МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Методические указания
по выполнению лабораторных работ
по дисциплине
«Системы обработки и передачи информации»
для студентов направления подготовки
15.04.04 Автоматизация технологических процессов и производств
направленность (профиль):
Информационно-управляющие системы

Невинномысск 2024 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Знание основ машинной арифметики необходимо любому специалисту, использующему в своей профессиональной деятельности цифровую технику. Современные кодировки и алгоритмы машинной математики рассматриваются в данных указаниях.

Правильное понимание излагаемого материала требует знакомства с различными системами счисления (особенно с двоичной) и основами логической алгебры. Необходимые сведения о них приведены в приложениях А и Б.

1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1 Форматы данных

Информация любой природы кодируется *двоичными словами* фиксированной длины. Эти слова разбиваются на *слоги*, имеющие длину ***8*** разрядов (один байт). *Длина слова всегда кратна длине слога*. Минимально различимые элементы кода, имеющие значение ***0*** или ***1***, называются **разрядами**. Совокупность устройств, служащих для размещения всех разрядов кода, называется **разрядной сеткой.**

Точка, отделяющая дробную часть, специально не отображается. Место, где она находится, определяется положением двоичных цифр числового кода относительно разрядной сетки. Способ размещения числа в разрядной сетке называется **форматом данных.** В настоящее время применяются два формата – с фиксированной и с плавающей точкой.

При кодировании данных в **формате с фиксированной точкой** предполагается, что эта точка размещается после младшего разряда сетки. *Так хранятся целые числа.*

Различают форматы чисел со знаком и без знака. В беззнаковом формате все разряды сетки используются для фиксации двоичных цифр кода. В знаковом формате старший разряд сетки используется

для кодирования знака числа (у положительных чисел его содержимое ***0***, у отрицательных чисел – ***1***). Условное изображение двоичного числа ***X*** длиной ***N*** разрядов в формате с фиксированной точкой показано на рисунке 1.1 (x_i – код двоичной цифры; S_X, ***sign*** – код знака числа; сверху от сетки показаны веса младшего и старшего двоичного разрядов числа).

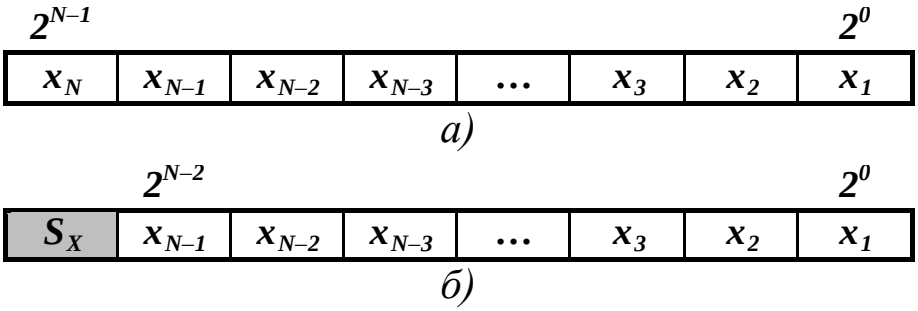


Рисунок 1.1 – Формат чисел с фиксированной точкой: (а) без знака; (б) со знаком

Для представления со знаком двоичного числа ***X*** разрядов обычно используются две кодировки.

В прямом коде $X_{Пр} = X$, если $X \geq 0$; $X_{Пр} = 2^{N-1} + |X|$, если $X < 0$. Для отрицательных чисел такое кодирование эквивалентно установке единицы в знаковый разряд. Нулевое значение в прямом коде представляется *неоднозначно* (различают «положительный» и «отрицательный» нули).

В дополнительном коде $X_{Доп} = X$, если $X \geq 0$; $X_{Доп} = 2^N - |X|$, если $X < 0$. Для отрицательных чисел это эквивалентно инверсии всех разрядов, включая знаковый, и добавлению единицы к младшему разряду полученного кода (разряд знака при этом обрабатывается по общим правилам; если из него возникает единица переноса, она игнорируется). Нулевое значение в дополнительном коде представляется *однозначно*.

Пример. *Представление двоичных чисел в различных кодах (**N** = **8**, нижним индексом обозначено основание системы счисления). Выделены знаковые разряды и единица переноса из старшего разряда,*

которая будет проигнорирована. Для прямого кода иллюстрируются неоднозначность представления нуля.

Прямой код:

$+27_{10} = \boxed{0}0011011_2$, $+0_{10} = \boxed{0}0000000_2$,

$-27_{10} = \boxed{1}0011011_2$, $-0_{10} = \boxed{1}0000000_2$.

Дополнительный код:

$+27_{10} = \boxed{0}0011011_2$, $+0_{10} = \boxed{0}0000000_2$,

$-27_{10} = \boxed{1}1100101_2$, $-0_{10} = \boxed{0}0000000_2$. ❶

• • •

В настоящее время наиболее употребительным является дополнительный код.

Диапазоны значений чисел в формате без знака и со знаком в дополнительном коде определяются следующими условиями

$$0 \le X \le 2^N - 1,$$
$$-2^{N-1} \le X_{Доп} \le 2^{N-1} - 1. \qquad (1.1)$$

Наименьшее число без знака имеет во всех разрядах нули, наибольшее – единицы. Наименьшее число в дополнительном коде содержит нули во всех разрядах, кроме $S_X = 1$. Наибольшее число кодируется всеми единицами, кроме $S_X = 0$. Ширина диапазонов в обоих случаях одинакова (это вытекает из правил комбинаторики: разрядная сетка конечной длины может воспроизвести строго определенное число сочетаний нулей и единиц). *Если значение числа выходит за границы допустимого диапазона, фиксируется состояние переполнения (**overflow**).*

Достоинствами чисел в фиксированном формате являются высокая эффективность их обработки и отсутствие погрешности представления в разрядной сетке. Недостатками являются относительно неширокий диапазон значений и невозможность решения большинства задач только в рамках целого типа данных.

Формат с фиксированной точкой может использоваться для **двоичного кодирования целых десятичных чисел** со знаком (***BCD***, от англ. ***Binary Coded Decimal*** – двоично-кодированное десятичное число). *Каждая десятичная цифра изображается четырехразрядным двоичным числом в коде* ***8-4-2-1*** *в диапазоне* $\boldsymbol{0000_2 \ldots 1001_2}$. Длина ***BCD***-чисел определяется характером решаемой задачи и может быть переменной (от ***1*** до ***16*** байт).

Десятичное число $\boldsymbol{X = S_X x_N x_{N-1} \ldots x_3 x_2 x_1}$ ($\boldsymbol{x_N \ldots x_1}$ – разряды модуля числа) может быть представлено в *зонном* или *упакованном* формате, как показано на рисунке 1.2 (***Z, zone*** – служебный код). Зонный формат применяется только для хранения данных, а упакованный – для их хранения и обработки.

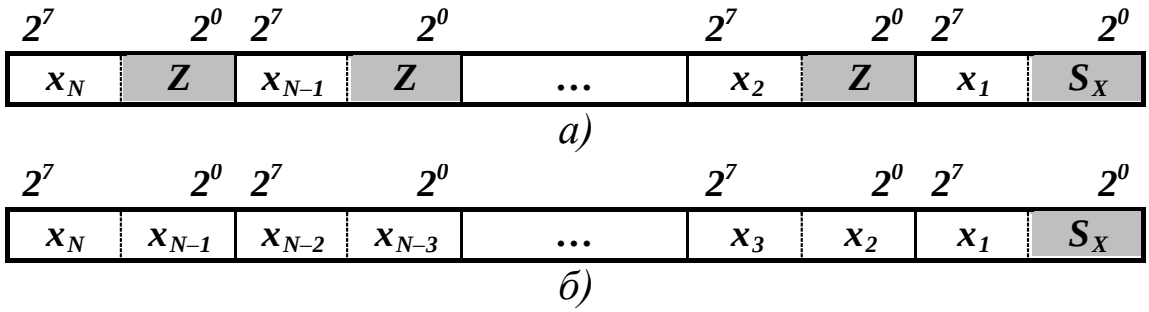


Рисунок 1.2 – Двоичное кодирование целых десятичных чисел: (а) в зонном формате; (б) в упакованном формате

Если четыре старших разряда последнего байта числа в упакованном формате окажутся свободными, их заполняют нулями. Коды знака и зоны располагаются в диапазоне $\boldsymbol{1010_2 \ldots 1111_2}$.

Пример. *Двоичное кодирование десятичных чисел в зонном и упакованном форматах. Выполнено разделение на байты. Выделены коды зоны и знака. Знак «плюс» кодируется значением* $\boldsymbol{1100_2}$*, знак «минус» –* $\boldsymbol{1101_2}$*, зона –* $\boldsymbol{1111_2}$*. Иллюстрируется несовпадение двоично-десятичного и двоичного дополнительного кодов числа.*

$-5749_{10} = \boxed{1}1101001100010\,11_2 = 5(Z)7(Z)4(Z)9(-) =$

$= 0101\boxed{1111}\ 0111\boxed{1111}\ 0100\boxed{1111}\ 1001\boxed{1101}_{2\text{-}10}$

$+5749_{10} = \boxed{\mathbf{0}}0010110011101012 = 05749(+) =$
$= 00000101\ 01110100\ 1001\boxed{\mathbf{1101}}_{2\text{-}10}$

• • •

BCD-коды чисел *не совпадают* с двоичными кодами, что усложняет их обработку. Однако такая форма представления повышает эффективность операций ввода и вывода данных за счет экономии времени, непроизводительно затрачиваемого на машинный перевод чисел из десятичной системы в двоичную и обратно.

При кодировании данных в **формате с плавающей точкой** предполагается, что эта точка размещается перед старшим разрядом сетки. Целая часть числа при этом теряется, а в дробной части (она может иметь бесконечную длину) сохраняется фиксированное число разрядов. Такая форма хранения требует специальной организации данных.

Числа с плавающей точкой представляются в виде

$$X = S_X M \cdot 2^P, \qquad (1.2)$$

где S_X – знак числа; M – **мантисса** или дробная часть числа (правильная двоичная дробь с фиксированной перед старшим разрядом точкой); 2 – характеристика числа, равная основанию системы счисления; P – **порядок** числа[1].

Структура кода числа в плавающем формате приведена на рисунке 1.3 (K – число разрядов порядка; p_i – разряды порядка; L – число разрядов мантиссы; m_i – разряды мантиссы).

---

[1] Термин «плавающая точка» отражает тот факт, что для разных чисел с одной и той же мантиссой положение точки в числе меняется («плавает») в зависимости от значения порядка. Например, в числе $500{,}0 = 0{,}5 \cdot 10^3$ точка в мантиссе располагается перед разрядом сотен, а в числе $5{,}0 = 0{,}5 \cdot 10^1$ – перед разрядом единиц.

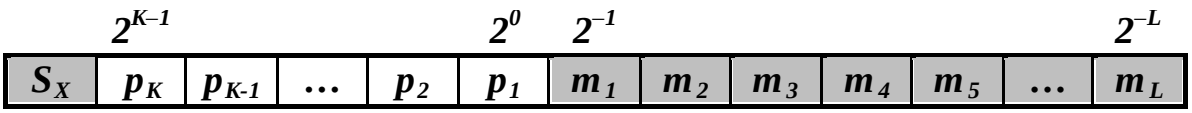


Рисунок 1.3 – Формат числа с плавающей точкой

Перед записью в разрядную сетку число нормализуется так, чтобы не осталось значащих цифр в его целой части и незначащих нулей в дробной. Это исключает неоднозначность представления одного и того же числа с различными порядками.

Пример. *Представления десятичного числа с различными порядками. Выделена нормализованная форма.*

$$27.95 \cdot 10^{1}, \quad 2.795 \cdot 10^{2}, \quad \mathbf{0.2795 \cdot 10^{3}}, \quad 0.02795 \cdot 10^{4}.$$

• • •

Порядок P является целым числом со знаком. Он хранится с избытком 2^{K-1}. Диапазон его значений $-2^{K-1} \ldots 2^{K-1} - 1$ при этом смещается и становится равным $0 \ldots 2^{K} - 1$. Изменение формы хранения упрощает обработку порядков (например, сравнение), которая сводятся к действиям над заведомо положительными числами. Соотношение смещенных P' и несмещенных P порядков одинаково (если $P_1 \geq P_2$, то $P'_1 \geq P'_2$), поэтому результаты операций будут правильными. Так как величина избытка *одинакова* для всех чисел, ее можно учитывать и вносить соответствующую коррекцию.

Примечание. *Часто значение порядка* -2^{K-1} *специально резервируется для обозначения неопределенного числа, а для порядков отводится диапазон* $-2^{K-1} + 1 \ldots 2^{K-1} - 1$. *Соответственно величина избытка становится равной* $2^{K-1} - 1$.

Дробная часть числа может храниться в различных кодах. **В прямом коде** $M_{Пр} = M$, если $M \geq 0$, иначе $M_{Пр} = 1 + |M|$, что соответствует установке единицы в знаковый разряд. В **дополнительном коде** $M_{Доп} = M$, если $M \geq 0$, иначе $M_{Доп} = 2 - |M|$. Для отрица-

тельных чисел это эквивалентно инверсии всех разрядов, включая знаковый, и добавлению единицы к младшему разряду полученного кода с игнорированием переноса из разряда знака (так же, как и для чисел в фиксированном формате). В прямом коде нулевое значение представляется неоднозначно. *Для хранения чисел, как правило, используется прямой код, а для их обработки – дополнительный.*

Пример. *Представление мантиссы двоичных чисел в различных кодах* ($L = 4$). *Выделены знаковые разряды и единица переноса из старшего разряда, которая будет проигнорирована. Для прямого кода иллюстрируются неоднозначность представления нуля.*

Прямой код:

$+0.75_{10} = \boxed{0}.1100_2$, $\quad +0.00_{10} = \boxed{0}.0000_2$,

$-0.75_{10} = \boxed{1}.1100_2$, $\quad -0.00_{10} = \boxed{1}.0000_2$.

Дополнительный код:

$+0.75_{10} = \boxed{0}.1100_2$, $\quad +0.00_{10} = \boxed{0}.0000_2$,

$-0.75_{10} = \boxed{1}.0100_2$, $\quad -0.00_{10} = \textcircled{1}\boxed{0}.0000_2$.

• • •

Если число нормализовано и отлично от нуля (этот случай распознается и обрабатывается особо), то для прямого кода разряды модуля мантиссы M находятся в диапазоне от 2^{-1} (единица в старшем разряде M и нули во всех остальных) до $1 - 2^{-L}$ (единицы во всех разрядах). Самый старший разряд дробной части *всегда равен единице*, поэтому хранение его считается излишним. Он исключается из разрядной сетки путем однократного сдвига M влево (что эквивалентно умножению на 2, см. п. 1.2). Чтобы абсолютное значение числа не изменилось, величину P уменьшают на единицу.

Примечание. *Из-за того, что старший разряд мантиссы не хранится, но всегда учитывается, он получил название «скрытого разряда нормализации».*

Пример. *Представление чисел в стандартном формате с плавающей точкой (**K** = **8**, **L** = **23**, избыток равен $2^{8-1} - 1 = 127$). Выделены разряды знака и мантиссы, цветом выделены скрытые разряды нормализации. Иллюстрируется смещение порядка и его уменьшение при сдвиге мантиссы влево.*

$$X = +0.75 = 0.5+0.25 = +(2^{-1}+2^{-2})\cdot 2^{0}$$

Значения	**S**	**P**	**M**
Исходные	0	00000000	11000000000000000000000
После смещения порядка	0	01111111	11000000000000000000000
После сдвига мантиссы	0	01111110	11000000000000000000000
С разделением на байты	0	0111111 0	1000000 00000000 00000000$_2$

$$X = -3.5 = -(2.0+1.0+0.5) = -(2^{-1}+2^{-2}+2^{-3})\cdot 2^{2}$$

Значения	**S**	**P**	**M**
Исходные	1	00000010	11100000000000000000000
После смещения порядка	1	10000001	11100000000000000000000
После сдвига мантиссы	1	10000000	11100000000000000000000
С разделением на байты	1	1000000 0	1100000 00000000 00000000$_2$

• • •

С учетом диапазонов для ***M*** и ***P*** область абсолютных значений чисел с плавающей точкой можно определить следующим образом

$$2^{-1}\cdot 2^{-2^{K-1}} \le |X| \le \left(1-2^{-L}\right)\cdot 2^{2^{K-1}-1}. \qquad (1.3)$$

Так как параметр ***L*** слабо влияет на входящие в эти соотношения величины, то можно сказать, что границы диапазона определяются количеством разрядов ***K***, отводимых на хранение порядка.

