руками без крайней необходимости. Каждая книга на своем корешке имеет шифр – многозначное условное число, в котором комбинацией цифр закодированы автор и название книги. Книги стоят на полках в специальных пронумерованных ячейках. Если книга нужна читателю, то с нее снимается фотокопия, а сама книга на руки не выдается и остается на своем месте. Каждая новая книга ставится в конец полки. Требуется организовать эффективную систему поиска и занесения книг в картотеку этой библиотеки.

Решение. Если бы не было новых поступлений книг, то библиотека, расположенная по возрастанию шифров книг, полностью решала бы задачу поиска. Описанный выше алгоритм вставки здесь уже не годится, поскольку при каждом новом поступлении книги концы картотеки (массива) постоянно движутся. Для эффективного решения этой задачи воспользуемся принципом библиотечного каталога, где на каждую книгу заводится карточка, указывающая два числа: шифр книги и порядковый номер книги на полке. Карточки будем хранить в специальной картотеке, упорядоченной по шифру книг. Рассмотрим, как в этой системе будут выполняться две основные библиотечные операции.

а). Занесение новой книги: книга ставится в конец полки, на нее заводится карточка, в которой проставляется шифр книги и номер занятого места на полке. Затем карточка вставляется в картотеку, не нарушая упорядоченности последней по шифрам книг.

б). Поиск книги по ее шифру: реализуется алгоритм поиска по картотеке, а затем обращение к книге по номеру места, записанного в ее карточке.

В решении этой задачи самой медленной операцией является занесение новой книги. Существует более сложная организация каталога, которая позволяет организовать и быстрый поиск, и быстрое занесение, правда, сам каталог при этом сложнее и занимает больше места.

Упражнения

14. В таблице T размером $m \times n$ совершите удаление строки с заданным ключом L. Предусмотрите два случая:
    а) таблица не упорядочена по ключу;
    б) таблица упорядочена по возрастанию ключей.
15. Дана таблица T размером $m \times n$ и массив-строка S длиной n. Поместите строку S в таблицу T на место строки с тем же самым ключевым словом. Предусмотрите случаи упорядочности и неупорядочности таблицы T по ключу.

СРАВНЕНИЕ И ОПТИМИЗАЦИЯ МАССИВОВ

Начнем с простого примера. Из двух чисел a и b выбрать наибольшее и присвоить переменной c значение нуль, если $a < b$, и единицы – в противном

случае. Решение этого примера очевидно (рис.19). Его принципиальная роль в организации информационных массивов видна из следующего обобщения.

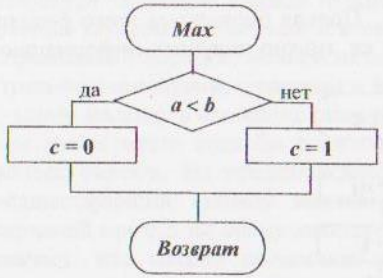

Рис.19

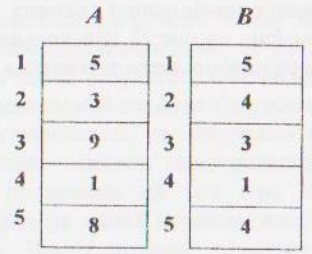

Рис. 20

Пусть A и B – два массива длины n. В каждом из них сверху вниз записано n – значное десятичное число (a и b), по одной цифре в ячейке. Алгоритм сравнения этих чисел должен переменной c присваивать значение 0 при $a < b$ и 1 – в противном случае. На рис. 20 изображены с конкретным числовым наполнением два массива длины 5. Так как 53918< 54314, то c =0. Общий алгоритм решения этой задачи приведен на рис. 21.

