Глава 5

Операторы и функции системы MATLAB

В этой главе…

- Арифметические и логические операторы
- Элементарные функции
- Функции для работы со значениями даты и времени
- Функции для выполнения побитовых операций
- Специальные математические функции
- Специальные символы
- Резюме
- Тесты
- Упражнения

Арифметические и логические операторы

Операторы — это неотъемлемая часть математических выражений, вычисление которых является одной из основных задач MATLAB как системы, созданной для выполнения численных расчетов.

В главе 2 вы познакомились с простейшими арифметическими операторами системы MATLAB, такими как оператор сложения (+), вычитания (–), умножения (*), деления (/) и возведения в степень (^). Из главы 4 узнали о применении в MATLAB операторов поэлементного умножения, деления и возведения в степень (.*, ./, .\ и .^). Таким образом, в MATLAB используются арифметические операторы двух типов — операторы, которые позволяют выполнять действия, соответствующие правилам матричного исчисления в математике, и операторы, служащие для выполнения поэлементных операций над массивами. Операторы для выполнения поэлементных действий предваряются точкой.

Поскольку операции поэлементного сложения и вычитания матриц не противоречат правилам матричного исчисления, для их выполнения используются операторы “+” и “–”. Операторов “.+” и “.–” в MATLAB не существует.

Напомним, что среди перечисленных операторов наибольший приоритет имеют операторы возведения в степень, а наименьший — операторы сложения и вычитания. Изменить приоритет операций можно, используя в выражении круглые скобки.

Помимо арифметических операторов, в MATLAB существуют операторы отношения и логические операторы. Приоритет этих операторов (кроме оператора логического отрицания) ниже, чем приоритет арифметических операторов.

Данная глава поможет вам расширить и обобщить полученные знания об операторах и функциях системы MATLAB.

Арифметические операторы и соответствующие им функции

Каждому из арифметических операторов в MATLAB соответствует определенная функция. В табл. 5.1 перечислены арифметические операторы системы MATLAB и соответствующие им функции.

Таблица 5.1. Арифметические операторы и функции системы MATLAB

Оператор	*Описание*	*Функция*
+	Сложение	plus
+	Унарный плюс	uplus
–	Вычитание	minus
–	Унарный минус	uminus
*	Матричное умножение	mtimes
.*	Поэлементное умножение массивов	times
^	Возведение матрицы в степень	mpower
.^	Поэлементное возведение массива в степень	power
/	Деление матриц слева направо	mrdivide
\	Деление матриц справа налево (обратное деление матриц)	mldivide
./	Поэлементное деление массивов слева направо	rdivide
.\	Поэлементное деление массивов справа налево	ldivide

Ниже мы рассмотрим несколько примеров использования арифметических функций (с примерами использования арифметических операторов вы могли ознакомиться в предыдущих главах). Допустим, имеются два массива `X` и `Y` одинаковых размеров.

```
>> X=[0 1 2; 5 3 6]
X =
     0     1     2
     5     3     6
>> Y=[9 8 4; 3 1 1]
Y =
     9     8     4
     3     1     1
```

С помощью функции `plus` можно выполнить сложение массивов.

```
>> plus(X,Y)
ans =
     9     9     6
     8     4     7
```

Функция `times` позволяет умножить элементы одного массива на соответствующие элементы другого.

```
>> times(X,Y)
ans =
     0     8     8
    15     3     6
```

Если нужно выполнить матричное умножение массивов X и Y, воспользуйтесь функцией mtimes. Однако чтобы операция умножения матриц имела смысл, не забудьте транспонировать одну из матриц.

```
>> mtimes(X,Y')
ans =
    16     3
    93    24
```

Посредством функции power можно возвести элементы одного массива в степени, равные соответствующим элементам другого массива.

```
>> power(X,Y)
ans =
     0     1    16
   125     3     6
```

А для того чтобы сделать все элементы массива отрицательными, воспользуйтесь функцией uminus.

```
>> uminus(Y)
ans =
    -9    -8    -4
    -3    -1    -1
```

С помощью функции rdivide выполним деление элементов массива X на соответствующие элементы массива Y.

```
>> rdivide(X,Y)
ans =
         0    0.1250    0.5000
    1.6667    3.0000    6.0000
```

Операторы / и \ (и соответствующие им функции mrdivide и mldivide) в MATLAB предназначены для решения систем линейных уравнений. Эта тема будет рассмотрена в главе 6, "Решение типичных математических задач".