Если число по модулю выходит за верхнюю границу диапазона, фиксируется переполнение (***overflow***). Если число становится по модулю меньше минимального допустимого значения, оно вырождается в ноль (***underflow***).

Дробная часть числа в общем случае является бесконечной двоичной дробью. При ее хранении разряды с номерами свыше L отбрасываются, а оставшаяся часть обрабатывается по правилам округления. Если $m_{L+1} = 1$, то последний хранимый разряд m_L будет увеличен на единицу (что эквивалентно добавлению единицы к старшему из отбрасываемых разрядов). Возникающие при округлении погрешности, как правило, имеют разные знаки. В большинстве случаев их влияние взаимно компенсируется. Среднее значение погрешности округления считается приближенно равным половине последнего хранимого разряда мантиссы, т. е. $\overline{\Delta} \approx 0{,}5 \cdot 2^{-L}$. Значение этого показателя целиком определяется числом разрядов L, отводимых для хранения дробной части числа.

Достоинствами чисел в плавающем формате являются их универсальность и широкий диапазон допустимых значений. Недостатками являются наличие погрешности представления в разрядной сетке ЭВМ и усложнение операций обработки таких данных.

1.2 Действия над числами в формате с фиксированной точкой

Над числами в фиксированном формате выполняются следующие виды операций.

СДВИГИ. Они заключаются в перемещении числа по разрядной сетке влево или вправо. Эти операции *очень эффективные*.

Логические сдвиги перемещают число без учета его знака. Выдвигаемые разряды при их выполнении теряются, на освободившиеся места в разрядной сетке записываются нули. **Арифметические сдвиги**, в отличие от логических, сохраняют знаковый разряд (при сдвиге вправо он воспроизводится). **Циклические сдвиги** или *ротации* вращают содержимое разрядной сетки, помещая выдвинутые разряды обратно в число.

Примечание. *Циклические сдвиги требуют временного хранения значения переносимой двоичной цифры. В реальных вычислительных системах для этого используется разряд специального регистра процессора (т. н. «флажок»* ***C, Carry*** *– в данном контексте означает «перенос»).*

Пример. *Логические сдвиги чисел в дополнительном коде влево и вправо на один разряд. Выделены знаковые разряды.*

X	=	$\boxed{0}0100101_2$	= $+37_{10}$	X	=	$\boxed{1}1011011_2$	= -37_{10}
$\Leftarrow X$	=	$\boxed{0}1001010_2$	= $+74_{10}$	$\Leftarrow X$	=	$\boxed{1}0110110_2$	= -74_{10}
$X \Rightarrow$	=	$\boxed{0}0010010_2$	= $+18_{10}$	$X \Rightarrow$	=	$\boxed{0}1101101_2$	= $+109_{10}$

• • •

Пример. *Арифметические сдвиги чисел в дополнительном коде влево и вправо на один разряд. Выделены знаковые разряды.*

X	=	$\boxed{0}0100101_2$	= $+37_{10}$	X	=	$\boxed{1}1011011_2$	= -37_{10}
$\Leftarrow X$	=	$\boxed{0}1001010_2$	= $+74_{10}$	$\Leftarrow X$	=	$\boxed{1}0110110_2$	= -74_{10}
$X \Rightarrow$	=	$\boxed{0}0010010_2$	= $+18_{10}$	$X \Rightarrow$	=	$\boxed{1}1101101_2$	= -19_{10}

• • •

Очевидно, что арифметический сдвиг двоичного числа влево на один разряд эквивалентен умножению его на два, сдвиг вправо на один разряд – делению нацело на два.

Пример. *Циклические сдвиги чисел в дополнительном коде влево и вправо на один разряд. Выделены знаковые разряды.*

X	=	$\boxed{0}0100101_2$	= $+37_{10}$	X	=	$\boxed{1}1011011_2$	= -37_{10}
$\Leftarrow X$	=	$\boxed{0}1001010_2$	= $+74_{10}$	$\Leftarrow X$	=	$\boxed{1}0110111_2$	= -73_{10}
$X \Rightarrow$	=	$\boxed{1}0010010_2$	= -110_{10}	$X \Rightarrow$	=	$\boxed{1}1101101_2$	= -19_{10}

• • •

ЛОГИЧЕСКИЕ ОПЕРАЦИИ. Они выполняются по законам алгебры логики *по отдельности* над каждым разрядом числа или

над каждой парой соответствующих разрядов чисел. Скорость выполнения операций *очень высокая.*

Основными считаются логические операции «***НЕ***» – инверсия (**¬*X*** и **¬*Y***), «***И***» – конъюнкция (***X*** ∧ ***Y***), «***ИЛИ***» – дизъюнкция (***X*** ∨ ***Y***), «***Исключающее ИЛИ***» – неравнозначность или сложение по модулю **2** (***X*** ⊕ ***Y***).

Пример. *Логические операции над числами в дополнительном коде. Выделены знаковые разряды.*

X	=	$\boxed{0}0011111_2$ = **+31**$_{10}$	***X***	=	$\boxed{0}1000001_2$ = **+65**$_{10}$
Y	=	$\boxed{1}1000111_2$ = **−57**$_{10}$	***Y***	=	$\boxed{1}0111111_2$ = **−65**$_{10}$
¬*X*	=	$\boxed{1}1100000_2$ = **−32**$_{10}$	**¬*X***	=	$\boxed{1}0111110_2$ = **−66**$_{10}$
¬*Y*	=	$\boxed{0}0111000_2$ = **+56**$_{10}$	**¬*Y***	=	$\boxed{0}1000000_2$ = **+64**$_{10}$
X ∧ ***Y***	=	$\boxed{0}0000111_2$ = **+7**$_{10}$	***X*** ∧ ***Y***	=	$\boxed{0}0000001_2$ = **+1**$_{10}$
X ∨ ***Y***	=	$\boxed{1}1011111_2$ = **−33**$_{10}$	***X*** ∨ ***Y***	=	$\boxed{1}1111111_2$ = **−1**$_{10}$
X ⊕ ***Y***	=	$\boxed{1}1011000_2$ = **−40**$_{10}$	***X*** ⊕ ***Y***	=	$\boxed{1}1111110_2$ = **−2**$_{10}$

• • •

С помощью логических операций выполняются проверка значений отдельных разрядов двоичных чисел и сброс (установка) их в нужное состояние. Для этого используются специально подобранные числа – «маски» (***masks***).

Пример. *Проверка состояния заданных разрядов чисел в дополнительном коде. Выделены знаковые разряды. Маски* ***M*** *подобраны так, чтобы сформировать нужный результат. Индексы обозначают номера разрядов.*

X	=	$\boxed{0}0101001_2$ = **+41**$_{10}$,	***X***	=	$\boxed{0}0000000_2$ = **0**$_{10}$,
Y	=	$\boxed{0}0100100_2$ = **+36**$_{10}$,	***Y***	=	$\boxed{0}0111000_2$ = **+56**$_{10}$,
M	=	$\boxed{0}0001000_2$ = **+8**$_{10}$,	***M***	=	$\boxed{1}1111111_2$ = **−1**$_{10}$,
X ∧ ***M***	=	$\boxed{0}0001000_2$ (***X***$_4$≠**0**),	***X*** ∧ ***M***	=	$\boxed{0}0000000_2$ (***X***=**0**),
Y ∧ ***M***	=	$\boxed{0}0000000_2$ (***Y***$_4$=**0**).	***Y*** ∧ ***M***	=	$\boxed{0}0111000_2$ (***Y***≠**0**).

• • •

Пример. *Получение нулевых и единичных значений групп разрядов числа в дополнительном коде. Выделены знаковые разряды. Маски подобраны так, чтобы сформировать нужный результат.*

$X = \boxed{0}0101011_2 = +43_{10},$		$X = \boxed{0}0011011_2 = +27_{10},$	
$M = \boxed{0}0001111_2 = +15_{10},$		$\neg X = \boxed{1}1100100_2 = -28_{10},$	
$X \wedge M = \boxed{0}0001011_2 = +11_{10},$		$X \oplus X = \boxed{0}0000000_2 = 0_{10},$	
$X \vee M = \boxed{0}0101111_2 = +47_{10}.$		$X \oplus \neg X = \boxed{1}1111111_2 = -1_{10}.$	

• • •

СЛОЖЕНИЕ И ВЫЧИТАНИЕ $Z = X \pm Y$. Для чисел в дополнительном коде они выполняются по одинаковым правилам. Вычитание заменяется сложением с отрицательным числом. Обработка дополнительных кодов чисел ведется, начиная с младших разрядов, с учетом формирующихся единиц переноса C. Разряды знака обрабатываются по общим правилам. Если при этом возникает единица переноса, она игнорируется.

Алгоритм. *Сложение* N*-разрядных чисел с одновременным формированием переноса. Индексы обозначают номера разрядов.*

Шаг 1. Сброс переноса $C_1 \leftarrow 0$.

Шаг 2. Для i от 1 до N выполнить $(C_{i+1}, Z_i) \leftarrow X_i + Y_i + C_i$.

Разряд суммы Z_i определяется сложением по модулю 2 разрядов слагаемых X_i и Y_i и переноса P_i (если $X_i + Y_i + C_i \geq 2$, то $Z_i = X_i + Y_i + C_i - 2$); значение P_{i+1} получается делением $X_i + Y_i + C_i$ на 2 нацело

$$Z_i = (X_i + Y_i + C_i) \bmod 2, \quad P_{i+1} = \left\lfloor \frac{X_i + Y_i + C_i}{2} \right\rfloor. \qquad (1.4)$$

Пример. *Сложение чисел в дополнительном коде. Выделены знаковые разряды и единицы переноса из старших разрядов, которые будут проигнорированы. Иллюстрируется формальное соблюдение*

правил двоичной арифметики.

X	=	$\boxed{0}0100100_2$	$= +36_{10}$
Y	=	$\boxed{0}0001110_2$	$= +14_{10}$
$X+Y$	=	$\boxed{0}0110010_2$	$= +50_{10}$

X	=	$\boxed{0}0100100_2$	$= +36_{10}$
Y	=	$\boxed{1}1110010_2$	$= -14_{10}$
$X+Y$	= ❶	$\boxed{0}0010110_2$	$= +22_{10}$

X	=	$\boxed{1}1011100_2$	$= -36_{10}$
Y	=	$\boxed{0}0001110_2$	$= +14_{10}$
$X+Y$	=	$\boxed{1}1101010_2$	$= -22_{10}$

X	=	$\boxed{1}1011100_2$	$= -36_{10}$
Y	=	$\boxed{1}1110010_2$	$= -14_{10}$
$X+Y$	= ❶	$\boxed{1}1001110_2$	$= -50_{10}$

• • •

При выполнении операций сложения (вычитания) может возникнуть **переполнение разрядной сетки**. «Положительное» переполнение распознается по наличию единицы переноса в знаковый разряд результата при отсутствии переноса из этого разряда. Нулевой знаковый разряд заменяется единицей, что искажает положительный результат. «Отрицательному» переполнению соответствует наличие переноса из знакового разряда результата при отсутствии переноса в этот разряд. Замена единицы в разряде знака нулем приводит к искажению отрицательного результата.

Пример. *Переполнение при операциях сложения и вычитания. Выделены знаковые разряды и единицы переноса из старших разрядов, которые будут проигнорированы. Иллюстрируется «положительное» и «отрицательное» переполнение.*

X	=	$\boxed{0}1111111_2$	$= +127_{10}$
Y	=	$\boxed{0}0000001_2$	$= +1_{10}$
$X+Y$	=	$\boxed{1}0000000_2$	$= -128_{10}$

X	=	$\boxed{1}0000000_2$	$= -128_{10}$
Y	=	$\boxed{1}1111111_2$	$= -1_{10}$
$X+Y$	= ❶	$\boxed{0}1111111_2$	$= +127_{10}$

• • •

УМНОЖЕНИЕ $Z = X \times Y$. Оно начинается с проверки значений сомножителей. Если хотя бы одно из них равно нулю, операция не выполняется, и произведению присваивается нулевое значение.

Нахождение произведения Z сводится к циклическому сумми-

рованию со сдвигом последовательно формируемых частичных произведений. На каждом шаге цикла анализируется очередная цифра множителя ***Y***. Если она равна единице, то к сумме частичных произведений добавляется множимое ***X***, иначе добавления не происходит. Цикл завершается сдвигом этой суммы относительно неподвижного множимого. Если умножение начинается с младших разрядов множителя, сдвиг выполняется вправо, а если со старших разрядов – влево.

Примечание. *При перемножении двух* ***N****-разрядных чисел за счет сдвигов на каждом этапе операции будет сформировано* ***2·N****-разрядное произведение. Для экономии аппаратных ресурсов в реальных цифровых устройствах младшая или старшая части произведения (это зависит от направления сдвига) будет занимать место множителя.*

Если умножение производится с отделением знака над абсолютными значениями сомножителей, то используется схема

$$Z = (S_X |X|) \times (S_Y |Y|) = (S_X \oplus S_Y)(|X| \times |Y|). \qquad (1.5)$$

Знак произведения определяется путем сложения знаковых разрядов сомножителей по модулю ***2***. Расчет модуля произведения обычно начинают со старшей цифры множителя, сдвигая сумму частичных произведений влево.

Алгоритм. *Умножение* ***N****-разрядных чисел с отделением знака. Индексы обозначают номера разрядов.*

Шаг 1. Обнуление суммы $\Sigma \leftarrow 0$.
Шаг 2. Для i от N до 1 выполнить шаги $3…4$.
Шаг 3. Арифметический сдвиг влево $\Leftarrow(|Y|, \Sigma)$.
Шаг 4. Если $|Y|_i=1$, то $\Sigma \leftarrow \Sigma+|X|$.
Шаг 5. Формирование результата $Z \leftarrow (S_X \oplus S_Y)(|Y|, \Sigma)$.

Пример. *Умножение чисел с отделением знака со сдвигом суммы*

частичных произведений влево. Анализ множителя начинается со старших цифр по значению разряда ***C****, вышедшему из сетки в процессе сдвига. При умножении разряды множителя* ***Y*** *заменяются старшими разрядами произведения. Выделены знаковые разряды и разряды суммы частичных произведений.*

$X = \boxed{0}011_2 = S_X|X| \rightarrow \boxed{0}\ 0011_2 = +3$

$Y = \boxed{1}110_2 = S_Y|Y| \rightarrow \boxed{1}\ 0010_2 = -2$

$S_Z = S_X \oplus S_Y = 0 \oplus 1 = 1$ *(произведение отрицательное)*

№	C	|Y|	Σ	|X|=0011	Комментарий
0	0	0010	0000		*Исходные значения*
1	0	0100	0000		⇦(|Y|,Σ), C = 0
2	0	1000	0000		⇦(|Y|,Σ), C = 0
3	1	0000	0000	+X	⇦(|Y|, Σ) C = 1 ⇒ Σ ← Σ + X
			0011		
		0000	0011		
4	0	0000	0110		⇦(|Y|,Σ), C = 0
		00000110			|Z| ← (|Y|,Σ)
		$\boxed{1}$1111010			Z ← (S_X ⊕ S_Y)|Z|

Произведение в дополнительном коде $\boxed{1}1111010_2 = -6_{10}$.

• • •

ДЕЛЕНИЕ $Z = X \div Y$. Операция начинается с проверки значений операндов. Если делитель равен нулю, фиксируется ошибочное состояние. Если делимое равно нулю, частному присваивается нулевое значение без выполнения операции.

