Федеральное агентство по образованию

Государственное образовательное учреждение
высшего профессионального образования

Воронежский государственный архитектурно-строительный университет

В.И. Гильмутдинов, А.Д. Кононов, А.А. Кононов

ИНФОРМАТИКА

Учебное пособие
для самостоятельной работы
студентов всех специальностей

Воронеж 2010

УДК 004.9 (07)
ББК 32.81 я 7
Г474

Рецензенты:
кафедра информационных систем
Воронежского государственного университета;
С.А.Чепелев, д. т. н., профессор Воронежской государственной лесотехнической академии

Гильмутдинов, В.И.
Г474 **Информатика** : учеб. пособие / В.И. Гильмутдинов, А.Д. Кононов, А.А. Кононов; Воронеж. гос. арх.-строит. ун-т. – Воронеж, 2010. – 56 с.

ISBN 978-5-89040-270-7

Рассматриваются вычислительные процедуры в ЭВМ, элементы булевой алгебры и алгебры контактных схем, функциональные элементы арифметико-логических устройств ЭВМ и их характеристики, операции на многоразрядном сумматоре, схемы типовых алгоритмов обработки информационных массивов. Приводятся примеры, после подробного разбора которых предлагаются упражнения для проверки усвоения студентами изучаемого материала. Учебное пособие предназначено для самостоятельной работы при изучении дисциплины «Информатика» студентами первого курса всех специальностей.

Ил. 50. Табл. 17. Библиогр.: 7 назв.

УДК 004.9 (07)
ББК 32.81 я 7

ISBN 978-5-89040-270-7

ВВЕДЕНИЕ

Успешная самостоятельная работа современного специалиста требует усиления подготовки в области использования вычислительной техники. В данной методической разработке рассматриваются вычислительные процедуры в ЭВМ, их арифметические основы, формы представления чисел, элементы булевой алгебры и алгебры контактных схем, функциональные элементы арифметико-логических устройств ЭВМ и их характеристики, операции на многоразрядном сумматоре.

Анализируются 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.

Учебное пособие предназначено для самостоятельной работы по программированию в рамках дисциплины «Информатика» для студентов всех специальностей.

Разделы 1, 2, 3 настоящего пособия соответствуют материалу первого семестра типовой программы по дисциплине «Информатика», а раздел 4 полезен для закрепления навыков программирования в рамках второго семестра изучения данной дисциплины.

Приведенные в тексте пособия 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 обсуждаются полученные результаты.

1. АРИФМЕТИЧЕСКИЕ ОСНОВЫ ЭВМ

Программа и числовая информация представляются в ЭВМ в двоичной системе счисления. Для записи команд в коде машины используются, кроме того, восьмеричная и шестнадцатеричная системы счисления. Настоящий раздел посвящается рассмотрению свойств и алгоритмов действий тех систем счисления, которые составляют арифметические основы современных ЭВМ.

1.1. Системы счисления и их свойства

Понятия «число» и «операции над числами» возникли из потребностей практической деятельности людей. На протяжении тысячелетий формы записи чисел претерпели большие изменения. Сначала многие народы имели свои отдельные системы записи чисел, каждая из которых отражала в той или иной степени тот факт, что число – простое количество объектов. Эту характерную особенность сохранила дошедшая до нас римская система счисления. В римской системе счисления нелегко проводить арифметические действия, очень сложно записывать большие числа. Последнее обстоятельство связано с тем

фактом, что число знаков (цифр), используемых для представления чисел в римской системе, ограничено и можно придумать такое большое число, что введенных знаков для изображения этого числа не будет хватать и потребуется дополнение основной символики. Системы счисления типа римской часто называют аддитивными (additio – сложение). Например, LXVI = 50+10+5+1=66.

Революционным моментом в совершенствовании записи чисел явилось создание десятичной системы счисления, в которой любое число представляется в виде последовательности цифр. Вклад каждой цифры в общее значение числа зависит не только от значения цифры, но и от места (позиции), которое занимает эта цифра среди других цифр числа. Отсюда происхождение термина «позиционная система счисления».

Позиционная система счисления потребовала введения в рассмотрение цифры (и числа) нуль! В римской системе счисления цифра нуль отсутствует. Появление десятичных дробей логически завершило построение десятичной системы счисления.

Отметим две важные особенности десятичной позиционной системы счисления, которые, впрочем, справедливы и для любой другой позиционной системы счисления.

Во-первых, краткость записи чисел по сравнению с непозиционными системами счисления. Например, в десятичной системе счисления, чтобы записать число k, необходимо не более $lgk+1$ цифр (позиций).

Во-вторых, возможность формального сведения арифметических действий над числами к действиям над их отдельными цифрами. Иногда мы просто не в состоянии прочитать слишком большие числа, но всегда можем получить их сумму, разность, произведение и частное, действуя по известным правилам. Эти правила - алгоритмы - носят общий характер, не зависящий от конкретного вида чисел и от самой системы счисления. Алгоритмы арифметических действий демонстрируют удивительную приспособленность позиционных систем счисления к арифметическим действиям в отличие от непозиционных систем. Эти алгоритмы оказались настолько «механическими», что их удается без особых затруднений передать для выполнения вычислительным устройствам. В частности, еще задолго до появления ЭВМ широкое применение в вычислительной практике получили арифмометры. Главная часть арифмометра – суммирующее устройство, а в суммирующем устройстве основное – это механизм сквозного переноса десятков. В устройстве арифмометра удалось все арифметические операции свести к выполнению действия сложения на том же суммирующем устройстве.

Рассмотрим свойства десятичной позиционной системы счисления. В любой позиционной системе для записи произвольного числа используется строго определенное количество цифр. В десятичной системе, как известно, введено десять цифр: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

В десятичной системе в каждой единице следующего (более старшего) разряда числа содержится десять единиц предыдущего разряда. Это отношение значений единиц соседних разрядов является основанием позиционной системы счисления.

В десятичной системе нуль и первые девять натуральных чисел обозначаются соответствующими цифрами. Всякое целое число N (например, четырехзначное) изображается в виде

$$N = a \cdot 10^3 + b \cdot 10^2 + c \cdot 10^1 + d \cdot 10^0, \qquad (1.1)$$

где a, b, c, d – есть цифры десятичной системы счисления.

Сокращенное число N изображается строкой $\overline{abcd}$. Коэффициенты a, b, c, d есть остатки последовательного деления числа N на основание десятичной системы счисления число **10**. Так, для числа **421** имеем

421 |	10	
40	42 |	10
21	40	4
20	2	
1		

Запись целого числа N в любой позиционной системе счисления с основанием n (будем указывать основание индексом справа внизу) обозначает представление этого числа в виде суммы степеней основания данной системы n с различными коэффициентами, меньшими n. Эти коэффициенты и являются цифрами в записи числа.

Докажем, что любое целое число N может быть записано в системе счисления с произвольным основанием n.

Найдем наибольшую степень среди последовательных степеней основания n: $n^0, n^1, n^2, \ldots, n^m$, содержащуюся в N, то есть такое m, чтобы $n^m \le N$, но $n^{m+1} > N$. Это можно сделать всегда. Разделим N на n^m. Если при этом частное равно a_m и остаток N_m, то

$$N = a_m n^m + N_m \ . \qquad (1.2)$$

Заметим, что $a_m < n$ и $N_m < n^m$. Полученный остаток N_m разделим на следующую степень основания n, т.е. n^{m-1}.

$$N_m = a_{m-1} n^{m-1} + N_{m-1} \ . \qquad (1.3)$$

Поставляя (1.3) в (1.2), получим

$$N = a_m n^m + a_{m-1} n^{m-1} + N_{m-1}, \qquad (1.4)$$

где $a_{m-1} < n$, $N_{m-1} < n^{m-1}$.

Будем продолжать делить получаемые остатки N_{m-1}, N_{m-2} …. на последующие степени основания n, т.е. n^{m-2}, n^{m-3}…, пока не получим

$$N_2 = a_1 n + N_1 \ . \qquad (1.5)$$

Если теперь соединить (1.4) и (1.5), то будем иметь

$$N = a_m n^m + a_{m-1} n^{m-1} + \ldots + a_1 n + a. \qquad (1.6)$$

Таким образом, мы представили число **N** как сумму степеней числа ***n*** с различными коэффициентами, меньшими числа ***n***, что и является доказательством возможности записи числа ***N*** в системе с основанием ***n*** в виде

$$N = \overline{a_m a_{m-1} a_{m-2} \ldots a_1 a_0}.$$

До сих пор речь шла о целых числах. Под десятичной дробью понимают представление дробного числа в виде суммы степеней ***10*** с отрицательными показателями. Так, запись $0{,}218_{10}$ означает:

$$2\cdot 10^{-1} + 1\cdot 10^{-2} + 8\cdot 10^{-3},$$

а запись $0{,}326_{8}$ означает число:

$$3\cdot 8^{-1} + 2\cdot 8^{-2} + 6\cdot 8^{-3};$$

эта дробь называется восьмеричной дробью.

Таким образом, в системе счисления с основанием ***n*** любое число ***N*** может быть записано в виде

$$N = \overline{(a_m a_{m-1} a_{m-2} \ldots a_0, a_{-1} a_{-2} \ldots a_{-p})_n}\ , \qquad (1.7)$$

где $a_m a_{m-1} \ldots a_0$ - целая часть числа,

$a_{-1} a_{-2} \ldots a_{-p}$ - дробная часть,

$a_m, a_{m-1}, \ldots, a_{-p}$ - цифры из набора ***0, 1, … , n – 1***.

Запись (1.7) изображает число

$$a_m n^m + a_{m-1} n^{m-1} + \ldots + a_0 n^0 + a_{-1} n^{-1} + \ldots + a_{-p} n^{-p}. \qquad (1.8)$$

Арифметические действия с многозначными числами (целыми и дробными) базируются на алгоритмах сложения и умножения однородных чисел и сводятся к переносу в случае переполнения из младших разрядов в старшие (и из старших разрядов в младшие – в случае недостатка при вычитании). Такая простота и удобство выполнения арифметических действий обусловлена изложенным выше принципом записи чисел в позиционной системе в виде строки цифр. Особенно наглядно указанная простота познается в сравнении: попробуйте перемножить два числа LXXVII и LXIX, записанных в римской (непозиционной) системе счисления.

У п р а ж н е н и я

1. Двухзначное десятичное число в сумме с числом, записанным теми же цифрами, но в обратном порядке, дает полный квадрат. Найдите все такие числа.
2. Трехзначное десятичное число оканчивается цифрой 3. Если эту цифру переместить через два знака влево, т.е. с этой цифры будет начинаться запись нового числа, то это новое число будет на единицу больше утроенного исходного числа. Найдите это число.

1.2. Восьмеричная и шестнадцатеричная системы счисления

Восьмеричной системой счисления называется позиционная система с основанием восемь. Для записи чисел в этой системе используется набор из восьми цифр 0, 1, 2, 3, 4, 5, 6, 7. Число восемь в восьмеричной системе изображается как 10_8.

Аналогично записи целого числа в десятичной позиционной системе счисления в восьмеричной системе число представляется в виде

$$\overline{abcd}_8 = a \cdot 8^3 + b \cdot 8^2 + c \cdot 8^1 + d \cdot 8^0, \qquad (1.9)$$

где ***a, b, c, d*** меньше ***8***.

Форма записи восьмеричного числа (1.9) подсказывает правила перевода значения числа из восьмеричной системы в десятичную: необходимо вычислить сумму последовательных степеней основания 8 в правой части записи (1.9).

Пример. Рассмотрим восьмеричное число 645_8. Распишем его по формуле (1.9):

$$645_8 = 6 \cdot 8^2 + 4 \cdot 8^1 + 5 \cdot 8^0 = 421_{10}.$$

Следовательно, число 421_{10} есть десятичный вид восьмеричного числа 645_8. Сделаем обратный перевод числа 421_{10} в восьмеричную систему, т.е. мы должны число 421_{10} представить в виде:

$$\overline{abc}_8 = a \cdot 8^2 + b \cdot 8^1 + c \cdot 8^0.$$

Выпишем последовательные степени основания ***8***: $8^0 = 1$, $8^1 = 8$, $8^2 = 64$, $8^3 = 512$. Число 421_{10} меньше 512, но больше 64. Значит, 8^2 – самая высокая степень ***8*** в числе 421_{10}. Разделим число ***42***1 на 8^2:

$$421_{10} = 6 \cdot 8^2 + 37. \qquad (1.10)$$

Полученный остаток разделим на следующую степень 8^1:

$$37_{10} = 4 \cdot 8 + 5. \qquad (1.11)$$

Собирая вместе (1.10) и (1.11), получим сумму степеней числа ***8*** с коэффициентами, меньше ***8***, которые и являются собственно цифрами в записи числа 421_{10} в восьмеричной системе счисления.

$$421_{10} = 6 \cdot 8^2 + 4 \cdot 8^1 + 5 \cdot 8^0 = 645_8.$$

В практике вычислений, однако, используется более рациональный способ перевода. Рассмотрим его на примере. Снова переводим число 421_{10} в восьмеричную систему счисления.

Разделим 421_{10} на ***8***: $421_{10} = 52 \cdot 8 + 5$.

Получили частное ***52*** и остаток ***5***. Остаток ***5*** показывает, что кроме ***52*** восьмерок число содержит еще ***5***, т.е. является последней цифрой восьмеричной записи данного числа (***c = 5***).

Разделив ***52*** на ***8***, узнаем, сколько в числе 52_{10} содержится ***8*** (частное):
$52_{10} = 6 \cdot 8 + 4$. Следовательно, ***a = 6***, ***b = 4***.
Окончательно:

$$421_{10} = 6\cdot 8^2 + 4\cdot 8^1 + 5\cdot 8^0 = 645_8 .$$

Описанный здесь переход удобно выполнять по следующей схеме:

$$\begin{array}{r|l} 421_{10} & 8 \\ \hline 40 & 52 \\ \hline 21 & 48 \\ \hline 16 & 4 \\ \hline 5 & \end{array} \qquad \begin{array}{r|l} 8 & \\ \hline 6 & \end{array} \qquad \rightarrow \; 645_8 .$$

Деление продолжается до тех пор, пока не получено частное, меньшее ***8***. Это частное представляет цифру, стоящую в самом старшем разряде. Остальные цифры получаются при записи остатков справа налево.

Убедимся теперь, что рассмотренный прием перевода чисел из одной системы счисления в другую является общим.

Пусть N - целое число, записанное в системе счисления с основанием a. Необходимо перевести его в систему с основанием n. Сначала разделим N на основание новой системы n (деление производим в системе счисления с основанием a):

$$N = b_0 \cdot n + a_0 . \qquad (1.12)$$

Далее разделим полученное частное b_0 на n:

$$b_0 = b_1 \cdot n + a_1 . \qquad (1.13)$$

Следующее частное b_1 снова делим на n:

$$b_1 = b_2 \cdot n + a_2 \qquad (1.14)$$

и т.д. Это последовательное деление продолжаем до тех пор, пока не получится частное, меньшее n – основания системы, в которую переводится число. Это последнее частное обозначим через a_m, т.е.

$$b_{m-2} = b_{m-1} n + a_{m-1} = a_m n + a_{m-1} . \qquad (1.15)$$

Подставив значение b_0 из (1.13) в (1.12), получим

$$N = (b_1 n + a_1)\, n + a_0 = b_1 n^2 + a_1 n + a_0 . \qquad (1.16)$$

В (1.16) подставим значение b_1 из (1.14):

$$N = ((b_2 n + a_2)\, n + a_1)\, n + a_0 = b_2 n^3 + a_2 n^2 + a_1 n^1 + a_0 . \qquad (1.17)$$

Продолжаем эти подстановки последующих значений b_2, b_3 … в (1.17) до тех пор, пока не дойдем до b_{m-2}, которое возьмем из (1.15). Тогда получим

$$N = a_m n^m + a_{m-1} n^{m-1} + \ldots + a_3 n^3 + a_2 n^2 + a_1 n^1 + a_0 . \qquad (1.18)$$

Цифрами, представляющими число N в системе с основанием n, будут являться остатки, получающиеся при последовательном делении N на n и записанные в обратном порядке. Обычно этот прием перевода называют *алгоритмом последовательного деления*.