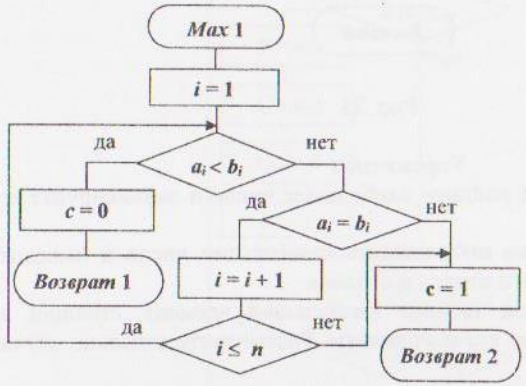

Рис. 21

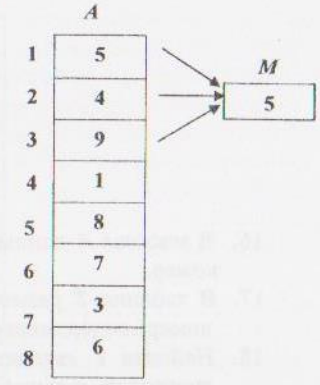

Рис. 22

Этот пример наглядно демонстрирует алгоритмическую природу десятичной системы счисления, позволяющую свести операции над числами к операциям над отдельными цифрами этих чисел.

В следующем примере из массива A длины n выберем наибольший элемент и запишем его в ячейку M. Решаем этот пример методом последовательного

испытания претендентов (рис.22). Сначала в ячейку запишем первый элемент массива A. Сравним с ним поочередно все элементы массива до тех пор, пока не попадется больший. Тогда его переписываем в ячейку M и сравнение по указанному принципу продолжаем. При достижении конца массива в ячейке M окажется наибольший элемент массива A. Полная блок-схема этого решения приведена на рис.23. Несколько усложнив ее, можно получить информацию о номере наибольшего элемента в массива.

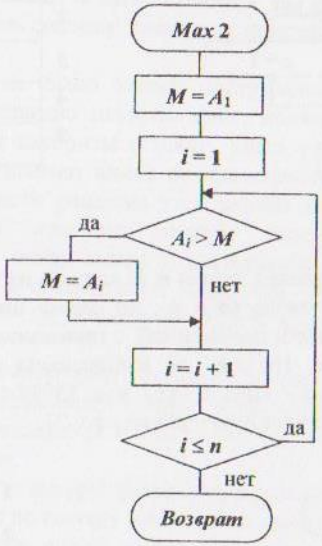

Рис. 23

Упражнения

16. В массиве A длины n найдите наибольшее число и зафиксируйте его номер.
17. В таблице T размером $m \times n$ найдите наименьшее число и выведите номера содержащих его строки и столбца.
18. Найдите в квадратной таблице наибольший элемент, стоящий на диагонали, и выведите его координаты. Рассмотрите отдельно случаи разных диагоналей.

УПОРЯДОЧЕНИЕ МАССИВОВ

Рассмотрим вопросы, связанные с получением полностью упорядоченных массивов. Упорядочим массив A длины n по убыванию всех его элементов. Простейшее решение, которое может быть здесь предложено, связано с последовательным применением алгоритма выбора наибольшего элемента – сначала из всего массива, потом – из оставшейся части, кроме уже выбранного,

и т.д. Рекомендуется самостоятельно построить алгоритм решения этой задачи, реализовав разобранную идею.

Далее, мы более подробно остановимся на другом способе, получившем в литературе название «метод пузырька». Идея метода состоит в следующем. Сравним по величине первый и второй элементы массива и, если между ними неправильный порядок, меняем их местами. Далее поступаем также со вторым и третьим элементами, с третьим и четвертым и т.д. В результате продвижения по всему массиву в его конце окажется самый маленький элемент («пузырек»). Если имела место хотя бы одна перестановка элементов, то повторяем весь процесс сначала. На предпоследнее место станет элемент, непосредственно предшествующий самому маленькому. Так поступаем до тех пор, пока очередной проход по всему массиву не потребует ни одной перестановки: это означает, что массив полностью упорядочен. Число полных проходов по массиву зависит от исходного расположения элементов, но не превышает числа n – длины массива. Блок-схема описанного процесса изображена на рис.24.

