МИР
программирования

Р. ХАГГАРТИ

Дискретная математика для программистов

Издание 2-е, исправленное

Перевод с английского
под редакцией С.А. Кулешова
с дополнениями А.А. Ковалева,
В.А. Головешкина, М.В. Ульянова

Допущено УМО вузов РФ по образованию в области прикладной математики в качестве учебного пособия для студентов высших учебных заведений, обучающихся по направлению подготовки «Прикладная математика»

ТЕХНОСФЕРА
Москва
2012

УДК 519.854
ББК 22.176
Х13
Х13 Хаггарти Р.
Дискретная математика для программистов
Издание 2-е, исправленное

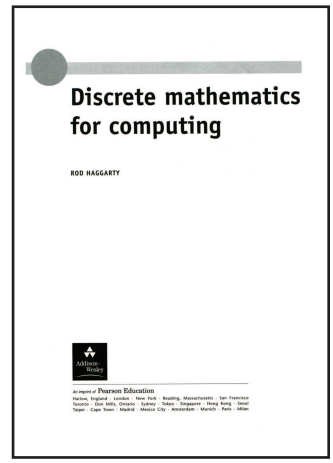
Москва: Техносфера, 2012. – 400 с., ISBN 978-5-94836-303-5

Основополагающее введение в дискретную математику, без знания которой невозможно успешно заниматься информатикой и программированием. Ни одно из многочисленных изданий по этой дисциплине, вышедших на русском языке, не читается с таким удовольствием и пользой. В доступной и весьма увлекательной форме автор рассказывает о фундаментальных понятиях дискретной математики – о логике, множествах, графах, отношениях и булевых функциях. Теория изложена кратко и иллюстрируется многочисленными простыми примерами, что делает ее доступной даже школьнику. После каждой главы (начиная со второй) рассматривается приложение описанных методов к информатике.

Дополнения в издании на русском языке посвящены актуальным задачам теории графов, рекурсивным алгоритмам, общей проблеме перебора и задачам целочисленного программирования.

Книга будет полезна студентам, изучающим курс дискретной математики, а также всем желающим проникнуть в технику написания и проверки корректности алгоритмов, включая программистов-практиков.

УДК 519.854
ББК 22.176



ISBN 978-5-94836-303-5
ISBN 0-201-73047-2 (англ.)

Содержание

Указатель обозначений

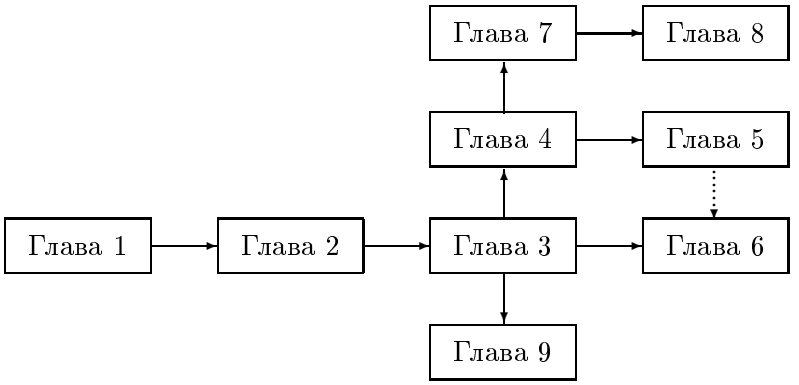
R^*	замыкание отношения R	75
E_x	класс эквивалентности элемента x	78
$x \prec y$	x — непосредственный предшественник y	80
проект	операция «проект»	87
соединение	операция «соединение»	88
выбор	операция «выбор»	89
R^{-1}	обратное отношение	91
$S \circ R$	композиция отношений R и S	92
MN	булево произведение матриц M и N	94
$f(x)$	образ элемента x	97
$f\colon A \longrightarrow B$	функция из A в B	97
$f(A)$	множество значений функции f	97
$f^{-1}\colon\longrightarrow A$	обратная функция	102
$g \circ f$	композиция функций f и g	104
$\lvert x \rvert$	модуль числа x	110
$\lfloor x \rfloor$	целая часть числа x	110
$P(n,\, k)$	число всех $(n,\, k)$-размещений без повторений	121
$C(n,\, k)$	число всех $(n,\, k)$-сочетаний без повторений	123
$O(g(n))$	класс функций, растущих не быстрее, чем $g(n)$	137
$\delta(v)$	степень вершины	143
$G = (V,\, E)$	граф с множеством вершин V и множеством ребер E	143
$c(G)$	число компонент связности	146
K_n	полный граф с n вершинами	148
МОД	минимальное остовное дерево	154
P	граф Петерсена	160
ПЕРТ	система планирования и руководства разработками	171
M^k	булево произведение k экземпляров матрицы M	176
M^*	матрица достижимости	176
$d[v]$	расстояние до вершины v	182
$\bar{p}$	отрицание булевой переменной p	195
$p \vee q$	дизъюнкция переменных p и q	195
$p \wedge q$	конъюнкция переменных p и q	195
НЕ–И	функция **НЕ–И**	200

Предисловие

Основная цель этой книги — рассказать об основной математической технике, необходимой студентам, изучающим информатику. Представленные здесь темы интересны и сами по себе, и в связи с их широкой применимостью как непосредственно в математике, так и в дисциплинах, использующих математический аппарат. В частности, формальные методы, применяемые в информатике, опираются на такие фундаментальные понятия дискретной математики, как логика, множества, отношения и функции.

Теория излагается преднамеренно кратко, а обсуждаемые здесь математические идеи вполне доступны студентам со скромной математической подготовкой. В многочисленных примерах обобщаются и развиваются ключевые идеи курса, а каждая глава, начиная со второй, снабжена приложением теории к практике. Приложения наглядно демонстрируют, как математика, о которой рассказывается в книге, решает задачи информатики. Каждая глава заканчивается набором упражнений, а чтобы поощрить читателя заниматься ими, полное решение приводится только в конце книги.

Основной материал книги появился при подготовке к чтению начального (годового) курса информатики в Оксфорде. Он рассчитан на 20 лекций. Зависимость глав друг от друга представлена на диаграмме, которая показывает, что существует некоторая свобода выбора очередности изучения материала. Это вместе с возможностью опускать отдельные приложения или заменять их альтернативными, делает книгу более гибкой и удобной для изучения.



Есть несколько доступных текстов по дискретной математике, охватывающих схожий материал. Их список для дальнейшего изучения предмета приведен в конце книги. Более продвинутые учебники по дискретной математике требуют большей математической зрелости, и я надеюсь, что читатели, успешно овладевшие содержанием настоящей книги, смогут изучать их более легко и уверенно.

Я хотел бы поблагодарить своих студентов, кто выдержал все трудности этого материала и чей рост собственных математических способностей поощрял меня писать книгу. Моя благодарность адресована также рецензентам предварительного варианта, сделавшим много полезных замечаний, и сотрудникам издательства «Pearson Education» за их усилия, предпринятые при оформлении текста. И наконец, моя признательность — супруге, за ее неизменную заботу и поддержку.

Род Хаггарти
Оксфорд
Март 2001

ГЛАВА 1

ВВЕДЕНИЕ

Дискретная математика и логика лежат в основе любого современного изучения информатики. Слово «дискретный» означает «составленный из отдельных частей», а дискретная математика имеет дело с совокупностями объектов, называемых множествами, и определенными на них структурами. Элементы этих множеств как правило изолированы друг от друга и геометрически не связаны. Действительно, большинство интересующих нас множеств конечны или, по крайней мере, счетны.