Если $n < a$, то делитель и все остатки окажутся однозначными числами, т.е. сразу получаются цифры в системе с основанием n . Примером этого может служить перевод из десятичной системы в восьмеричную.

Если $n > a$, то делитель и некоторые остатки нельзя представить в системе с основанием a однозначным числом. Поэтому их следует заменить новыми цифрами, которых, естественно, нет в системе с основанием n. В качестве такого примера рассмотрим перевод из десятичной системы в шестнадцатеричную систему счисления. Для записи чисел в ней используется шестнадцать цифр – десять обычных десятичных цифр и еще шесть дополнительных обозначений в виде заглавных латинских букв: A, B, C, D, E, F.

Основание шестнадцатеричной системы счисления обозначается как 10_{16}. Так же, как и в десятичной и восьмеричной системах счисления, целое четырехзначное число в шестнадцатеричной системе расписывается в виде строки

$$\overline{abcd} = a \cdot 16^3 + d \cdot 16^2 + c \cdot 16^1 + d \cdot 16^0, \qquad (1.19)$$

где a, b, c, d меньше 16.

Рассмотрим перевод из десятичной системы в шестнадцатеричную.

Пример. Переведем число 421_{10} в шестнадцатеричную систему. Применяем алгоритм последовательного деления:

$$\begin{array}{r|l} 421_{10} & 16 \\ \hline 32 & 26 \\ \hline 101 & 16 \\ 96 & \\ \hline 5 & \end{array} \qquad \begin{array}{r|l} 16 & \\ \hline 1 & \end{array} \qquad \rightarrow 1A5.$$

В нашем примере получен один остаток, равный 10_{10}. Это значение цифры в шестнадцатеричном числе, поэтому его надо представить в виде A. Обратный перевод производится простым суммированием строки (1.19).

Пример. Переведем число $A7F5$ в десятичную систему:

$$\overline{A7F5} = A \cdot 16^3 + 7 \cdot 16^2 + F \cdot 16^1 + 5 \cdot 16^0 =$$

$$= 10 \cdot 16^3 + 7 \cdot 16^2 + 15 \cdot 16^1 + 5 \cdot 16^0 =$$

$$= 10 \cdot 4096 + 7 \cdot 256 + 15 \cdot 16 + 5 = 42997_{10}.$$

При переводе дробной части числа её надо умножать на новое основание и последовательно выписывать целые части произведений.

Пример. Перевести число $0,6_{10}$ в восьмеричную систему счисления

$$0,6_{10} = 0,(4631)_8$$

$$\begin{array}{r|l} 0,6 & \\ \hline 4,8 & 8 \\ \hline 6,4 & 8 \\ \hline 3,2 & 8 \\ \hline 1,6 & 8 \\ \hline 4,8 & 8 \\ \hline 6,4 & \end{array}$$

и т.д.

Обратный переход:

$$0,4631_8 \approx 4 \cdot 8^{-1} + 6 \cdot 8^{-2} + 3 \cdot 8^{-3} + 1 \cdot 8^{-4} + \ldots = 1/2 + 3/32 + \ldots \approx 0,6_{10}.$$

Все вышеизложенное позволяет кратко сформулировать правила перевода чисел.

1. При переводе ***целого числа*** из системы счисления с основанием ***a*** в систему счисления с основанием ***n*** необходимо число в системе счисления с основанием ***a*** разделить на основание ***n***. Полученный остаток является младшей цифрой числа в новой системе с основанием ***n***. Деление продолжается до тех пор, пока частное не станет меньше делителя.

2. При переводе ***правильной дроби*** из системы счисления с основанием ***a*** в систему счисления с основанием ***n*** необходимо дробь в системе с основанием ***a*** умножить на основание ***n***. Полученная целая часть произведения является старшим дробным разрядом числа в системе счисления с основанием ***n***. Дробную часть уточняют до тех пор, пока не получат требуемое количество знаков после запятой.

У п р а ж н е н и я

1. Переведите следующие числа из десятичной системы счисления в восьмеричную и шестнадцатеричную системы: 149_{10}; 1023_{10}; 1500_{10}; $13,4_{10}$.
2. Переведите следующие восьмеричные числа в десятичную систему счисления: 225_8; 1777_8; 2734_8; 100_8; $0,345_8$; $43,08_8$; $31,7_8$.
3. Переведите следующие шестнадцатеричные числа в десятичную систему счисления: 100_{16}; $2ABC_{16}$; $67E_{16}$; $D305C_{16}$; $3A,C_{16}$.

1.3. Двоичная система счисления

Двоичной системой счисления называется позиционная система с основанием, равным двум. Двоичная система нашла широкое применение в ЭВМ. В ЭВМ любое двоичное число представляется с помощью элементов, имеющих два устойчивых состояния: одно из этих состояний принимается за единицу, другое – за нуль.

Представление двоичного числа в ЭВМ можно смоделировать с помощью наборов обыкновенных электрических лампочек, если условиться считать, что зажженная лампочка обозначает ***1***, погашенная ***0***. На рис. 1.1 изображен некий набор лампочек, на котором «записано» двоичное число $\boldsymbol{110100101}_{2}$.

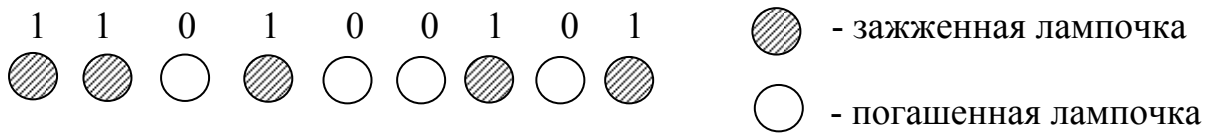


Рис. 1.1

Как обычно для позиционной системы, целое число (например, четырехзначное) может быть расшифровано по формуле

$$\overline{abcd}_{2} = a \cdot 2^{3} + b \cdot 2^{2} + c \cdot 2^{1} + d \cdot 2^{0}, \qquad (1.20)$$

где ***a, b, c, d*** – числа, меньшие ***2***.

Прочитаем, какое десятичное число изображено на рис. 1.1:

$$110100101_{2} = 1 \cdot 2^{8} + 1 \cdot 2^{7} + 0 \cdot 2^{6} + 1 \cdot 2^{5} + 0 \cdot 2^{4} + 0 \cdot 2^{3} + 1 \cdot 2^{2} + 0 \cdot 2^{1} + 1 \cdot 2^{0} = 421_{10}.$$

Таким образом, вычисление суммы последовательных степеней основания ***2*** позволяет перевести двоичное число в десятичную систему счисления.

Составим сравнительную таблицу (табл.1.1) десятичных, восьмеричных, шестнадцатеричных и двоичных чисел от 0 до 15. Эта таблица понадобится при переводе чисел из десятичной системы в двоичную.

Таблица 1.1

Десятичная система	Восьмеричная система	Шестнадцатеричная система	Двоичная система	Десятичная система	Восьмеричная система	Шестнадцатеричная система	Двоичная система
0	0	0	0000	8	10	8	1000
1	1	1	0001	9	11	9	1001
2	2	2	0010	10	12	A	1010
3	3	3	0011	11	13	B	1011
4	4	4	0100	12	14	C	1100
5	5	5	0101	13	15	D	1101
6	6	6	0110	14	16	E	1110
7	7	7	0111	15	17	F	1111

Первые восемь чисел сделаем четырехразрядными добавлением нулей слева. Полученные четырехзначные двоичные строки будем называть двоичными тетрадами.

Пример. Переведем число $\boldsymbol{421}_{10}$ в двоичную систему, воспользовавшись способом последовательного деления этого числа на основание двоичной сис-

темы. Выполняя такой перевод, целесообразно операции вычитания при последовательном делении подробно не записывать

$$
\begin{array}{r|l}
421_{10} & 2 \\
\hline
1 & 210 \\
 & \hline
 & 0 \quad 105 \\
 & \qquad\quad \hline
 & \qquad\quad 1 \quad 52 \\
 & \qquad\qquad\quad \hline
 & \qquad\qquad\quad 0 \quad 26 \\
 & \qquad\qquad\qquad\quad \hline
 & \qquad\qquad\qquad\quad 0 \quad 13 \\
 & \qquad\qquad\qquad\qquad\quad \hline
 & \qquad\qquad\qquad\qquad\quad 1 \quad 6 \\
 & \qquad\qquad\qquad\qquad\qquad\quad \hline
 & \qquad\qquad\qquad\qquad\qquad\quad 0 \quad 3 \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\quad \hline
 & \qquad\qquad\qquad\qquad\qquad\qquad\quad 1 \quad 1
\end{array}
\longrightarrow 110100101_2 .
$$

Этот громоздкий перевод из десятичной системы счисления в двоичную можно значительно упростить, если воспользоваться шестнадцатеричной системой счисления как промежуточной. Благодаря тому, что основание шестнадцатеричной системы счисления является степенью с целым показателем основания двоичной системы, т.е. $16 = 2^4$, перевод из шестнадцатеричной системы в двоичную осуществляется по очень простому правилу, заключающемуся в том, что в шестнадцатеричном числе все цифры надо заменить соответствующими им двоичными тетрадами из табл. 1.1.

Пример. Переведем шестнадцатеричное число **C503D** в двоичную систему. Заменим все шестнадцатеричные цифры этого числа на соответствующие двоичные тетрады:

C	5	0	3	D
1100	0101	0000	0011	1101

Теперь, собирая эти тетрады в последовательность цифр, имеем двоичное число 1100010100000011101_2.

Пример. Переведем двоичное число $11\ 1010\ 0111\ 1111_2$ в шестнадцатеричную систему. Заметим, что при разбиении этого числа на тетрады справа, возможно, одна тетрадь (последняя) получается неполной. Ее дополняют приписыванием слева незначащих нулей

0011	1010	0111	1111_2
3	A	7	F

Получаем число $3A7F_{16}$.

Упрощение перевода из десятичной системы в двоичную достигается, если сначала десятичное число переводится в шестнадцатеричную систему, а затем шестнадцатеричное число заменяется двоичным числом.

Пример. По упрощенному алгоритму перевести число 421_{10} в двоичную систему счисления.

$$
\begin{array}{r|l}
421 & 16 \\
\hline
5 & 26 \\
 & \hline
 & 10 \quad 1
\end{array}
\longrightarrow 1A5_{16} = 0001\ \ 1010\ \ 0101_2 = 110100101_2
$$

Естественно, что для упрощения перевода из десятичной системы в двоичную и обратно можно также использовать и восьмеричную систему счисления $(8 = 2^3)$. В этом случае вместо тетрад мы будем использовать двоичные триады, соответствующие первым восьми цифрам в табл.1.1.

Пример. Используя восьмеричную систему как промежуточную, переведем двоичное число $1011010101_{\,2}$ в десятичную систему. Разбиваем его на триады, начиная справа; последнюю неполную триаду дополняем до полной приписыванием нулей слева.

$$1011010101_{\,2} = 001\ \ 011\ \ 010\ \ 101_{\,2} = 1325_{\,8} - 1\cdot 8^{\,3} + 3\cdot 8^{\,2} + 2\cdot 8^{\,1} + 5\cdot 8^{\,0} =$$
$$= 512 + 192 + 16\ + 5 - 725_{\,10}.$$

Рассмотрим теперь особенности перевода двоичных дробей в десятичную систему счисления. Для этого необходимо вспомнить выражение (1.8) для записи дроби в системе счисления с основанием n.

Пример. Переведем дробь $0{,}101101_{\,2}$ в десятичную систему счисления, используя (1.8) при $n = 2$.

$$0{,}101101_{\,2} = 1\cdot 2^{\,-1} + 0\cdot 2^{\,-2} + 1\cdot 2^{\,-3} + 1\cdot 2^{\,-4} + 0\cdot 2^{\,-5} + 1\cdot 2^{\,-6} =$$
$$= \frac{1}{2} + \frac{1}{8} + \frac{1}{16} + \frac{1}{64} = \frac{32+8+4+1}{64} = \left(\frac{45}{64}\right)_{10}.$$

Если теперь мы воспользуемся восьмеричной системой счисления как промежуточной, то существенно сократим процесс перевода :

$$0{,}101101_2 = 0{,}55_8 = 5\cdot 8^{\,-1} + 5\cdot 8^{\,-2} = \frac{5}{8} + \frac{5}{64} = \left(\frac{45}{64}\right)_{10}.$$

Из этого примера, в частности, следует, что

$$0{,}1_2 = \left(\frac{1}{2}\right)_{10} = 0{,}5_{\,10}\,.$$

Именно это равенство положено в основу алгоритма перевода десятичных дробей в двоичную систему счисления. Для такого перевода необходимо выполнить следующие операции:

1. Определить, будет ли переводимое число (десятичная дробь) больше или меньше $0{,}5_{10}$. Это выясняется весьма просто: данная дробь удваивается, если она больше половины, то после умножения на 2 в целой части появляется 1; если меньше половины, то в целой части будет нуль. Содержимое целой части после этого умножения и дает первую цифру после запятой в записи двоичной дроби.

2. Остальные цифры в двоичной записи определяются аналогично. Отметим лишь, что в следующих умножениях на 2 участвуют только дробные части от удвоенного перед этим числа.

Пример. Переведем десятичную дробь $0{,}375_{10}$ в двоичную систему счисления:

0,	375
	2
0,	750
	2
1,	500
	2
1,	0

Собрав все цифры, полученные в целой части произведений, имеем

$$0{,}375_{10} = 0{,}011_2$$

Следует заметить, что при изменении основания системы счисления конечные дроби при одном основании могут не иметь конечного представления при другом основании.

Пример. При переводе конечной десятичной дроби $0{,}6_{10}$ в двоичную систему мы не получим ее конечного представления:

0,	6
	2
1,	2
	2
0,	4
	2
0,	8
	2
1,	6
	2
1,	2
	2
0,	4
	2
0,	8
	2
1,	6

В результате имеем:

$$0{,}6_{10} = 0{,}10011001\ldots_2 = 0{,}(1001)_2.$$

Как видим, из конечной десятичной дроби получена двоичная периодическая дробь.

У п р а ж н е н и я

1. Переведите следующие десятичные числа в двоичную систему счисления: $0{,}5$; $0{,}625$; $0{,}999$; $0{,}125$; $0{,}3$; $0{,}02$.
2. Переведите следующие двоичные числа в восьмеричную и шестнадцатеричную системы счисления: 101011_2 ; 111010011_2 ; $0{,}101_2$; $111{,}11_2$; $100{,}001_2$.
3. Переведите двоичные числа, указанные в предыдущем задании, в десятичную систему счисления, используя для такого перевода восьмеричную и шестнадцатеричную системы счисления.
4. Было 11 яблок. После того, как каждое из них разрезали пополам, стало 110 половинок. В какой системе счисления вели счет?