Как уже упоминалось в главе 1, в новой версии MATLAB повышена надежность вычислений с числами двойной точности (double) и расширена поддержка других типов данных: целочисленных и одинарной точности (single), а также вычислений с ними. Обработку новых типов данных можно выполнять без перевода их в числа двойной точности, что значительно повышает производительность и уменьшает объем используемой памяти. В частности, над данными с одинарной точностью и целочисленными данными теперь допустимы почти все перечисленные выше арифметические операции. Подробнее о поддерживаемых в MATLAB типах данных читайте в главе 9, "Типы данных".

Операторы отношения

Операторы отношения используются для поэлементного сравнения двух операндов, в качестве которых могут выступать числа, векторы или матрицы. При этом сравниваемые векторы или матрицы должны иметь одинаковые размеры. Перечень операторов отношения с соответствующими им функциями представлен в табл. 5.2.

Таблица 5.2. Операторы отношения и их функции

Оператор	*Название*	*Функция*
==	Равно	eq
~=	Не равно	ne
<	Меньше	lt

Окончание табл. 5.2

Оператор	*Название*	*Функция*
>	Больше	gt
<=	Меньше или равно	le
>=	Больше или равно	ge

Результатом операции отношения может быть “истина” или “ложь”. В MATLAB “истина” обозначается логической единицей (1), “ложь” — логическим нулем (0). В выражениях, вводимых в командном окне системы MATLAB, операторы отношения могут использоваться наряду с арифметическими операторами. Рассмотрим использование операторов отношения на следующем примере.

```
>> x=1; y=2; z=3;
>> ((x+y)==z)+(y<z)+(x<=y)
ans =
     3
```

Используя функции отношения, это выражение можно переписать в следующем виде: `eq((x+y),z)+lt(y,z)+le(x,y)`.

Приведенное выражение состоит из трех частей: `(x+y)==z`, `y<z` и `x<=y`. При x=1, y=2 и z=3 эти выражения оказываются истинными, т.е. результатом выполнения каждого из них является 1. В итоге переменной `ans` присваивается значение, равное сумме результатов трех выражений, т.е. 3 (1+1+1).

Как уже упоминалось, приоритет операторов отношения ниже, чем приоритет арифметических операторов. Но в нашем примере за счет использования круглых скобок мы добились того, чтобы в первую очередь выполнялись операции отношения и лишь затем операция суммирования. Если же круглые скобки опустить, результат будет совершенно иным.

```
>> x+y==z+y<z+x<=y
ans =
     1
```

Например, при поэлементном сравнении двух массивов одинаковых размеров с помощью операторов отношения результат будет представлен в виде массива того же размера, состоящего из нулей и единиц.

```
>> A=[1 0 3; -2 5 -6];
>> B=[8 -9 1; 7 2 2];
>> A>B
ans =
     0     1     1
     0     1     0
```

Если же один из операндов является массивом, а другой скаляром, MATLAB “расширяет” скаляр до размеров данного массива и производит их поэлементное сравнение. Иными словами, команды

```
>> X=3; Y=[2 4 7; 3 8 1]; X<Y
```

и

```
>> X=3*ones(2,3); Y=[2 4 7; 3 8 1]; X<Y
```

приведут к одинаковому результату:

```
ans =
     0     1     1
     0     1     0
```

Если вы попытаетесь сравнить массивы разных размеров, то получите сообщение об ошибке.

Применение операторов отношения для сравнения комплексных операндов имеет следующие особенности. При использовании операторов <, >, <= и >= сравниваются только действительные части комплексных чисел, а при использовании операторов == и ~= учитываются как действительные, так и мнимые части комплексных операндов.

Логические операторы

Логические операторы предназначены для выполнения поэлементных логических операций над массивами одинаковых размеров. Логические операторы и соответствующие им функции MATLAB приведены в табл. 5.3.

Таблица 5.3. Логические операторы и функции

Оператор	*Название*	*Функция*
&	Логическое И	and
|	Логическое ИЛИ	or
~	Логическое НЕ	not
	Исключающее ИЛИ	xor

Операции И и ИЛИ являются бинарными (т.е. выполняются над двумя операндами), а операция НЕ — унарной (однооперандной).

Выполнение логических операций над массивами одинаковых размеров состоит в поэлементном применении логических операторов к элементам массивов, в результате чего получается массив такого же размера, состоящий из нулей и единиц.

При выполнении логических операций “истинными” считаются операнды, не равные нулю, а “ложными” — операнды, равные нулю. При этом результатом операции И будет 1, если оба операнда не равны нулю, и 0, если хотя бы один из операндов нулевой. Операция ИЛИ дает 1, если хотя бы один операнд не равен нулю. А операция “исключающее ИЛИ” выдает 1 лишь тогда, когда один из операндов равен нулю, а другой не равен, в остальных случаях ее результатом будет 0. И наконец, в результате операции НЕ получится 1, если ее единственный операнд равен нулю, и 0 в противном случае.