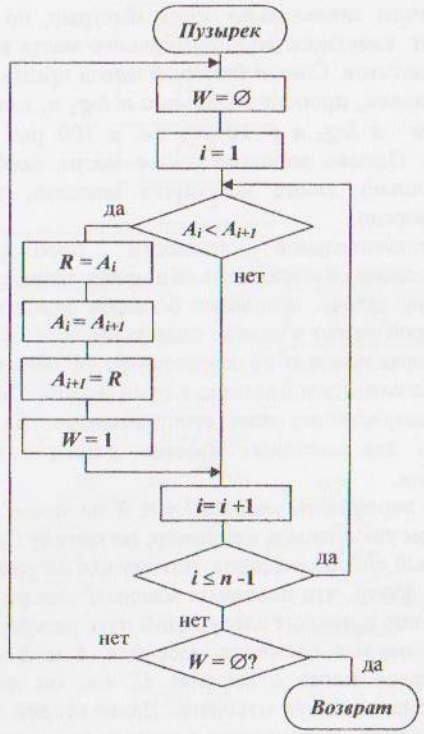

Рис. 24

Укажем две особенности этого алгоритма:

1) запоминание факта перестановки элементов обеспечивает переменная W ($W = 1$ – надо повторять проход; $W = 0$ – повторять не надо, массив полностью упорядочен);

2)условием правильной работы алгоритма служит условие $n \geq 2$.

Следует заметить, что метод последовательного выделения наибольшего элемента и метод пузырька относятся к числу медленных методов упорядочения элементов массива: общее число оборотов цикла, а, значит, и время работы пропорциональны квадрату длины массива (оценивая грубо, n прогонов по массиву и n сравнений и перестановок при каждом прогоне). Эта величина оказывается очень большой даже для не очень больших массивов (так, n^2=1000000 для n=1000). Однако несомненным достоинством этих методов, помимо простоты реализации, является их нетребовательность к дополнительному месту для записи промежуточных результатов: нужны лишь две рабочие ячейки для организации перестановки двух элементов и запоминания факта ее проведения.

Существуют методы значительно более быстрые, но они все в той или иной степени требуют заметного дополнительного места в памяти для записи промежуточных результатов. Самый быстрый метод предполагает общее число сравнений и перестановок, пропорциональное $n \log_2 n$, где n – длина массива (для $n = 1000$ имеем $n \log_2 n \approx 10000$, т.е. в 100 раз меньше, чем n^2 в предыдущем случае). Однако дополнительное место, необходимое для этого метода, пропорционально длине исходного массива, т.е. само по себе представляет целый массив.

В силу исключительной важности проблемы упорядочения информационных массивов ей посвящена обширная литература /1, 4/. Здесь мы рассмотрим лишь одну задачу, имеющую большое самостоятельное значение, способ решения которой лежит в основе самого быстрого упорядочения. Пусть даны две массива, упорядоченных по возрастанию их элементов: A длины n и B длины m. Необходимо слить эти массивы в один массив C общей длины $m + n$, упорядоченный по возрастанию всех его элементов. На рис.25 в качестве примера изображены два исходных массива длины 5 и 4 с конкретным числовым наполнением.

Можно было бы переписать массивы A и B на место массива C, а потом провести упорядочение последнего, например, по методу пузырька. Однако это крайне нерациональный способ решения, потому что он требует много времени и не использует того факта, что исходные массивы уже упорядочены. К более рациональному решению приводит следующий путь рассуждений.