5. В какой системе счисления верны следующие равенства:
   а) ***11 : 110 = 0,1***; б) ***12 + 22 = 100***; в) ***70 – 1 = 68***;
   г) ***1- 0,1 = 0,1***; д) ***0,01 + 0,1 = 0,11*** ?
6. Переведите десятичную дробь ***0,9*** в шестнадцатеричную систему счисления, а затем - в двоичную систему. Сколько двоичных разрядов в записи двоичной дроби следует определить, чтобы при обратном переводе обеспечить точность приближения до ***0,005*** ?

1.4. Двоичная арифметика

Арифметические действия в двоичной системе производятся по обычным для позиционных систем алгоритмам.

Сложение многозначных двоичных чисел происходит по разрядам, начиная с младшего. Таблица двоичного сложения исключительно проста:

0 + 0 = 0 ***0 + 1 = 1*** ***1 + 0 = 1*** ***1 + 1 = 10***

Пример сложения двух многозначных двоичных чисел:

$$\begin{array}{r} 11110101 \\ +\ 1001101 \\ \hline 101000010 \end{array}$$

В отличие от умножения в десятичной системе счисления двоичное умножение отличается существенным своеобразием. Если десятичная таблица умножения состоит из 100 строк, то двоичная таблица умножения предельно удобна для запоминания:

0 · 0 = 0 ***0 · 1 = 0*** ***1 · 0 = 0*** ***1 ·1 = 1***

Эта удивительная простота двоичной таблицы умножения позволяет на самом деле умножение многозначных двоичных чисел заменить последовательным сложением со сдвигом множимого с самим собой. Величина сдвига определяется содержимым очередного разряда множителя. Если содержимое какого-либо разряда множителя равно ***0***, то это значит, что соответствующее произведение равно ***0***. При вычислениях это произведение не пишут, а сразу под предыдущим произведением подписывают следующее, сдвинутое влево сразу на 2 разряда.

Пример двоичного умножения:

$$\begin{array}{r} 11011 \\ \cdot\ 1101 \\ \hline 11011 \\ +\ 11011 \\ \hline 11011 \\ \hline 101011111 \end{array}$$

Видно, что процесс умножения сводится к последовательному сложению множимого с самим собой с определенным сдвигом. С точки зрения электроники моделирование такого процесса несложно. Именно последнее является од-

ним из веских аргументов привлекательности двоичной системы счисления для использования ее в ЭВМ.

Рассмотрим обратные арифметические действия – вычитание и деление – в двоичной системе счисления.

Пример.

$$\begin{array}{r} 11000011 \\ -\ 1101001 \\ \hline 1011010 \end{array} \qquad (1.21)$$

Сравните двоичное вычитание с десятичным вычитанием:

$$\begin{array}{r} 523 \\ -\ 87 \\ \hline 436 \end{array}$$

Известно, что при устном счете это же вычитание можно сделать проще. Если вычитаемое ***87*** заменить его десятичным дополнением ***100 – 87 = 13***, то тогда действие вычитания можно представить как сложение уменьшаемого с дополнением с последующим уменьшением старшего разряда (до которого было сделано дополнение) на ***1***.

1) $\begin{array}{r} 100 \\ -\ 87 \\ \hline 13 \end{array}$ 2) $\begin{array}{r} 523 \\ +\ 13 \\ \hline 536 \end{array}$ 3) $\begin{array}{r} 536 \\ -\ 1 \\ \hline 436 \end{array}$

В двоичной системе счисления вычитание можно тоже представить в виде сложения двоичного уменьшаемого с двоичным дополнением вычитаемого.

Двоичным дополнением положительного двоичного числа называют такое положительное двоичное число, которое при сложении с данным числом дает в сумме число с единственной ***1*** в следующем старшем разряде.

Пример. Число ***1100101*** имеет двоичное дополнение ***11011***, так как

$$\begin{array}{r} 1100101 \\ +\ 11011 \\ \hline 10000000 \end{array}$$

В этом случае говорят, что двоичное дополнение сделано до единицы в восьмом разряде.

Используя специфику двоичной арифметики, нахождение двоичного дополнения можно осуществить по более простому алгоритму. Оказывается, вполне достаточно в данном числе заменить все нули единицами, а единицы – нулями и к полученному таким образом числу прибавить ***1***:

данное число ***1100101***

$$\begin{array}{r} 0011010 \\ +\ 1 \\ \hline 11011 \end{array}$$

– его двоичное дополнение.

Итак, чтобы двоичное вычитание представить сложением, необходимо к уменьшаемому прибавить двоичное дополнение вычитаемого, а из полученной суммы вычесть единицу того разряда, до которого было сделано двоичное дополнение вычитаемого.

Пример. Повторим действие (1.21), следуя изложенному выше алгоритму.

1. Нахождение двоичного дополнения:

$$\begin{array}{r} 1101001 \\ 0010110 \\ +\quad\ 1 \\ \hline 10111 \end{array}$$

Двоичное дополнение вычитаемого сделано до ***1*** в восьмом разряде.

2. Сложение уменьшаемого с двоичным дополнением вычитаемого:

$$\begin{array}{r} 11000011 \\ +\quad 10111 \\ \hline 11011010 \end{array}$$

3. Вычитание ***1*** из восьмого разряда, т.е. разряда, до которого было сделано дополнение:

$$\begin{array}{r} 11011010 \\ -\ 1 \\ \hline 1011010 \end{array}$$

Полученный результат совпадает с ранее вычисленной разностью (1.21).

Двоичное деление рассмотрим на примере

$$\begin{array}{rl|l} 10111010111 & & 1101 \\ -\ 1101 & & 1110011 \\ \hline 10100 & & \\ -\ 1101 & & \\ \hline 1111 & & \\ -\ 1101 & & \\ \hline 10011 & & \\ -\ 1101 & & \\ \hline 1101 & & \\ -\ 1101 & & \\ \hline 0 & & \end{array}$$

Проверить полученный результат двоичного деления можно умножением частного на делитель.

У п р а ж н е н и я

1. Произведите сложение следующих десятичных чисел в двоичной системе счисления: 44_{10} и 81_{10}; 35_{10} и 104_{10}; 77_{10} и 15_{10} .
2. Произведите умножение следующих восьмеричных чисел в двоичной системе счисления: 47_8 и 12_8 ; 37_8 и 24_8 ; 64_8 и 15_8.
3. Проверьте результаты двоичного умножения предыдущего упражнения с помощью двоичного деления.
4. Произведите сложение следующих шестнадцатеричных чисел:

   $\overline{7ABC}_{16}$ и $\overline{E90C}_{16}$; $\overline{EEE3}_{16}$ и $\overline{CACAB}_{16}$.

1.5. Формы представления чисел (форматы данных)

В современных ЭВМ применяют естественную (с фиксированной десятичной точкой) или нормальную (с плавающей десятичной точкой) формы записи чисел.

Естественная форма представления чисел. При естественной форме число представляется в виде целой части числа и отделенной от нее десятичной точкой дробной части. Место десятичной точки фиксировано, т.е. для целой и дробной частей числа отводится вполне определенное количество разрядов.

Например, если в некотором формате вывода для целой и дробной частей числа (без учета знака) отводится по ***3*** десятичных разряда, то число ***741,3*** будет представляться в виде: ***+ 741,300***, а число ***–1,327*** – в виде: ***- 001,327***. Очевидно, что в этом формате можно представить числа, по абсолютной величине расположенные в диапазоне от ***000,001*** до ***999,999***. Все числа, меньшие ***0,001***, представляются в виде нуля, который называют *машинным нулем*. Наименьшее число ***0,001*** определяет точность представления чисел.

Нормальная форма представления чисел. Всякое число можно представить в виде произведения двух сомножителей:

$$N = m \cdot n^{p},$$

где m – мантисса числа (дробное число, меньшее единицы);

p – порядок числа (целое число);

n – основание системы счисления (целое число).

Например, число ***3,25*** в нормальной форме можно записать следующим образом:

$$3{,}25 = 0{,}325 \cdot 10^{1} = 0{,}0325 \cdot 10^{2} = 0{,}00325 \cdot 10^{3} = \ldots \ .$$

Порядок числа может быть отрицательный, например,

$$0{,}00325 = 0{,}325 \cdot 10^{-2}.$$

Порядок определяет положение десятичной точки в числе. Величина порядка говорит, на сколько разрядов влево (в случае отрицательного порядка)

или вправо (в случае положительного порядка) необходимо сдвинуть десятичную точку в мантиссе числа, чтобы получить число в естественной форме.

Число ***3,25*** в нормальной форме может быть записано с различными мантиссами и порядками, равными ***1, 2, 3*** и т.д. Первое число из этих чисел называется *нормализованным числом*, а остальные – *ненормализованными числами.*

Условие нормализации числа ***N*** можно записать следующим образом:

$$(1/n) \le |m| < 1,$$

где ***n*** – основание системы счисления.

У нормализованного числа первая цифра мантиссы отлична от нуля. Числа в памяти машины желательно хранить в нормализованном виде, так как у ненормализованных чисел могут теряться младшие разряды мантиссы из-за конечного количества разрядов для ее записи. При выполнении арифметических действий над числами, представленными в нормальной форме, может получиться результат, мантисса которого больше единицы. Такое число можно нормализовать, сдвинув мантиссу вправо и увеличив порядок числа. Таким образом, при нормальной форме представления чисел может иметь место нарушение нормализации, которое легко устраняется.

Нормализация чисел в машине производится как самой машиной автоматически после выполнения арифметических действий, так и по специальным командам.

Числа в нормальной форме записывают с помощью двух групп цифр, характеризующих мантиссу и порядок чисел с соответствующими знаками. При этом мантисса может записываться с помощью цифр, стоящих после десятичной точки. Например, если для записи мантиссы отводится четыре разряда, а для записи порядка два разряда, то числа $+0{,}325 \cdot 10^{1}$ и $-0{,}325 \cdot 10^{-2}$ будут представляться в виде ***+.3250 E+01*** и ***- .3250 E -02***, ***E*** означает ***10*** в соответствующей степени.

Диапазон представления чисел в нормальной форме значительно шире, чем диапазон представления в естественной форме, что является преимуществом нормальной формы представления чисел.

Недостатком нормальной формы по сравнению с естественной является необходимость дополнительных разрядов для хранения порядка. Кроме того, алгоритм выполнения арифметических действий над числами с плавающей десятичной точкой более сложный, что приводит к увеличению времени его выполнения и усложнению реализующей его аппаратуры.

Рассмотренные здесь примеры записи чисел в обеих формах представляли собой примеры записи десятичных чисел. По этим же правилам записи могут быть представлены числа в системе счисления с произвольным основанием.

2. ВВЕДЕНИЕ В АЛГЕБРУ ВЫСКАЗЫВАНИЙ

Электронная техника использует элементы, обладающие двумя устойчивыми состояниями, поэтому наиболее простой для реализации является двоичная арифметика, использующая всего две цифры: ***0*** и ***1*** для записи любого числа. С этим же связано и то, что математической основой вычислительной техники является алгебра высказываний, с введения в которую мы продолжаем знакомство с устройствами ЭВМ.

2.1. Элементы булевой алгебры

Известно, что высказывание – это предложение, относительно которого имеет смысл говорить, что оно истинно или ложно. В алгебре высказываний интересуются только значением истинности высказывания, отвлекаясь от его содержания. Каждому верному высказыванию приписывают значение истинности ***1*** (истинно), каждому неверному высказыванию – значение истинности ***0*** (ложно).

В алгебре высказываний вводится, как и в обычной алгебре, ряд операций, которые производятся над высказываниями. В результате этих операций получаются новые высказывания.

Соединение двух высказываний ***A*** и ***B*** с помощью союза ***И*** называется *логическим умножением*, или *конъюнкцией этих высказываний.*

Эта операция дает новое высказывание, обозначаемое $A \wedge B$**.** Высказывание $A \wedge B$ считается истинным в том и только в том случае, когда истинны одновременно оба высказывания ***A*** и ***B***, т.е. значение истинности высказывания

$A \wedge B$ определяется таблицей истинности (табл. 2.1).

Соединение двух высказываний с помощью союза ИЛИ называется *логическим сложением,* или *дизъюнкцией.*

Дизъюнкция высказываний ***A*** и ***B*** есть новое высказывание, обозначаемое $A \vee B$. Дизъюнкция $A \vee B$ считается истинной в том и только в том случае, когда истинно хотя бы одно из высказываний ***A*** или ***B***, таким образом значение истинности дизъюнкции ***A*** и ***B*** определяется в зависимости от значения истинности ***A*** и ***B*** (табл. 2.2).

Заметим, что термины «конъюнкция» и «дизъюнкция» относятся не только к самим операциям, но и к их результатам.

Определение конъюнкции и дизъюнкции легко распространить на любое конечное число аргументов. Например, конъюнкция истинна тогда и только тогда, когда истинны все входящие в нее высказывания. Дизъюнкция истинна тогда и только тогда, когда истинно хотя бы одно из входящих в нее высказываний.

Присоединение частицы НЕ или слов НЕВЕРНО, ЧТО к высказыванию ***A*** дает новое высказывание, называемое *отрицанием высказывания* ***A***. Оно обозначается $\overline{A}$ и читается «не ***A***». Если высказывание ***A*** истинно, то его от-

рицание ложно, и наоборот. Таким образом, значения истинности отрицания высказывания A определяются табл. 2.3.

Таблица 2.1

A	B	$A \wedge B$
0	0	0
0	1	0
1	0	0
1	1	1

Таблица 2.2

A	B	$A \vee B$
0	0	0
0	1	1
1	0	1
1	1	1

Таблица 2.3

A	$\overline{A}$
0	1
1	0

Еще одним примером операции над высказываниями является равнозначность.

Равнозначностью двух высказываний A и B служит новое высказывание $A \sim B$, оно считается истинным тогда и только тогда, когда логические значения высказываний A и B совпадают между собой.

Таким образом, значения истинности высказывания $A \sim B$ определяются таблицей истинности (табл. 2.4).

В алгебре высказываний с помощью введенных в рассмотрение логических операций можно из простых высказываний составлять сложные.

Пример. Важным примером сложного высказывания является отрицание равнозначности $\overline{A \sim B}$ двух высказываний A и B. Значения истинности высказывания $A \sim B$ определяются табл. 2.5.

Таблица 2.4

A	B	$A \sim B$
0	0	1
0	1	0
1	0	0
1	1	1

Таблица 2.5

A	B	$\overline{A \sim B}$
0	0	0
0	1	1
1	0	1
1	1	0

Пример. Пусть A, B, C – высказывания. Многократным применением рассмотренных операций можно образовывать из них новые высказывания любой сложности. Из трех высказываний A, B, C составим высказывание $\overline{A} \vee (B \wedge C)$. Скобки указывают, как и в обычной алгебре, порядок выполнения операций. Составим таблицу истинности высказывания $\overline{A \vee (B \wedge C)}$.