Более наглядное представление о работе логических операторов дает табл. 5.4, где в качестве операндов используются значения 0 и 1.

Таблица 5.4. Работа логических операторов

x	*y*	*x&y* *and(x,y)*	*x|y* *or(x,y)*	*~x* *not(x)*	*xor(x,y)*
0	0	0	0	1	0
0	1	0	1	1	1
1	0	0	1	0	1
1	1	1	1	0	0

Ниже приведены примеры использования логических операций.

```
>> A=[2 5; 0 3];
>> B=[7 1; -4 0];
>> A&B
ans =
     1     1
```

```
     0     0
>> A|B
ans =
     1     1
     1     1
>> xor(A,B)
ans =
     0     0
     1     1
>> ~A
ans =
     0     0
     1     0
```

Кроме перечисленных в табл. 5.3, программа MATLAB включает следующие логические функции:

- `any` — возвращает 1, если в заданном векторе есть хотя бы один ненулевой элемент; если аргументом является матрица, результат выдается в виде вектора, элементы которого соответствуют столбцам заданной матрицы;
- `all` — возвращает 1, если среди элементов исходного вектора нет нулевых; если аргументом является матрица, результат выдается в виде вектора, элементы которого соответствуют столбцам заданной матрицы.

Приведем пример.

```
>> C=[1 0 3; 7 0 0; 2 0 10]
C =
     1     0     3
     7     0     0
     2     0    10
>> any(C)
ans =
     1     0     1
>> all(C)
ans =
     1     0     0
```

Приоритет операций

Если выражение включает как арифметические, так и логические операции, порядок выполнения этих операций зависит от их приоритета. Приоритет операций можно изменить с помощью круглых скобок.

Приоритеты операций системы MATLAB в порядке убывания представлены в следующем списке.

1. Круглые скобки `()`.
2. Транспонирование (`.'`), транспонирование с комплексным сопряжением (`'`), возведение в степень (`^`), поэлементное возведение в степень (`.^`).
3. Унарный плюс (`+`), унарный минус (`–`), логическое отрицание (`~`).
4. Умножение и деление (`.*`, `./`, `.\`, `*`, `/`, `\`).
5. Сложение и вычитание (`+` и `–`).
6. Операции отношения (`<`, `<=`, `>`, `>=`, `==`, `~=`).
7. Логическое И (`&`).
8. Логическое ИЛИ (`|`).

Элементарные функции

В главе 2 вы уже познакомились с некоторыми элементарными функциями, доступными в MATLAB. Это тригонометрические и гиперболические, степенные, логарифмические и экспоненциальные функции, функции округления и вычисления остатка от деления, а также функции для работы с комплексными аргументами.

Аргументами элементарных функций могут быть действительные либо комплексные числа, а также массивы. Если в качестве аргумента функции задан массив, в результате получится массив того же размера и типа, элементы которого будут равны функциям от соответствующих элементов исходного массива.

Далее мы более детально опишем элементарные функции системы MATLAB и изучим их работу на примерах.

Тригонометрические и гиперболические функции

Рассмотрим встроенные тригонометрические и гиперболические функции системы MATLAB, а также обратные к ним функции (табл. 5.5). Аргументы одних тригонометрических функций задаются в радианах, а других — в градусах. И соответственно, одни обратные тригонометрические функции возвращают результат в радианах, а другие — в градусах. Различить их можно следующим образом. Имена функций, которые работают со значениями, заданными в градусах, имеют окончание `d` (от английского слова *degree* — градус), а у тех функций, которые работают со значениями в радианах, такого окончания нет.

Таблица 5.5. Тригонометрические и гиперболические функции MATLAB

Функция	*Описание*
sin(x), sind(x)	Синус
cos(x), cosd(x)	Косинус
tan(x), tand(x)	Тангенс
cot(x), cotd(x)	Котангенс
sec(x), secd(x)	Секанс
csc(x), cscd(x)	Косеканс
asin(x), asind(x)	Арксинус (для действительных чисел из диапазона [–1,1] возвращает действительные числа в диапазоне от $-\pi/2$ до $\pi/2$, а для действительных чисел вне этого диапазона — комплексные значения)
acos(x), acosd(x)	Арккосинус (для действительных чисел из диапазона [–1,1] возвращает действительные числа в диапазоне от 0 до π), а для действительных чисел вне этого диапазона — комплексные значения)
atan(x), atand(x)	Арктангенс (для действительных чисел возвращает значения в диапазоне от $-\pi/2$ до $\pi/2$)
acot(x), acotd(x)	Арккотангенс
asec(x), asecd(x)	Арксеканс
acsc(x), acscd(x)	Арккосеканс
sinh(x)	Гиперболический синус числа x
cosh(x)	Гиперболический косинус
tanh(x)	Гиперболический тангенс
coth(x)	Гиперболический котангенс
sech(x)	Гиперболический секанс
csch(x)	Гиперболический косеканс