Деление сводится к последовательному вычитанию делителя сначала из делимого, а затем из образующихся частичных остатков. Как правило, делитель остается неподвижным, а делимое и частичные остатки сдвигаются влево. В результате операции формируется частное Z и остаток от деления Δ.

Деление чисел с отделением знака производится по схеме

$$Z = \frac{S_X |X|}{S_Y |Y|} = (S_X \oplus S_Y)\frac{|X|}{|Y|}. \qquad (1.6)$$

Знак частного находится так же, как и знак произведения. Ниже рассмотрен алгоритм деления с восстановлением остатка при неподвижном делителе.

Алгоритм. *Деление N-разрядных чисел с отделением знака с восстановлением остатка при неподвижном делителе. Индексы обозначают номера разрядов.*

Шаг 1. Обнуление остатка $\Delta \leftarrow 0$.
Шаг 2. Для i от N до 1 выполнить шаги **3…5.**
Шаг 3. Арифметический сдвиг влево $\Leftarrow(\Delta, |X|)$.
Шаг 4. $\Delta \leftarrow \Delta - |Y|$.
Шаг 5. Если $\Delta \geq 0$, то $|Z|_i \leftarrow 1$,
иначе $|Z|_i \leftarrow 0$ и $\Delta \leftarrow \Delta + |Y|$.
Шаг 6. Формирование результата $Z \leftarrow (S_X \oplus S_Y)|Z|$.

Пример. *Деление чисел с отделением знака с восстановлением остатка. В качестве примера использованы четырехразрядные числа. Выделены знаковые разряды, исходные разряды остатка, сформированные разряды частного и единицы переноса из старших разрядов, которые будут проигнорированы. Иллюстрируется формирование цифр частного.*

$X = \boxed{0}111_2 = S_X|X| \rightarrow \boxed{0}\,0111_2 = +7$
$Y = \boxed{1}110_2 = S_Y|Y| \rightarrow \boxed{1}\,0010_2 = -2$
$S_Z = S_X \oplus S_Y = 0 \oplus 1 = 1$ *(частное отрицательное)*

№	$\|Y\|=\boxed{0}010$ $-\|Y\|=\boxed{1}110$	Δ	$\|X\|$	$\|Z\|$	*Комментарии*
0		$\boxed{0}000$	0111	????	*Исходные значения*
1		$\boxed{0}000$	111		$\Leftarrow(\Delta, \|X\|)$
	$-Y$	$\boxed{1}110$			$\Delta \geq 0 \Rightarrow \Delta \leftarrow \Delta - \|Y\|$
		$\boxed{1}110$	111	$0???$	$\Delta < 0 \Rightarrow \|Z\|_4 \leftarrow 0$
	$+Y$	$\boxed{0}010$			$\Delta < 0 \Rightarrow \Delta \leftarrow \Delta + \|Y\|$

	❶	0000	111		
2	−Y	0001	11		⇦(Δ, |X|) Δ≥0 ⇒ Δ ← Δ−|Y|
		1111	11	00??	Δ<0 ⇒ |Z|₃ ← 0
	+Y	0010			Δ<0 ⇒ Δ ← Δ+|Y|
	❶	0001	11		
3	−Y	0011	1		⇦(Δ, |X|) Δ≥0 ⇒ Δ ← Δ−|Y|
	❶	0001	1	001?	Δ≥0 ⇒ |Z|₂ ← 1
4	−Y	0011			⇦(Δ, |X|) Δ≥0 ⇒ Δ ← Δ−|Y|
	❶	0001		0011	Δ≥0 ⇒ |Z|₁ ← 1
		0001		1101	Z ← (S_X ⊕S_Y)|Z|

Частное в дополнительном коде $1101_2 = -3_{10}$, *остаток* $0001_2 = 1_{10}$.

• • •

1.3 Действия над двоично-кодированными числами

Над такими числами выполняются следующие виды операций.

СЛОЖЕНИЕ И ВЫЧИТАНИЕ $Z = X \pm Y$. Эти действия производятся над числами в упакованном формате по одинаковым правилам путем последовательной обработки десятичных цифр. Знаковые тетрады не обрабатываются.

Применяемый для ***BCD***-чисел код ***8-4-2-1*** является *аддитивным* (сумма кодов двух цифр равна коду их суммы), но не *самодополняющимся* (инверсия кода десятичной цифры не дает кода ее дополнения до девяти). Так как значения десятичных цифр лежат в диапазоне ***0 … 9***, а в пределах двоичной тетрады можно сформировать число от ***0*** до ***15***, то единицы десятичного переноса автоматически не формируются. Результат может требовать коррекции

По правилам десятичной арифметики цифра суммы Z_i определяется сложением по модулю ***10*** соответствующих цифр слагаемых X_i и Y_i и переноса C_i (если $X_i + Y_i + C_i \geq 10$, то $Z_i = X_i + Y_i + C_i -$

10). Значение C_{i+1} получается делением суммы X_i, Y_i и C_i на ***10*** нацело

$$Z_i = (X_i + Y_i + C_i) \mod 10, \quad P_{i+1} = \left\lfloor \frac{X_i + Y_i + C_i}{10} \right\rfloor. \qquad (1.7)$$

Чтобы эти правила выполнялись, после поразрядного сложения модулей ***X*** и ***Y*** каждая десятичная цифра суммы Z_i представляется в коде «**с избытком шесть**» (он получается путем прибавления к каждой десятичной цифре Z_i величины $0110_2 = 6$). При этом формируются разряды $Z_{Избi}$ и переносы C_{i+1}

$$Z_{Избi} = (Z_i + 6) \mod 16, \quad C_{i+1} = \left\lfloor \frac{Z_i + 6}{16} \right\rfloor. \qquad (1.8)$$

Если при переводе в код «с избытком шесть» формируется перенос, цифра суммы представляется в естественном ***BCD***-коде. Если перенос не формируется, цифра суммы представляется в коде «с избытком шесть». Она должна быть уменьшена на $0110_2 = 6$ (это можно реализовать путем подсуммирования $1010_2 = 10 = -6_{Доп}$ с одновременным игнорированием возникающего переноса).

Пример. *Сложение двоично-кодированных чисел. Знак кодируется значением* 1100_2. *Показаны прямые коды данных, код суммы «с избытком шесть» и межтетрадные переносы (↩). Выделены коды знаков чисел и единицы переноса, которые будут проигнорированы.*

X	$= 0010\ 0011\ 0101\ \boxed{1100}_{2\text{-}10} = +235_{10}$	*Данные*
Y	$= 0001\ 1001\ 1000\ \boxed{1100}_{2\text{-}10} = +198_{10}$	
X+Y	$= 0011\ 1100\ 1101\ \boxed{1100}_{2}$	
	$0110\ 0110\ 0110\ 0000_{2\text{-}10}$	*Избыток*
$(X+Y)_{Избб}$	$= 1010\ 0011\ 0011\ 1100_{2\text{-}10Избб}$	
	↩ ↩	
	$1010\ 0000\ 0000\ 0000$	*Коррекция*
X+Y	$= 0100\ 0011\ 0011\ \boxed{1100}_{2\text{-}10} = +433_{10}$	*Результат*
	❶↩	

• • •

При вычитании отрицательное число ***Y*** представляется в дополнительном коде и складывается с ***X***. Добавление избытка в этом случае выполняется неявно при получении дополнительного кода вычитаемого.

Если из старшей двоичной тетрады разности возникает перенос, она считается положительной и исправляется по тем же правилам, что и сумма. Если из старшей тетрады переноса не возникает, разность считается отрицательной и представленной в дополнительном коде. В этом случае требуется соответствующая установка ее знака и перевод в прямой код. Коррекции подлежат двоичные тетрады, из которых при выполнении операции вычитания *возникал* перенос.

Пример. *Вычитание двоично-кодированных чисел. Знак «плюс» кодируется значением* $\mathbf{1100_2}$*, знак «минус» –* $\mathbf{1101_2}$*. Показаны прямые, дополнительные коды данных и межтетрадные переносы (↼). Выделены коды знаков чисел и единицы переноса, которые будут проигнорированы.*

X	$0010\ 0011\ 0101\ \boxed{1100}_{2\text{-}10} = +235_{10}$	*Данные*
Y	$0001\ 1001\ 1000\ \boxed{1101}_{2\text{-}10} = -198_{10}$	
$Y_{Доп}$	$1110\ 0110\ 1000\ \boxed{1101}_{2\text{-}10Доп} = -198_{10}$	
$X+Y_{Доп}$	❶ $0000\ 1001\ 1101\ \boxed{1100}_{2\text{-}10}$	$Z \geq 0$
	$0000\ 1010\ 1010\ 0000$	*Коррекция*
$X-Y$	$0000\ 0011\ 0111\ \boxed{1100}_{2\text{-}10} = +37_{10}$	*Результат*
	❶↼ ❶↼	

X	$0001\ 1001\ 1000\ \boxed{1100}_{2\text{-}10} = +198_{10}$	*Данные*
Y	$0010\ 0011\ 0101\ \boxed{1101}_{2\text{-}10} = -235_{10}$	
$Y_{Доп}$	$1101\ 1100\ 1011\ \boxed{1101}_{2\text{-}10Доп} = -235_{10}$	
$X+Y_{Доп}$	$1111\ 0110\ 0011\ \boxed{1100}_{2\text{-}10Доп}$	$Z<0$
	↼ ↼	
	$0000\ 1001\ 1101\ \boxed{1101}_{2\text{-}10}$	*Прямой код* Z
	$0000\ 1010\ 1010\ 0000$	*Коррекция*

$X-Y$ = 0000 0011 0111 **1101**$_{2-10}$ = -37_{10}	*Результат*
❶↵ ❶↵	

• • •

УМНОЖЕНИЕ $Z = X \times Y$. Это действие выполняется над модулями ***BCD***-чисел. Знак произведения определяется так же, как и при умножении двоичных чисел.

Умножение сводится к многократному сложению частичных произведений. При этом последовательно анализируются цифры множителя Y, начиная со старшей. На каждом шаге операции множимое X прибавляется к сумме частичных произведений Y_i раз, после чего сумма сдвигается влево на одну двоичную тетраду. Эти действия повторяются, пока все цифры Y не будут обработаны.

ДЕЛЕНИЕ $Z = X \div Y$. В нем также участвуют модули ***BCD***-чисел. Операция производится путем многократного вычитания десятичных цифр подобно тому, как это делается при обычном делении. Знак частного определяется, как и знак произведения.

1.4 Действия над числами в формате с плавающей точкой

Арифметические операции над числами с плавающей точкой выполняются по отдельности над разрядами мантиссы и порядка по правилам обработки данных в фиксированном формате. Считается, что точка в мантиссе находится перед старшим разрядом, а в порядке – после младшего. Основные операции следующие.

НОРМАЛИЗАЦИЯ. Она используется для приведения мантиссы числа к стандартному диапазону $2^{-1} \le |M| < 1$. Нормализация выполняется при определенных сочетаниях знакового и старших разрядов дробной части (для положительных чисел – **00…**, для отрицательных – **11…**). Если $M = 0$, операция не выполняется.

Нормализация производится автоматически путем арифметического сдвига мантиссы влево до тех пор, пока в старшем разряде ее

модуля не окажется единица. Чтобы сохранить величину числа, при каждом сдвиге порядок уменьшается на единицу.

Пример. *Нормализация числа с плавающей точкой (**K** = **4**, **L** = **7**, порядок показан без избытка). Выделены разряды знака и мантиссы (условно размещены до разрядов порядка). Цветом выделены скрытые разряды нормализации.*

До операции	$X = \boxed{0}0001000\ 0111_2 = +2^{-4}\cdot 2^{7} = +8.0_{10}$
После операции	$X = \boxed{0}1000000\ 0100_2 = +2^{-1}\cdot 2^{4} = +8.0$

До операции	$X = \boxed{1}1111000\ 0111_2 = -2^{-4}\cdot 2^{7} = -8.0_{10}$
После операции	$X = \boxed{1}1000000\ 0100_2 = -2^{-1}\cdot 2^{4} = -8.0_{10}$

• • •

Обратной операцией является **денормализация**, выполняемая путем арифметического сдвига мантиссы вправо с одновременным увеличением порядка на единицу при каждом сдвиге.

СЛОЖЕНИЕ И ВЫЧИТАНИЕ $Z = X \pm Y$. Они выполняются по одинаковым правилам в процессе последовательной обработки дополнительных кодов мантисс после выравнивания порядков. Если $X = M_X \cdot 2^{P_X}$, $Y = M_Y \cdot 2^{P_Y}$ и $P_X \geq P_Y$, то используется следующая схема операции

$$Z = M_X \cdot 2^{P_X} \pm M_Y \cdot 2^{P_Y} = \left(M_X \pm M_Y \cdot 2^{P_Y - P_X}\right) \cdot 2^{P_X} = \left(M_X \pm \frac{M_Y}{2^{P_X - P_Y}}\right) \cdot 2^{P_X} = \left(M_X \pm \frac{M_Y}{2^{\Delta P}}\right) \cdot 2^{P_X}. \quad (1.9)$$

Мантисса меньшего числа арифметически сдвигается вправо на ΔP разрядов, что эквивалентно делению на $2^{\Delta P}$.

Сложение (вычитание) начинается с определения разности $P_X - P_Y$. Если она отрицательна, то порядком результата считается P_X, а денормализации подлежит число Y. Если $P_X - P_Y < 0$, то порядком результата считается P_Y, а денормализуется X. Если $P_X - P_Y = 0$, то

операция выполняется непосредственно. Мантисса результата определяется в ходе поразрядной обработки дополнительных кодов мантисс M_X и M_Y как чисел с фиксированной точкой. При необходимости результат нормализуется. Для хранения его дополнительный код преобразуется в прямой.

Пример. *Сложение и вычитание чисел с плавающей точкой (**K** = **4**, **L** = **7**, порядок показан без избытка). Выделены разряды знака, мантиссы (условно размещены до разрядов порядка). Цветом выделены скрытые разряды нормализации. Показаны единицы переноса, которые будут проигнорированы.*

Сложение двух чисел:

X	=	$\boxed{0}1000000\ 0110_2 = 2^{-1}\cdot 2^6 = 32_{10}$	*Исходные данные*
Y	=	$\boxed{0}1000000\ 0010_2 = 2^{-1}\cdot 2^2 = 2_{10}$	
X	=	$\boxed{0}1000000\ 0110_2 = 2^{-1}\cdot 2^6 = 32_{10}$	*Выравнивание порядков*
Y	=	$\boxed{0}0000100\ 0010_2 = 2^{-5}\cdot 2^6 = 2_{10}$	
$X+Y$	=	$\boxed{0}1000100\ 0010_2 =$	*Нормализованный результат*
	=	$(2^{-1}+2^{-5})\cdot 2^6 = 34$	

Вычитание двух чисел:

X	=	$\boxed{0}1000000\ 0110_2 = 2^{-1}\cdot 2^6 = 32_{10}$	*Исходные данные*
Y	=	$\boxed{1}1000000\ 0010_2 = -2^{-1}\cdot 2^2 = -2_{10}$	
$Y_{доп}$	=	$\boxed{1}1000000\ 0010_2 = -2^{-1}\cdot 2^2 = -2_{10}$	
X	=	$\boxed{0}1000000\ 0110_2 = 2^{-1}\cdot 2^6 = 32_{10}$	*Выравнивание порядков*
$Y_{доп}$	=	$\boxed{1}1111100\ 0110_2 = -2^{-5}\cdot 2^6 = -2_{10}$	
$X+Y_{доп}$	=	❶ $\boxed{0}0111100\ 0110_2 =$	*Ненормализованный результат*
	=	$(2^{-2}+2^{-3}+2^{-4}+2^{-5})\cdot 2^6 = 30_{10}$	
$X-Y$	=	$\boxed{0}1111000\ 0101_2 =$	*Нормализованный результат*
	=	$(2^{-1}+2^{-2}+2^{-3}+2^{-4})\cdot 2^5 = 30_{10}$	

• • •

УМНОЖЕНИЕ $Z = X \times Y$ чисел $X = M_X \cdot 2^{P_X}$, $Y = M_Y \cdot 2^{P_Y}$ обычно выполняется в прямом коде с отделением знака по схеме

$$Z = \left(S_X \left|M_X\right| \cdot 2^{P_X}\right) \times \left(S_Y \left|M_Y\right| \cdot 2^{P_Y}\right) = \\ = \left(S_X \oplus S_Y\right)\left(\left|M_X\right| \times \left|M_Y\right|\right) \cdot 2^{P_X + P_Y}. \qquad (1.10)$$

Если множимое или множитель равны нулю, результату присваивается нулевое значение без выполнения операции. Порядок произведения находится суммированием порядков P_X и P_Y. Мантисса произведения определяется перемножением M_X и M_Y со сдвигом суммы частичных произведений вправо. Так как $0{,}1_2 \le |M_X| < 1$ и $0{,}1_2 \le |M_Y| < 1$, то $0{,}01_2 \le |M_Z| < 1$, т. е. произведение должно быть нормализовано. Произведению присваивается знак «плюс», если знаки X и Y одинаковы, или «минус», если эти знаки разные.