В первых трех столбцах табл. 2.6 выписаны все возможные наборы комбинаций значений истинности высказываний A, B, C. В следующих столбцах выписываются значения истинности последовательно выполняемых операций и окончательного результата $\overline{A \vee (B \wedge C)}$.

Таблица 2.6

A	B	C	$\overline{A}$	$B \wedge C$	$\overline{A} \vee (B \wedge C)$
0	0	0	1	0	1
0	0	1	1	0	1
0	1	0	1	0	1
0	1	1	1	1	1
1	0	0	0	0	0
1	0	1	0	0	0
1	1	0	0	0	0
1	1	1	0	1	1

Нетрудно заметить, что зависимость числа строк в таблице от числа исходных столбцов n всегда равняется 2^n .

Если два сложных высказывания принимают одни и те же значения при любых наборах значений входящих в них высказываний, то они называются *эквивалентными* и соединяются знаком ≡.

Сложное высказывание, значение истинности которого равно 1 при любых значениях истинности входящих в него простых высказываний, называется *тождественно-истинным.*

Например, высказывание $A \vee \overline{A}$ есть тождественно-истинное высказывание, что следует из таблицы истинности (табл. 2.7).

Таблица 2.7

A	$\overline{A}$	$A \vee \overline{A}$
0	1	1
1	0	1

Таблица 2. 8

A	$\overline{A}$	$A \wedge \overline{A}$
0	1	0
1	0	0

Сложное высказывание, значение истинности которого равно нулю при любых значениях истинности входящих в него простых высказываний, называется *тождественно-ложным.*

Так, высказывание $A \wedge \overline{A}$ есть тождественно-ложное высказывание, что следует из его таблицы истинности (табл. 2.8).

У п р а ж н е н и я

1. Составьте таблицы истинности следующих высказываний:

   а) $\overline{A} \wedge B$; б) $A \wedge \overline{B}$; в) $\overline{A} \wedge \overline{B}$; г) $(\overline{A} \wedge B) \wedge (A \wedge \overline{B})$.

2. Составьте таблицы истинности высказываний:

   а) $A \vee (B \wedge C)$; б) $(A \vee B) \wedge C$; в) $A \wedge (B \vee C)$;
   г) $(A \vee B) \wedge (A \vee C)$.

2.2. Алгебра контактных схем

Современные микропроцессорные устройства допускают общее представление происходящих в них функционально-логических процессов на уровне анализа простых электротехнических конструкций.

Рассмотрим участок электрической цепи, разветвляющейся на два направления. В каждом из направлений установлен выключатель, который в дальнейшем будем называть *контактом.* Изобразим (рис.2.1) этот участок цепи с параллельным расположением контактов. Контакт может быть в одном из двух состояний: замкнут (ток через него проходит) или разомкнут (ток не проходит). Очевидно, что если оба контакта будут замкнуты, то по этому участку цепи электрический ток проходит. Но ток пройдет также и в том случае, если хотя бы один из контактов будет замкнут. Если же разомкнутыми будут одновременно оба контакта, то электрический ток через этот участок цепи не пройдет. Табл. 2.9 определяет зависимость прохождения тока через данный участок цепи от всех возможных комбинаций состояний контактов ***X*** и ***Y***. Примем такие обозначения: **1** соответствует замкнутому состоянию контакта, **0** – разомкнутому. Состояние участка цепи, если он пропускает ток, обозначим **1**, если ток не пропускается, то **0**.

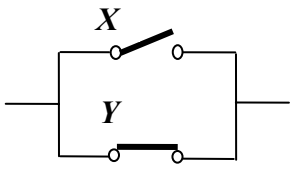


Рис. 2.1

Таблица 2.9

X	***Y***	Состояние участка цепи
0	0	0
0	1	1
1	0	1
1	1	1

Табл. 2.9 совпадает с таблицей истинности дизъюнкции двух высказываний $A \vee B$ (табл. 2.2). Это соответствие позволяет состояние участка цепи с параллельно соединенными контактами представлять как дизъюнкцию $X \vee Y$.

Рассмотрим другой участок цепи, на котором последовательно расположены два контакта ***X*** и ***Y*** (рис. 2.2). Нетрудно заметить, что по этому участку электрический ток будет проходить в одном единственном случае, когда одновременно будут замкнуты оба контакта (табл. 2.10).

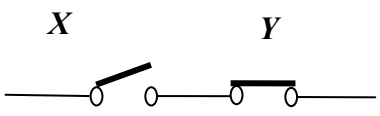


Рис. 2.2

Таблица 2.10

X	***Y***	Состояние участка цепи
0	0	0
0	1	0
1	0	0
1	1	1

Полученная табл. 2.10 совпадает с таблицей истинности конъюнкции двух высказываний ***A*** и ***B*** (табл.2.1). Это соответствие позволяет состояние участка

цепи с последовательно соединенными контактами представить как конъюнкцию $X \wedge Y$.

В курсе физики разбирается устройство электромагнитного реле (рис. 2.3).

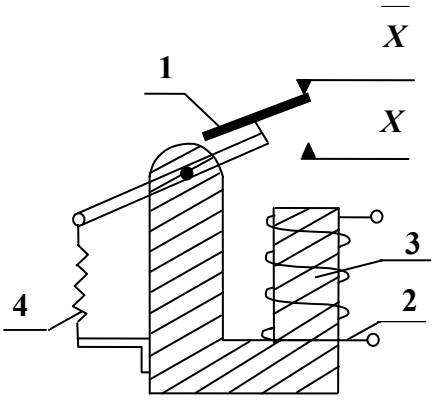


Рис. 2.3

Принцип его действия прост: при возбуждении обмотки катушки реле 2 подвижный контакт 1 притягивается к якорю 3 реле и таким образом замыкает контакт X (этот контакт называется *замыкающим*); в это время контакт $\overline{X}$ оказывается разомкнутым (его называют *размыкающим*). Когда обмотка катушки обесточена, пружина 4 возвращает подвижный контакт в исходное положение, при котором он замыкает контакт X и размыкает контакт X. Контакт X называют *инверсией контакта* X.

Каждой контактной схеме, составленной из параллельного и последовательного соединений размыкающих и замыкающих контактов реле, соответствует некоторая логическая функция. Эта функция выражается формулой, состоящей из простых высказываний и их отрицаний с использованием операций дизъюнкции и конъюнкции. На этом соответствии построено применение математического аппарата алгебры высказываний в анализе, упрощении и синтезе контактных схем.

Рассмотрим некоторые задачи, связанные с установлением этого соответствия.

Пример. Составим сложное высказывание, которое соответствует контактной схеме, изображенной на рис. 2.4

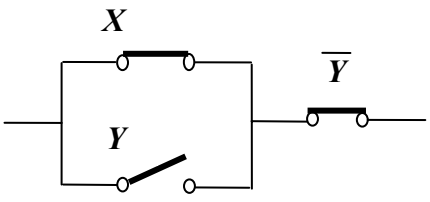


Рис. 2.4

Таблица 2.11

X	Y	$\overline{Y}$	$X \vee Y$	$F(X,Y)$
0	0	1	0	0
0	1	0	1	0
1	0	1	1	1
1	1	0	1	0

Здесь контакты X и Y объединены параллельно между собой и последовательно по отношению к контакту Y. Следовательно, сложное высказывание, соответствующее этой схеме, имеет вид $(X \vee Y) \wedge Y$, т.е. состояние контактной схемы описывается логической функцией $F(X, Y)$, представленной формулой

$$F(X, Y) = (X \vee Y) \wedge Y.$$

По этой формуле можно составить таблицу истинности 2.11. Действительно, эта контактная схема пропускает ток в единственном случае, когда контакт Y разомкнут, а контакт X замкнут.

Пример. Рассмотрим теперь обратную ситуацию: по данной логической формуле требуется построить контактную схему. Дана логическая функция

$$F(X, Y) = X \wedge (\overline{X} \vee \overline{Y}).$$

Контакты $\overline{X}$ и $\overline{Y}$, связанные в формуле операцией дизъюнкции, на схеме должны быть соединены параллельно. Контакт X и участок цепи $\overline{X} \vee \overline{Y}$ должны быть соединены последовательно, т.е. приходим к контактной схеме (рис. 2.5).

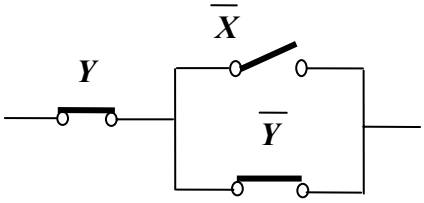


Рис. 2.5

Приведенный в предыдущих примерах разбор работы контактной схемы, построение формулы, ее описывающей, и определение условий, при которых данная схема пропускает или не пропускает ток, называют *анализом контактной схемы.*

Следующим видом применения алгебры контактных схем является упрощение контактных схем, которое сводится к упрощению соответствующей ей логической формулы и заключается в преобразовании исходной формулы в более простую (как и в обычной алгебре). В основе преобразования формул алгебры высказываний лежат свойства основных логических операций, вытекающие из определений отрицания, конъюнкции и дизъюнкции:

1. $\overline{\overline{X}} \equiv X$ - закон двойного отрицания.
2. $X \vee Y \equiv Y \vee X$ - коммутативность дизъюнкции.
3. $X \wedge Y \equiv Y \wedge X$ - коммутативность конъюнкции.
4. $(X \vee Y) \vee Z \equiv X \vee (Y \vee Z)$ – ассоциативность дизъюнкции.
5. $(X \wedge Y) \wedge Z \equiv X \wedge (Y \wedge Z)$ – ассоциативность конъюнкции.
6. $X \wedge (Y \vee Z) \equiv (X \wedge Y) \vee (X \wedge Z)$ – дистрибутивность конъюнкции относительно дизъюнкции.
7. $X \vee (Y \wedge Z) \equiv (X \vee Y) \wedge (X \vee Z)$ – дистрибутивность дизъюнкции относительно конъюнкции.
8. $X \vee X \equiv X$.
9. $X \wedge X \equiv X$.
10. $X \vee 0 \equiv X$.
11. $X \wedge 1 \equiv X$.
12. $X \wedge 0 \equiv 0$.
13. $X \vee 1 \equiv 1$.
14. $X \vee \overline{X} \equiv 1$.

15. $X \wedge \overline{X} \equiv 0$.

16. $\overline{X \wedge Y} \equiv \overline{X} \vee \overline{Y}$.

17. $\overline{X \vee Y} \equiv \overline{X} \wedge \overline{Y}$.

Эти свойства легко доказать, например, при помощи подстановки на место X, Y, Z всевозможных наборов их значений, т.е. при помощи составления таблиц истинности и установления совпадения значений результирующих столбцов.

Справедливость этих свойств легко усматривается и из содержательного толкования самих операций на уровне контактных схем. Например, свойство 8: $X \vee X \equiv X$. Контактная схема, состоящая из двух параллельно соединенных контактов X , ведет себя так же, как один контакт X. Аналогично свойство 9: $X \wedge X \equiv X$. Два последовательно соединенных контакта при их одновременном замыкании или размыкании ведут себя так же, как один контакт X .

В ряде записанных формул, раскрывающих свойства логических операций, стоят 0 и 1, которые следует рассматривать как логические константы. 0 – это постоянно разомкнутый контакт, независимый элемент контактной схемы. Так, свойство 10: $X \vee 0 \equiv X$ показывает, что замыкание или размыкание контактной схемы зависит лишь от контакта X, а свойство 12: $X \wedge 0 \equiv 0$ показывает, что последовательное соединение постоянно разомкнутого контакта *«0»* всегда приводит к размыканию всей контактной схемы.

1 – это постоянно замкнутый контакт. Свойство 13: $X \vee 1 \equiv 1$ показывает, что при параллельном соединении постоянно замкнутого контакта 1 с контактом X схема всегда будет замкнута независимо от состояния контакта X. Свойство 11**:** $X \wedge 1 \equiv X$ показывает, что если в последовательной цепи один контакт постоянно замкнут, то состояние цепи определяется состоянием другого контакта. Свойство 14: $X \vee \overline{X} \equiv 1$ – если каждый раз при размыкании одного параллельного контакта другой замыкается, то цепь всегда остается замкнутой. Свойство 15: $X \wedge \overline{X} \equiv 0$ − если при замыкании одного из последовательных контактов другой размыкается, то цепь всегда остается разомкнутой.

Используя таблицы истинности, докажем свойство 16: $X \wedge Y \equiv \overline{X} \vee \overline{Y}$.

Составим таблицы истинности (табл. 2.12 и табл.2.13).

Таблица 2.12

X	Y	$X \wedge Y$	$\overline{X \wedge Y}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

Таблица 2.13

X	Y	$\overline{X}$	$\overline{Y}$	$\overline{X} \vee \overline{Y}$
0	0	1	1	1
0	1	1	0	1
1	0	0	1	1
1	1	0	0	0

Из сравнения таблиц видно, что при одинаковом заполнении исходных столбцов X и Y результирующие столбцы этих таблиц $\overline{X \wedge Y}$ и $\overline{X} \vee \overline{Y}$ совпадают.

Используя свойства логических операций 1 – 17, проведем упрощение контактных схем путем преобразования соответствующих им формул.

Пример. Рассмотрим контактную схему, изображенную на рис 2.6. Ей соответствует такая формула: $F(X, Y) = (X \vee \overline{Y}) \wedge X$.

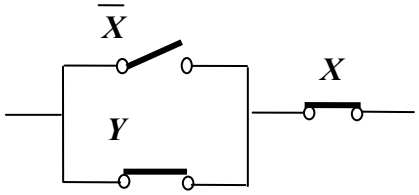


Рис. 2.6

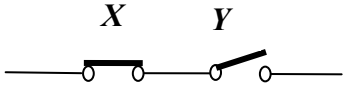


Рис. 2.7

Проведем преобразование этой формулы, последовательно используя свойства 3, 6, 15, 2, 10:

$$F(X,Y) = (\overline{X} \vee Y) \wedge X \equiv X \wedge (\overline{X} \vee Y) \equiv (X \wedge \overline{X}) \vee (X \wedge Y) \equiv 0 \vee (X \wedge Y) \equiv (X \wedge Y) \vee 0 \equiv X \wedge Y.$$

Следовательно, логические функции, выражаемые формулами $(\overline{X} \vee Y) \wedge X$ и $X \wedge Y$, соответствуют эквивалентным контактным схемам. Это позволяет заменить контактную схему рис. 2.6 более простой схемой рис. 2.7. Заметим, что эквивалентность этих контактных схем может быть также установлена с помощью таблиц истинности.

Пример. Требуется произвести анализ и упрощение контактной схемы, представленной на рис. 2.8. Здесь и далее контакты для простоты изображения заменим кружочками.