Окончание табл. 5.5

Функция	*Описание*
asinh(x)	Гиперболический арксинус
acosh(x)	Гиперболический арккосинус
atanh(x)	Гиперболический арктангенс
acoth(x)	Гиперболический арккотангенс
asech(x)	Гиперболический арксеканс
acsch(x)	Гиперболический арккосеканс

Ниже приведено несколько примеров.

```
>> z=[0.5 1 2.5]
z =
    0.5000    1.0000    2.5000
>> cos(z)
ans =
    0.8776    0.5403   -0.8011
>> sec(z)
ans =
    1.1395    1.8508   -1.2482
>> atan(z)
ans =
    0.4636    0.7854    1.1903
>> sinh(z)
ans =
    0.5211    1.1752    6.0502
>> asech(z)
ans =
    1.3170    0    0 + 1.1593i
```

Экспоненциальные функции

Экспоненциальные функции MATLAB представлены следующими функциями (табл. 5.6).

Таблица 5.6. Экспоненциальные функции

Функция	*Описание*
exp(x)	Экспонента числа x
log(x)	Натуральный логарифм
log10(x)	Десятичный логарифм
log2(x)	Логарифм по основанию 2
pow2(x)	Возведение числа 2 в степень x
sqrt(x)	Квадратный корень
nextpow2(x)	Степень, в которую нужно возвести число 2, чтобы получить ближайшее число, большее или равное x; если x — вектор, эта функция возвращает значение, равное значению функции nextpow2(length(x))
expm1(x)	Вычисление точного значения выражения exp(x)-1 для маленьких значений x
log1p(x)	Вычисление точного значения выражения log(1+x) для маленьких значений x
nthroot(x, n)	Вычисление вещественного корня степени n для вещественного значения x (значения x и n должны быть вещественными, n — скаляром)

Ниже приведены примеры некоторых функций, перечисленных в табл. 5.6.

```
>> a=[1 4.5 3 8]
a =
    1.0000    4.5000    3.0000    8.0000
>> exp(a)
ans =
  1.0e+003 *
    0.0027    0.0900    0.0201    2.9810
>> log10(a)
ans =
         0    0.6532    0.4771    0.9031
>> pow2(a)
ans =
    2.0000   22.6274    8.0000  256.0000
>> nextpow2(a)
ans =
     2
```

Функции для работы с комплексными числами

Для работы с комплексными данными в MATLAB предусмотрены специальные функции, перечисленные в табл. 5.7.

Таблица 5.7. Функции комплексного аргумента

Функция	*Описание*
abs(x)	Возвращает модуль комплексного числа x
angle(x)	Возвращает аргумент комплексного числа x (в радианах, в диапазоне от $-\pi$ до π)
conj(x)	Возвращает число, комплексно-сопряженное относительно x
imag(x)	Возвращает мнимую часть комплексного числа x
real(x)	Возвращает действительную часть комплексного числа x
complex(A,B)	Создает комплексное число A+Bi по заданной действительной и мнимой части (A и B могут быть скалярами либо массивами одинакового размера и типа)

Чтобы понять работу описанных функций, изучите следующие примеры.

```
>> z=[5+7i, 2i, 9-4i];
>> abs(z)
ans =
    8.6023    2.0000    9.8489
>> angle(z)
ans =
    0.9505    1.5708   -0.4182
>> conj(z)
ans =
   5.0000 - 7.0000i        0 - 2.0000i   9.0000 + 4.0000i
>> real(z)
ans =
     5     0     9
>> imag(z)
ans =
     7     2    -4
```

Функции округления и вычисления остатка от деления

В табл. 5.8 описаны функции округления и нахождения остатка от деления.

Таблица 5.8. Функции округления и вычисления остатка от деления

Функция	*Описание*
fix(x)	Округление до ближайшего целого в сторону нуля
floor(x)	Округление до ближайшего целого в сторону отрицательной бесконечности
ceil(x)	Округление до ближайшего целого в сторону положительной бесконечности
round(x)	Округление до ближайшего целого
mod(x)	Остаток от целочисленного деления с учетом знака
rem(x)	Остаток от целочисленного деления по модулю
sign(x)	Определение знака числа

Вот несколько примеров.

```
>> b=[1.95 8.17 -4.2];
>> fix(b)
ans =
     1     8    -4
>> floor(b)
ans =
     1     8    -5
>> ceil(b)
ans =
     2     9    -4
>> round(b)
ans =
     2     8    -4
>> mod(b,2)
ans =
    1.9500    0.1700    1.8000
>> rem(b,3)
ans =
    1.9500    2.1700   -1.2000
>> sign(b)
ans =
     1     1    -1
```

Функции для работы со значениями даты и времени

В MATLAB существует ряд функций, предназначенных для работы со значениями даты и времени. Такие функции, в частности, возвращают информацию о текущих дате и времени, а также преобразуют даты из одного допустимого формата в другой.