Эта область математики привлекается для решения задачи на компьютере в терминах аппаратных средств и программного обеспечения с привлечением организации символов и манипуляции данными. Современный цифровой компьютер — по существу конечная дискретная система. Понимания того, как такая машина работает, можно достигнуть, если представить машину как дискретную математическую систему. Поэтому наша главная цель при изучении дискретной математики — приобрести инструменты и технику, необходимые для понимания и проектирования компьютерных систем. Когда и как использовать эти инструменты и технику — основа раздела математики, известного как математическое моделирование.

В настоящей главе мы бросим взгляд на процесс моделирования и применим стандартный алгоритм к решению практической задачи. Выбранный пример проиллюстрирует не только вид математики, о которой идет речь в этой книге, но и ее использование при решении насущных задач. Затем мы разовьем паскалеподобный[1] псевдокод в качестве средства выражения алгоритмов, вводимых далее, для однозначной трактовки их команд.

1.1. Моделирование

Процесс математического моделирования на диаграмме можно представить так, как показано на рис. 1.1.
В качестве примера моделирования рассмотрим следующую задачу:

> *Расстояние (в милях) между шестью шотландскими городами: Абердин, Эдинбург, Форт Уильям, Глазго, Инвернесс*

[1] Pascal — язык программирования высокого уровня. — *Прим. перев.*

и Перт дано в табл. 1.1. Требуется найти дорожную сеть минимальной длины, связывающую все шесть городов.

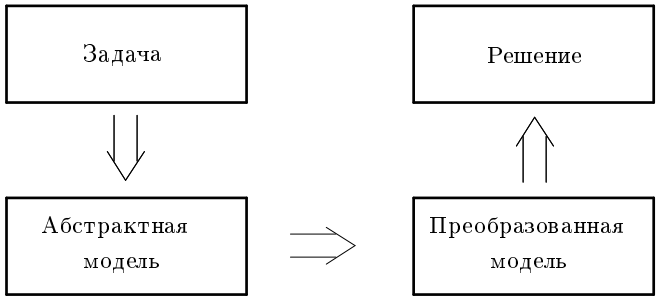


Рисунок 1.1. Схема моделирования

Сама таблица является абстрактной моделью реальной задачи. Однако для нашего решения мы преобразуем ее в геометрическую модель.

Таблица 1.1

	Абердин	Эдинбург	Форт Уильям	Глазго	Инвернесс	Перт
Абердин	—	120	147	142	107	81
Эдинбург	120	—	132	42	157	45
Форт Уильям	147	132	—	108	66	105
Глазго	142	42	108	—	168	61
Инвернесс	107	157	66	168	—	112
Перт	81	45	105	61	112	—

Мы нарисуем *граф*, чьи *вершины* обозначают города, а *ребра* — дороги их связывающие. Более подробно о графах рассказано в главе 7. Каждое ребро нашего графа, изображенного на рис. 1.2, снабжено *весом*, который означает расстояние между соответствующими городами согласно табл. 1.1.

Для решения поставленной задачи с помощью подходящего *алгоритма* (последовательности однозначных инструкций, выполнение которых влечет решение за конечное время), мы построим новый граф, имеющий минимальный общий вес, в котором все шесть городов будут соединены дорогами.

Алгоритм Прима

Шаг 1 Выберите произвольную вершину и ребро, соединяющее ее с ближайшим (по весу) соседом.

Шаг 2 Найдите не присоединенную (еще) вершину, ближе всего лежащую к одной из присоединенных, и соедините с ней.

Шаг 3 Повторяйте шаг 2 до тех пор пока все вершины не будут присоединены.

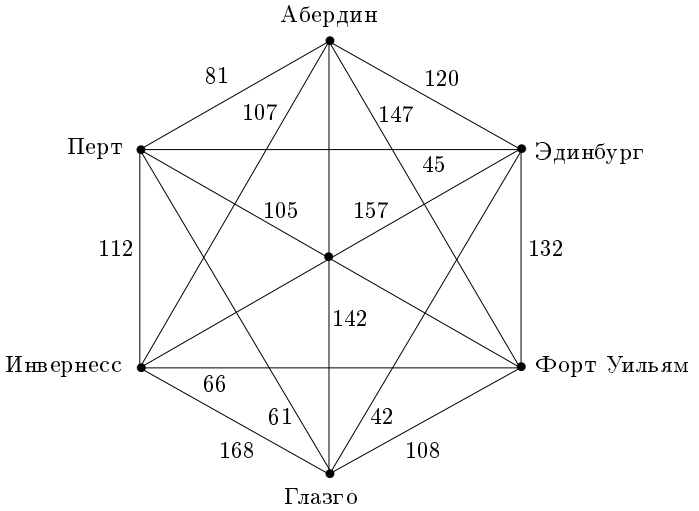


Рисунок 1.2.

На рисунках 1.3, 1.4 и 1.5 изображена последовательность графов, которая получается в результате применения алгоритма Прима, если начинать с вершины Перт. Последний граф (с общим весом 339) представляет собой минимальную сеть дорог, охватывающую все шесть городов.

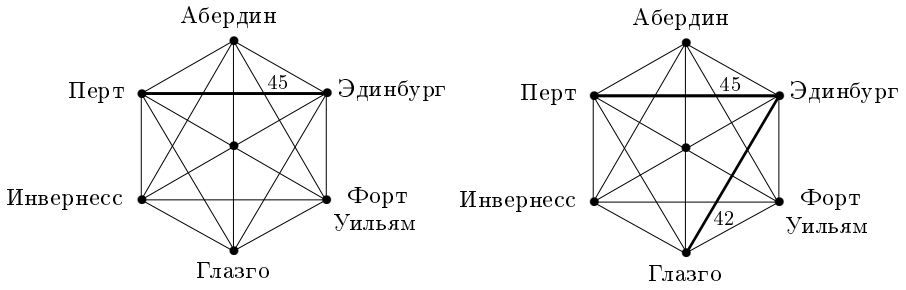


Рисунок 1.3.

Алгоритм, который мы применяли, написан на обычном русском языке. Разговорный язык может оказаться слишком многоречивым, неоднозначным и, в следствие этого, не соответствующим запутанной проблеме. Мы могли бы написать программу для компьютера,

реализующую алгоритм, но какой язык выбрать? Кроме того, язык программирования зачастую скрывает истинный смысл алгоритма от неопытного читателя! Подходящий компромисс в этой ситуации — использовать так называемый *псевдокод*, состоящий из небольшого числа структурных языковых элементов вместе с русскоподобным описанием действий реализуемого алгоритма. О нем идет речь в следующем параграфе.

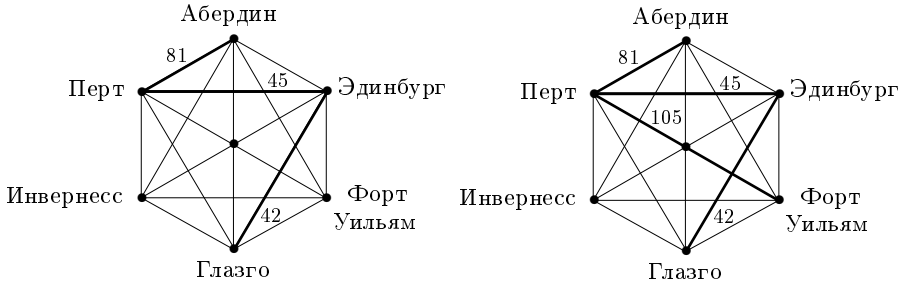


Рисунок 1.4.