Возникновение при умножении отрицательного переполнения порядка означает, что результат по модулю стал меньше минимально допустимого значения и выродился в ноль. Возникающее положительное переполнение порядка может быть ликвидировано в процессе нормализации произведения. Если этого сделать не удается, фиксируется ошибочное состояние.

ДЕЛЕНИЕ $Z = X \div Y$ чисел $X = M_X \cdot 2^{P_X}$, $Y = M_Y \cdot 2^{P_Y}$ выполняется в прямом коде с отделением знака по схеме

$$Z = \frac{S_X \left|M_X\right| \cdot 2^{P_X}}{S_Y \left|M_Y\right| \cdot 2^{P_Y}} = \left(S_X \oplus S_Y\right) \frac{\left|M_X\right|}{\left|M_Y\right|} \cdot 2^{P_X - P_Y}. \qquad (1.11)$$

Если делимое равно нулю, частному присваивается нулевое значение без выполнения операции. Если делитель равен нулю, фиксируется ошибочное состояние. Порядок частного равен разности порядков делимого и делителя. Мантисса частного равна частному от деления мантисс делимого и делителя. Частное нормализуется, и ему присваивается знак «плюс», если знаки сомножителей одинаковы, или «минус», если эти знаки разные.

Деление мантисс может выполняться с восстановлением остатка. Если $M_X > M_Y$, то мантисса частного образуется с переполнением. Во избежание этого перед операцией делимое денормализуют

путем арифметического сдвига M_X на один разряд вправо.

Возникновение при делении отрицательного переполнения порядка означает, что частное выродилось в ноль. При положительном переполнении порядка фиксируется ошибочное состояние.

1.5 Распространение погрешности при вычислениях

Числа с плавающей точкой представляются приближенно. При выполнении операций над ними в сформированный результат вносится погрешность, которая сама будет участвовать в вычислительном процессе и влиять на его протекание. Такой эффект называется **распространением** или **накоплением погрешности.**

Математическую операцию, в которой участвуют приближенные числа X и Y, можно обозначить, как $Z = f(X,Y)$. Если $\overline{X}$ и $\overline{Y}$ – точные значения чисел, $\overline{\Delta}_X$ и $\overline{\Delta}_Y$ – средние погрешности их представления, то приближенные значения чисел определятся, как $X = \overline{X} + \overline{\Delta}_X$ и $Y = \overline{Y} + \overline{\Delta}_Y$. Погрешности $\overline{\Delta}_X$ и $\overline{\Delta}_Y$ имеют случайный характер, и знак ошибки определения результата Z известен очень редко. Поэтому для оценки точности математической операции определяется вероятностная характеристика – средний квадрат отклонения точного значения $\overline{Z}$ от расчетного Z

$$\Delta_Z^2 = \{Z - \overline{Z}\}^2 = \{f(X,Y) - f(\overline{X},\overline{Y})\}^2. \qquad (1.12)$$

После разложения функции $f(X,Y)$ в ряд Тейлора в окрестности точки $(\overline{X},\overline{Y})$ по параметрам $\overline{\Delta}_X$ и $\overline{\Delta}_Y$ и исключения из этого разложения составляющих со степенями частных производных выше первой получается приближенное выражение

$$f(X,Y) \approx f(\overline{X},\overline{Y}) + \overline{\left[\frac{\partial f}{\partial X}\right]} \cdot \overline{\Delta}_X + \overline{\left[\frac{\partial f}{\partial Y}\right]} \cdot \overline{\Delta}_Y. \qquad (1.13)$$

Средние значения частных производных определяют *чувствительность результата операции к изменениям величин* X *и* Y. С

учетом (1.13) выражение (1.12) примет вид

$$\Delta_Z^2 = \left\{ \overline{\left[\frac{\partial f}{\partial X}\right]} \cdot \overline{\Delta}_X + \overline{\left[\frac{\partial f}{\partial Y}\right]} \cdot \overline{\Delta}_Y \right\}^2, \qquad (1.14)$$

$$\Delta_Z^2 = \overline{\left[\frac{\partial f}{\partial X}\right]}^2 \cdot \overline{\Delta}_X^2 + \overline{\left[\frac{\partial f}{\partial Y}\right]}^2 \cdot \overline{\Delta}_Y^2 + 2 \cdot \overline{\left[\frac{\partial f}{\partial X}\right]} \cdot \overline{\left[\frac{\partial f}{\partial Y}\right]} \cdot \overline{\Delta_X \cdot \Delta_Y}. \qquad (1.15)$$

Так как независимые величины $\overline{\Delta}_X$ и $\overline{\Delta}_Y$ распределены случайным образом симметрично относительно нулевого значения, то среднее значение их произведения $\overline{\Delta_X \cdot \Delta_Y}$ равно нулю. Тогда абсолютная погрешность математической операции равна

$$\Delta_Z = \sqrt{\overline{\left[\frac{\partial f}{\partial X}\right]}^2 \cdot \overline{\Delta}_X^2 + \overline{\left[\frac{\partial f}{\partial Y}\right]}^2 \cdot \overline{\Delta}_Y^2}. \qquad (1.16)$$

Относительная погрешность математической операции равна

$$\varepsilon_Z = \frac{\Delta_Z}{\overline{Z}} = \frac{1}{f(\overline{X}, \overline{Y})} \cdot \sqrt{\overline{\left[\frac{\partial f}{\partial X}\right]}^2 \cdot \overline{\Delta}_X^2 + \overline{\left[\frac{\partial f}{\partial Y}\right]}^2 \cdot \overline{\Delta}_Y^2}. \qquad (1.17)$$

Полная погрешность результата (с учетом погрешности его представления в разрядной сетке $\overline{\Delta}_Z$) равна

$$\tilde{\Delta}_Z = \sqrt{\Delta_Z^2 + \overline{\Delta}_Z^2}. \qquad (1.18)$$

Формулы для расчета абсолютных и относительных погрешностей основных операций приведены в таблице 1.1.

Таблица 1.1 – Погрешности математических операций

Операция $f(X,Y)$	$\frac{\partial f}{\partial X}$	$\frac{\partial f}{\partial Y}$	***Абсолютная погрешность***	***Относительная погрешность***
$X \pm Y$	1	1	$\sqrt{\overline{\Delta}_X^2 + \overline{\Delta}_Y^2}$	$\frac{\sqrt{\overline{\Delta}_X^2 + \overline{\Delta}_Y^2}}{\lvert X \pm Y \rvert}$

$X \times Y$	$\overline{Y}$	$\overline{X}$	$\sqrt{\overline{Y}^2 \cdot \overline{\Delta}_X^2 + \overline{X}^2 \cdot \overline{\Delta}_Y^2}$	$\sqrt{\frac{\overline{\Delta}_X^2}{\overline{X}^2} + \frac{\overline{\Delta}_Y^2}{\overline{Y}^2}}$
$\frac{X}{Y}$	$\frac{1}{\overline{Y}}$	$-\frac{\overline{X}}{\overline{Y}^2}$	$\sqrt{\frac{\overline{\Delta}_X^2}{\overline{Y}^2} + \frac{\overline{X}^2 \cdot \overline{\Delta}_Y^2}{\overline{Y}^4}}$	$\sqrt{\frac{\overline{\Delta}_X^2}{\overline{X}^2} + \frac{\overline{\Delta}_Y^2}{\overline{Y}^2}}$

Пример. *Определение погрешности математических операций над числами* $X = 2{,}3$ *и* $Y = 1{,}7$*, представленных приближенно с погрешностями* $\overline{\Delta}_X = \overline{\Delta}_Y = 10^{-7}$.

Операция	***Абсолютная погрешность***	***Относительная погрешность***
$X + Y$	$1.41 \cdot 10^{-7}$	$3.54 \cdot 10^{-8}$
$X - Y$	$1.41 \cdot 10^{-7}$	$2.36 \cdot 10^{-7}$
$X \times Y$	$2.86 \cdot 10^{-7}$	$7.32 \cdot 10^{-8}$
X / Y	$8.64 \cdot 10^{-8}$	$7.32 \cdot 10^{-8}$

• • •

Погрешность конечного результата вычислений можно определить последовательно, учитывая погрешности каждой выполненной операции. Если эта величина известна заранее, то можно решить обратную задачу – выяснить, насколько точными должны быть исходные данные.

2 СОДЕРЖАНИЕ ПРАКТИЧЕСКИХ ЗАНЯТИЙ

На практических занятиях студент должен решить следующие задачи.

1. Представить два заданных целых числа ***x*** и ***y*** в двоичном дополнительном коде, используя разрядную сетку длиной ***8*** разрядов.

Выполнить над дополнительными кодами чисел операции:

- логического, арифметического и циклического сдвигов;
- инверсии, конъюнкции, дизъюнкции и сложения по модулю ***2***;
- сложения и вычитания;
- умножения с отделением знака (сформировать ***16*** разрядов произведения);
- деления с отделением знака с восстановлением остатка (сформировать ***8*** разрядов частного).

Оценить достоверность полученных результатов и проанализировать возможное наличие переполнения разрядной сетки.

2. Представить эти же целые числа в ***BCD***-коде, используя зонный и упакованный формат. Значения кодов знака и зоны выбрать самостоятельно. Длину кода выбрать из условия представления всех десятичных цифр.

Выполнить над упакованными ***BCD***-кодами чисел операции сложения и вычитания. Оценить достоверность полученных результатов.

3. Представить два заданных числа с дробной частью ***a*** и ***b*** в двоичном дополнительном коде, используя мантиссу длиной ***8*** разрядов и порядок длиной ***4*** разряда. Числа нормализовать и определить погрешность их представления в разрядной сетке (по согласованию с преподавателем допускается не учитывать смещение порядка и отображать скрытый разряд нормализации).

Выполнить над дополнительными кодами чисел операции:

- сложения и вычитания;

- умножения с отделением знака;
- деления с отделением знака с восстановлением остатка.

Обработку мантисс и порядков чисел выполнить раздельно по правилам, принятым для чисел в формате с фиксированной точкой.

Полученные результаты нормализовать и определить погрешность их вычисления и представления в разрядной сетке. Оценить достоверность полученных результатов и проанализировать возможное наличие переполнения разрядной сетки.

Исходные данные для самостоятельного решения задач приведены в приложении В.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Как кодируются числа в формате с фиксированной точкой? Что определяет диапазон чисел в фиксированном формате?
2. Как хранятся числа со знаком в формате с фиксированной точкой в прямом, обратном и дополнительном коде?
3. Какие особенности имеет двоично–десятичное кодирование? Как кодируются двоично–десятичные числа в зонном и упакованном формате?
4. Как кодируются логические, символьные и строковые данные, графическая и звуковая информация?
5. Как кодируются числа в формате с плавающей точкой? Что определяет диапазон значений чисел в плавающем формате?
6. Как хранятся порядок и мантисса числа в формате с плавающей точкой? Что понимается под смещением порядка и скрытым разрядом мантиссы?
8. Как хранится мантисса числа со знаком в формате с плавающей точкой в прямом, обратном и дополнительном коде?
1. Как выполняются логические, арифметические и циклические сдвиги над числами в фиксированном формате? Каким арифметическим операциям эквивалентны арифметические сдвиги?
2. Как выполняются логические операции над числами в фик-

сированном формате? Для чего применяются логические операции?

3. Как выполняются операции сложения и вычитания над числами в фиксированном формате со знаком? Что такое «положительное» и «отрицательное» переполнение?

4. Как выполняется умножение чисел в фиксированном формате с отделением знака?

5. Как выполняется умножение чисел в фиксированном формате без отделения знака? Что такое псевдопроизведение? По каким правилам оно корректируется?

6. Как выполняется деление чисел в фиксированном формате в восстановлением и без восстановления остатка?

7. Как выполняются операции сложения и вычитания над двоично-кодированными числами? По каким правилам производится коррекция результата?

8. Как выполняются операции умножения и деления над двоично-кодированными числами?

9. Как получается двоично-десятичное изображение двоичного числа? Как производится преобразование двоично-десятичного числа в двоичную систему?

10. Как выполняются операции нормализации и денормализации над числами в плавающем формате?

11. Как выполняются операции сложения и вычитания над числами в плавающем формате?

12. Как выполняются операции умножения и деления над числами в плавающем формате?

13. В каких случаях при обработке чисел в плавающем формате нарушаются правила классической арифметики? Как этого избежать?

14. Что понимается под эффектом распространения погрешности при вычислениях?

15. Как определяется погрешность результата операции, в которой участвуют приближённо представленные числа?

СПИСОК РЕКОМЕНДУЕМЫХ ИСТОЧНИКОВ

1. Информатика : Учебник [Текст] / Под ред. проф. Н.В. Макаровой. – М. : Финансы и статистика, 2000. – 768 с. : ил. – ISBN 5-279-02202-0.

2. Острейковский, В.А. Информатика [Текст] : Учеб. для вузов / Острейковский В.А. – М. : Высш. шк., 2007. – 511 с. : ил. – ISBN 978-5-06-003533-9 ; ISBN 5-06-003533-06.

3. Савельев, А.Я. Основы информатики [Текст] : Учебник для вузов / Савельев А.Я. – М. : Изд-во МГТУ им. Н.Э. Баумана, 2001. – 328 с. : ил. – (Информатика в техническом университете). – ISBN 5-7038-1515-0.

4. Брой, Манфред. Информатика. Основополагающее введение [Текст]. В 4 ч. Ч. 2. Вычислительные структуры и машинно-ориентированное программирование / Брой М. ; Пер. с нем. – М. : Диалог-МИФИ, 1996. – 224 с. – ISBN 3-540-56969-3 (нем.) ; ISBN 5-86404-074-6 (русск.).

5. Введение в информатику [Электронный ресурс] / В.М. Казиев. – Интернет-Университет Информационных технологий. – Электрон. текст. дан. – Режим доступа: http//www.INTUIT.ru, свободный.

6. Практическая информатика [Электронный ресурс] / Е.А. Роганов. – Интернет-Университет Информационных технологий. – Электрон. текст. дан. – Режим доступа: http//www.INTUIT.ru, свободный.

7. Основы информатики и программирования [Электронный ресурс] / Е.А. Роганов. – Интернет-Университет Информационных технологий. – Электрон. текст. дан. – Режим доступа: http//www.INTUIT.ru, свободный.

8. Логические и арифметические основы и принципы работы ЭВМ [Электронный ресурс] / В.О. Чуканов, В.В. Гуров. – Интернет-Университет Информационных технологий. – Электрон. текст. дан. – Режим доступа: http//www.INTUIT.ru, свободный.

9. Основы теории информации и криптографии [Электронный ресурс] / В.В. Лидовский. – Интернет-Университет Информационных технологий. – Электрон. текст. дан. – Режим доступа: http//www.INTUIT.ru, свободный.

10. Информатика в семи томах [Электронный ресурс] / А.А. Красилов. – «Интеллсист» Интеллектуальные системы общего назначения. – Электрон. текст. дан. – Режим доступа: http//www.intellsyst.ru/publications, свободный.