Система MATLAB поддерживает три различных формата представления даты: строковый формат '*дд-ммм-гггг*'; внутренний числовой формат (дата представляется в виде порядкового числа — номера текущего дня, который отсчитывается от 1-го января 0000 года); и векторный формат [*год месяц день час минута секунда*].

Для пользователя более привычными и наглядными являются строковый и векторный форматы. MATLAB же оперирует с датами как с порядковыми числами. При этом время рассматривается как дробная часть от порядкового номера дня. Например, если порядковый номер 73 1871 соответствует дате 17 октября 2003 года, то номер 37 745,75 указывает на момент времени, соответствующий 18 часам 17 октября 2003 года.

Это можно проверить с помощью специальных функций, преобразующих дату из одного формата в другой:

- `datenum` — преобразует строковый формат даты во внутренний числовой формат;
- `datestr` — преобразует внутренний числовой формат даты в строковый;
- `datevec` — преобразует дату, заданную во внутреннем числовом либо в строковом формате, в векторный формат.

```
>> datenum('17-Oct-2005')
ans =
732602
>> datestr(732602.75)
ans =
17-Oct-2005 18:00:00
>> datevec(732602.75)
ans =
  2005       10       17       18        0        0
```

Функция `datestr` может принимать второй аргумент, позволяющий изменять вид строки с датой (по умолчанию этот аргумент равен 1, что соответствует строке '*дд-ммм-гггг*'). Выбрав команду `help datestr`, можно ознакомиться со всеми возможными вариантами строковых форматов.

Функция `clock`, например, возвращает текущую дату в векторном формате [*год месяц день час минута секунда*].

```
>> a=clock
a =
  1.0e+003 *
   2.0050   0.0040   0.0040   0.0140        0   0.0353
```

Текущая дата будет более наглядной, если функцию `clock` задать в виде аргумента функции округления `fix`.

```
>> a=fix(clock)
a =
     2005        4        4       14        0       57
```

В данном примере переменной `a` присваивается вектор, первый элемент которого соответствует текущему году (а именно, 2005), второй — месяцу (апрель), третий — числу (4), а четвертый, пятый и шестой — часу, минуте и секунде соответственно (14:00:57).

Функция `now`, например, возвращает дату и время во внутреннем числовом формате, т.е. в виде одного дробного числа, целая часть которого означает дату, а дробная — время.

```
>> format long;
>> now
ans =
    7.324065858320834e+005
```

Чтобы вывести дату в строковом формате, нужно воспользоваться функцией `date`.

```
>> date
ans =
04-Apr-2005
```

Функция `calendar` выводит в командное окно календарь на текущий месяц в виде таблицы из семи столбцов.

```
>> calendar
                  Apr 2005
     S     M    Tu     W    Th     F     S
     0     0     0     0     0     1     2
```

```
     3     4     5     6     7     8     9
    10    11    12    13    14    15    16
    17    18    19    20    21    22    23
    24    25    26    27    28    29    30
     0     0     0     0     0     0     0
```

Чтобы отобразить календарь какого-либо другого месяца и года, нужно задать функцию `calendar` с двумя аргументами (первый из них будет означать год, а второй — номер месяца). Например, отобразим календарь на март 2005 года.

```
>> calendar(2005,3)
                  Mar 2005
     S     M    Tu     W    Th     F     S
     0     0     1     2     3     4     5
     6     7     8     9    10    11    12
    13    14    15    16    17    18    19
    20    21    22    23    24    25    26
    27    28    29    30    31     0     0
     0     0     0     0     0     0     0
```

Особый интерес в MATLAB представляет пара функций `tic` и `toc`, позволяющих вычислить время выполнения системой той или иной операции. Если в командной строке набрать приведенную ниже последовательность команд:

```
tic, операция, toc
```

где *операция* — это команда или набор команд, то будет отображен не только результат выполнения заданной операции, но и приблизительное время ее выполнения (в секундах). При этом функция `tic` запускает секундомер, а функция `toc` выводит вычисленное время. Это время зависит от нескольких различных факторов и может изменяться, поэтому для получения более точных значений времени следует провести несколько измерений, а результаты усреднить.

Функции `tic` и `toc` очень полезны, когда нужно выбрать наиболее эффективную реализацию решения одной и той же задачи.

К функциям, служащим для работы со значениями даты и времени, относятся также функции `cputime`, `etime`, `weekday` и `eomday`, информацию о которых можно получить, воспользовавшись справочной системой MATLAB.