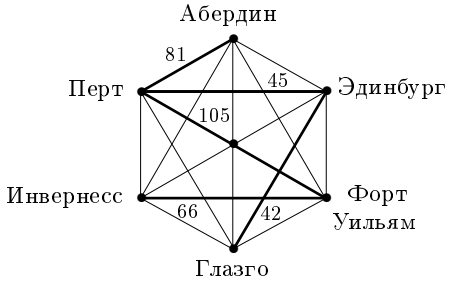


Рисунок 1.5.

1.2. Псевдокод

Мы будем использовать псевдокод, основанный на Паскале. Алгоритм в нем выглядит следующим образом.

begin
операторы исполняемых действий
операторы, управляющие порядком выполнения
end

Строительными блоками алгоритмического языка являются операторы, которые можно разбить на две категории: операторы присваивания и управляющие операторы.

Оператор присваивания приписывает переменным определенные величины и имеют такую общую форму:

имя переменной := *выражение*

Пример 1.2.1. (Алгоритм сложения двух чисел, *First* и *Second*, и присвоение результата переменной *Sum*.)

begin
Input First and Second;
Sum:= *First* + *Second;*
end

Управляющий оператор определяет порядок, в котором должны выполняться шаги алгоритма. Операторы управления бывают трех типов:

- составные операторы;
- условные операторы;
- оператор цикла.

Составные операторы представляют собой список операторов, которые должны выполняться как отдельная команда в том порядке, в котором они записаны. Составные операторы имеют следующий вид:

begin
оператор 1;
оператор 2;
.....
оператор n;
end

Пример 1.2.2. (Алгоритм обмена значений двух переменных: *One* и *Two*.)

begin
Input One and Two;
Temp := *One*;
One := *Two*;
Two := *Temp*;
end

Чтобы проследить за работой алгоритма, предположим, что начальные значения переменных *One* и *Two* равны 5 и 7 соответственно, и обратимся к табл. 1.2.

Таблица 1.2

	Temp	*One*	*Two*
Строка 1	—	5	7
Строка 2	5	5	7
Строка 3	5	7	7
Строка 4	5	7	5

Условные операторы позволяют делать выбор между двумя альтернативными ситуациями. Они записываются в виде **if-then** или **if-then-else**. На псевдокоде условные операторы изображают так:

begin
  if *условие* **then** *оператор*
end

или так:

begin
  if *условие* **then** оператор 1
    else *оператор 2*
end

Пример 1.2.3. (Алгоритм вычисления модуля числа n и присвоение результата переменной *abc*.)

begin
  Input n;
  if $n < 0$ **then** $abc := -n$
    else $abc := n$;
  Output abc;
end

В этом алгоритме оператор, стоящий во второй строке, выполняется при отрицательных значениях переменной n, а в третьей — при положительных (и нулевом). Можно написать и другой алгоритм, решающий ту же задачу, но не использующий **else**:

begin
  Input n;
  if $n < 0$ **then** $n := -n$;
  $abc := n$;
  Output abc;
end

Здесь оператор во второй строчке выполняется только при отрицательных значениях n и игнорируется при любом другом значении.

В последнем случае выполняется оператор, записанный в третьей строке.

Оператор цикла или просто *цикл* может иметь одну из форм записи:

for $X := A$ **to** Z **do** оператор; (1)

while выражение **do** оператор; (2)

repeat
оператор 1;
оператор 2;
.....
оператор n;
until условие. (3)

Здесь X — переменная, а A и Z — ее начальное и конечное значения.

В случае (1) цикл повторяется определенное число раз. Его разновидность выглядит следующим образом:

for *всех элементов множества* **do** *оператор*

В случае (2) цикл выполняется не определенное число раз, а до тех пор, пока выражение, о котором в нем идет речь, остается верным. Как только выражение становится ложным, цикл заканчивается.

И наконец, в последней ситуации (3) цикл выполняется до тех пор, пока конечное условие остается ложным. Единственное различие между (2) и (3) заключается в том, что в последнем цикл выполнится по крайней мере один раз, поскольку истинность условия в нем проверяется *после* каждого прохода цикла.

Пример 1.2.4. (Алгоритм вычисления суммы квадратов первых n натуральных чисел.)

```
begin
    sum := 0;
    for i := 1 to n do
        begin
            j := i * i;
            sum := sum + j;
        end
    Output sum;
end
```

Проследим алгоритм в случае $n = 4$, записав результаты в табл. 1.3

Таблица 1.3

	i	j	Sum
Перед выполнением цикла	—	—	0
Первый проход цикла	1	1	1
Второй проход цикла	2	4	5
Третий проход цикла	3	9	14
Четвертый проход цикла	4	16	30

Выводимый результат: $sum = 30$.

Пример 1.2.5. (Алгоритм выделения графа с минимальным общим весом, связывающего все вершины в данном связном взвешенном графе.)

```
begin
    v := произвольная вершина;
    u := ближайшая соседняя вершина;
    связать v и u;
    while остаются неприсоединенные вершины do
        begin
            u := неприсоединенная вершина, ближайшая
                 к одной из присоединенных вершин;
            соединить u с ближайшей
            из присоединенных вершин;
        end
end
```

Это — написанная на псевдокоде версия алгоритма Прима, с которым мы познакомились ранее.

Замечание. *Связным называется такой граф, в котором существует путь (по ребрам) между любыми двумя вершинами (подробнее об этом см. главу 7, стр. 146).*

Превращение алгоритма в работающую программу — дело программирования или курса структуры данных, поэтому мы не будем обсуждать этот процесс в нашей книге. Однако мы познакомимся со множеством алгоритмов, некоторые из которых представлены в форме псевдокода, а другие оформлены как математические теоремы. Доказательство истинности теорем — необходимая и далеко нетривиальная часть математического процесса. Аналогично необходимо проверять корректность написанного на псевдокоде алгоритма. Например, откуда мы можем знать, что алгоритм из примера 1.2.5 действительно дает минимальную сеть дорог?

В том случае, когда есть несколько различных алгоритмов, решающих одну и ту же задачу, возникает вопрос: какой из них является более эффективным? В упражнении 1.5 приведен еще один алгоритм, суммирующий квадраты натуральных чисел (как и в примере 1.2.4). Оба работают. Но какой это делает быстрее, использует при этом меньше памяти? Короче говоря, какой из этих алгоритмов является наилучшим?

Обе эти проблемы: корректности и эффективности алгоритмов — будут обсуждаться в последующих главах после того, как мы освоим необходимый для этого аппарат дискретной математики.

Набор упражнений 1

1.1. Граф на рисунке рис. 1.6 изображает сеть дорог, связывающих семь деревень. Расстояние между деревнями задано в милях. Используя алгоритм Прима, найдите сеть дорог минимальной общей длины, охватывающую все деревни.

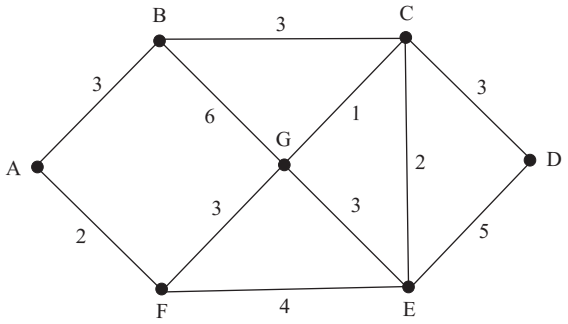


Рисунок 1.6.