11. Основы машинной арифметики [Электронный ресурс]. – Учебные курсы. Дистанционное образование и обучение в Интернет. – Электрон. текст. дан. – Режим доступа: http//www.myLearn.ru, свободный.

12. Методические указания к оформлению научно-технической документации для специальности 230201.65 (071900) – Информационные системы и технологии [Текст] / Д.В. Болдырев, Л.Г. Гордиенко. – Невинномысск : изд-во НТИ (филиала) ГОУ ВПО «СевКавГТУ», 2008. – 45 с.

13. ГОСТ 7.1-2003. Библиографическая запись. Библиографическое описание. Общие требования и правила составления [Текст]. – Взамен ГОСТ 7.1-84, ГОСТ 7.16-79, ГОСТ 7.18-79, ГОСТ 7.34-81, ГОСТ 7.40-82 ; введ. 2004-07-01. – Минск : Межгос. совет по стандартизации, метрологии и сертификации ; М. : Изд-во стандартов, 2003. – 62 с. – (Система стандартов по информации, библиотечному и издательскому делу).

ПРИЛОЖЕНИЕ А

Основные сведения о системах счисления

Система счисления – совокупность приемов и правил изображения чисел цифровыми знаками. Системы счисления делятся на непозиционные и позиционные.

В **непозиционных системах счисления** значение цифры не зависит от ее положения в числе. Их недостаток – большое число различных знаков и сложность обработки чисел. Примером служит римская система. Цифры в ней обозначаются знаками: ***I – 1, V – 5, X – 10, L – 50, C – 100, D – 500, M – 1000***. Для записи чисел используют правило: каждая меньшая цифра, поставленная справа от большей, прибавляется к ее значению, поставленная слева – вычитается из него. Так, ***IV*** обозначает ***4, VI – 6, XL – 40, LX – 60*** и т. п.

В **позиционных системах счисления** значение цифры зависит от ее места в изображении числа (например, в десятичном числе выделяют разряды единиц, десятков, сотен и т. д.). Позиционные системы счисления более удобны для вычислений.

Основание (базис) позиционной системы счисления q – количество знаков, используемых для изображения разрядов числа. **Вес разряда** числа – это отношение вида $q^i / q^o = q^i$, где i – номер разряда при счете справа налево. Количество разрядов в записи числа определяет его **длину.**

За основание можно принять любое целое число, поэтому возможно формирование множества позиционных систем счисления. В цифровой технике наибольшее распространение получили двоичные, восьмеричные и шестнадцатеричные системы. В системе с основанием $q = 2$ используют цифры ***0*** и ***1***, с основанием $q = 8$ – ***0 … 7***, с основанием $q = 16$ – цифры ***0 … 9*** и буквы ***A … F***, которые

обозначают шестнадцатеричные цифры ***10 … 15***. Двоичная система является основной из-за простоты и высокой скорости выполнения арифметических операций над двоичными числами. Восьмеричная и шестнадцатеричная системы являются вспомогательными.

В позиционной системе счисления с основанием q любое число $X(q)$ можно представить следующим образом

$$X(q) = a_n \cdot q^n + a_{n-1} \cdot q^{n-1} + \ldots + a_0 \cdot q^0 + \\ + a_{-1} \cdot q^{-1} + \ldots + a_{-m} \cdot q^{-m}, \qquad \text{(A.1)}$$

где a_i – разрядный коэффициент (цифра); n, m – количество разрядов в целой и дробной частях числа. На практике используют сокращенную запись чисел $X(q) = a_n a_{n-1} \ldots a_1 a_0 a_{-1} \ldots a_{-m}$.

При переводе целой части десятичного числа в систему с основанием q ее делят на q, фиксируя остаток. Операцию деления продолжают, используя предыдущий результат в качестве делимого. Перевод прекращают при получении частного, равного нулю. Остатки от деления, записанные в обратном порядке, дадут изображение целой части числа в системе с основанием q.

При переводе дробной части десятичного числа в систему с основанием q ее умножают на q, отделяя от произведения целую часть. Операцию умножения с отделением целой части продолжают, используя дробную часть предыдущего результата в качестве сомножителя. Перевод прекращают при получении заданного числа разрядов дробной части. При этом исходят из условия, что точность числа в системе с основанием q должна соответствовать точности десятичного числа. Целые части произведений, записанные в прямом порядке, дадут изображение дробной части числа в системе с основанием q.

Пример. *Перевод десятичного числа* ***458,91*** *в шестнадцатеричную, восьмеричную и двоичную системы. Число цифр в дробной части*

выбрано из условия точности (**0,01**) *для шестнадцатеричной системы –* **2** (**$16^{-2} \approx 0{,}004$**), *для восьмеричной системы –* **3** (**$8^{-3} \approx 0{,}002$**), *для двоичной системы –* **7** (**$2^{-7} \approx 0{,}008$**). *Первые два столбца таблиц содержат промежуточные частные и остатки от деления при переводе целой части числа, вторые два столбца – целые и дробные части промежуточных произведений при переводе дробной части числа в систему счисления с новым основанием.*

Целая часть исходного числа 458						***Дробная часть исходного числа 0,91***					
q = 2		***q = 8***		***q = 16***		***q = 2***		***q = 8***		***q = 16***	
Целая часть частного	*Остаток от деления на* **2**	*Целая часть частного*	*Остаток от деления на* **8**	*Целая часть частного*	*Остаток от деления на* **16**	*Целая часть произведения*	*Дробная часть произведения*	*Целая часть произведения*	*Дробная часть произведения*	*Целая часть произведения*	*Дробная часть произведения*
458	**–**	***458***	**–**	***458***	**–**	**–**	***0,91***	**–**	***0,91***	**–**	***0,91***
229	*0*	*57*	*2*	*28*	*10*	*1*	*0,82*	*7*	*0,28*	*14*	*0,56*
114	*1*	*7*	*1*	*1*	*12*	*1*	*0,64*	*2*	*0,24*	*8*	*0,96*
57	*0*	*0*	*7*	*0*	*1*	*1*	*0,28*	*1*	*0,92*	*15*	*0,36*
28	*1*					*0*	*0,56*	*7*	*0,36*		
14	*0*					*1*	*0,12*				
7	*0*					*0*	*0,24*				
3	*1*					*0*	*0,48*				
1	*1*					*0*	*0,96*				
0	*1*					*1*	*0,92*				

Результат: $\mathbf{458{,}91_{10} = 111001010{,}1110100_2 = 712{,}766_8 = 1CA{,}ED_{16}}$.

• • •

Для получения десятичного эквивалента двоичного, восьмеричного или шестнадцатеричного изображений применяют формулу (А.1). В качестве $\boldsymbol{q}$ используют соответственно **2**, ***8*** или ***16***, в качестве $\boldsymbol{a_i}$ – значения соответствующих цифр.

Взаимный перевод двоичных, восьмеричных и шестнадцатеричных чисел упрощен, так как все эти системы имеют основание,

кратное степени двойки. Для перевода двоичного числа в шестнадцатеричную (восьмеричную) системы его целую и дробную части, начиная от точки, делят на тетрады (триады), заполнив недостающие разряды слева и справа от числа незначащими нулями. Каждая из тетрад (триад) представляется соответствующей шестнадцатеричной (восьмеричной) цифрой. Результат перевода формируется из полученных цифр с учетом их положения в числе. Для обратного перевода шестнадцатеричного (восьмеричного) числа в двоичную систему каждую его цифру в целой и дробной части изображают четырехзначным (трехзначным) двоичным числом. Результат перевода получается объединением тетрад (триад) целой и дробной части. Незначащие нули в начале и конце числа могут исключаться. Перевод из шестнадцатеричной системы в восьмеричную и обратно производится, как правило, через двоичную систему.

Пример. *Перевод двоичных чисел в восьмеричную и шестнадцатеричную системы и обратно.*

$101\ 010\ 001\ 000,000\ 100\ 001\ 111_2 = \mathbf{5210,0417_8}$

$1010\ 1000\ 1000,0001\ 0000\ 1111_2 = \mathbf{A88,10F_{16}}$

$C3,93_{16} = \mathbf{1100\ 0011,1001\ 0011_2}$

$20,24_8 = \mathbf{010\ 000,010\ 100_2}$

$F2,3A_{16} = 011110010,001110100_2 = \mathbf{362,164_8}$

• • •

ПРИЛОЖЕНИЕ Б

Основные положения алгебры логики

Алгебра логики – раздел математической логики, называемая исчислением высказываний.

Логическое высказывание – всякое предложение, содержащее смысл утверждения (истинности) или отрицания (ложности). Если соответствующая высказыванию логическая величина принимает значение «истина» при любых условиях, высказывание считается *абсолютно истинным*. Если такая величина при любых условиях принимает значение «ложь», высказывание считается *абсолютно ложным*. Если истинность высказывания не зависит от истинности других высказываний, оно считается *простым*. Если истинность высказывания определяется истинностью других высказываний, оно считается *сложным*.

Логическая переменная – величина, определяющая состояние некоторой системы или определяющая ее принадлежность к некоторому классу состояний. Она принимает значение «ложь» или «истина», которым ставятся в соответствие ***0*** или ***1***.

Логическая (переключающая) функция – это зависимость вида $y = f(x_1, x_2, ..., x_n)$ (где y, x_i, $i = 1, \ldots, n$ – логические переменные). Она задается в виде соответствия значений функции некоторому набору (комбинации) значений аргументов. Аргумент x_i считается *действительным*, если функция изменяется при его изменении. Если такого изменения не происходит, аргумент считается *фиктивным*. Использование фиктивных параметров позволяет сокращать или расширять набор аргументов логической функции.

Логические функции считаются *различными*, если их значения различаются хотя бы для одного такого набора. При числе аргумен-

тов функции n число комбинации значений аргументов $N = 2^n$, а общее число логических функций $M = 2^N$.

Логические функции являются сложными высказываниями, а их аргументы – простыми или сложными высказываниями. Элементарными считаются функции двух переменных. Функции большего числа переменных строятся *методом суперпозиции*. Их аргументы сами могут быть функциями, т. е. если $y = f(z_1, z_2)$, $z_1 = f(x_1, x_2)$, $z_2 = f(x_3, x_4)$, то $y = f(x_1, x_2, x_3, x_4)$.

Функции двух переменных приведены в таблице Б.1 (считается, что операции выполняются над x_1 и x_2 поразрядно).

Таблица Б.1 – Логические функции двух переменных

№	x_1:0011 x_2:0101	$f(x_1,x_2)$	**Наименование**
0	0000	0	*Константа* **0** *(«всегда ложно»)*
1	0001	$x_1 \cdot \wedge \cdot x_2$	*Конъюнкция* x_1 *и* x_2 *(«И»)*
2	0010	$\overline{x_1 \to x_2}$	*Запрет по* x_2 *(«неверно, что, если* x_1*, то* x_2*»)*
3	0011	x_1	*Повторение* x_1
4	0100	$\overline{x_2 \to x_1}$	*Запрет по* x_1 *(«неверно, что, если* x_2*, то* x_1*»)*
5	0101	x_2	*Повторение* x_2
6	0110	$x_1 \oplus x_2$	*Неравнозначность* x_1 *и* x_2
7	0111	$x_1 \cdot \vee \cdot x_2$	*Дизъюнкция* x_1 *и* x_2 *(«ИЛИ»)*
8	1000	$x_1 \downarrow x_2$	*Стрелка Пирса («НЕ–ИЛИ», «ни* x_1*, ни* x_2*»)*
9	1001	$x_1 \equiv x_2$	*Равнозначность* x_1 *и* x_2
10	1010	$\bar{x}_2$	*Отрицание* x_2
11	1011	$x_2 \to x_1$	*Импликация от* x_2 *к* x_1 *(«если* x_2*, то* x_1*»)*
12	1100	$\bar{x}_1$	*Отрицание* x_1
13	1101	$x_1 \to x_2$	*Импликация от* x_1 *к* x_2 *(«если* x_1*, то* x_2*»)*
14	1110	x_1 / x_2	*Штрих Шеффера («НЕ–И», «не* x_1*, или не* x_2*»)*
15	1111	1	*Константа* **1** *(«всегда истинно»)*

ПРИЛОЖЕНИЕ В

Варианты заданий для самостоятельного решения

№	x	y	a	b	№	x	y	a	b
1	29	19	0,9	5,0	26	66	12	4,9	3,7
2	38	28	1,8	4,0	27	57	26	3,8	4,6
3	47	37	2,7	3,1	28	48	33	2,7	4,6
4	56	46	3,6	2,1	29	39	45	1,6	5,5
5	65	55	4,5	1,2	30	20	54	0,5	0,5
6	64	14	5,4	9,2	31	21	15	0,1	0,4
7	53	23	6,3	8,3	32	32	24	1,2	1,4
8	42	32	7,2	7,3	33	43	33	2,3	1,3
9	31	41	8,1	6,4	34	54	42	3,4	2,3
10	20	50	9,0	9,4	35	65	51	4,5	2,2
11	20	10	9,9	8,5	36	66	10	5,9	3,2
12	31	22	8,8	7,5	37	57	29	6,8	3,1
13	42	34	7,7	6,6	38	48	38	7,7	4,1
14	53	46	6,6	5,6	39	39	47	8,6	4,0
15	64	58	5,5	4,7	40	20	56	9,5	5,0
16	65	11	4,4	3,7	41	22	18	9,4	5,9
17	56	23	3,3	2,8	42	34	26	8,3	6,8
18	47	35	2,2	1,8	43	46	34	7,2	6,7
19	38	47	1,1	0,9	44	58	42	6,1	7,6
20	29	59	0,1	0,2	45	60	50	5,0	7,5
21	21	19	9,5	1,9	46	61	19	4,0	8,4
22	32	20	8,4	1,9	47	53	27	3,1	8,3
23	43	38	7,3	2,8	48	45	35	2,2	9,2
24	54	41	6,2	2,8	49	37	43	1,3	9,1
25	65	57	5,1	3,7	50	29	51	0,4	1,1

ОБРАБОТКА ИНФОРМАЦИИ В СИСТЕМАХ УПРАВЛЕНИЯ

к практическим занятиям для магистров направления подготовки 15.04.04. — Автоматизация технологических процессов и производств

Составитель *Д.В. Болдырев*

Редактор

Подписано в печать Формат 60×84 1/16

Уч.-изд. л. Усл. печ. л. Тираж 50 экз. Заказ №

Невинномысский технологический институт (филиал)

ФГАОУ ВО «Северо-Кавказский федеральный университет»

Типография

Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Невинномысский технологический институт

Кафедра информационных систем, электропривода и автоматики

СИСТЕМЫ ОБРАБОТКИ И ПЕРЕДАЧ И ИНФОРМАЦИИ

Методические указания к выполнению контрольной работы
Направление подготовки 15.04.04
«Автоматизация технологических процессов и производств»
Направленность (профиль) «Информационно-управляющие системы»

Невинномысск 2024

Настоящие методические указания предназначены для магистров направления подготовки 15.04.04 — Автоматизация технологических процессов и производств. Они разработаны в соответствии федеральным государственным образовательным стандартов и образовательной программой направления.

В методических указаниях приведены основные сведения об организации самостоятельной работы студентов, предложена примерная структура контрольной работы, кратко изложено содержание ее разделов, даны необходимые теоретические сведения и приведен список рекомендуемых источников.

Составитель *канд. техн. наук, доцент Д.В. Болдырев*

Отв. редактор *канд. техн. наук, доцент А.А. Евдокимов*

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Выполнение контрольной работы по дисциплине «Обработка информации в системах управления» имеет целью систематизацию, закрепление и расширение теоретических и практических знаний об обработке информации в системах управления с использованием современных информационных технологий, развитие навыков самостоятельного принятия проектных решений.

Выпускник по направлению 15.04.04 — Автоматизация технологических процессов и производств должен знать правила создания, внедрения, функционирования, применения информационных технологий и информационных систем, обеспечивающих поддержку работы специалиста в области управления; принципы информатизации в сфере управления предприятием и организацией.