Функции для выполнения побитовых операций

Программа MATLAB включает ряд функций, предназначенных для выполнения побитовых операций над целыми положительными числами.

Например, для представления целых положительных чисел в двоичном виде служит функция `dec2bin`.

```
>> dec2bin(9)
ans =
1001
```

Обратную операцию можно выполнить посредством функции `bin2dec` (двоичное число, заданное в качестве аргумента этой функции, следует заключить в апострофы).

```
>> bin2dec('1001')
ans =
     9
```

Для перевода десятичного числа в шестнадцатеричное используется функция `dec2hex`.

```
>> dec2hex(216)
ans =
D8
```

Обратная операция производится с помощью функции hex2dec.

```
>> hex2dec('D8')
ans =
   216
```

Аргументами рассмотренных функций могут быть и массивы, элементы которых являются целыми положительными числами. Например, если в качестве аргумента задана матрица, результат выдается по столбцам.

```
>> X=[12 35 8; 31 5 2];
>> dec2hex(X)
ans =
0C
1F
23
05
08
02
```

Функция bitmax возвращает максимальное допустимое целое число (без знака) для используемого компьютера. Данное значение определяется для комбинации, когда все биты установлены. На компьютерах с IEEE-арифметикой это число равно 2^{53}–1.

```
>> bitmax
ans =

  9.0072e+015
```

Для поразрядной логической обработки данных применяются функции bitand (поразрядное И), bitor (поразрядное ИЛИ) и bitxor (поразрядное исключающее ИЛИ). Указанные функции выполняют традиционные логические операции И, ИЛИ и "исключающее ИЛИ" над каждой парой битов своих операндов (работа операторов И, ИЛИ и "исключающее ИЛИ" описана в табл. 5.4). При этом более короткий операнд слева дополняется нулями. Операндами этих функций могут быть целые положительные числа, меньшие bitmax.

Ниже приведены примеры побитовых операций для x=55 и y=25.

```
>> x=55; y=25;
>> bitand(x,y)
ans =
    17
>> bitor(x,y)
ans =
    63
>> bitxor(x,y)
ans =
    46
```

Работа функций поразрядной обработки станет более понятной, если исходные и результирующие значения представить в двоичном виде, воспользовавшись функцией dec2bin.

Кроме того, в MATLAB существуют функции, позволяющие прочитать значение определенного разряда или установить его в требуемое значение (0 или 1). Это функции bitget и bitset. Например, чтобы получить значение 4-го разряда числа 358 (которое в двоичном представлении равно 101100110), нужно выбрать следующую команду.

```
>> bitget(358,4)
ans =
     0
```

При этом номер разряда не должен превышать число 52.

А чтобы установить в единицу 4-й разряд числа 358, выберем такую команду.

```
>> bitset(358,4,1)
ans =
   366
```

В результате получилось число 366, которое в двоичном виде соответствует значению 101101110.

Специальные математические функции

Кроме элементарных функций, пакет MATLAB включает целый ряд специальных математических функций, которые встречаются в задачах математической физики. Такие функции являются решениями некоторых дифференциальных уравнений или обозначениями интегралов определенного вида.

Аргументами специальных математических функций могут быть как отдельные числа, так и массивы чисел. Если аргументом спецфункции является массив, в результате получится массив того же размера, элементы которого будут преобразованы в соответствии с заданной функцией.

Далее мы рассмотрим некоторые специальные математические функции, доступные в MATLAB.

Например, функции Эйри (`airy`) формируют пару линейно-независимых решений линейного дифференциального уравнения следующего вида.

$$\frac{d^2w}{dz^2} - zw = 0$$

Функции Эйри в MATLAB задаются следующим образом:

- `w=airy(z)` — функция Эйри первого порядка;
- `w=airy(k,z)` — результат зависит от значений k: при k=1 возвращается производная функции Эйри первого порядка; при k=2 возвращается функция Эйри второго порядка; при k=3 возвращается производная функции Эйри второго порядка;
- `[w,ierr]=airy(k,z)` — во втором выходном аргументе возвращается информация о вычислении значений функции Эйри (`ierr=0` — функция Эйри успешно вычислена; `ierr=1` — неверно заданы входные аргументы; `ierr=2` — переполнение, ответ равен `Inf`; `ierr=3` — частичная потеря точности при вычислениях; `ierr=4` — полная потеря точности при вычислениях, так как z слишком большое; `ierr=5` — вычислительный процесс не сходится, результат будет `NaN`).