Сравниваем два первых элемента массивов A и B и меньший из них переписываем на первое место в массиве C, т.к. он наименьший из всех исходных данных элементов двух массивов. Далее второй элемент массива, из которого взят наименьший, сравнивается с первым элементом другого массива. Меньший из них переписывается уже на второе место массива C, при этом в массиве, из которого он взят, для дальнейшего сравнения берется следующий элемент, а в соседнем массиве – тот же самый. Процесс повторяется до тех пор,

	A
1	3
2	7
3	8
4	10
5	15

	B
1	4
2	6
3	8
4	16

	C	
1	3	(A)
2	4	(B)
3	6	(B)
4	7	(A)
5	8	(B)
6	8	(A)
7	10	(A)
8	15	(A)
9	16	(B)

Рис. 25

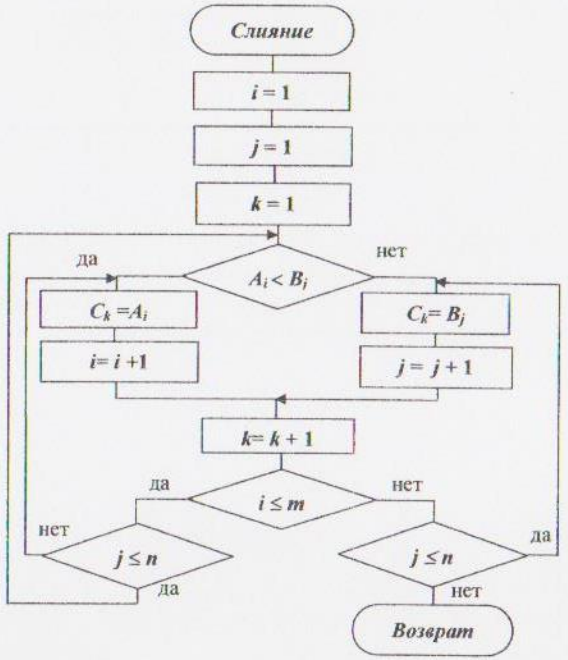

Рис. 26

пока не будут полностью переписаны все элементы хотя бы из одного массива, тогда оставшиеся элементы другого массива просто переписываются в конец результирующего массива. Блок-схема алгоритма приведена на рис.26. Общее

число повторений цикла этого алгоритма равно $m+n$, т.е. длине результирующего массива. Это значит, что более быстрого алгоритма в принципе составить невозможно.

Упражнение

19. Составить блок-схему алгоритма упорядочения массива A длины n способом последовательного выбора наибольших элементов.
20. Пусть A и B – два массива, упорядоченных по возрастанию и имеющих длины m и n. «Слейте» эти массивы в упорядоченный массив с записью последнего на месте массива A («влейте» массив B в массив A).

БИБЛИОГРАФИЧЕСКИЙ СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1. Информатика [Текст]: учебник: рек. МО РФ /под ред. Н.В. Макаровой. – 3-е изд., перераб. – М., Финансы и статистика, 2005.- 765с.

2. Информатика: практикум по технологии работы на компьютере. Учеб. пособие /Макаров Н.В. и др./ под ред. Макаровой Н.В. Изд. 3-е, финансы и статистика, 2005 – 255с.

3. Информатика и информационные технологии [Текст]: учеб. пособие /И.Г. Лесничная, И.В. Миссинг, Ю.Д. Романова, В.И. Шестаков. –М.:Эксмо, 2005. -542с.

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

ОГЛАВЛЕНИЕ

Работа с информационными массивами

Методические указания к индивидуальным занятиям
по курсу «Информатика» для студентов
1-го курса всех специальностей

Составители: д.т.н., проф. Виктор Петрович Авдеев,
к.т.н., доц. Владимир Исламович Гильмутдинов,
к.ф.-м.н., доц. Александр Давыдович Кононов,
к.т.н., доц. Кононов Андрей Александрович

Подписано в печать 25.06.2007. Формат 60×84 1/16. Уч.-изд. л. 1,2.
Усл. – печ. л. 1,3. Бумага писчая. Тираж 200 экз. Заказ № 360.

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