1.2. Найдите результаты вычислений следующего алгоритма в случаях

(а) $n = 3$;

(б) $n = 5$.

```
begin
    f := 1;
    Input n;
    for i := 1 to n do
        f := f * i;
    Output f;
end
```

Что получится на выходе алгоритма при произвольном натуральном числе n?

1.3. Проследите за изменением значений переменных i и j в следующем алгоритме при $m = 3$ и $n = 4$:

```
begin
    Input m, n;
    i := 1;
    j := m;
    while i ≠ n do
        begin
            i := i + 1;
            j := j * m;
        end
    Output j;
end
```

Опишите на словах выходные данные этого алгоритма при произвольных целых m и $n > 0$. Что получится при $n = 0$?

1.4. Найдите целые числа, получающиеся в результате работы следующего алгоритма:

```
begin
    first := 1;
    Output first;
    second := 1;
    Output second;
    next := first + second;
    while next < 100 do
        begin
            Output next;
            first := second;
            second := next;
            next := first + second;
        end
end
```

Опишите полученную последовательность чисел в терминах ее членов.

1.5. Проследите эволюцию значений переменных l, sum и k в алгоритме, приведенном на следующей странице при $n = 6$.

```
begin
    Input n;
    k := 1;
    l := 0;
    sum := 0;
    while k < 2n do
        begin
            l := l + k;
            sum := sum + l;
            k := k + 2;
        end
    Output sum;
end
```

Опишите результат работы алгоритма при вводе произвольного натурального значения n.

1.6. Проследите работу алгоритма на примере сети дорог, из упр. 1.1. Какой получился результат?

```
begin
    Упорядочите ребра графа по убыванию веса
    и пронумеруйте их числами: 1, 2, 3, ... и т. д.;
    m := число вершин;
    остаток := число ребер;
    текущее := 1;
    while остаток > m − 1 do
        begin
            if удаление ребра с номером «текущее»
               не нарушает связности графа then
                  begin
                      удалить ребро «текущее»;
                      остаток := остаток − 1;
                  end;
            текущее := текущее + 1;
        end
end
```

Краткое содержание главы

Дискретная математика представляет собой математический аппарат и технику, необходимую для проектирования и понимания вычислительных систем.

Математическое моделирование — это процесс, привлекающий математику для решения реальных практических задач.

Граф (модель) данной сети дорог между городами состоит из набора **вершин**, изображающих города, соединенных друг с другом (взвешенными) **ребрами**, обозначающими дороги.

Алгоритм — это последовательность однозначных команд, выполнение которых влечет решение поставленной задачи за конечное время.

Алгоритм Прима может быть использован для выделения сети ребер минимального общего веса, соединяющей все вершины данного взвешенного графа.

Псевдокодом называется набор структурных элементов языка, подходящий для выражения алгоритма в однозначных терминах.

Оператор присваивания присваивает переменным определенные значения.

Управляющий оператор определяет порядок, в котором должны выполняться шаги алгоритма.

Составной оператор представляет собой список инструкций (операторов), которые должны выполняться как отдельная команда в том порядке, в котором они записаны.

Условный оператор дает возможность сделать выбор между альтернативными возможностями.

Оператор цикла или просто **цикл** позволяет выполнить определенный набор команд подходящее число раз.

ГЛАВА 2

ЛОГИКА И ДОКАЗАТЕЛЬСТВО

Логика необходима в любой формальной дисциплине и состоит из правил получения обоснованного вывода (заключения). Логику можно выделить из контекста тех дисциплин, в которых она используется, и изучать как отдельный раздел науки. Акцент в этой главе будет сделан именно на логике, лежащей в основе неоспоримых рассуждений и доказательств.

Мы познакомимся с логикой высказываний, имеющей дело с истинностью (или ложностью) простых описательных утверждений, что можно рассматривать как короткое введение в логику предикатов. Скажем сразу, что предикатами принято называть утверждения, содержащие переменные величины[1]. Кроме того, в этой главе описаны различные методы доказательств (прямое рассуждение, метод «от противного» и обратное рассуждение), снабженные простыми примерами проверки фактов о четных и нечетных числах, иллюстрирующими методологию рассуждений. Наконец, мы рассмотрим сильный метод доказательства, называемый методом математической индукции.

После упражнений, размещенных в конце главы, мы встретимся с первыми приложениями изучаемых методов к информатике. В них мы увидим, как логические методы доказательств используются при проверке корректности алгоритмов.

2.1. Высказывания и логика

Стандартными блоками формальной логики являются высказывания. *Высказыванием* называется утверждение, которое имеет значение истинности, т. е. может быть **истинным** (обозначается буквой И) или **ложным** (обозначается Л). Например,

- земля плоская;
- Сара — доктор;
- 29 — простое число.

[1]Здесь нужно сказать, что логика предикатов обобщает логику высказываний, и мы ею тоже займемся.

Каждое из высказываний можно обозначить своей буквой. Пусть, например, P обозначает высказывание «земля плоская», Q — «Сара — доктор» и R — «29 — простое число».

Используя такие логические операции, как **не**, **или**, **и**, можно построить новые, так называемые *составные высказывания*, компануя более простые. Например,

- (**не** P) — это высказывание «земля не плоская»;
- (P **или** Q) — «земля плоская или Сара — доктор»;
- (P **и** Q) — «земля плоская и Сара — доктор».

Пример 2.1. Обозначим через P высказывание «логика — забава», а через Q — «сегодня пятница». Требуется выразить каждое из следующих составных высказываний в символьной форме.

(а) Логика — не забава, и сегодня пятница.

(б) Сегодня не пятница, да и логика — не забава.

(в) Либо логика — забава, либо сегодня пятница.

Решение.

(а) (**не** P) **и** Q.

(б) (**не** P) **и** (**не** Q).

(в) P **или** Q.

Чтобы уметь определять значение истинности составных высказываний, нам необходимо разобраться со смыслом логических операций, т. е. какой эффект они оказывают на истинностное значение простых высказываний. Это можно аккуратно сделать с помощью так называемых *таблиц истинности*.

Отрицанием произвольного высказывания P называется высказывание вида (**не** P), чье истинностное значение строго противоположно значению P. Определяющая таблица истинности отрицания высказывания приведена в табл. 2.1.

Таблица 2.1

P	(**не** P)
И	Л
Л	И

Конъюнкцией или *логическим умножением* двух высказываний P и Q называют составное высказывание вида (P **и** Q). Оно принимает истинное значение только в том случае, когда истинны обе его составные части. Такое определение хорошо согласуется с обычным пониманием союза «и» в разговорном языке. Соответствующая таблица истинности — табл. 2.2.

Таблица 2.2

P	Q	(P **и** Q)
И	И	И
И	Л	Л
Л	И	Л
Л	Л	Л

Дизъюнкцией или *логическим сложением* двух высказываний P и Q называется составное высказывание (P **или** Q). Оно истинно, если хотя бы одна из ее составных частей имеет истинное значение, что в некотором смысле также согласуется с обыденным пониманием союза «или». Другими словами, (P **или** Q) означает, что «или P, или Q, или и то, и другое». Таблица истинности дизъюнкции обозначена как табл. 2.3.

Таблица 2.3

P	Q	(P **или** Q)
И	И	И
И	Л	И
Л	И	И
Л	Л	Л

Пример 2.2. Что можно сказать об истинности составного высказывания: «либо луна делается из зеленого сыра и Генрих VIII имел шесть жен, или не верно, что дронт[1] вымер»?