Выпускник должен уметь строить и анализировать информационные модели объектов управления; использовать современные программные и технические средства для решения задач управления и принятия решения.

Выпускник должен владеть умением самостоятельно приобретать с помощью информационных технологий и использовать в практической деятельности новые знания и умения; навыками выбора оптимальных решений при создании продукции, разработке автоматизированных технологий и производств, средств технического и аппаратно-программного обеспечения.

1 ОРГАНИЗАЦИЯ РАБОТЫ НАД КОНТРОЛЬНОЙ РАБОТОЙ

Выполнение контрольной работы является одной из форм самостоятельной работы студентов, предусмотренной рабочей программой дисциплины.

Требования к содержанию, объему и структуре контрольной работы определяются кафедрой и регламентируются соответствующими методическими указаниями. Уровень работы должен соот-

ветствовать современному уровню развития систем управления.

На весь период работы назначается руководитель, контролирующий соответствие выполненной работы установленным требованиям и оказывающий студенту помощь в решении специальных вопросов. Руководство работой над контрольной работой осуществляется путем индивидуальных или групповых консультаций, расписание которых доводится до студентов.

Не позднее двух недель со дня начала занятий в семестре студенту выдается письменное задание утвержденной формы, в котором указываются даты выдачи и представления выполненной работы к защите. Ответственность за своевременность выдачи задания несет руководитель. В течение указанного срока студент обязан явиться на кафедру или непосредственно к преподавателю для получения задания. В случае неявки студента в установленные сроки руководитель в трехдневный срок извещает об этом декана факультета для принятия соответствующих мер. Срок защиты контрольной работы при этом не изменяется.

Руководитель может при необходимости изменить задание или исходные данные. Эти изменения не должны приводить к значительному увеличению объема самостоятельной работы студента.

Руководитель обязан проводить систематические беседы со студентом и давать ему по мере надобности консультации в соответствии с утвержденным графиком, рекомендовать студенту необходимую литературу, справочные и архивные материалы, проверять выполнение задания (по частям и в целом).

Студент обязан в установленные сроки являться на консультацию к руководителю, по требованию руководителя представлять ему выполненные разделы контрольной работы.

За принятые решения отвечает автор — студент. Если эти решения принципиально неверные, руководитель вправе не допускать контрольную работу к защите.

При защите контрольной работы студент делает краткое сообщение, в котором освещает основные вопросы, рассмотренные в

задании. При защите допускается демонстрация действующих образцов, макетов и разработанных программных модулей. После сообщения студент должен ответить на вопросы руководителя.

Контрольная работа оцениваются отметками «зачтено» или «не зачтено». Оценка определяется качеством выполнения и оформления пояснительной записки, уровнем доклада и ответов на поставленные вопросы. Задание, получившее оценку «не зачтено», может быть допущено к повторной защите после доработки.

Невыполнение контрольной работы может служить основанием для выставления студенту оценки «не зачтено» (при аттестации в форме зачета) или его отстранению от сдачи экзамена по соответствующей дисциплине.

2 СОСТАВ И СТРУКТУРА КОНТРОЛЬНОЙ РАБОТЫ

2.1 Содержание контрольной работы

Контрольная работа выполняется студентом по одному из индивидуальных вариантов, приведенных в приложении А.

В контрольной работе необходимо разработать программное приложение, выполняющее функции обхода системы односторонних или двусторонних дорог с соблюдением поставленных условий.

2.2 Структура контрольной работы

Контрольная работа состоит из пояснительной записки и демонстрационного программного обеспечения.

Пояснительная записка должна в краткой и четкой форме раскрывать творческие замыслы автора. Она должна включать анализ проблемной ситуации, сделанные выводы, обоснование принятых решений. Изложение материала должно сопровождаться необходимыми рисунками, графиками, диаграммами, схемами, таблицами и

программами вычислений.

Пояснительная записка выполняется компьютерным способом в соответствии с правилами оформления научно-технической документации [1]. Она должна содержать:

- титульный лист;
- задание;
- содержание;
- основную часть;
- список использованных источников;
- приложения.

Содержание основной части записки приведено ниже.

Введение

1 Разработка технического задания
  1.1 Общая постановка задачи
  1.2 Определение требований к программе
  1.3 Предварительный выбор метода решения задачи
  1.4 Определение требований к системе
    1.4.1 Требования к системе в целом
    1.4.2 Требования к техническому обеспечению
    1.4.3 Требования к программному обеспечению
2 Проектирование программного приложения
  2.1 Разработка технологии обработки информации
  2.2 Разработка структуры и формы представления данных
  2.3 Разработка алгоритма решения задачи
  2.4 Разработка программы решения задачи
    2.4.1 Выбор инструментальных средств
    2.4.2 Разработка структуры программы
    2.4.3 Проектирование программных модулей
    2.4.4 Проектирование программного интерфейса
    2.4.5 Обеспечение надежности программы
  2.5 Определение конфигурации технических средств

2.6 Тестирование программы

2.6.1 Общие сведения

2.6.2 Процесс тестирования программы

2.6.3 Оценка надежности программы

Заключение

Список использованных источников оформляется по правилам, приведенным в [2].

В приложения выносятся схемы алгоритмов, листинги программных модулей, распечатки результатов работы программ, основные экранные формы и другие материалы вспомогательного характера, размещение которых в основной части записки нецелесообразно или затрудняет ее восприятие.

Разработанное программное обеспечение представляется в электронной форме. Защита контрольной работы должна сопровождаться демонстрацией его работоспособности.

2.3 Содержание разделов пояснительной записки

ВВЕДЕНИЕ (до 1 страницы). Роль программного обеспечения в системах управления. Краткие сведения о конкретной задаче, оценка необходимости ее решения и области применения полученных результатов.

РАЗРАБОТКА ТЕХНИЧЕСКОГО ЗАДАНИЯ НА ПРОЕКТИРОВАНИЕ (до 5 страниц).

ОБЩАЯ ПОСТАНОВКА ЗАДАЧИ. Формулировка задачи, оценка ее сложности и новизны.

ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ К ПРОГРАММЕ. Требования к функциональности и надежности программы, ее информационной и программной совместимости с другими программными средствами. Ограничения на область применения программы. Требования к средствам разработки.

ПРЕДВАРИТЕЛЬНЫЙ ВЫБОР МЕТОДА РЕШЕНИЯ ЗАДАЧИ. Методы решения аналогичных задач. Выбор метода решения задачи. Оценка допущений и ограничений, связанных со сделанным выбором.

ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ К СИСТЕМЕ. Требования к системе в целом. Общие требования к эксплуатации программы, квалификации персонала, эргономике и технической эстетике, защите информации от несанкционированного доступа и от внешней среды, сохранности информации при авариях, патентной чистоте проектных решений, унификации и стандартизации. **Требования к техническому обеспечению.** Технические условия эксплуатации программы. **Требования к программному обеспечению.** Виды системного и прикладного программного обеспечения, необходимого для функционирования программы.

ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

(до 15 страниц).

РАЗРАБОТКА ТЕХНОЛОГИИ ОБРАБОТКИ ИНФОРМАЦИИ. Проектирование процессов ввода информации. Предварительная подготовка вводимых данных. Обеспечение достоверности вводимых данных. Выбор режима работы программы (интерактивный или пакетный). Проектирование основных операций обработки данных. Формирование выходных данных на основе входной и внутренней информации. Разработка структуры технологического процесса обработки информации.

РАЗРАБОТКА СТРУКТУРЫ И ФОРМЫ ПРЕДСТАВЛЕНИЯ ДАННЫХ. Определение структуры и содержания информации. Определение характеристик входных данных (формат, описание, способ получения), внутренних данных (формат, описание) и выходных данных (формат, описание, способ представления).

РАЗРАБОТКА АЛГОРИТМА РЕШЕНИЯ ЗАДАЧИ. Обоснование выбора общего алгоритма решения задачи. Разработка схем решения информационных задач.

РАЗРАБОТКА ПРОГРАММЫ РЕШЕНИЯ ЗАДАЧИ. Выбор инструментальных средств. Обоснование выбора языка и среды программирования. **Разработка структуры программы.** Определение функций программы и ее связей с другими программами. Обоснование выбора подхода к построению структуры программы (предметного, функционального или проблемного). Определение функций составных частей программы и связей между ними. **Проектирование программных модулей.** Разработка спецификаций модулей (прототипов, интерфейсных элементов, внутренних программных объектов). **Проектирование программного интерфейса.** Выбор типа, формы и сценария диалога с пользователем. Проектирование графа диалога. Определение перечня сообщений. Определение требований к эргономике программного обеспечения. Выбор типа интерфейса. Проектирование экранных форм. Описание визуальных компонентов интерфейса. **Обеспечение надежности программы.** Оценка уязвимости информации. Обеспечение защиты от сбоев. Предотвращение несанкционированного получения, уничтожения или модификации информации.

ОПРЕДЕЛЕНИЕ КОНФИГУРАЦИИ ТЕХНИЧЕСКИХ СРЕДСТВ. Описание минимальных и рекомендуемых требований к характеристикам технических средств. Выбор технических средств, необходимых для работы программы.

ТЕСТИРОВАНИЕ ПРОГРАММЫ. Общие сведения. Порядок вызова программы с соответствующего носителя. Контрольные данные для тестирования. Оценка полноты тестовых данных. **Процесс тестирования программы.** Порядок выполнения основных операций при эксплуатации программы. **Оценка надежности программы.** Описание результатов решения контрольных задач. Оценка вероятности безотказной работы программы.

ЗАКЛЮЧЕНИЕ (до 1 страницы). Выводы, определение вариантов использования созданного программного обеспечения и оценка возможных направлений совершенствования.

3 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

3.1 Алгоритмическое представление графа

Способы представления структуры графа в программах различаются объемом занимаемой памяти и скоростью решения задач. Выбор любого из них оказывает принципиальное влияние на эффективность алгоритмов.

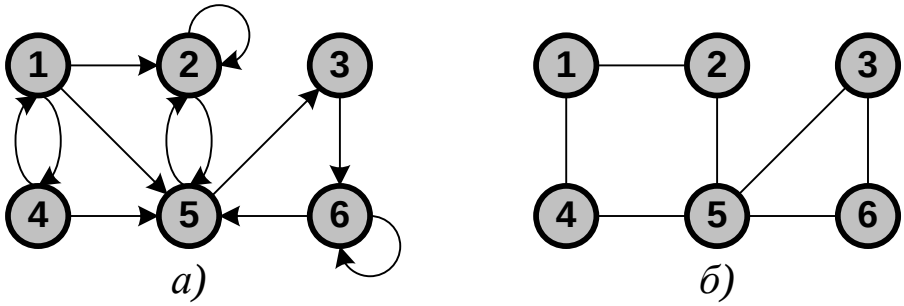


Рисунок 3.1 — Ориентированный (а) и неориентированный (б) графы, используемые в качестве примеров

1. Представление с помощью матрицы инцидентности. Матрица имеет размерность $n \times m$ (***n*** — число вершин, ***m*** — число ребер). Для ориентированного графа ее столбец, соответствующий дуге ***(u, v)***, содержит ***—1*** в строке ***u***, ***1*** в строке ***v*** и нули во всех остальных строках (петлю, т. е. дугу вида ***(u, u)*** представляют иным значением в строке ***u***, например ***2***). Для неориентированного графа столбец, соответствующий ребру ***{u, v}***, содержит ***1*** в строках ***u*** и ***v*** и нули в остальных строках.

С алгоритмической точки зрения это один из худших способов представления графа. Для хранения сильно разреженной структуры используется $n \times m$ элементов. Неудобен также доступ к информации (ответ на элементарные вопросы «существует ли данное ребро», «к каким вершинам ведут ребра из данной вершины» требует перебора всех столбцов матрицы).

	(1, 2)	(1, 4)	(1, 5)	(2, 2)	(2, 5)	(3, 6)	(4, 1)	(4, 5)	(5, 2)	(5, 3)	(6, 5)	(6, 6)
1	–1	–1	–1	0	0	0	1	0	0	0	0	0
2	1	0	0	2	–1	0	0	0	1	0	0	0
3	0	0	0	0	0	–1	0	0	0	1	0	0
4	0	1	0	0	0	0	–1	–1	0	0	0	0
5	0	0	1	0	1	0	0	1	–1	–1	1	0
6	0	0	0	0	0	1	0	0	0	0	–1	2

а)

	{1, 2}	{1, 4}	{2, 5}	{3, 5}	{3, 6}	{4, 5}	{5, 6}
1	1	1	0	0	0	0	0
2	1	0	1	0	0	0	0
3	0	0	0	1	1	0	0
4	0	1	0	0	0	1	0
5	0	0	1	1	0	1	1
6	0	0	0	0	1	0	1

б)

Рисунок 3.2 — Матрица инцидентности: (а) для графа с рисунка 3.1 (а); (б) для графа с рисунка 3.1 (б)

2. Представление с помощью матрицы смежности. Матрица имеет размерность $n \times n$. Ее элемент ***u***—й строки и ***v***—го столбца равен ***1***, если вершина ***v*** смежна с вершиной ***u***. В противном случае он равен ***0***. (Очевидно, что для неориентированного графа матрица смежности симметрична.)

	1	2	3	4	5	6
1	0	1	0	1	1	0
2	0	1	0	0	1	0
3	0	0	0	0	0	1
4	1	0	0	0	1	0
5	0	1	1	0	0	0
6	0	0	0	0	1	1

а)

	1	2	3	4	5	6
1	0	1	0	1	0	0
2	1	0	0	0	1	0
3	0	0	0	0	1	1
4	1	0	0	0	1	0
5	0	1	1	1	0	1
6	0	0	1	0	1	0

б)

Рисунок 3.3 — Матрица смежности: (а) для графа с рисунка 3.1 (а); (б) для графа с рисунка 3.1 (б)

Основное преимущество матрицы смежности — возможность сразу получить ответ на вопрос, существует ли ребро ***{u, v}***. Основной недостаток — независимость ее размера от числа ребер (необходимо всегда размещать $n \times n$ элементов).

3. Представление с помощью списка смежности. Каждый элемент списка содержит номера смежных вершин.

1	1	1	2	2	3	4	4	5	5	6	6
2	4	5	2	5	6	1	5	2	3	5	6

а)

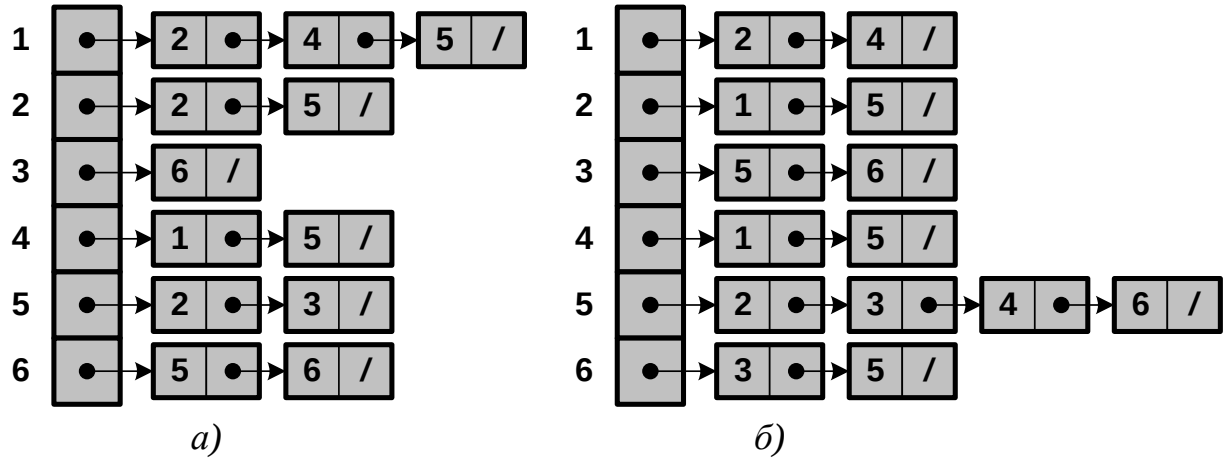
1	1	2	2	3	3	4	4	5	5	5	5	6	6
2	4	1	5	5	6	1	5	2	3	4	6	3	5

б)

Рисунок 3.4 — Список смежности: (а) для графа с рисунка 3.1 (а); (б) для графа с рисунка 3.1 (б)

Этот метод более экономный (особенно в случае неплотных графов, когда $m << n \times n$), так как расход памяти пропорционален $2 \times m$. Однако, доступ к информации неудобен (ответ на вопрос «к каким вершинам ведут ребра из данной вершины» требует перебора элементов списка). Для более эффективного решения задач множество таких пар необходимо упорядочить лексикографически.