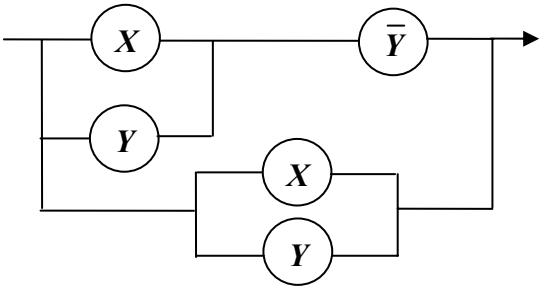


Рис. 2.8

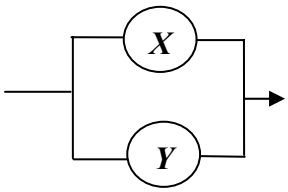


Рис. 2.9

Соответствующая этой схеме формула имеет вид:

$$F(X, Y) = ((X \vee Y) \wedge \overline{Y}) \vee (X \vee Y).$$

Используя последовательно свойства 11, 6, 13, 11, проведем упрощение этой формулы:

$$F(X,Y) = ((X \vee Y) \wedge \overline{Y}) \vee (X \vee Y) \equiv ((X \vee Y) \wedge \overline{Y}) \vee (X \vee Y) \wedge 1) \equiv$$
$$\equiv (X \vee Y) \wedge (\overline{Y} \vee 1) \equiv (X \vee Y) \wedge 1 \equiv X \vee Y.$$

Контактная схема, соответствующая формуле $F(X,Y) = X \vee Y$ (рис. 2.9) и состоящая всего из двух контактов, эквивалентна исходной контактной схеме

(рис. 2.8), в которой было 5 контактов. Построение контактной схемы по соответствующей ей формуле есть один из этапов синтеза контактных схем.

Синтез контактной схемы заключается в построении таблицы истинности по заданным условиям работы будущей схемы, в составлении формулы по этим условиям, в конструировании контактной схемы в соответствии с полученной формулой. В данном примере мы в результате упрощения исходной формулы получили $F(X,Y) = X \vee Y$. По этой формуле и была сконструирована упрощенная схема рис. 2.9.

Пример. Упростим контактную схему, изображенную на рис. 2.10. Данной контактной схеме соответствует формула:

$$F(X,Y,Z) = ((X \vee Y) \wedge Y) \wedge (Z \vee \overline{X})) \vee (Y \wedge (\overline{Z} \vee \overline{Y})).$$

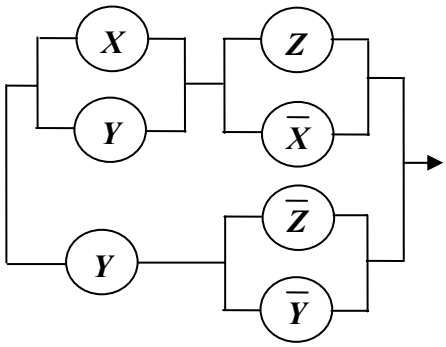


Рис. 2.10

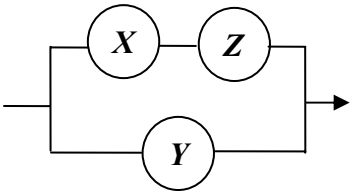


Рис. 2.11

Проведем преобразование этой формулы, используя свойства 6, 15, 10, 2, 11, 13:

$$F(X,Y,Z) \equiv ((X \vee Y)) \wedge (Z \vee \overline{X})) \vee (Y \wedge (Z \vee \overline{Y})) \equiv$$

$$\equiv (X \wedge (Z \vee \overline{X})) \vee (Y \wedge (Z \vee \overline{X})) \vee (Y \wedge (\overline{Z} \vee \overline{Y})) \equiv$$

$$\equiv ((X \wedge Z) \vee (X \wedge \overline{X})) \vee ((Y \wedge Z) \vee (Y \wedge \overline{X})) \vee ((Y \wedge \overline{Z}) \vee (Y \wedge \overline{Y})) \equiv$$

$$\equiv ((X \wedge Z) \vee 0) \vee (Y \wedge Z) \vee (Y \wedge \overline{X}) \vee ((Y \wedge \overline{Z}) \vee 0)) \equiv$$

$$\equiv (X \wedge Z) \vee (Y \wedge Z) \vee (Y \wedge \overline{X}) \vee (Y \wedge \overline{Z}) \equiv$$

$$\equiv ((X \wedge Z) \vee (Y \wedge \overline{X})) \vee ((Y \wedge Z) \vee (Y \wedge \overline{Z})) \equiv$$

$$\equiv ((X \wedge Z) \vee (Y \wedge \overline{X})) \vee (Y \wedge (Z \vee \overline{Z})) \equiv$$

$$\equiv (X \wedge Z) \vee (Y \wedge \overline{X}) \vee (Y \wedge 1) \equiv$$

$$\equiv (X \wedge Z) \vee (Y \wedge (\overline{X} \vee 1)) \equiv (X \wedge Z) \vee (Y \wedge 1) \equiv (X \wedge Z) \vee Y.$$

Конструируем по формуле $F(X,Y,Z) \equiv (X \wedge Z) \vee Y$ контактную схему (рис. 2.11). Если в исходной схеме было 7 контактов, то в упрощенной их стало только 3.

У п р а ж н е н и я

1. Произведите построение контактных схем по формулам:

а) $F(X,Y) \equiv X \vee (Y \wedge \overline{X})$;

б) $F(X,Y) \equiv (X \vee Y) \wedge \overline{X}$;

в) $F(X,Y) \equiv (X \vee \overline{Y}) \wedge (\overline{X} \vee Y)$;

г) $F(X,Y) \equiv (X \vee \overline{X} \vee Y) \wedge (X \vee X \vee \overline{Y})$.

2. Произведите анализ и упрощение контактных схем, изображенных на рис. 2.12, 2.13, 2.14.

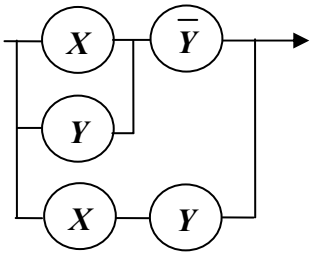


Рис. 2.12

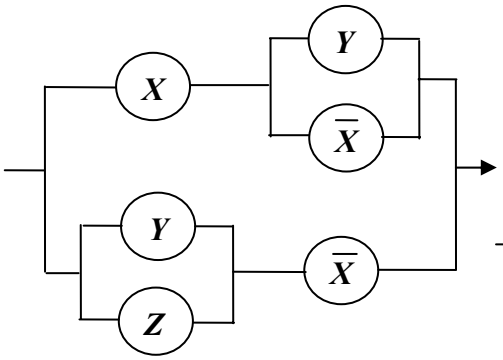


Рис. 2.13

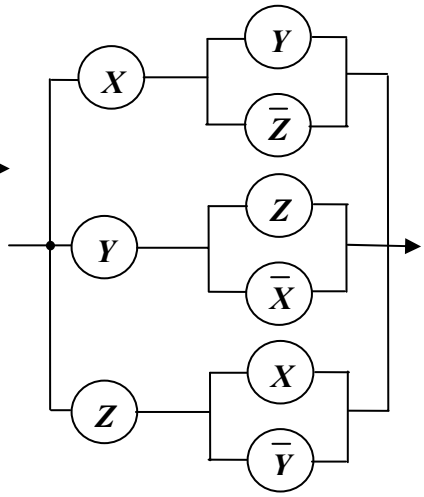


Рис. 2.14

3. ФУНКЦИОНАЛЬНЫЕ ЭЛЕМЕНТЫ АРИФМЕТИКО-ЛОГИЧЕСКОГО УСТРОЙСТВА

Ясно, что скорость замыкания и размыкания контактов реле, из которых составляется контактные схемы, невелика – около 10 срабатываний в секунду. Быстродействие современных ЭВМ порядка $10^6 – 10^{10}$ операций в секунду достигается благодаря использованию микроэлектронных схем, скорость срабатывания которых в миллионы раз быстрее, чем скорость аналогичных релейно-контактных схем. Мы не будем рассматривать физические основы этих электронных устройств, реализующих основные логические операции. Для нас важно другое: эти электронные устройства, получая значения истинности отдельных простых высказываний (в виде, например, электрических сигналов: 1 - наличие сигнала, 0 - его отсутствие), могут выдавать значения истинности конъюнкции, дизъюнкции, отрицания. Устройства такого рода называются *функциональными элементами.*

3.1. Характеристики функциональных элементов

Этот материал важен для понимания общих принципов функционирования арифметико-логического устройства (АЛУ) ЭВМ.

Элемент ***НЕ*** реализует отрицание: он имеет один вход и один выход. Если на вход подается сигнал, то на выходе сигнал всегда отсутствует; если на входе сигнала нет, то на выходе сигнал есть (рис. 3.1).

Элемент ***И*** реализует конъюнкцию: он имеет два или более входов и один выход. На выходе сигнал появляется тогда и только тогда, когда на все входы поданы сигналы (рис. 3.2).

Элемент ***ИЛИ*** реализует дизъюнкцию: он имеет два или более входов и один выход. На выходе сигнал появляется тогда и только тогда, когда хотя бы на один его вход подан сигнал (рис. 3.3).

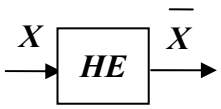


Рис. 3.1

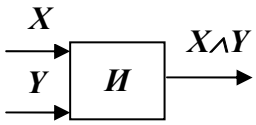


Рис. 3.2

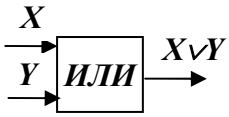


Рис. 3.3

Из функциональных элементов, соединяя их между собой (выход одного со входом другого), можно составлять функциональные схемы, реализующие сложные логические формулы. Каждой логической формуле можно поставить в соответствие функциональную схему.

Пример. Составим функциональную схему, реализующую логическую формулу:

$$F(X,Y) = (\overline{X} \vee Y) \wedge X.$$

Функциональная схема будет иметь два входа, на которые будут подаваться сигналы, соответствующие значениям ***X*** и ***Y***. На этой схеме будет один выход, на котором будет появляться сигнал, соответствующий значению заданной логической функции при данных значениях ***X*** и ***Y***.

В функциональной схеме будет столько элементов, сколько операций в самой формуле. В примере их три: отрицание, дизъюнкция и конъюнкция, т.е. и в функциональной схеме будет три элемента (рис. 3.4)

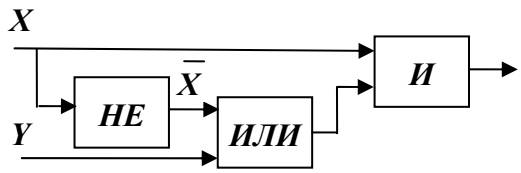


Рис. 3.4

Пример. Рассмотрим задачу синтеза функциональной схемы одноразрядного двоичного сумматора с двумя входами.

Основным счетным узлом АЛУ ЭВМ является многоразрядный сумматор, выполняющий непосредственно сложение многозначных двоичных чисел. Отметим еще раз, что другие арифметические операции в двоичной системе счисления могут быть сведены к двоичному сложению, т.е. они тоже могут выполняться на двоичном сумматоре.

Многоразрядный сумматор состоит из одноразрядных сумматоров, осуществляющих двоичное сложение в каждом разряде и перенос единицы в старший разряд, если в этом возникает потребность.

Построим одноразрядный сумматор на два входа, в котором осуществляется сложение в разряде единиц, т.е. не предполагается наличие третьего входа для переноса единицы из младшего разряда.

Прежде всего определим условия работы такой функциональной схемы, в которой при подаче на два ее входа сигналов X и Y, соответствующих значениям двоичных слагаемых в разряде единиц (двоичные слагаемые, принимающие значения только 0 и 1, можно считать значениями истинности некоторых высказываний), на двух выходах появлялись бы сигналы, соответствующие значению суммы в этом разряде S_1 и значению переноса в старший разряд S_2, т.е. $X + Y = S_2 S_1$.

Запишем все возможные комбинации слагаемых X и Y при сложении:

$$\begin{array}{r} 0 \\ +\,0 \\ \hline 00 \end{array} \qquad \begin{array}{r} 0 \\ +\,1 \\ \hline 01 \end{array} \qquad \begin{array}{r} 1 \\ +\,0 \\ \hline 01 \end{array} \qquad \begin{array}{r} 1 \\ +\,1 \\ \hline 10 \end{array}$$

Очевидно, что на выходе S_2 сигнал будет появляться тогда и только тогда, когда будут поданы сигналы на оба входа X и Y: на входе S_1 сигнал будет, если подан сигнал на один из входов X или Y. Объединим результаты четырех случаев двоичного сложения в табл. 3.1

Таблица 3.1

X	Y	S_2	S_1
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

В полученной таблице истинности два результирующих столбца по числу выходов функциональной схемы: S_2 и S_1. Будем считать, что S_1 и S_2 есть логические функции от X и Y. Эти логические функции $S_1(X,Y)$ и $S_2(X,Y)$ необходимо выразить в виде формул. Сравнив значения результирующего столбца S_2 с таблицей конъюнкции (табл. 2.1), приходим к выводу, что логическая функция S_2 является конъюнкцией X и Y.

$$S_2(X,Y) = X \wedge Y.$$

Составить формулу для представления S_1 намного труднее. Тем не менее существует алгоритм составления логических формул по таблицам истинности, который заключается в следующем.

Из таблицы истинности выбираются наборы переменных X и Y, для которых соответствующие значения результирующего столбца равны 1 (т.е. те значения X и Y, для которых $S_1(X,Y) = 1$). Такие наборы выбираются из второй и третьей строк таблицы: 0,1 и 1,0.

Для каждого такого набора находим конъюнкцию X и Y или их отрицаний, истинную при этих значениях X и Y:

для набора $(0,1)$: $\overline{X} \wedge Y$,

для набора $(1,0)$: $X \wedge \overline{Y}$.

Дизъюнкция этих двух конъюнкций $\overline{X} \wedge Y$ и $X \wedge \overline{Y}$ будет истинной лишь в тех случаях, когда истинна логическая функция $S_1(X,Y)$. Значит, сложное высказывание $(X \wedge \overline{Y}) \vee (X \wedge Y)$ представляет собой одно из возможных выражений искомой логической функции $S_1(X,Y)$. Следовательно, формула для $S_1(X,Y)$ составлена:

$$S_1(X,Y) = (\overline{X} \wedge Y) \vee (X \wedge \overline{Y}).$$

Для проверки правильности полученной формулы составим таблицу истинности 3.2

Таблица 3.2

X	Y	$\overline{X}$	$\overline{Y}$	$\overline{X} \wedge Y$	$X \wedge \overline{Y}$	$S_1(X,Y)$
0	0	1	1	0	0	0
0	1	1	0	1	0	1
1	0	0	1	0	1	1
1	1	0	0	0	0	0

Совпадение результирующих столбцов табл. 3.1 и табл. 3.2 доказывает правильность составленной формулы для $S_1(X,Y)$.

Переходим к последнему этапу синтеза одноразрядного двоичного сумматора. Его функциональная схема должна представлять собой некоторое устройство с двумя входами X и Y и двумя выходами S_1 и S_2. Схематически это устройство представлено на рис. 3.5.

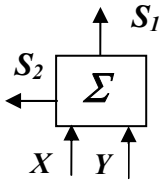


Рис. 3.5

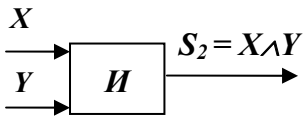


Рис. 3.6

Для простоты построения это устройство сначала будем представлять как бы состоящим из двух отдельных частей: в одной части вырабатывается сигнал S_1, в другой - сигнал S_2.

Функциональная схема первой части, соответствующая формуле

$$S_2(X,Y) = X \wedge Y,$$

приведена на рис. 3.6.

Функциональная схема второй части, соответствующая формуле

$$S_1(X,Y) = (\overline{X} \wedge Y) \vee (X \wedge \overline{Y}),$$

состоит из пяти функциональных элементов (по числу логических операций) и представлена на рис. 3.7.