Решение. Обозначим через P высказывание «луна делается из зеленого сыра», через Q — «Генрих VIII имел шесть жен» и через R — «дронт вымер». Символьная запись данного высказывания имеет вид: (P **и** Q) **или** (**не** R). Известно, что высказывание P ложно, а Q и R истинны. Поэтому высказывание (P **и** Q) **или** (**не** R) имеет такое истинностное значение: (Л **и** И) **или** Л, что эквивалентно Л.

[1] Дронт — вымершая птица отряда голубеобразных, обитавшая на островах Индийского океана и истребленная в XVII–XVIII в.в. завезенными туда свиньями. — *Прим. перев.*

Два составных высказывания, построенные из одних и тех же простых утверждений, но разными путями, могут принимать одинаковые значения истинности на любом возможном наборе значений истинности своих составных частей. Такие высказывания называются *логически эквивалентными*.

Пример 2.3. Показать, что высказывание (**не** (P **и** (**не** Q))) логически эквивалентно утверждению ((**не** P) **или** Q).

Решение. Заполним совместную таблицу истинности (табл. 2.4) для составных высказываний:

$$R = (\textbf{не}\ (P\ \textbf{и}\ (\textbf{не}\ Q))) \qquad \text{и} \qquad S = ((\textbf{не}\ P)\ \textbf{или}\ Q).$$

Вспомогательные колонки используются для построения обоих выражений из P и Q.

Таблица 2.4

P	Q	**не** P	**не** Q	P **и** (**не** Q)	R	S
И	И	Л	Л	Л	И	И
И	Л	Л	И	И	Л	Л
Л	И	И	Л	Л	И	И
Л	Л	И	И	Л	И	И

Две последние колонки таблицы идентичны. Это означает, что высказывание R логически эквивалентно высказыванию S.

Важно изучить еще один тип логического оператора, результатом которого является *условное высказывание*. Примером такого высказывания является следующее: «если завтра будет суббота, то сегодня — пятница». При определении истинностного значения условного высказывания, необходимо различать фактическую истину и логическую.

Рассмотрим высказывание «если P, то Q». В том случае, когда предпосылка P истинна, мы не можем получить лигически корректного заключения, если Q ложно. Однако если посылка P ложна, мы имеем логически корректное высказывание и когда Q ложно, и когда оно истинно.

Пример 2.4. Пусть P — (ложное) высказывание $1 = 5$, Q — (тоже ложное) высказывание $3 = 7$ и R — (истинное) утверждение $4 = 4$. Показать, что условные высказывания: «если P, то Q» и «если P, то R», — оба истинны.

Решение. Если $1 = 5$, то, прибавляя 2 к обеим частям равенства, мы получим, что $3 = 7$. Следовательно, высказывание «если P, то Q»

справедливо. Вычтем теперь из обеих частей равенства $1 = 5$ число 3 и придем к $-2 = 2$. Поэтому $(-2)^2 = 2^2$, т. е. $4 = 4$. Таким образом, «если P, то R» тоже верно.

В логике условное высказывание «если P, то Q» принято считать ложным только в том случае, когда *предпосылка* P истинна, а *заключение* Q ложно. В любом другом случае оно считается истинным.

Используя символ импликации «$\Rightarrow$», мы пишем $P \Rightarrow Q$ для обозначения условного высказывания «если P, то Q». Такая запись читается как «из P следует Q» или, «P влечет Q», или «P достаточно для Q», или «Q необходимо для P».

Таблица истинности импликации приведена в табл. 2.5.

Таблица 2.5

P	Q	$(P \Rightarrow Q)$
И	И	И
И	Л	Л
Л	И	И
Л	Л	И

Пример 2.5. Высказывание $((\textbf{не}\ Q) \Rightarrow (\textbf{не}\ P))$ называется *противоположным* или *контрапозитивным* к высказыванию $(P \Rightarrow Q)$. Показать, что $((\textbf{не}\ Q) \Rightarrow (\textbf{не}\ P))$ логически эквивалентно высказыванию $(P \Rightarrow Q)$.

Решение. Рассмотрим совместную таблицу истинности (табл. 2.6).

Таблица 2.6

P	Q	**не** P	**не** Q	$(P \Rightarrow Q)$	$((\textbf{не}\ Q) \Rightarrow (\textbf{не}\ P))$
И	И	Л	Л	И	И
И	Л	Л	И	Л	Л
Л	И	И	Л	И	И
Л	Л	И	И	И	И

Поскольку два последних столбца этой таблицы совпадают, то и высказывания, о которых идет речь, логически эквивалентны.

2.2. Предикаты и кванторы

Логика высказываний применяется к простым декларативным высказываниям, где базисные высказывания — либо истинны, либо ложны. Утверждения, содержащие одну и более переменных, могут

быть верными при некоторых значениях переменных и ложными при других.

Предикатом называется утверждение, содержащее переменные, принимающее значение истины или лжи в зависимости от значений переменных. Например, выражение «x — целое число, удовлетворяющее соотношению $x = x^2$» является предикатом, поскольку оно истинно при $x = 0$ или $x = 1$ и ложно в любом другом случае.

Логические операции можно применять и к предикатам. В общем случае истинность составного предиката в конечном счете зависит от значений входящих в него переменных. Однако существуют некоторые, еще незнакомые Вам логические операторы (называемые *кванторами*), применение которых к предикатам превращает последние в ложные или истинные высказывания.

Пример 2.6. Какие из следующих высказываний истинны, а какие ложны?

(а) Сумма внутренних углов любого треугольника равна 180°.

(б) У всех кошек есть хвост.

(в) Найдется целое число x, удовлетворяющее соотношению $x^2 = 2$.

(г) Существует простое четное число.

Решение.

(а) Истинно.

(б) Ложно. У бесхвостой[1] кошки хвоста нет.

(в) Ложно.

(г) Истинно. Число 2 является и простым, и четным.

В примере 2.6 мы имеем дело с набором объектов и утверждениями о том, что некоторое свойство имеет место *для всех* рассматриваемых объектов, или что *найдется* (*существует*) по крайней мере один объект, обладающий данным свойством.

Выражения «для всех» и «найдется» («существует») называются кванторами и обозначаются, соответственно, $\forall$ и $\exists$. Включая в предикат кванторы, мы превращаем его в высказывание. Поэтому предикат с кванторами может быть истинным или ложным.

[1] Бесхвостая кошка — разновидность домашней кошки. — *Прим. перев.*

Пример 2.7. Обозначим через $P(x)$ предикат «x — целое число и $x^2 = 16$». Выразите словами высказывание: $\exists\, x :\ P(x)$ и определите его истинностное значение.

Решение. Высказывание $\exists\, x :\ P(x)$ означает, что найдется целое число x, удовлетворяющее уравнению $x^2 = 16$. Высказывание, конечно, истинно, поскольку уравнение $x^2 = 16$ превращается в верное тождество при $x = 4$. Кроме того, $x = -4$ — также решение данного уравнения. Однако нам не требуется рассуждать о знаке переменной x, чтобы проверить истинность высказывания $\exists\, x :\ P(x)$.

Пример 2.8. Пусть $P(x)$ — предикат: «x — вещественное число и $x^2+1 = 0$». Выразите словами высказывание: $\exists\, x :\ P(x)$ и определите его истинностное значение.

Решение. Данное высказывание можно прочитать так: существует вещественное число x, удовлетворяющее уравнению $x^2 + 1 = 0$. Поскольку квадрат любого вещественного числа неотрицателен, т. е. $x^2 \geqslant 0$, мы получаем, что $x^2 + 1 \geqslant 1$. Следовательно, утверждение $\exists\, x :\ P(x)$ ложно.