Для примера вычислим значения функции Эйри второго порядка.

```
>> k=2; z=96;
>> [f,ierr]=airy(k,z)
f =
  3.8786e+271
ierr =
     0
```

Функции Бесселя являются решениями дифференциального уравнения

$$z^2\frac{d^2y}{dx^2} + z\frac{dy}{dz} + (z^2 - v^2)y = 0$$

для вещественных v. Функции Бесселя делятся на функции первого, второго и третьего рода. В MATLAB для вычисления функций Бесселя предусмотрены следующие функции.

- `besselj(nu,z)` — функция Бесселя первого рода. Аргумент `nu` — вещественное число, определяющее порядок функции, аргумент `z` может принимать комплексные значения.

Если z и nu представляют собой массивы одинаковых размеров, результат вычисления функции будет массивом того же размера. Если один из аргументов — массив, а другой — число, это число "расширяется" до размеров массива, и результат представляет собой массив, соразмерный с исходным. Если же один из аргументов — вектор-строка, а другой — вектор-столбец, результатом будет двухмерный массив значений функции.

- `bessely(nu,z)` — функция Бесселя второго рода.
- `besselh(nu,k,z)` — функция Бесселя третьего рода, известная также, как функция Ганкеля. Результат зависит от значений k: при k=1 возвращается функция Ганкеля первого рода, при k=2 возвращается функция Ганкеля второго рода.

Приведенные функции можно также вызывать со вторым выходным аргументом, в котором будет отображена информация о найденном значении функции (см. выше функцию `airy`).

В качестве примера вычислим значения функции Бесселя первого рода.

```
>> N=[0 1; 2 3]; Z=[2+3i, 6; 5-i, 27];
>> [f,ierr]=besselj(N,Z)
f =
  -0.4695 - 4.3138i  -0.2767
   0.0304 + 0.3929i  -0.1459
ierr =
     0     0
     0     0
```

Во втором выходном аргументе `ierr` возвращаются нули, значит, функция успешно вычислена для всех входных значений.

Другие специальные функции MATLAB представлены в табл. 5.9.

Таблица 5.9. Другие специальные функции MATLAB

Функция	*Описание*
besseli	Модифицированная функция Бесселя первого рода
besselk	Модифицированная функция Бесселя второго рода
beta	Бета-функция
betainc	Неполная бета-функция
betaln	Логарифм бета-функции
gamma	Гамма-функция
gammainc	Неполная гамма-функция
gammaln	Логарифм гамма-функции
ellipj	Эллиптические функции Якоби
ellipke	Полный эллиптический интеграл
expint	Функция экспоненциального интеграла
erf	Функция ошибок
erfc	Дополнительная функция ошибок
erfcx	Масштабированная дополнительная функция ошибок
erfinv	Обратная функция ошибок
factorial	Функция вычисления факториала

Окончание табл. 5.9

Функция	*Описание*
gcd	Наибольший общий делитель
lcm	Наименьшее общее кратное
legendre	Обобщенная функция Лежандра
rat	Представление вещественного числа в виде рациональной дроби

Чтобы вывести соответствующий раздел справочной системы MATLAB с подробной информацией о требуемой специальной функции (а также о любой другой функции) и примерами ее использования, введите в командной строке команду `doc` *`функция`*, задав вместо слова *`функция`* имя нужной функции.

Специальные символы

К числу операторов системы MATLAB относятся также специальные символы (табл. 5.10). Они позволяют формировать различные объекты входного языка, а также языка программирования MATLAB.

Таблица 5.10. Специальные символы

Символ	*Описание*
:	Двоеточие служит для формирования подвекторов и подматриц из векторов и матриц, а также для создания числовых последовательностей
()	Круглые скобки применяются для задания порядка выполнения операций в арифметических выражениях, указания аргументов функции и указания индексов элемента вектора или матрицы
[]	Квадратные скобки предназначены для формирования векторов и матриц
{}	Фигурные скобки служат для формирования массивов ячеек
.	Десятичная точка применяется для отделения дробной части чисел от целой, а также для выделения полей структур
..	Две точки означают переход по дереву каталогов на один уровень вверх
...	Три и более точек в конце строки означают продолжение этой строки
;	Используется для разделения строк матриц (в круглых скобках), а также для подавления вывода на экран результатов вычислений (в конце операторов)
,	Запятая служит для разделения индексов элементов матрицы и аргументов функции, а также наряду с символом точки с запятой используется для разделения операторов в строке
%	Знаком процента предваряются текстовые комментарии (такие комментарии MATLAB игнорирует)
!	Восклицательный знак свидетельствует о вводе команды операционной системы
=	Знак равенства является символом присваивания значений в арифметических выражениях
'	Апостроф (одиночная кавычка) является символом транспонирования; текст, заключенный в апострофы, представляется как вектор символов с компонентами, являющимися ASCII-кодами символов
[,]	Горизонтальная конкатенация матриц
[;]	Вертикальная конкатенация матриц

Перечисленные в табл. 5.10 специальные символы относятся к следующим категориям: `colon` (:), `paren` (()), [], {}), `punct` (., .., ..., ,, ;, %, !, =), `ctranspose` ('), `horzcat` ([,]) и `vertcat` ([;]). Иными словами, чтобы получить справочную информацию о специальных символах MATLAB и синтаксисе их применения, введите в командной строке команду `help` или `doc`, а затем имя нужной категории.