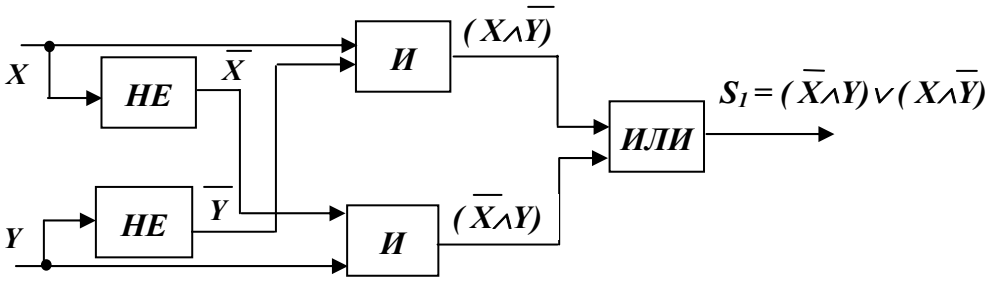


Рис. 3.7

Соединим теперь эти две полученные функциональные схемы в одну. На рис. 3.8 показан один из вариантов такого соединения. Эта функциональная схема и есть одноразрядный сумматор на два входа.

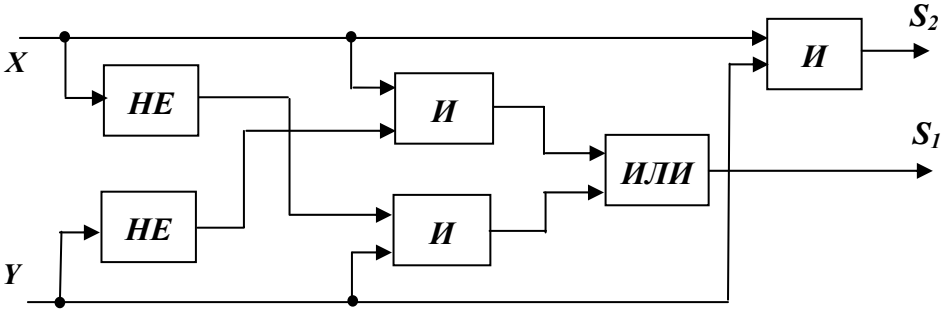


Рис. 3.8

У п р а ж н е н и е

Чтобы проверить, действительно ли эта схема реализует таблицу двоичного сложения, проследите преобразование сигналов X и Y, придавая им значения, равные 0 или 1 (таких наборов всего четыре). Убедитесь в совпадении результатов проверки каждого отдельного набора X и Y с соответствующими значениями результирующих столбцов табл. 3.1.

3.2. Двоичное сложение на многоразрядном сумматоре

Сконструированный только что двоичный сумматор может быть использован лишь в разряде единиц, т.к. он не имеет третьего входа для единицы переноса из младшего разряда. Для сложения в других разрядах необходимы сумматоры на три входа. Мы не будем повторять все этапы построения такого сумматора. В принципе оно не отличается от приведенного в предыдущем разделе.

Ввиду того, что этот сумматор на три входа имеет более универсальное использование, введем следующие обозначения (рис. 3.9):

X_i - значение i -го разряда слагаемого X,

Y_i - значение i -го разряда слагаемого Y,

C_{i-1} - значение переноса из соседнего младшего разряда,

S_i - значение разряда суммы (i-й разряд),

C_i - значение переноса в соседний старший разряд.

Логические функции S_i и C_i от X_i, Y_i, C_{i-1} задаются табл. 3.3

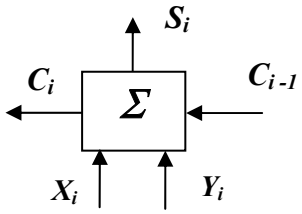


Рис. 3.9

Таблица 3.3

Вход			Выход	
X_i	Y_i	C_{i-1}	C_i	S_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Эту таблицу истинности двух логических функций $S_i(X_i, Y_i, C_{i-1})$ и $C_i(X_i, Y_i, C_{i-1})$ реализует функциональная схема (рис.3.10).

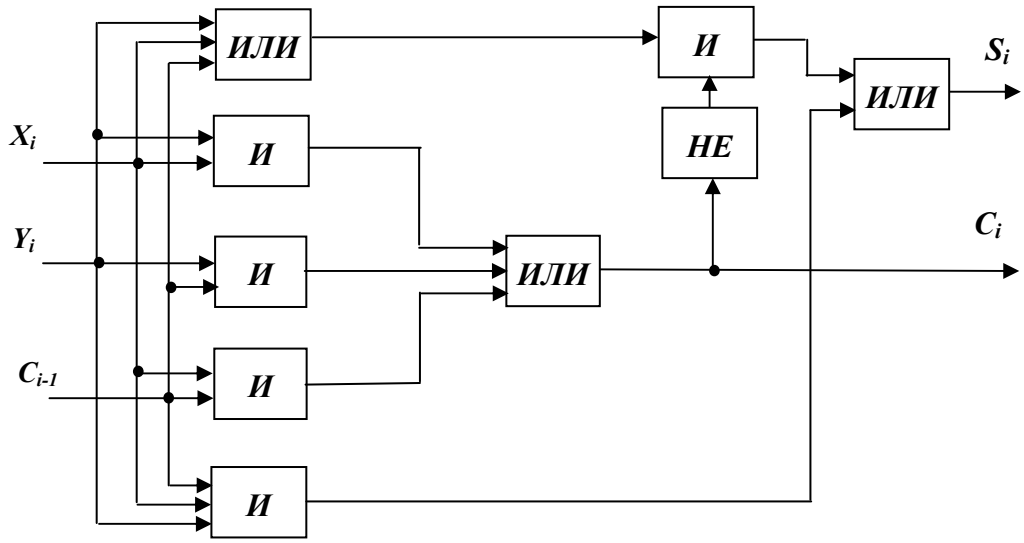


Рис. 3.10

Последовательно соединяя несколько одноразрядных сумматоров на три входа (выход ***C*** одного со входом ***C*** другого), можно составить многоразрядные двоичные сумматоры, осуществляющие сложение многозначных двоичных чисел.

Пример. Сложим два двоичных числа ***X = 11110*** и ***Y = 11011*** на пятиразрядном сумматоре (рис. 3.11)

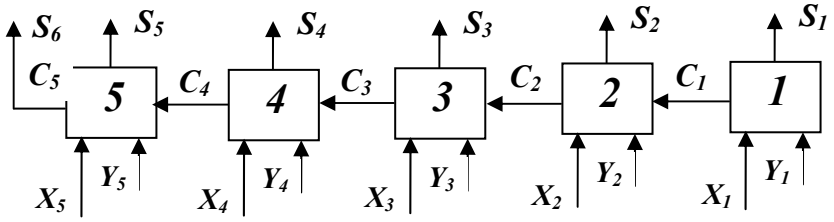


Рис. 3.11

На входы ***X*** сумматора подадим следующие сигналы, соответствующие значениям разрядов первого слагаемого ***X***:

$$X_1 = 0, \quad X_2 = 1, \quad X_3 = 1, \quad X_4 = 1, \quad X_5 = 1,$$

а на входы ***Y*** – сигналы, соответствующие значениям разрядов второго слагаемого ***Y***:

$$Y_1 = 1, \quad Y_2 = 1, \quad Y_3 = 0, \quad Y_4 = 1, \quad Y_5 = 1.$$

Сложение на многоразрядном сумматоре начинается с разряда единиц, т.е. в первом одноразрядном сумматоре, на входы которого поданы сигналы $X_1 = 0$ и
$Y_1 = 1$. В результате преобразования этих сигналов на функциональной схеме (рис. 3.8) будем иметь на выходе $S_1 = 1$, на выходе переноса $C_1 = 0$.

Затем сложение производится на следующем одноразрядном сумматоре, на входы которого подаются сигналы $X_2 = 1$, $Y_2 = 1$, $C_1 = 0$. Согласно табл. 3.3 на функциональной схеме (рис. 3.10) будем иметь на выходе $S_2 = 0$, на выходе переноса $C_2 = 1$.

В третьем сумматоре, на входы которого поданы сигналы $X_3 = 1$, $Y_3 = 0$, $C_2 = 1$, согласно табл. 3.3 будем иметь на выходе $S_3 = 0$, на выходе переноса $C_3 = 1$.

Аналогично для четвертого сумматора согласно табл. 3.3 будет $S_4 = 1$, $C_4 = 1$, а для пятого - $S_5 = 1$, $C_5 = 1$. Так как в нашем пятиразрядном сумматоре нет шестого одноразрядного сумматора, то, чтобы не пропадало значение переноса в шестой разряд, которое вырабатывается в пятом сумматоре, этот выход целесообразно сделать шестым разрядом суммы.

Итак, получено начальное представление о том, как выполняется двоичное сложение в арифметических устройствах ЭВМ. Другие арифметические операции выполняются с помощью функциональных схем, в основе которых лежит сумматор, реализующий сложение со сдвигом.

У п р а ж н е н и я

1. Составьте формулы логических функций $C_i (X_i, Y_i, C_{i-1})$ и $S_i (X_1, Y_i, C_{i-1})$ по таблице истинности 3.3.
2. Рассмотрите преобразование сигналов X_i, Y_i, C_{i-1} на функциональной схеме (рис. 3.10). Значения X_i, Y_i, C_{i-1} взять из табл. 3.3.

4. ОБРАБОТКА МАССИВОВ ИНФОРМАЦИИ

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, 7/.

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.

4.1. Перемещение массивов

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.4.1). Решение этой задачи представлено на рис.4.2.

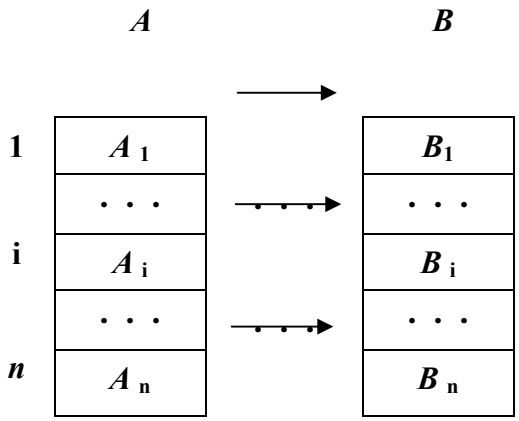


Рис. 4.1

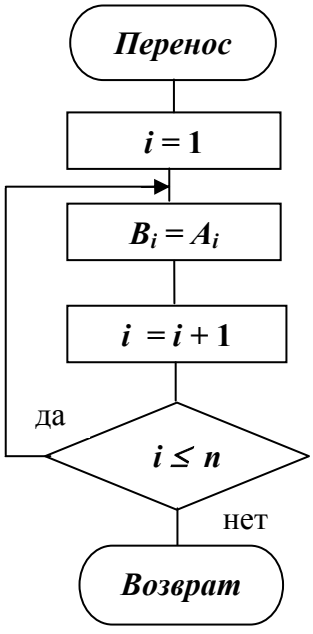


Рис. 4.2

У п р а ж н е н и я

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***.

4.2. 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. 4.3).

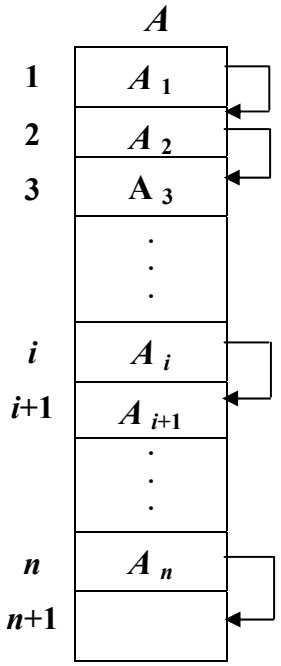


Рис. 4.3

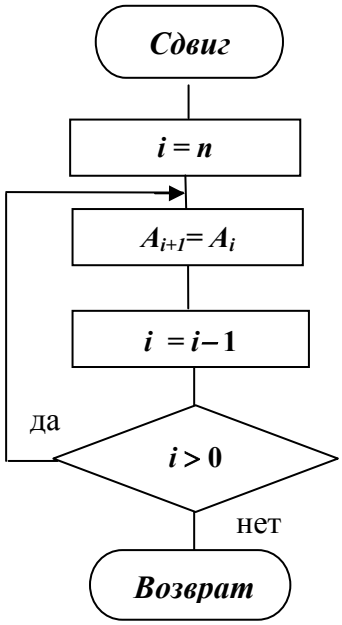


Рис. 4.4

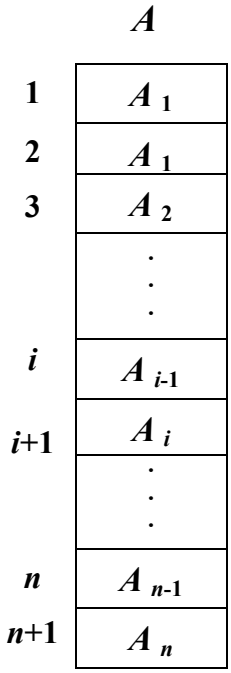


Рис. 4.5

Мы не можем начать решение этой задачи с первого элемента массива, потому что тогда безвозвратно сотрется значение второго элемента. Начинать приходится с конца (рис. 4.4). В результате выполнения этого алгоритма массив ***A*** приобретает вид, показанный на рис. 4.5. Обратим внимание на то обстоятельство, что первым элементом массива ***A*** по-прежнему является старое значение A_1: сам по себе он не уничтожается.

У п р а ж н е н и я

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

4.3. 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. 4.6).

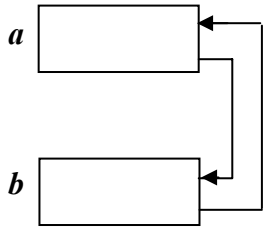


Рис. 4.6

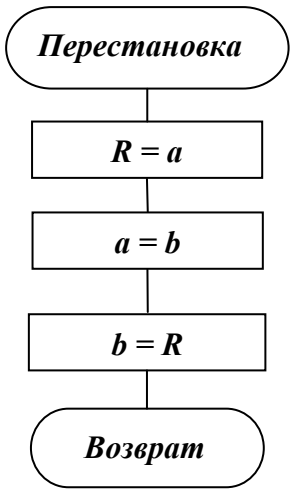


Рис. 4.7

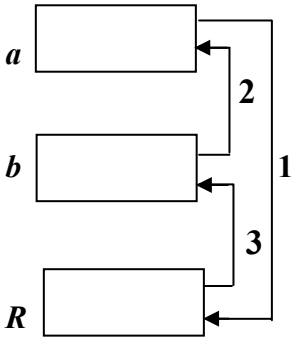


Рис. 4.8

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. 4.7. Для большей наглядности схему потоков информации по этому алгоритму изобразим порядковыми номерами стрелок (рис. 4.8).

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. 4.9 изображен массив с некоторым конкретным числовым наполнением и две вспомогательные ячейки с исходными данными.