Отрицание высказывания из примера 2.8 записывается в следующем виде: **не** $\exists\, x :\ P(x)$. Это, естественно, истинное высказывание, которое означает, что не существует вещественного числа x, удовлетворяющего условию $x^2 + 1 = 0$. Иными словами, каково бы ни было вещественное x, $x^2 + 1 \neq 0$. В символьной форме это можно записать как $\forall\, x$ **не** $P(x)$.

Для общего предиката $P(x)$ есть следующие логические эквивалентности[2]:

$$\textbf{не}\ \exists\, x :\ P(x) \Leftrightarrow \forall\, x\ \textbf{не}\ P(x);$$

$$\textbf{не}\ \forall\, x\ P(x) \Leftrightarrow \exists\, x :\ P(x).$$

Как показывает следующий пример, некоторые трудности возникают, когда в высказывании участвует более одного квантора.

Пример 2.9. Предположим, что x и y — вещественные числа, а $P(x,\, y)$ обозначает предикат $x + y = 0$. Выразите каждое из высказываний словами и определите их истинность.

(а) $\forall\, x\ \exists\, y :\ P(x,\, y)$;

(б) $\exists\, y :\ \forall\, x\ P(x,\, y)$.

[2] В символьной форме логически эквивалентные высказывания обозначаются значком «$\Leftrightarrow$». — *Прим. перев.*

Решение.

(а) Высказывание $\forall x\, \exists y:\ P(x, y)$ говорит о том, что для любого вещественного числа x найдется такое вещественное число y, что $x+y=0$. Оно, очевидно, верно, поскольку какое бы число x мы ни взяли, число $y = -x$ обращает равенство $x + y = 0$ в верное тождество.

(б) Высказывание $\exists y:\ \forall x\ P(x, y)$ читается следующим образом: существует такое вещественное число y, что для любого вещественного числа x выполнено равенство $x + y = 0$. Это, конечно, не так: не существует вещественного числа y, обладающего указанным свойством. Следовательно, высказывание ложно.

2.3. Методы доказательств

При доказательстве теорем применяется логическая аргументация. Доказательства в информатике — неотъемлемая часть проверки корректности алгоритмов. Необходимость доказательства возникает, когда нам нужно установить истинность высказывания вида $(P \Rightarrow Q)$. Существует несколько стандартных типов доказательств, включающих следующие:

1. Прямое рассуждение. Предполагаем, что высказывание P истинно и показываем справедливость Q. Такой способ доказательства исключает ситуацию, когда P истинно, а Q — ложно, поскольку именно в этом и только в этом случае импликация $(P \Rightarrow Q)$ принимает ложное значение (см. табл. 2.5 на стр. 27).

2. Обратное рассуждение. Предполагаем, что высказывание Q ложно и показываем ошибочность P. То есть, фактически, прямым способом проверяем истинность импликации $((\textbf{не}\ Q) \Rightarrow (\textbf{не}\ P))$, что согласно примеру 2.5, логически эквивалентно истинности исходного утверждения $(P \Rightarrow Q)$.

3. Метод «от противного». Предположив, что высказывание P истинно, а Q ложно, используя аргументированное рассуждение, получаем противоречие. Этот способ опять-таки основан на том, что импликация $(P \Rightarrow Q)$ принимает ложное значение только тогда, когда P истинно, а Q ложно.

Пример 2.10. Покажите прямым способом рассуждений, что произведение xy двух нечетных целых чисел x и y всегда нечетно.

Решение. Прежде всего заметим, что любое нечетное число, и в частности x, можно записать в виде $x = 2m + 1$, где m — целое число. Аналогично, $y = 2n + 1$ с некоторым целым n.

Значит, произведение

$$xy = (2m + 1)(2n + 1) = 4mn + 2m + 2n + 1 = 2(2mn + m + n) + 1$$

тоже является нечетным числом.

Пример 2.11. Пусть n — натуральное число. Покажите, используя обратный способ доказательства, что если n^2 нечетно, то и n нечетно.

Решение. Отрицанием высказывания о нечетности числа n^2 служит утверждение «n^2 четно», а высказывание о четности n является отрицанием утверждения «число n нечетно». Таким образом, нам нужно показать прямым способом рассуждений, что четность числа n влечет четность его квадрата n^2.

Так как n четно, то $n = 2m$ для какого-то целого числа m. Следовательно, $n^2 = 4m^2 = 2(2m^2)$ — четное число.

Пример 2.12. Методом «от противного» покажите, что решение уравнения $x^2 = 2$ является иррациональным числом, т. е. не может быть записано в виде дроби с целыми числителем и знаменателем.

Решение. Здесь нам следует допустить, что решение x уравнения $x^2 = 2$ рационально, т. е. записывается в виде дроби $x = \frac{m}{n}$ с целыми m и n, причем $n \neq 0$. Предположив это, нам необходимо получить противоречие либо с предположением, либо с каким-то ранее доказанным фактом.

Как известно, рациональное число неоднозначно записывается в виде дроби. Например, $x = \frac{m}{n} = \frac{2m}{2n} = \frac{3m}{3n}$ и т. д. Однако можно считать, что m и n не имеют общих делителей. В этом случае неоднозначность записи пропадает.

Итак, предполагаем дополнительно, что дробь $x = \frac{m}{n}$ несократима (m и n не имеют общих делителей). По условию число x удовлетворяет уравнению $x^2 = 2$. Значит, $\left(\frac{m}{n}\right)^2 = 2$, откуда $m^2 = 2n^2$.

Из последнего равенства следует, что число m^2 четно. Следовательно, m тоже четно (см. упр. а(б)) и может быть представлено в виде $m = 2p$ для какого-то целого числа p. Подставив эту информацию в равенство $m^2 = 2n^2$, мы получим, что $4p^2 = 2n^2$, т. е. $n^2 = 2p^2$. Но тогда n тоже является четным числом. Таким образом, мы показали, что как m, так и n — четные числа. Поэтому они обладают

общим делителем 2. Если же теперь вспомнить, что мы предполагали отсутствие общего делителя у числителя и знаменателя дроби $\frac{m}{n}$, то увидим явное противоречие.

Найденное противоречие приводит нас к однозначному выводу: решение уравнения $x^2 = 2$ не может быть рациональным числом, т. е. оно иррационально.

2.4. Математическая индукция

Компьютерную программу в информатике называют *правильной* или *корректной*, если она делает то, что указано в ее спецификации. Несмотря на то, что тестирование программы может давать ожидаемый результат в случае каких-то отдельных начальных данных, необходимо доказать приемами формальной логики, что правильные выходные данные будут получаться при любых вводимых начальных значениях. О доказательствах такого сорта будет рассказано в приложении, размещенном в конце этой главы.

Проверка корректности алгоритма, содержащего циклы, нуждается в довольно мощном методе доказательства, который называется «*математическая индукция*». Продемонстрируем преимущества этого важного метода, доказав корректность следующего рекуррентного алгоритма, определяющего максимальный элемент из набора a_1, a_2, a_3, ..., a_n натуральных чисел.

```
begin
    i := 0;
    M := 0;
        while i < n do
            begin
                j := j + 1;
                M := max(M, a);
            end
end
```

Действие алгоритма на наборе данных: $a_1 = 4$, $a_2 = 7$, $a_3 = 3$ и $a_4 = 8$ прослежено в табл. 2.7.