4. Представление графа с помощью списков инцидентности. Эта структура каждой вершине $v \in V$ ставит в соответствие список $LIST[v]$, в который включаются (в произвольном порядке) ссылки на все смежные с v вершины.



а) *б)*

Рисунок 3.5 — Списки инцидентности: (а) для графа с рисунка 3.1 (а); (б) для графа с рисунка 3.1 (б)

Основное достоинство списка инцидентности — простота модификации при изменении структуры графа. Расход памяти, необходимой для представления графа, будет иметь порядок $n + m$.

2.2 Поиск в графах

Поиском считается систематический перебор вершин графа. Основные требования к нему следующие:

- алгоритм решения конкретной задачи должен *легко «погружаться»* в процедуру поиска;
- каждое ребро должно анализироваться *конечное число раз*;
- систематическая обработка информации для каждой вершины графа должна выполняться *один раз*.

Если при посещении вершины структура графа не меняется, то наиболее полезными оказываются следующие способы поиска.

Поиск в глубину (DFS — depth first search) производится в следующем порядке:

- посещается в общем случае произвольная *новая* (еще не обработанная) вершина v; она перестает быть *новой*, что отмечается свойством $done[v]$;
- выполняется *рекурсивное обращение* к процедуре поиска для каждой смежной с v *новой* вершины u_j, причем u_j также становится исходной точкой поиска; обход всех ребер (v, u_j) выполняется с учетом установленного для них порядка;
- если новых смежных с v вершин не существует, то производится возврат в вершину, предшествующую v.

Алгоритмы основной программы поиска в глубину в произвольном, не обязательно связном графе и рекурсивной процедуры «погружения» ***DFS*** представлены ниже.

```
for v ∈ V do done[v] ← False
for v ∈ V do if not done[v] then DFS( v )
```

```
DFS( v )
  Выполнить действия над вершиной v.
  done[v] ← True
  for u ∈ LIST[v] do if not done[u] then DFS( u )
```

Устранить рекурсию можно с помощью стека [1—5], в котором запоминаются последовательность просмотренных вершин. Нерекурсивный алгоритм поиска в глубину представлен ниже (символ ⇐ обозначает включение в структуру и исключение из нее).

```
for v ∈ V do done[v] ← False
for v ∈ V do if not done[v] then
  Выполнить действия над вершиной v
  STACK ← ∅; STACK ⇐ v; done[v] ← True
  while STACK ≠ ∅ do
    v ⇐ STACK; STACK ⇐ v // Считывание верхушки стека
    for u ∈ LIST[v] do if not done[u] then break
    if u = 0 then v ⇐ STACK // Список просмотрен
    else
      Выполнить действия над вершиной u
      STACK ⇐ u; done[u] ← True
```

На каждом шаге условного цикла последняя обработанная вершина запоминается в стеке ***STACK***. Ее список инцидентности ***LIST*[*v*]** просматривается до обнаружения необработанного узла, над которым выполняется необходимые действия, или до конца. После этого использованная вершина удаляется из стека. *Чем раньше посещается вершина, тем позже она используется.* Порядок обхода графа при поиске в глубину показан на рисунке.

Поиск в ширину (BFS — breadth first search) начинается с некоторой в общем случае произвольной новой вершины. После этого просматриваются все ее соседи, затем соседи соседей и т. д.

Поиск основывается на замене стека очередью ***QUEUE*** [1—5]. *Чем раньше посещается вершина, тем раньше она используется* (удаляется из очереди). Алгоритм поиска в произвольном, не обязательно связном графе приведен ниже.

```
for v ∈ V do done[v] ← False
for v ∈ V do if not done[v] then
 QUEUE ← ∅; QUEUE ⇐ v; done[v] ← True
 while QUEUE ≠ ∅ do
   v ⇐ QUEUE
   Выполнить действия над вершиной v
   for u ∈ LIST[v] do if not done[u] then
     QUEUE ⇐ u; done[u] ← True
```

Условный цикл работает так же, как и при поиске в глубину (вместо стека используется очередь).

Порядок обхода графа при поиске в глубину и в ширину показан на рисунке 3.6.

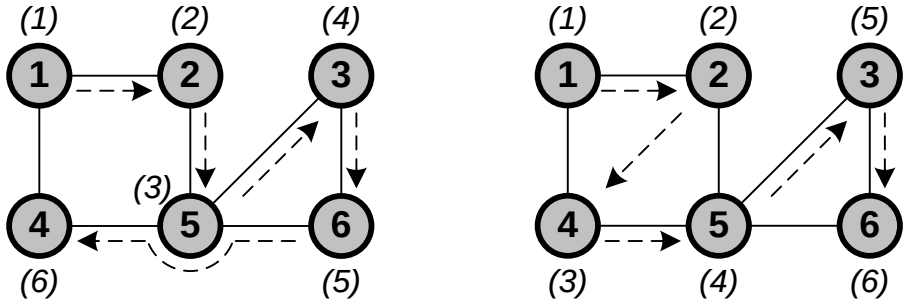


Рисунок 3.6 — Очередность просмотра вершин графа: (а) при поиске в глубину; (б) при поиске в ширину

Оба вида поиска могут быть использованы для *нахождения путей в графе*. Достаточно начать обход из вершины ***v*** и вести его до посещения вершины ***u***. Поиск в глубину является универсальным средством решения большого числа практических задач.

2.3 Циклы

Для любого графа можно определить минимальное **множество фундаментальных циклов**, ни один из которых не может быть получен линейной комбинацией остальных. Любой цикл графа может быть представлен симметрический разностью[♣] (сложением по модулю ***2***) некоторого числа элементов этого множества. Количество фундаментальных циклов называется *циклическим рангом* или *цикломатическим числом* графа. Оно равно $m - n + 1$.

Фундаментальные циклы графа $G = \langle V, E \rangle$ строятся относительно его каркаса T. Добавление в T любого ребра из множества $E \setminus T$ порождает в точности один цикл (см. рисунок 3.7).

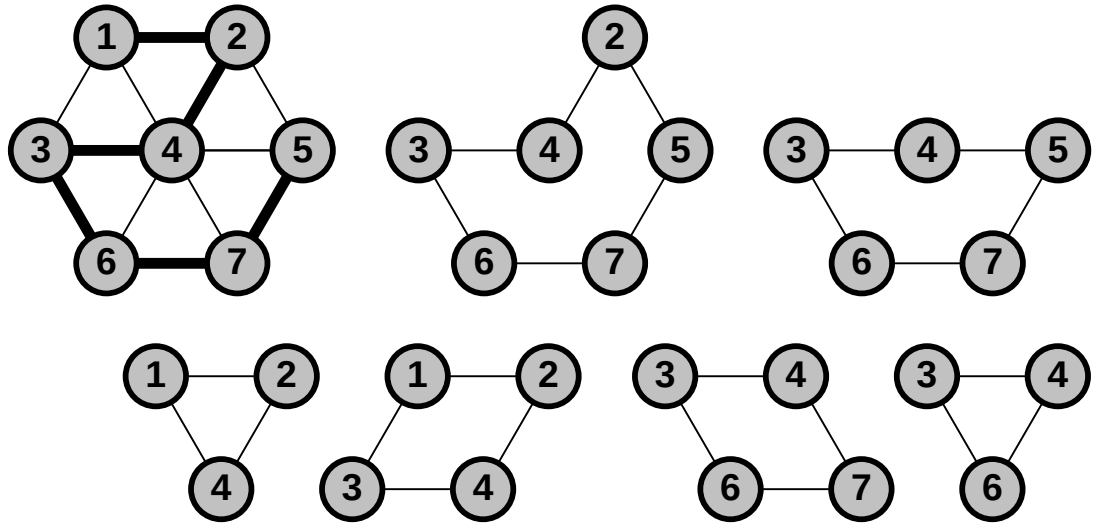


Рисунок 3.7 — Каркас графа, построенный поиском в глубину от вершины ***1***, и множество фундаментальных циклов графа

Алгоритм нахождения множества фундаментальных циклов, основанный на поиске в глубину, строит каркас графа. Цикл порождает ребро, соединяющее новую вершину с той, которая была пройдена ранее.

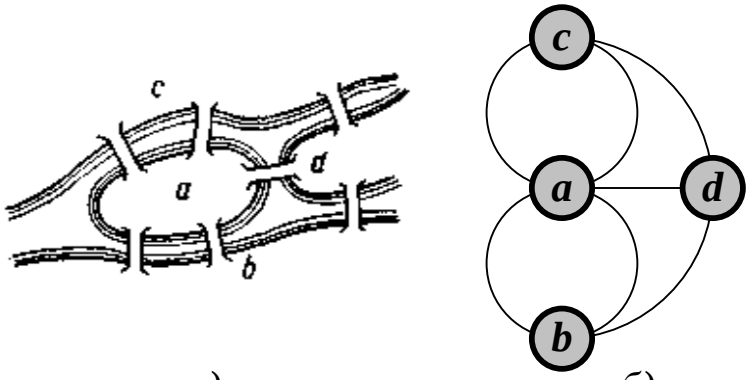
♣ Симметрической разностью двух множеств A и B называется операция $A \oplus B = (A \cup B) \setminus (A \cap B)$.

```
for v ∈ V do ord[v] ← 0; k ← 0
STACK ← ∅; s ← 0
DFS( v )

DFS( v )
  k ← k + 1; ord[v] ← k
  s ← s + 1; STACK ⇐ v
  for u ∈ LIST[v] do
    if ord[u] = 0 then DFS( u ) else
    if (u ≠ STACK[s – 1]) & (ord[u] < ord[v]) then
      Отображение STACK[j] от j = s до STACK[j] = u
  s ← s – 1; v ⇐ STACK
```

В процессе поиска вершины нумеруются в том порядке, в котором они включаются в каркас (что отмечается свойством ***ord***). Каждая новая вершина ***v*** помещается в стек ***STACK***, из которого она удаляется после использования. Стек всегда содержит последовательность вершин от ***v*** до корня стягивающего дерева. Анализируемое ребро ***{v, u}*** замыкает цикл, если вершина ***u*** рассматривалась ранее (***ord*[*v*] > *ord*[*u*] > *0***), и она не предшествует ***v*** (не является предпоследним элементом ***STACK***). Цикл, замыкаемый ребром ***{v, u}***, представляется верхней группой элементов стека.

Цикл, проходящий через каждое ребро связного графа один раз, называется **эйлеровым** (по названию известной задачи о кенигсбергских мостах; Леонардом Эйлером было доказано, что она не имеет решения). Чтобы такой цикл существовал, в графе *не должно быть вершин нечетной степени*.



а) *б)*

Рисунок 3.8 — Иллюстрация к задаче о кенигсбергских мостах: (а) план города; (б) граф

Для построения эйлерового цикла используется модифицированный алгоритм ***DFS*** (последовательность вершин сохраняется в стеке ***STACK***).

```
DFS( v )
 for u ∈ LIST[v] do
   LIST[v] ← LIST[v] \ {u}
   LIST[u] ← LIST[u] \ {v}
   DFS( u )
 STACK ⇐ v
```

Поиск начинается от вершины ***v***. Образующие путь ребра удаляются из графа. Если дальнейшее удлинение пути невозможно (***LIST*[*v*] = ∅**), последняя пройденная вершина переносится в ***STACK***, и поиск продолжается от предыдущей вершины. Пример эйлерового цикла показан на рисунке 3.9 (а).

Если связный граф содержит не более двух вершин нечетной степени, в нем существует *эйлеров путь*, проходящий через каждое ребро по одному разу. Вершины нечетной степени всегда являются его началом и концом. Пример эйлерового пути показан на рисунке 3.9 (б).

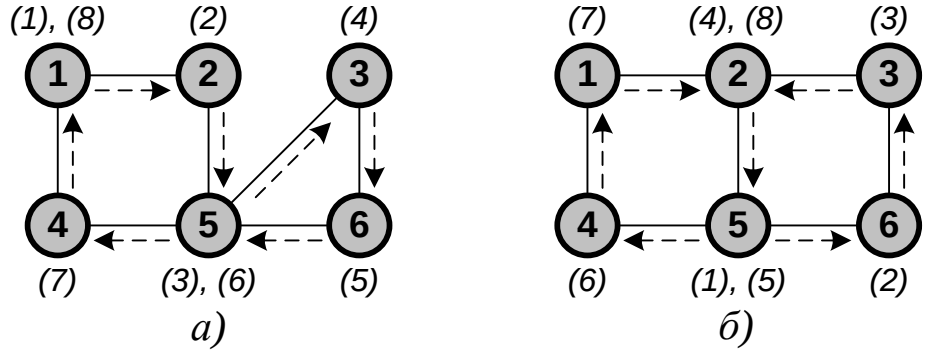


Рисунок 3.9 — Специальные случаи обхода графа: (а) эйлеров цикл; (б) эйлеров путь

Простой цикл, проходящий через каждую вершину (но не обязательно через каждое ребро) связного графа один раз, называется **гамильтоновым** (по названию известной задачи о кругосветном путешествии, придуманной Уильямом Гамильтоном [1—5]). *Гамильтонов путь* определяется очевидным образом. Простые условия существования гамильтоновых циклов не известны. Универсальный алгоритм их построения в произвольном графе за разумное время отсутствует. Для сокращения числа шагов перебора возможных вариантов используют технику «программирования с возвратом». Ее идея заключается в следующем.

Пусть искомое решение имеет вид последовательности вершин $<x_1,\ldots\ x_n>$, которая вначале является пустой. Если уже имеется частное решение $<x_1,\ldots\ x_{k-1}>$, и обнаруживается новая допустимая вершина x_k, то последовательность расширяется до $<x_1,\ldots\ x_k>$. В противном случае производится возврат (англ.: *backtracking*) к найденному ранее частному решению, и продолжается поиск приемлемой, но еще не рассмотренной вершины x_k.

Алгоритмы основной программы построения всех возможных гамильтоновых циклов в графе от вершины v_0 и рекурсивной процедуры расширения последовательности решений **<x[1],... x[k—1]>** ***НМ*** представлены ниже.

```
for v ∈ V do done[v] ← False
```

```
x[1] ← v₀; done[v₀] ← True; HM( 2 )

HM( k )
 for y ∈ LIST[k – 1] do
   if (k = n + 1) & (y = v₀) then
     Использовать <x[1],... x[n], v₀>
   else if not done[y] then
     x[k] ← y; done[y] ← True; HM( k + 1 )
     done[y] ← False // "Возврат"
```

Пример гамильтонова цикла показан на рисунке 3.10 (а). Технология его построения путем поиска в дереве возможных решений отражена на рисунке 3.10 (б).