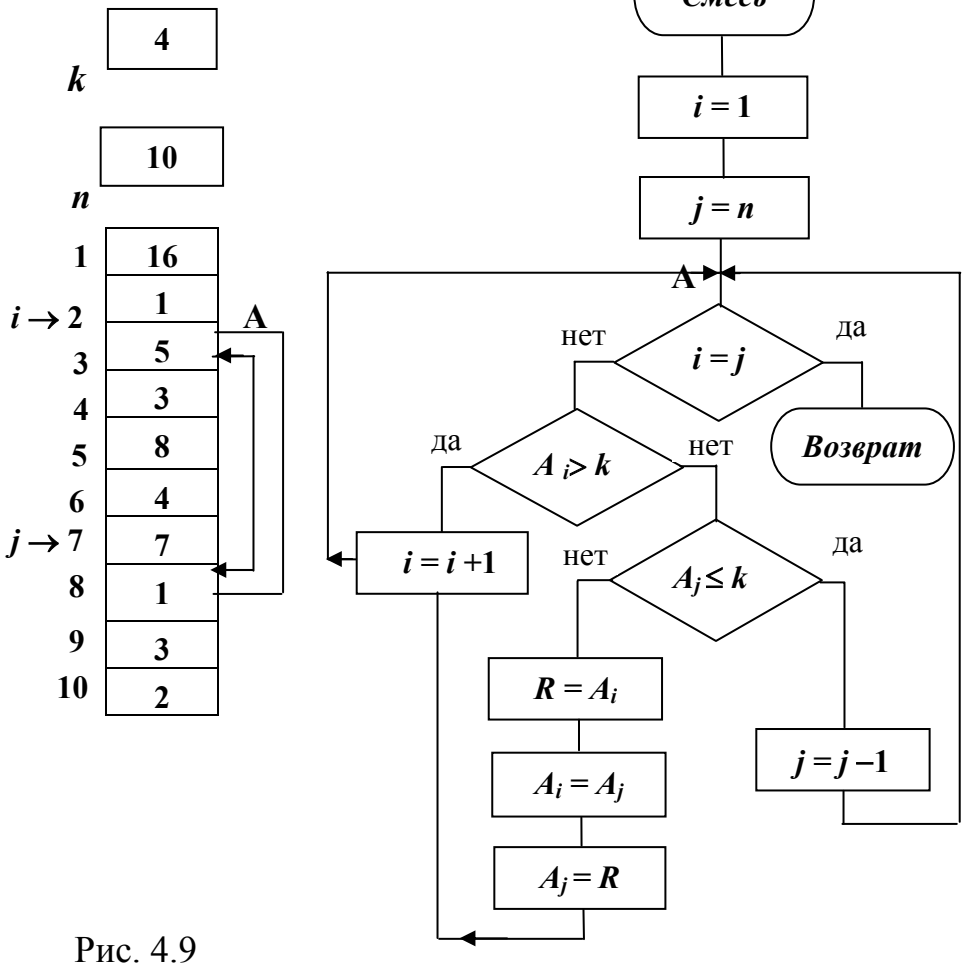


Рис. 4.9

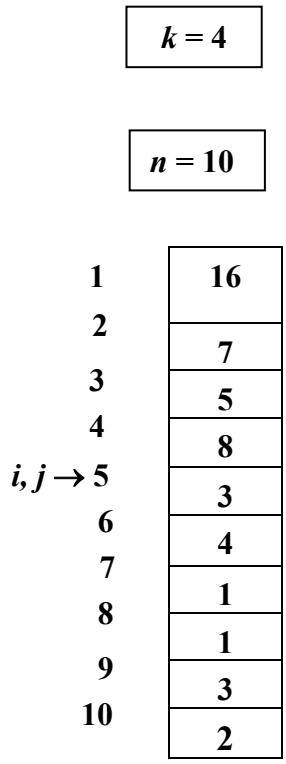


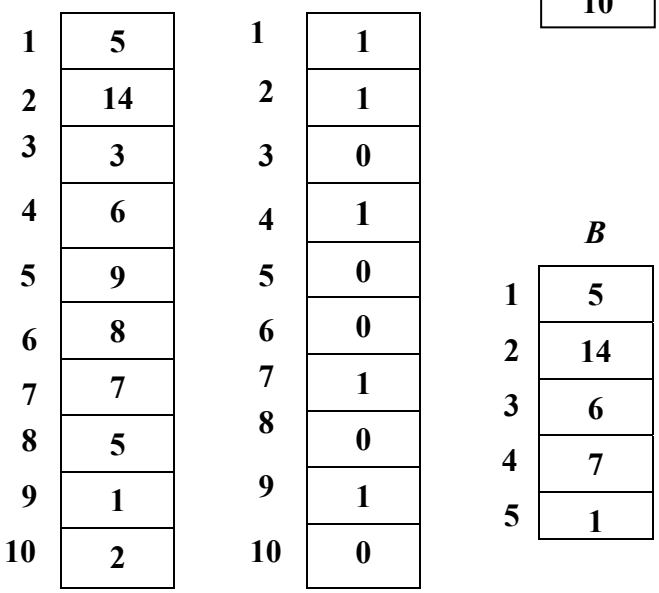
Рис. 4.11

Рис. 4.10

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. 4.9 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. 4.9 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. 4.10 изображена блок-схема алгоритма, реализующего указанную идею. Результирующий массив для примера, заданного на рис. 4.9, приводится на рис. 4.11.

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. 4.12.



	A	Q
1	5	1
2	14	1
3	3	0
4	6	1
5	9	0
6	8	0
7	7	1
8	5	0
9	1	1
10	2	0

	B
1	5
2	14
3	6
4	7
5	1

Рис. 4.12

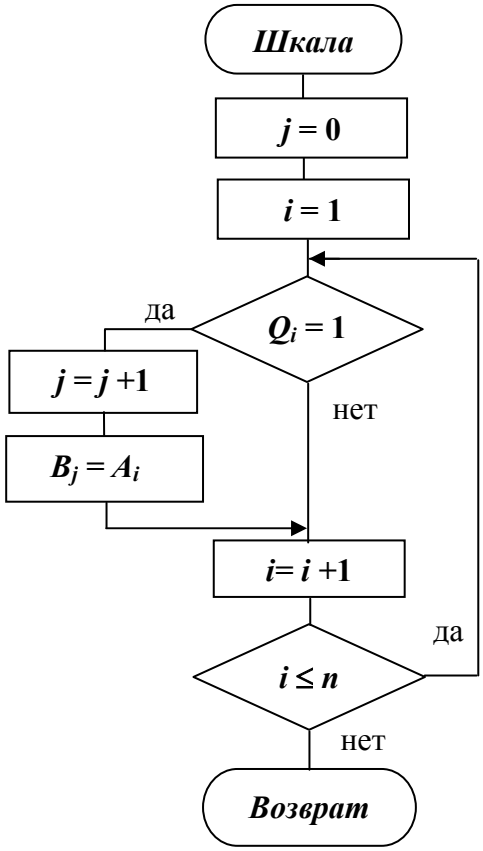


Рис. 4.13

Массив Q за его специфическую роль часто называют *логической шкалой,* или *маской*. Блок-схема решения этой задачи приведена на рис. 4.13. Длиной результирующего массива B (рис. 4.12) будет последнее по порядку значение индекса j.

У п р а ж н е н и я

1. Запишите в обратном порядке элементы массива A длины n.
2. В квадратной таблице размером $n \times n$ поменяйте местами строки и столбцы (транспонирование матрицы).
3. 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.
4. Осуществите сжатие массива A длины n, удалив из него все нулевые элементы.
5. Осуществите сжатие массива A длины n, удалив из него каждый k-й элемент.
6. В квадратной таблице размером $n \times n$ поменяйте местами i-ю и j-ю строки, а затем i-й и j-й столбцы.

4.4. 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. 4.14. Обратите внимание, что в этой блок-схеме имеются «Возврат 1» и «Возврат 2».

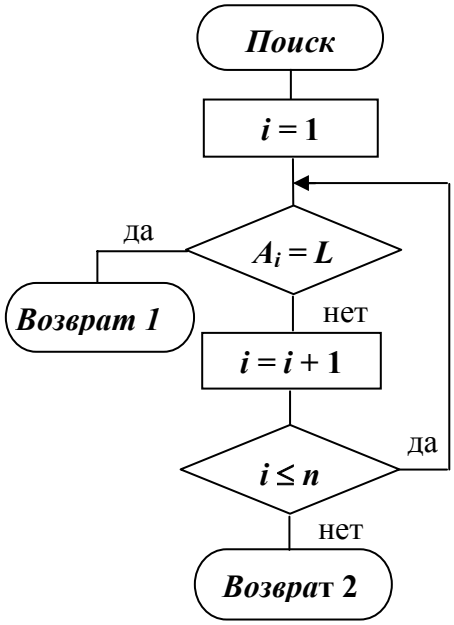


Рис. 4.14

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

Рис. 4.15

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 × n,*** т.е. двумерный массив ***B***, содержащий ***m*** строк и ***n*** столбцов (рис. 4.15). 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. 4.14, только проверка условия $A_i = L$ заменяется проверкой $B_{i1} = L$. После найденного номера строки ***i*** можно сразу обра-

щаться 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. 4.16). 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.

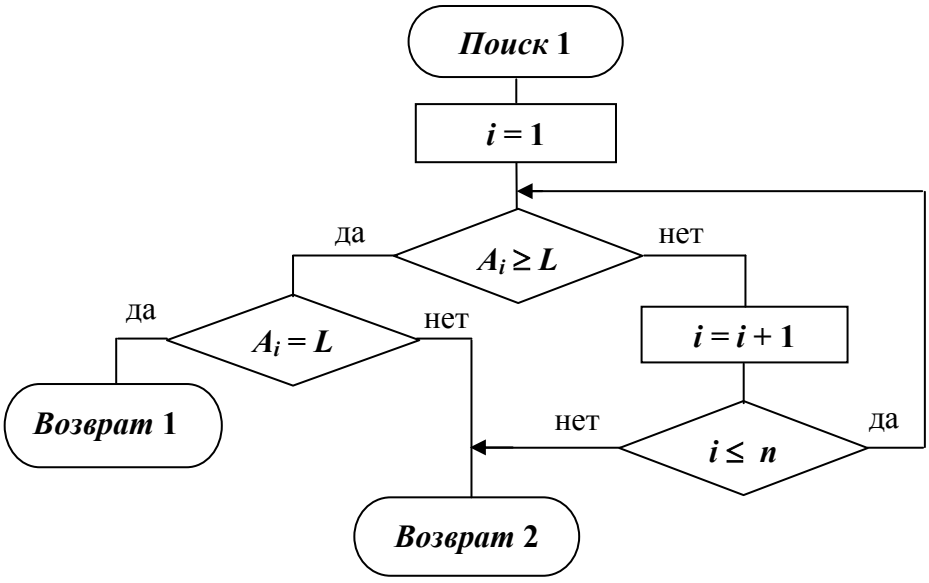


Рис. 4.16

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, стоящего 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. 4.17), сложнее предыдущего. Основным объектом его обработки является суживающийся фрагмент

массива ***A***. Введем следующие обозначения: ***i*** - начало фрагмента, ***j*** – конец фрагмента, ***k*** – полудлина фрагмента.

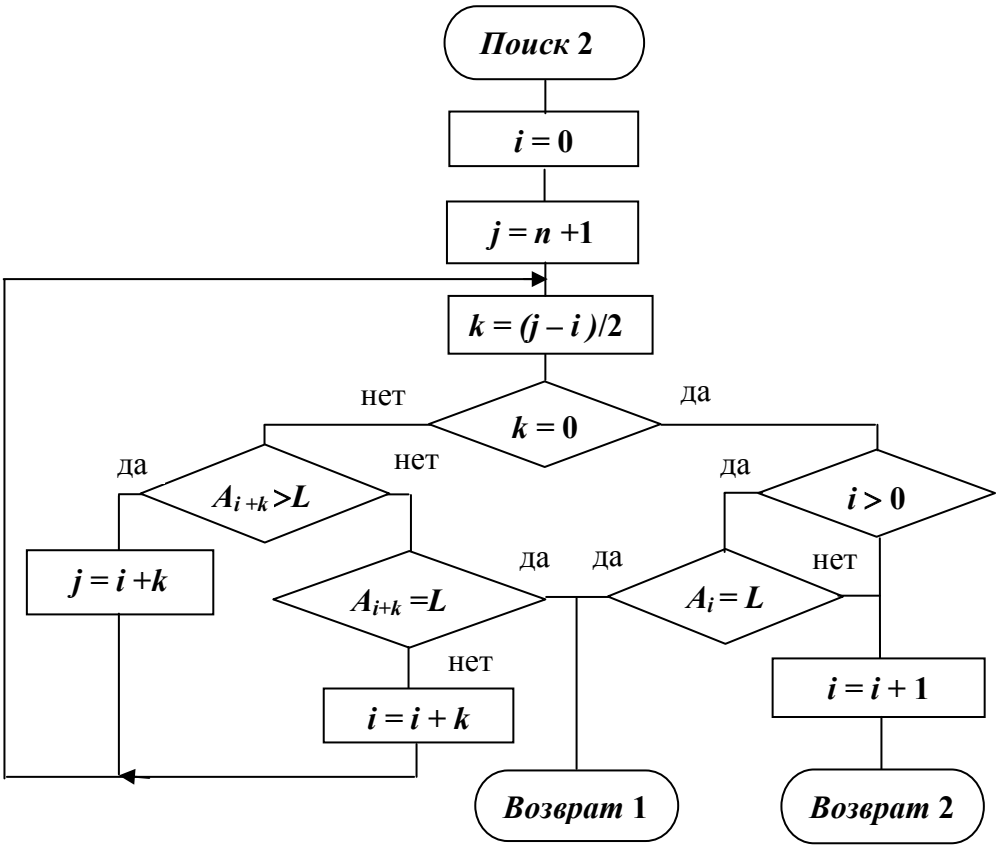


Рис. 4.17

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

У п р а ж н е н и я

1. 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.
2. 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.
3. Выполните предыдущее упражнение в предположении, что массив упорядочен по возрастанию своих элементов.

4.5. 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. 4.18). 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 при организации вставки нового элемента, т.к. при этом необходим сдвиг в среднем на **n/2** элементов массива. Создадим теперь более жесткие условия.

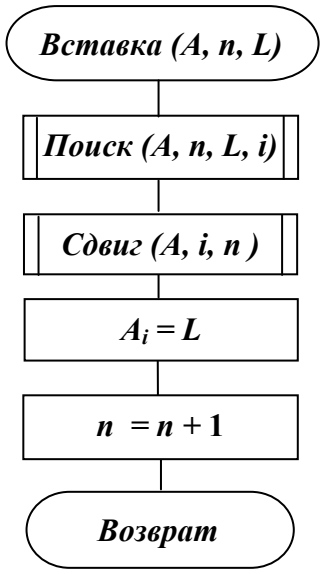


Рис. 4.18

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 специаль-

ных 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.

1. 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.
2. 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.

У п р а ж н е н и я

1. 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.
2. 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.

4.6. 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. 4.19). Его принципиальная роль в организации информационных массивов видна из следующего обобщения.

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. 4.20 изображены с конкретным числовым наполнением два массива длины 5. Так как 53918< 54314, то ***c*** =**0**. Общий алгоритм решения этой задачи приведен на рис. 4.21.