Таблица 2.7

j	M	$j < 4$?
0	0	Да
1	4	Да
2	7	Да
3	7	Да
4	8	Нет

В качестве выходных данных мы получили $M = 8$, что безусловно правильно. Заметим, что после каждого прохода цикла переменная M равна наибольшему из чисел набора, просмотренных к этому моменту.

Но будет ли алгоритм работать правильно при любом вводимом наборе чисел длины n?

Рассмотрим вводимый набор $a_1, a_2, a_3, \ldots, a_n$ длины n и обозначим через M_k значение переменной M после k-го прохода цикла.

1. Если мы вводим набор a_1 длины 1, то цикл сделает только один проход и M присвоится наибольшее значение из 0 и a_1, которым, очевидно, будет a_1 (натуральные числа больше 0). В этом простом случае вывод будет правильным.

2. Если после k-го прохода цикла M_k — наибольший элемент из набора $a_1, a_2, \ldots, a_k$, то после следующего прохода M_{k+1} будет равно $\max(M_k, a_{k+1})$, т. е. максимальному элементу набора $a_1, a_2, \ldots, a_k, a_{k+1}$.

В п. 1 мы показали, что алгоритм работает правильно при любом вводимом наборе длины 1. Поэтому согласно п. 2, он будет правильно работать и на любом наборе длины 2. Вновь применяя п. 2 рассуждений, мы убеждаемся, что алгоритм работает правильно и на любых наборах длины 3, и т. д. Таким образом, алгоритм правильно работает на любых наборах произвольной длины n, т. е. он корректен.

На формальном языке использованный метод доказательства выглядит следующим образом.

Принцип математической индукции

Пусть $P(n)$ — предикат, определенный для всех натуральных чисел n.
Предположим, что

1. *$P(1)$ истинно и*
2. *$\forall k \geqslant 1$ импликация $(P(k) \Rightarrow P(k+1))$ верна.*

Тогда $P(n)$ истинно при любом натуральном значении n.

Пример 2.13. Докажите по индукции, что равенство

$$1 + 2 + \cdots + n = \frac{n(n+1)}{2}$$

выполнено при всех натуральных n.

Решение. Пусть $P(n)$ — предикат $1+2+\cdots+n=\frac{n(n+1)}{2}$.

В случае $n=1$ левая часть равенства — просто 1, а вычисляя правую часть, получаем

$$\frac{1(1+1)}{2}=1.$$

Следовательно, $P(1)$ истинно.

Предположим теперь, что равенство $1+2+\cdots+k=\frac{k(k+1)}{2}$ имеет место для какого-то натурального числа k. Тогда

$$\begin{aligned}1+2+\cdots+k+(k+1) &= (1+2+\cdots+k)+(k+1)= \\ &= \frac{k(k+1)}{2}+(k+1)= \\ &= \frac{1}{2}\big(k(k+1)+2(k+1)\big)= \\ &= \frac{1}{2}\big((k+2)(k+1)\big)= \\ &= \frac{(k+1)(k+2)}{2}.\end{aligned}$$

Таким образом, при любом натуральном k импликация

$$P(k)\Rightarrow P(k+1)$$

справедлива. Значит, по принципу математической индукции, предикат $P(n)$ имеет истинное значение при всех натуральных n.

Пример 2.14. Методом математической индукции докажите, что 7^n-1 делится на 6 при любом натуральном показателе n.

Решение. Прежде всего напомним, что целое число a делится на целое число b тогда и только тогда, когда выполняется равенство $a=mb$ при каком-то целом числе m. Например, 51 делится на 17, поскольку $51=3\cdot 17$. Кроме того, для наших рассуждений потребуется простое свойство делимости чисел, которое утверждает, что сумма делящихся на b чисел делится на b.

Пусть $P(n)$ обозначает предикат «7^n-1 делится на 6».

При $n=1$ имеем

$$7^n-1=7-1=6,$$

т. е. предикат $P(1)$ имеет истинное значение.

Предположим, что 7^k-1 делится на 6 при каком-то натуральном k. Тогда

$$\begin{aligned} 7^{k+1} - 1 &= 7(7^k) - 1 = \\ &= 7(7^k - 1) + 7 - 1 = \\ &= 7(7^k - 1) + 6. \end{aligned}$$

Так как $7^k - 1$ делится на 6, то по упомянутому свойству делимости сумма $7(7^k - 1) + 6$ тоже делится на 6.

Итак, $7^{k+1} - 1$ делится на 6, так что при любом натуральном k импликация $(P(k) \Rightarrow P(k+1))$ истинна.

Индуктивным рассуждением мы доказали истинность предиката $P(n)$ для всех натуральных n.

Пример 2.15. Последовательность целых чисел x_1, x_2, ..., x_n определена рекуррентной формулой:

$$x_1 = 1 \qquad \text{и} \qquad x_{k+1} = x_k + 8k \quad \text{при } k \geqslant 1.$$

Доказать, что имеет место формула: $x_n = (2n-1)^2$ для всех $n \geqslant 1$.

Решение. Предикат $x_n = (2n-1)^2$ обозначим через $P(n)$. Если $n = 1$, то $(2n-1)^2 = (2-1)^2 = 1$, что показывает истинность высказывания $P(1)$.

Допустим теперь, что $x_k = (2k-1)^2$ для некоторого $k \geqslant 1$. Тогда

$$\begin{aligned} x_{k+1} &= x_k + 8k = \\ &= (2k-1)^2 + 8k = \\ &= 4k^2 + 4k + 1 = \\ &= (2k+1)^2. \end{aligned}$$

Мы видим, что $x_{k+1} = \big(2(k+1)-1\big)^2$ и поэтому истинность импликации $(P(k) \Rightarrow P(k+1))$ доказана при всех $k \geqslant 1$. Следовательно, согласно индуктивному принципу, предикат $P(n)$ превращается в истинное высказывание при любом натуральном значении переменной n.

Набор упражнений 2

2.1. Пусть P, Q и R — определенные следующим образом высказывания:

P: Я умираю от жажды.

Q: Мой стакан пуст.
R: Сейчас три часа.

Запишите каждое из следующих высказываний как логическое выражение, включающее P, Q и R.

(а) Я умираю от жажды и мой стакан не пуст.

(б) Сейчас три часа, а я умираю от жажды.

(в) Если сейчас три часа, то я умираю от жажды.

(г) Если я умираю от жажды, то мой стакан пуст.

(д) Если я не умираю от жажды, то мой стакан не пуст.

2.2. Обозначим через P высказывание: «розы красные», а через Q — «фиалки синие». Запишите каждое из следующих высказываний:

(а) если розы не красные, то фиалки не синие;

(б) розы красные или фиалки не синие;

(в) либо розы красные, либо фиалки синие (но не одновременно)

как логическое выражение.

Используя таблицы истинности, докажите логическую эквивалентность высказываний (а) и (б).

2.3. Составные высказывания, принимающие истинные значения при любых истинностных значениях своих компонент, называются *тавтологиями*. С помощью таблиц истинности найдите тавтологии среди следующих высказываний:

(а) **не** (P **и** (**не** P));

(б) $P \Rightarrow$ (**не** P);

(в) (P **и** ($P \Rightarrow Q$)) $\Rightarrow Q$.

2.4. Покажите, что высказывание $(P \Rightarrow Q) \Rightarrow R$ логически эквивалентно высказыванию ((**не** P) $\Rightarrow R$) **и** ($Q \Rightarrow R$).