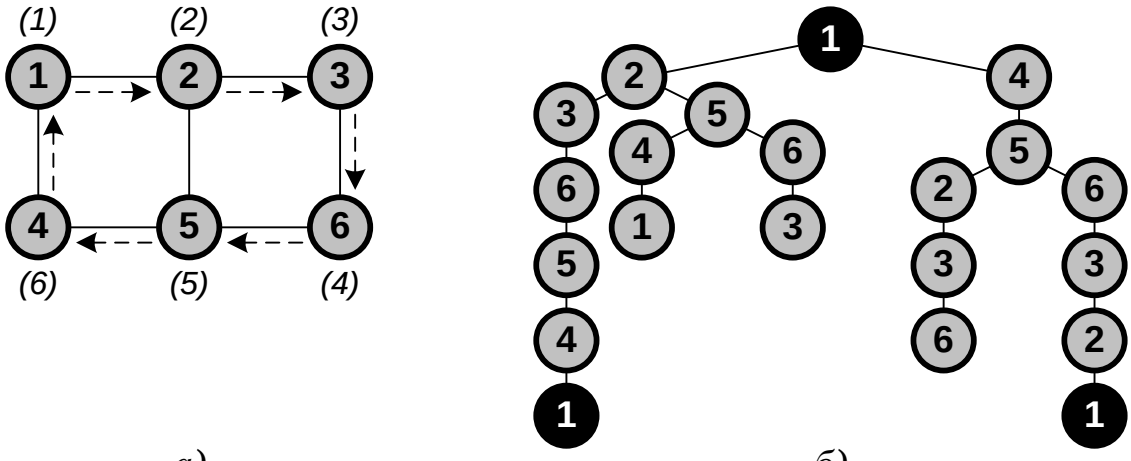


Рисунок 3.10 — Гамильтонов цикл в графе (а) и дерево поиска всех гамильтоновых циклов графа (б)

2.4 Кратчайшие пути

Пусть каждой дуге ***(u, v)*** ориентированного графа приписан вес ***a(u, v)***, определяющий его длину (***a(u, v)*** = ∞, если отсутствует путь из ***u*** в ***v***; ***a(u, u)*** = ∞, если отсутствует петля ***(u, u)***). Тогда можно поставить задачу поиска **кратчайшего пути между фиксированными вершинами *s*** (*source*) и ***t*** (*target*). Если ***s*** ∈ ***V*** — начало после-

довательности вершин $<v_0, v_1, v_2, \ldots v_k>$, а $t \in V$ — ее конец, то длина такого пути (*расстояние от* ***s*** *до* ***t***) определится как сумма:

$$d(s,t) = \sum_{i=1}^{k} a(v_{i-1}, v_i).$$

Если не существует ни одного пути из ***s*** в ***t***, то $d(s, t) = \infty$.

Если каждый контур графа имеет положительную длину, то кратчайший путь всегда будет простым. Если в графе существует контур отрицательной длины, расстояние между некоторыми вершинами становится неопределенным

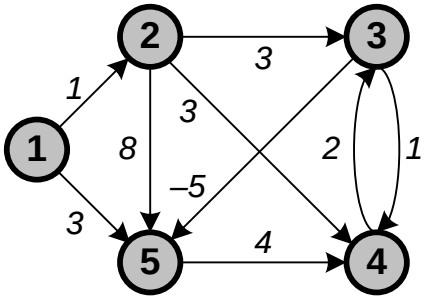
Кратчайшие пути определяются следующим образом. Для произвольных $s, t \in V$ $(s \neq t)$ существует такая вершина v, что $d(s, t) = d(s, v) + a(v, t)$ (это предпоследняя вершина произвольного кратчайшего пути из ***s*** в ***t***). Далее можно найти вершину u, для которой $d(s, v) = d(s, u) + a(u, v)$, и т. д. При положительной длине всех контуров созданная таким образом последовательность $t, v, u, \ldots$ не содержит повторений и оканчивается вершиной ***s***. При обращении очередности она определяет кратчайший путь из ***s*** в ***t***.

Большинство алгоритмов нахождения кратчайших путей между двумя фиксированными вершинами используют матрицу весов дуг $A[u, v]$ $(u, v \in V)$. С ее помощью вычисляются некоторые верхние ограничения $D[v]$ на расстояния от ***s*** до всех вершин $v \in V$. Каждый раз, когда устанавливается, что $D[v] > D[u] + A[u, v]$, оценка $D[v]$ улучшается: $D[v] \leftarrow D[u] + A[u, v]$. Процесс прерывается, когда дальнейшее улучшение ни одного из ограничений невозможно. Очевидно, что значение каждой из переменных $D[v]$ равно $d(s, v)$.

Ниже приведен алгоритм Форда (*Ford*) и Беллмана (*Bellman*) вычисления кратчайших расстояний $D[v]$ от источника ***s*** до всех вершин графа $v \in V$. Предполагается, что граф *не содержит контуров отрицательной длины*. Для каждой вершины v определяется номер ее предшественницы в кратчайшем пути $P[v]$, который используется при построении этого пути.

```
// Начальные оценки
for v ∈ V do
 D[v] ← A[s, v]; P[v] ← s
D[s] ← 0; P[s] ← 0
// Поиск кратчайших путей
for k ← 1 to n – 2 do
 for v ∈ V / {s} do for u ∈ V fo
   if D[v] > D[u] + A[u, v] then
     D[v] ← D[u] + A[u, v]
     P[v] ← u
// Отображение кратчайшего пути
v ← t
while P[v] ≠ 0 do
 Отобразить v
 v ← P[v]
Отобразить s
```

Работа алгоритма Форда-Беллмана проиллюстрирована на рисунке 3.11 (исходной является вершина ***1***).



$$A = \begin{pmatrix} \infty & 1 & \infty & \infty & 3 \\ \infty & \infty & 3 & 3 & 8 \\ \infty & \infty & \infty & 1 & -5 \\ \infty & \infty & 2 & \infty & \infty \\ \infty & \infty & \infty & 4 & \infty \end{pmatrix}$$

k	*D*[*1*]	*D*[*2*]	*D*[*3*]	*D*[*4*]	*D*[*5*]
0	***0***	***1***	∞	∞	***3***
1	***0***	***1***	***4***	***4***	***—1***
2	***0***	***1***	***4***	***3***	***—1***
3	***0***	***1***	***4***	***3***	***—1***

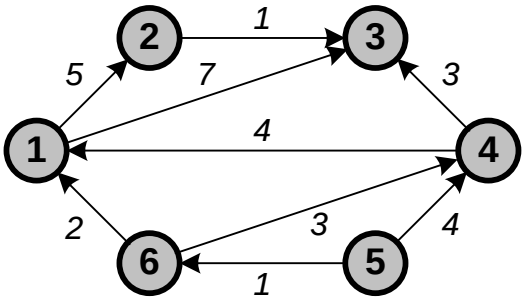
Рисунок 3.11 — Работа алгоритма Форда-Беллмана

Если *веса всех дуг неотрицательны*, то можно использовать эффективный алгоритм Дейкстры (*Dijkstra*), работа которого проиллюстрирована на рисунке 3.12 (исходной является вершина ***5***).

```
// Начальные оценки
for v ∈ V do
 D[v] ← A[s, v]; done[v] ← False; P[v] ← s
D[s] ← 0; done[s] ← True; P[s] ← 0
u ← s
// Поиск кратчайших путей
for k ← 1 to n – 1 do
 for v ∈ V do if not done[v] then
   if A[u, v] ≠ ∞ then
     if D[v] > D[u] + A[u, v] then
       D[v] ← D[u] + A[u, v]
       P[v] ← u
 minDist ← ∞
 for v ∈ V do if not done[v] then
   if minDist > D[v] then minDist ← D[v]; u ← v
 done[u] ← True
// Отображение кратчайшего пути
v ← t
while P[v] ≠ 0 do
 Отобразить v
 v ← P[v]
Отобразить s
```

Эффективный алгоритм Флойда (*Floyd*) и Уоршалла (*Warshall*), использующий технику динамического программирования, вычисляет **кратчайшие расстояния между всеми парами вершин** ориентированного графа (*допустимы дуги с отрицательным весом, но не контура отрицательной длины*). Матрица ***D*[*u, v*]** хранит огра-

ничения на расстояния между вершинами. Матрица предшествования ***P*[*u*, *v*]** используется для построения кратчайших путей.



$$A = \begin{pmatrix} \infty & 5 & 7 & \infty & \infty & \infty \\ \infty & \infty & 1 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty \\ 4 & \infty & 3 & \infty & \infty & \infty \\ \infty & \infty & \infty & 4 & \infty & 1 \\ 2 & \infty & \infty & 3 & \infty & \infty \end{pmatrix}$$

k	*D*[*1*]	*D*[*2*]	*D*[*3*]	*D*[*4*]	*D*[*5*]	*D*[*6*]
0	∞	∞	∞	***4***	***0***	***1***
1	∞	∞	∞	***4***	***0***	***1***
2	***3***	∞	∞	***4***	***0***	***1***
3	***3***	***8***	***10***	***4***	***0***	***1***
4	***3***	***8***	***7***	***4***	***0***	***1***
5	***3***	***8***	***7***	***4***	***0***	***1***

Рисунок 3.12 — Работа алгоритма Дейкстры

```
// Начальные оценки
for i ← 1 to n do
 for j ← 1 to n do
   D[i, j] ← A[i, j]
   if (A[i, j] ≠ ∞) & (i ≠ j)
     then P[i, j] ← i
     else P[i, j] ← 0
 D[i, i] ← 0
 P[i, i] ← 0
// Поиск кратчайших путей
for k ← 1 to n do
  for i ← 1 to n do
```

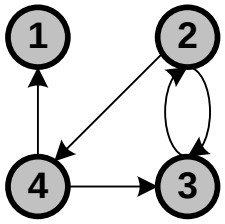
```
    for j ← 1 to n do
  if D[i, j] > D[i, k] + D[k, j] then
    D[i, j] ← D[i, k] + D[k, j]
    P[i, j] ← P[k, j]
// Отображение кратчайшего пути
v ← t
while P[s, v] ≠ 0 do
 Отобразить v
 v ← P[s, v]
Отобразить s
```

Алгоритм Флойда-Уоршалла можно использовать для решения вопроса о существовании пути между любой парой вершин ориентированного графа ***G(V, E)***. Отношение достижимости вершин называется **транзитивным замыканием** графа. При его вычислении считают, что все дуги имеют единичный вес. Если существует путь из ***u*** в ***v***, то ***D*[*u, v*] < *n***, иначе ***D*[*u, v*] = ∞**.

На практике в алгоритме вместо арифметических действий используют высокоэффективные логические операции. Если существует путь из ***u*** в ***v***, то элемент транзитивного замыкания ***D*[*u, v*] = *1***, иначе ***D*[*u, v*] = *0***.

```
for i ← 1 to n do
 for j ← 1 to n do
   if (A[i, j] ≠ ∞) or ((i, j) ∈ E)
     then D[i, j] ← 1
     else D[i, j] ← 0
for k ← 1 to n do
 for i ← 1 to n do
   for j ← 1 to n do
     D[i, j] ← D[i, j] ∨ (D[i, k] & D[k, j])
```

Пример графа и матрицы, вычисленные с помощью этого алгоритма, показаны на рисунке 3.13.



$$D^{(0)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix}, \; D^{(1)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix}, \; D^{(2)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix},$$

$$D^{(3)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}, \; D^{(4)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}.$$

Рисунок 3.13 — Ориентированный граф и матрицы, вычисленные алгоритмом построения транзитивного замыкания

При положительных весах ребер задача поиска кратчайших путей в неориентированном графе легко сводится к аналогичной задаче для ориентированного графа. Достаточно заменить каждое ребро ***{u, v}*** двумя дугами ***(u, v)*** и ***(v, u)*** с таким же весом. Однако при наличии ребер с неположительным весом это может приводить к возникновению контуров с неположительной длиной.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Методические указания к оформлению научно-технической документации / Д.В. Болдырев, Л.Г. Гордиенко. — Невинномысск : изд-во НТИ (филиала) ГОУ ВПО «СевКавГТУ», 2008. — 45 с.

2. ГОСТ 7.1-2003. Библиографическая запись. Библиографическое описание. Общие требования и правила составления [Текст]. — Взамен ГОСТ 7.1-84, ГОСТ 7.16-79, ГОСТ 7.18-79, ГОСТ 7.34-81, ГОСТ 7.40-82 ; введ. 2004-07-01. — Минск : Межгос. совет по стандартизации, метрологии и сертификации ; М. : Изд-во стандартов, 2003. — 62 с. — (Система стандартов по информации, библиотечному и издательскому делу).

ПРИЛОЖЕНИЕ А

Варианты заданий

1. Задана система односторонних дорог. Найти путь, соединяющей города ***A*** и ***B***, который не проходит через заданное множество городов.

2. Задана система двусторонних дорог. Определить, является ли она трехсвязной.

Рекомендации: *система двусторонних дорог будет считаться трехсвязной, если для любой четверки разных городов* ***A***, ***B***, ***C*** *и* ***D*** *существует два различных пути из* ***A*** *в* ***D****, причем один из них проходит через* ***B****, а другой — через* ***C****.*

3. Задана система двусторонних дорог. Для каждой пары городов определить длину кратчайшего пути между ними.

4. Задана система двусторонних дорог. Найти два города и соединяющий их путь, который проходит через каждую из дорог ровно один раз.

5. Задана система двусторонних дорог. Найти замкнутый путь длиной не более ***100*** км, проходящий через каждую дорогу ровно один раз.

6. Задана система двусторонних дорог, причем для любой пары городов можно указать соединяющий их путь. Найти такой город, для которого сумма расстояний до остальных городов минимальна.

7. Задана система односторонних дорог. Определить, есть ли в ней город, из которого можно добраться до каждого из остальных городов, проезжая не более ***100*** км.

8. Задана система односторонних дорог. Определить, есть ли в ней город, куда можно попасть из любого другого города, проезжая

не более ***100*** км.

9. Задана система двусторонних дорог. Определить, можно ли, построив какие-нибудь новые три дороги заданной длины, из заданного города добраться до каждого из остальных городов, проезжая не более ***100*** км.

10. Задана система двусторонних дорог. Определить, можно ли, закрыв какие-нибудь три дороги, добиться того, чтобы из города ***A*** нельзя было попасть в город ***B***.

11. Задана система двусторонних дорог, связывающих фиксированное число городов. Для заданного ***N*** определить периферию столицы.

Рекомендации: *периферией считается множество городов, расстояние от которых до выделенного города больше* ***N***.

12. Задана система односторонних дорог. Определить, можно ли проехать из города ***A*** в город ***B*** таким образом, чтобы посетить город ***C*** и не проезжать никакой дороги более одного раза.

13. Задана система двусторонних дорог. За проезд по каждой из них взимается пошлина. Найти путь из города ***A*** в город ***B*** с минимальной величиной ***S + P***, где ***S*** — сумма длин дорог пути, а ***P*** — сумма пошлин проезжаемых дорог.

14. Заданы две системы двусторонних железных и шоссейных дорог с одним и тем же множеством городов. Найти минимальный по длине путь из города ***A*** в город ***B***, который может проходить как по железным, так и по шоссейным дорогам, и места пересадок с одного вида транспорта на другой на этом пути.

15. Задана система односторонних дорог. Найти длину самого длинного простого пути от города ***A*** до города ***B***.

ПРИЛОЖЕНИЕ Б

Список рекомендуемой литературы

1 Липский В. Комбинаторика для программистов. — М.: Мир, 1988. — 204 с.

2 Кнут Д. Искусство программирования на ЭВМ: в 3-х кн. — М.-СПб.-Киев: Вильямс, 2001. — info@citforum.ru.
Кн. 1. Основные алгоритмы. — 2001. — 814 с.
Кн. 2. Получисленные алгоритмы. — 2001. — 782 с.
Кн. 3. Сортировка и поиск. — 2001. — 764 с.

3 Стивенс Р. Готовые алгоритмы. — М.: ДМК, 2001. — 760 с.

4 Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы. Построение и анализ. — М.: МЦНМО, 2001. — 960 с.

5 Топп У., Форд У. Структуры данных в C++. — М.: БИНОМ, 2000. — 816 с.

6 Новиков Ф.А. Дискретная математика для программистов. — СПб.: Питер, 2001. — 304 с.

ОБРАБОТКА ИНФОРМАЦИИ В СИСТЕМАХ УПРАВЛЕНИЯ

Методические указания к выполнению контрольной работы для магистров направления подготовки 15.04.04 — Автоматизация технологических процессов и производств

Составитель *Д.В. Болдырев*

Редактор

Подписано в печать Формат 60×84 1/16
Уч.-изд. л. 2,25 Усл. печ. л. 2,3 Тираж 50 экз. Заказ №
Невинномысский технологический институт (филиал)
ФГАОУ ВПО «Северо-Кавказский федеральный университет»

Типография