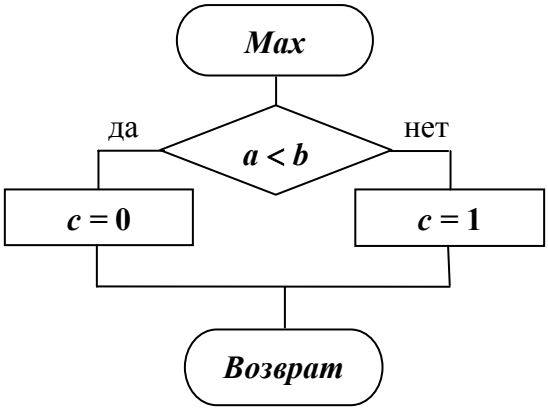


Рис. 4.19

	A		B
1	5	1	5
2	3	2	4
3	9	3	3
4	1	4	1
5	8	5	4

Рис. 4.20

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. 4.22). 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. 4.23. Несколько усложнив ее, можно получить информацию о номере наибольшего элемента в массиве.

A

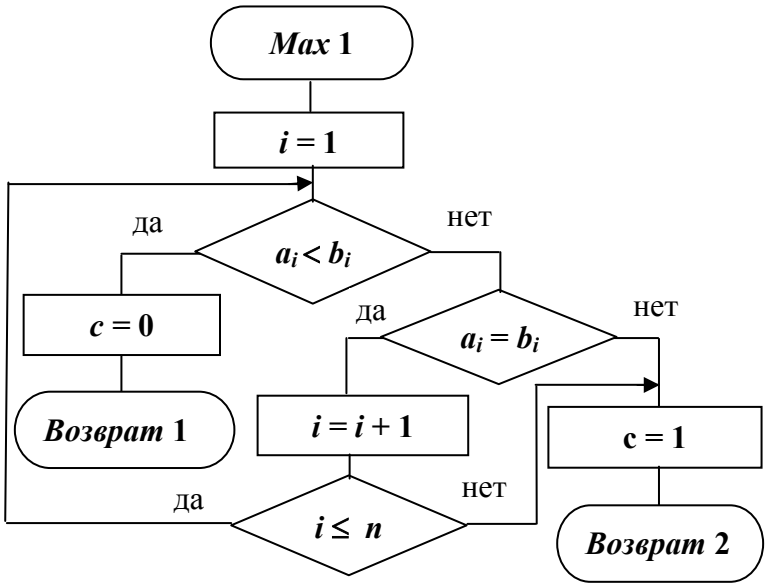


Рис. 4.21

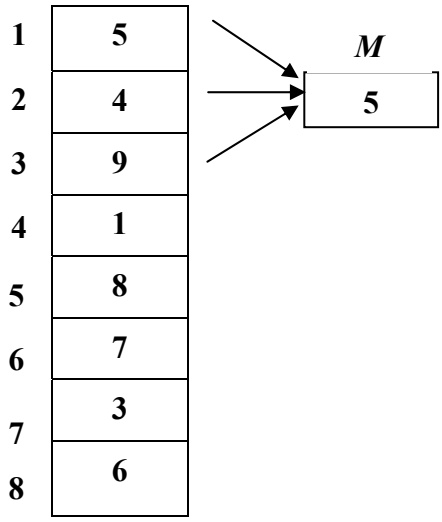


Рис. 4.22

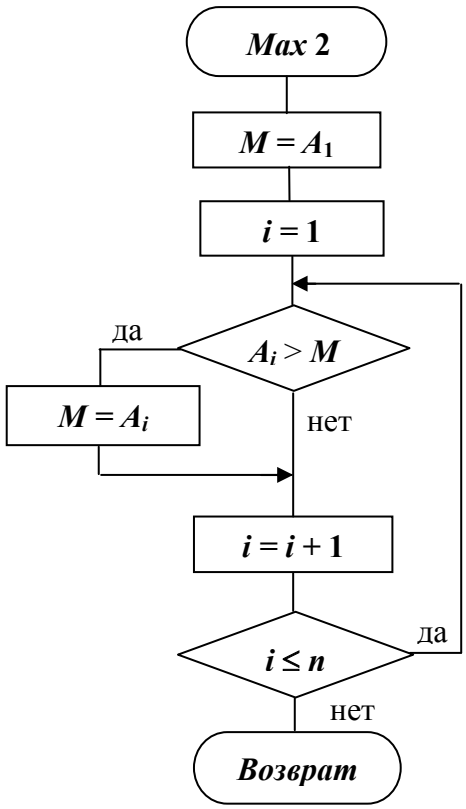


Рис. 4.23

У п р а ж н е н и я

1. В массиве ***A*** длины ***n*** найдите наибольшее число и зафиксируйте его номер.

2. В таблице ***T*** размером ***m*×*n*** найдите наименьшее число и выведите номера содержащих его строки и столбца.
3. Найдите в квадратной таблице наибольший элемент, стоящий на диагонали, и выведите его координаты. Рассмотрите отдельно случаи разных диагоналей.

4.7. 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. 4.24.

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* ≥ 2**.

Следует заметить, что метод последовательного выделения наибольшего (наименьшего) 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 ра-

бочие ячейки для организации перестановки двух элементов и запоминания факта ее проведения.

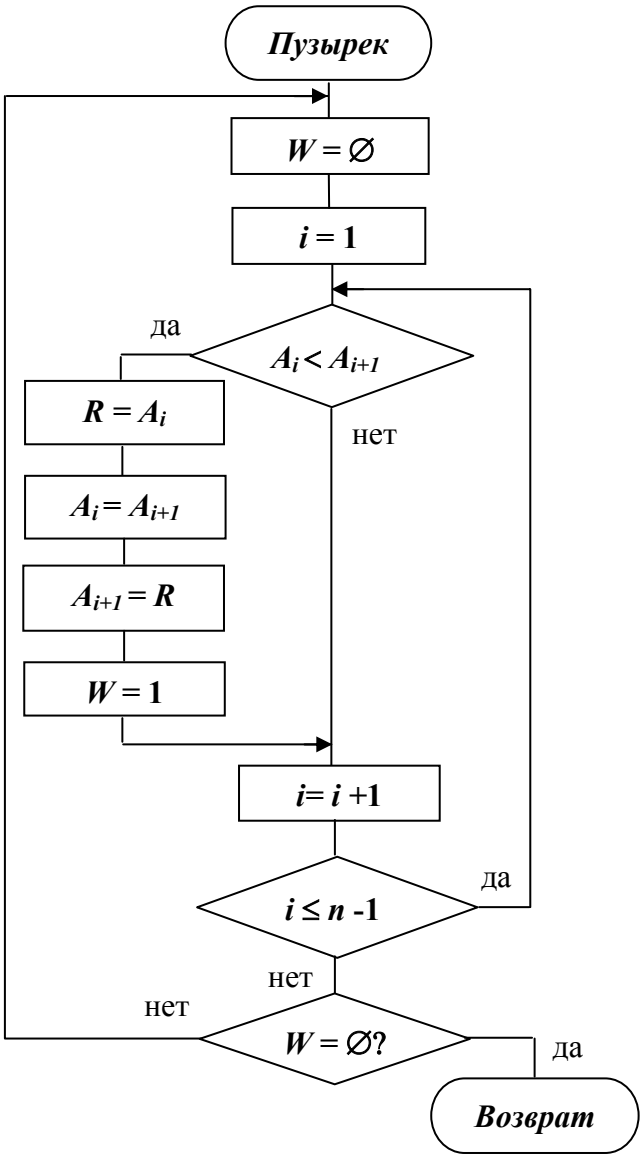


Рис. 4.24

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, 2, 3, 7/. Здесь мы рассмотрим лишь одну задачу, имеющую большое самостоятельное значение, способ решения которой лежит в основе самого быстрого упорядочения. Пусть да-

ны 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. 4.25 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, 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. 4.26. Общее число повторений цикла этого алгоритма равно ***m***+***n***, т.е. длине результирующего массива. Это значит, что более быстрого алгоритма в принципе составить невозможно.

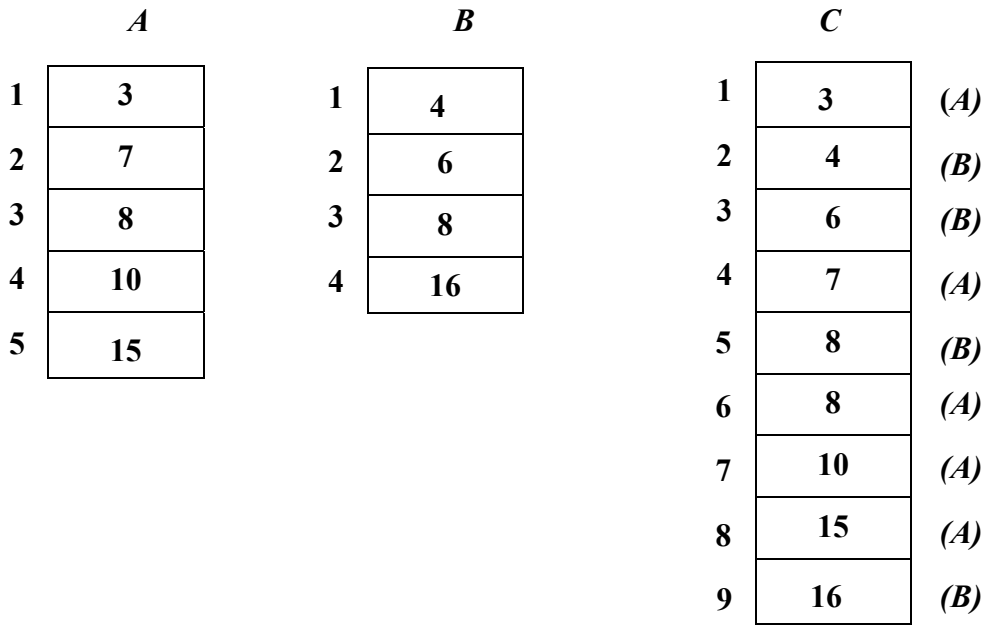


Рис. 4.25

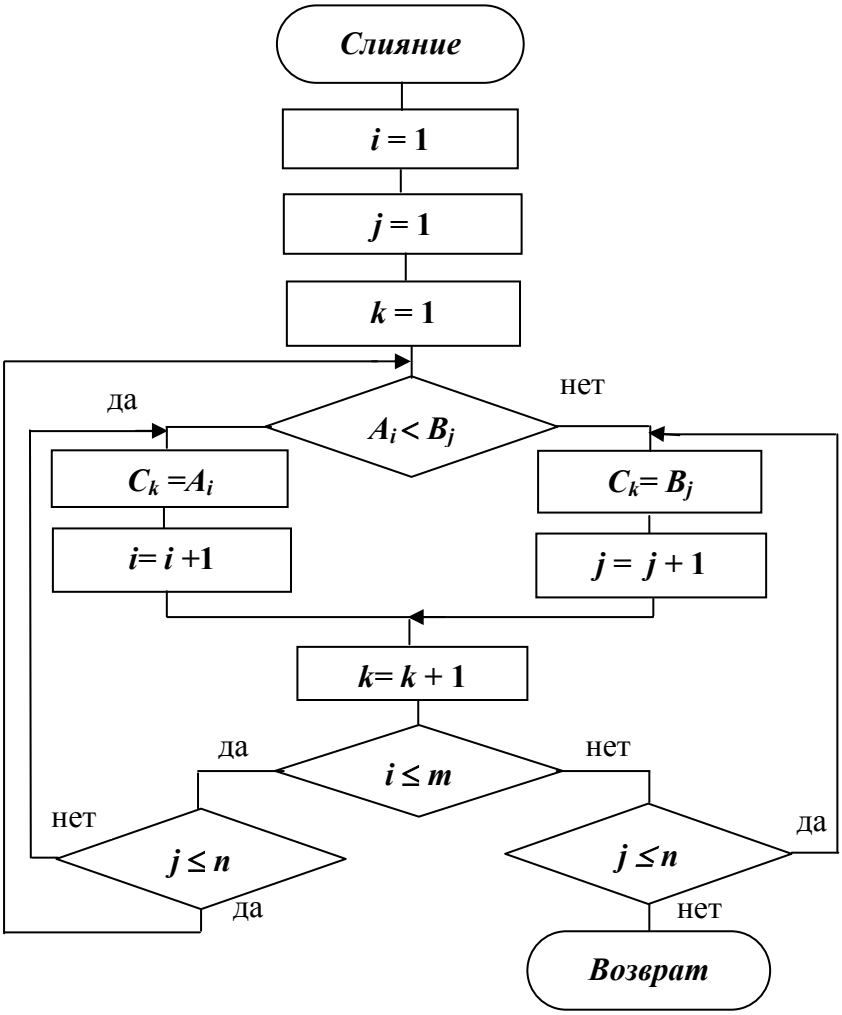


Рис. 4.26

У п р а ж н е н и я

1. Составить блок-схему алгоритма упорядочения массива ***A*** длины ***n*** способом последовательного выбора наибольших элементов.
2. 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***).

Заключение

В соответствии с государственными образовательными стандартами построение общеобразовательного курса «Информатика» таково, что вначале рассматриваются общие вопросы информатики, а затем программирование и методы вычислений. Распределение часов по дисциплине не позволяет в рамках аудиторных занятий подробно рассмотреть все темы учебного плана на достаточном уровне, и в этой ситуации максимально продуктивно должны быть использованы часы внеаудиторной работы.

Поэтому учебное пособие ставит целью помочь студентам детально разобраться в алгебре контактных схем и элементах булевой алгебры, изучить характеристики функциональных элементов и двоичное сложение на многоразрядном сумматоре, арифметические основы ЭВМ и работу с информационными массивами. Знание этих разделов может позволить специалистам те или иные задачи практики перевести в область формальных логических преобразований и, упростив решение, оптимизировать схемотехнику вычислительного устройства.

Учебное пособие должно составлять необходимое информационное звено между аудиторными занятиями и самостоятельной работой студентов в учебном процессе.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1. Могилев, А.В. Информатика. / А.В. Могилев, Н.И. Пак, Е.К. Хеннер. - М.: Academia.- 2001.-816 с.
2. Информатика: учебник: рек. МО РФ /под ред. Н.В. Макаровой. -3-е изд., перераб. - М.: Финансы и статистика, 2005.-765 с.
3. Информатика: Практикум по технологии работы на компьютере. учеб. пособие / под ред. Макаровой Н.В. - 3-е изд.- М.:Финансы и статистика, 2005. – 255 с.
4. Лесничная , И.Г. Информатика и информационные технологии: учеб. пособие /И.Г. Лесничная, И.В. Миссинг, Ю.Д. Романова, В.И. Шестаков. - М.: Эксмо, 2005. -542с.
5. Острейковский, В.А. Информатика: учеб. для вузов. / В.А. Острейковский. - М.: Высш. шк., 2000.-511с.
6. Цилькер, Б.Я. Организация ЭВМ и систем: учеб. для вузов. / Б.Я. Цилькер. - СПб.: Питер.- 2007. – 434с.
7. Лабораторный практикум по информатике: учеб. пособие /под ред. В.А. Острейковского. -М.: Высш. шк., 2003.-275с.

ОГЛАВЛЕНИЕ

Учебное издание

Гильмутдинов Владимир Исламович,
Кононов Александр Давыдович,
Кононов Андрей Александрович

Информатика

Учебное пособие для самостоятельной работы студентов всех специальностей

Редактор Акритова Е.В.

Подписано в печать 16.02.2010. Формат 60×84 1/16. Уч.-изд. л. 3,4. Усл.-печ. л. 3,5.
Бумага писчая. Тираж 260 экз. Заказ №

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