2.5. Обозначим через x слово «кошка», а через $P(x)$ предикат «у x есть усы». Запишите каждое из высказываний в символьной форме:

(а) усы есть у всех кошек;

(б) найдется кошка без усов;

(в) не бывает кошек с усами.

Запишите отрицание высказывания (б) в символьной форме, а отрицание высказывания (в) запишите как символами, так и словами.

2.6. Пусть $P(x)$ означает «x высокий», а $Q(x)$ — «x толстый», где x — какой-то человек. Прочитайте высказывание:

$$\forall\, x\ (P(x)\ \textbf{и}\ Q(x)).$$

Найдите его отрицание среди следующих утверждений:

(а) найдется некто короткий и толстый;

(б) нет никого высокого и худого;

(в) найдется некто короткий или худой.

2.7. (а) Прямым рассуждением докажите истинность высказывания:

$$n \text{ и } m \text{ — четные числа} \Rightarrow n+m \text{ — число четное.}$$

(б) Дайте обратное доказательство высказывания:

$$n^2 \text{ — четное число} \Rightarrow n \text{ — четное.}$$

(в) Методом «от противного» докажите, что

$$n+m \text{ — нечетное число} \Rightarrow \text{одно из слагаемых является четным, а другое — нечетным.}$$

2.8. Докажите каждое из высказываний методом математической индукции.

(а) $1+5+9+\cdots+(4n-3)=n(2n-1)$ для всех натуральных чисел n.

(б) $1^2+2^2+\cdots+n^2=\frac{1}{6}n(n+1)(2n+1)$ для всех натуральных чисел n.

(в) $\frac{1}{1\cdot3}+\frac{1}{3\cdot5}+\cdots+\frac{1}{(2n-1)\cdot(2n+1)}=\frac{n}{2n+1}$ для всех натуральных чисел n.

(г) Число n^3-n делится на 3 при всех натуральных значениях числа n.

(д) $1\cdot1!+2\cdot2!+\cdots+n\cdot n!=(n+1)!-1$ для всех натуральных чисел n.

(Символ $n!$ читается как «n факториал» и обозначает произведение всех натуральных чисел от 1 до n включительно: $n!=1\cdot2\cdot3\cdots(n-1)\cdot n$.)

2.9. Последовательность натуральных чисел $x_1, x_2, \ldots, x_n$ определяется рекуррентной формулой

$$x_1 = 1 \qquad \text{и} \qquad x_{k+1} = \frac{x_k}{x_k + 2} \quad \text{при } k \geqslant 1.$$

Вычислите x_2, x_3 и x_4. Докажите по индукции, что

$$x_n = \frac{1}{2^n - 1}$$

для всех $n \geqslant 1$.

2.10. Последовательность натуральных чисел $x_1, x_2, \ldots, x_n$ определяется рекуррентной формулой

$$x_1 = 1,\ x_2 = 2 \qquad \text{и} \qquad x_{k+1} = 2x_k - x_{k-1} \quad \text{при } k > 1.$$

Вычислите x_3, x_4 и x_5. Найдите общую формулу для x_n и докажите ее истинность индуктивным методом.

Краткое содержание главы

Логика представляет собой набор правил для получения обоснованных выводов.

Высказыванием называется утверждение, имеющее истинностное значение, т. е. оно может быть *истинным* или *ложным*.

Составное высказывание может быть построено из других с помощью логических операций. Наиболее употребительными операциями являются **и**, **или**, **если ... то** и **не**.

В табл. 2.8 сведены **таблицы истинности** логических операций **и**, **или** и $\Rightarrow$.

Таблица 2.8

P	Q	P **и** Q	P **или** Q	$(P \Rightarrow Q)$
И	И	И	И	И
И	Л	Л	И	Л
Л	И	Л	И	И
Л	Л	Л	Л	И

Два составных высказывания называются **логически эквивалентными**, если они принимают одинаковые значения истинности на любом наборе истинностных значений своих составных частей.

Высказывание о свойствах переменной x называют **предикатом** и обозначают, например, так: $P(x)$.

Для всех ($\forall$) и **существует** ($\exists$) — это **кванторы**.

При доказательстве **прямым рассуждением** утверждения вида $(P \Rightarrow Q)$ из предположения об истинности P выводят истинность Q.

Обратное рассуждение в доказательстве основано на логической эквивалентности высказываний (**не** $Q \Rightarrow$ **не** P) и $(P \Rightarrow Q)$.

Метод доказательства импликации $(P \Rightarrow Q)$, при котором из предположения о ложности Q и истинности P приходят к противоречию, называют методом **«от противного»**.

Математическая индукция полезна при доказательстве высказывания, истинного для всех натуральных чисел.

Принцип математической индукции — это следующая теорема:

> *Пусть $P(n)$ — предикат, определенный для всех натуральных n.*
> *Предположим, что*
>
> 1. $P(1)$ *истинно и*
> 2. $\forall k \geqslant 1$ *импликация* $(P(k) \Rightarrow P(k+1))$ *верна.*
>
> *Тогда $P(n)$ истинно при любом натуральном значении n.*

Приложение. Корректность алгоритмов

Чтобы доказать корректность алгоритма (иными словами, убедиться, что он делает именно то, что и предусмотрено), нам нужно проверить все изменения переменных, в нем используемых *до*, *в течение* и *после* работы алгоритма. Эти изменения и условия можно рассматривать как небольшие утверждения или предикаты.

Пусть P — предикат, истинный для входных данных алгоритма A, и Q — предикат, описывающий условия, которым должны удовлетворять выходные данные. Высказывание $\{P\}\,A\,\{Q\}$ означает, что «если работа алгоритма A начинается с истинного значения предиката P, то она закончится при истинном значении Q». Предикат P называется *входным условием* или *предусловием*, а Q — *выходным*

условием или *постусловием*. Высказывание $\{P\}\,A\,\{Q\}$ само является предикатом. Поэтому доказательство корректности алгоритма A равносильно доказательству истинности $\{P\}\,A\,\{Q\}$. Для простых алгоритмов это делается достаточно прямолинейно.

Задача 1. Докажите корректность алгоритма ***Разность***.

Разность
begin
$z := x - y;$
end

Решение. В данном случае предусловием P являются равенства: $x = x_1$ и $y = y_1$. Постусловие Q — это $z = x_1 - y_1$. Предикат

$$\{P\}\,\textbf{\textit{Разность}}\,\{Q\}$$

читается как «если $x = x_1$ и $y = y_1$, то $z = x_1 - y_1$». Истинность последнего предиката легко проверяется подстановкой $x = x_1$ и $y = y_1$ в тело алгоритма, содержащего переменные z, x и y. С формальной точки зрения соотношения: $z = x - y$, $x = x_1$ и $y = y_1$ влекут тождество $z = x_1 - y_1$.

Когда в алгоритме A происходит много различных действий с переменными, мы разбиваем его на подходящие отрезки $A_1, \ldots, A_n$ и доказываем цепочку утверждений вида

$$\{P\}\,A_1\,\{Q_1\},\ \{Q_1\}\,A_2\,\{Q_2\},\ \ldots,\ \{Q_{n-1}\}\,A_n\,\{Q\},$$

где постусловие любого отрезка служит предусловием следующего.

Задача 2. Докажите правильность алгоритма ***«Квадратный многочлен»***.

Квадратный многочлен
$\{x$ — *вещественное число*$\}$
begin
$y := ax;$
$y := (y + b)x;$
$y := y + c;$
end
$\{y = ax^2 + bx + c\}$

Решение. Разобьем алгоритм на кусочки, зафиксировав при этом обозначения пред- и постусловий.