Резюме

Эта глава содержит информацию об основных операторах и функциях системы MATLAB, достаточную для вычисления выражений любой степени сложности. Мы рассмотрели арифметические и логические операторы, а также соответствующие им функции. Соответствие функций операторам — это одно из основных положений программирования. Кроме того, в главе описаны элементарные функции, функции для работы со значениями даты и времени, функции для выполнения поразрядных операций, а также дан обзор специальных математических функций MATLAB и специальных символов.

Подведем итоги:

- в MATLAB существуют арифметические операторы двух типов —матричные и поэлементные;
- для изменения приоритета операторов в выражении используются круглые скобки;
- в MATLAB "истина" обозначается логической единицей (1), "ложь" — логическим нулем (0);
- логические операторы (кроме оператора логического отрицания) имеют самый низкий приоритет;
- MATLAB поддерживает три формата представления даты и времени: строковый, внутренний числовой и векторный формат.

В следующей главе вы познакомитесь со встроенными средствами MATLAB, используемыми для решения различных математических задач.

Тесты

Выберите правильные ответы на приведенные ниже вопросы.

1. Для чего используются операторы ".+" и ".–":
   - **а)** для выполнения поэлементного сложения и вычитания;
   - **б)** для сложения и вычитания матриц;
   - **в)** таких операторов в MATLAB не существует.
2. Среди арифметических операторов наибольший приоритет имеют:
   - **а)** операторы возведения в степень;
   - **б)** операторы сложения и вычитания.
   - **в)** операторы умножения и деления.
3. Можно ли использовать операторы отношения для поэлементного сравнения двух матриц:
   - **а)** да;
   - **б)** нет.

4. Могут ли операторы отношения использоваться в выражениях, вводимых в командном окне системы MATLAB, наряду с арифметическими операторами:
   а) да;
   б) нет.
5. Результатом логической операции "исключающее ИЛИ" будет 1 лишь в том случае:
   а) когда оба операнда равны нулю;
   б) когда оба операнда не равны нулю;
   в) когда один из операндов равен нулю, а другой не равен.
6. Какое из утверждений является верным:
   а) приоритет логических операторов (кроме оператора логического отрицания) ниже, чем приоритет арифметических операторов;
   б) приоритет логических операторов (кроме оператора логического отрицания) выше, чем приоритет арифметических операторов;
   в) вычисление выражений всегда происходит слева направо, независимо от приоритета операторов.
7. В каком формате возвращает дату функция `clock`:
   а) во внутреннем числовом формате;
   б) в векторном формате;
   в) в строковом формате.
8. Какая функция преобразует внутренний числовой формат даты в строковый:
   а) `datenum`;
   б) `datestr`;
   в) `datevec`.
9. Для установки разряда числа в требуемое значение применяется функция:
   а) `bitget`;
   б) `bitset`;
   в) `setbit`.
10. Функция `besselh` предназначена для вычисления:
   а) функции Ганкеля;
   б) функции Бесселя первого рода;
   в) функции Бесселя второго рода.

Упражнения

1. Расставьте следующие операции в порядке возрастания их приоритетов: сложение (`+`), возведение в степень (`^`), логическое ИЛИ (`|`), поэлементное деление (`./`), унарный минус (`-`), меньше (`<`), логическое отрицание (`~`).
2. Перепишите выражение `(a*b~=c)+(a>=c)+(a^c<=b)`, используя вместо операторов отношения соответствующие функции отношения. Проверьте правильность выполнения полученного выражения, присвоив переменным `a` и `b` конкретные числовые значения.

3. С помощью специальных функций отобразите на экране текущую дату в строковом формате, а также календарь на текущий месяц. Преобразуйте эту дату сначала в векторный, а затем во внутренний числовой формат.
4. Определите среднее время, которое система тратит на отображение календаря на текущий месяц.
5. Выполните поразрядные операции И, ИЛИ и "исключающее ИЛИ" над числами 39 и 20. Проверьте правильность вычислений, представив исходные значения и результаты поразрядных операций над ними в двоичном виде.
6. Постройте графики функций Бесселя первого рода для значений порядка nu=0, nu=1, nu=2, nu=3 и nu=4 на промежутке от 0 до 20 с шагом изменения 0,01.