Компьютерные технологии в научных исследованиях и образовании

Юрий Анатольевич Богоявленский, заведующий кафедрой Информатики и математического обеспечения, к.т.н., доцент, ybgv

Выражения

Индексные выражения

Арифметические операторы

Операторы сравнения

Выражения присваивания

Операторы приращения (increment/decrement)

Приоритет операторов

Выражения

Выражение состоит из связанных знаками операций констант, имен, переменных, массивов или их элементов и вызвов функций.

Индексные выражения

Индексное выражение позволяет выбирать элементы или подмассивы массивов (в частности векторов и матриц). Оно может иметь тип скаляра, вектора, диапазона или представляется специальным оператором **‘:’**, который выбирает целые строки, столбцы или фрагменты более высокой размерности.

Индексное выражение состоит из заключенных в круглые скобки `M` подвыражений, разделенных запятыми, каждое из которых последовательно соответствует измерениям массива. Т.е. первое после скобки подвыражение соответствует первому измерению — строкам в

случае матрицы, второе — столбцам и т. д. Значение M определяет размерность индексного выражения.

В простейшем случае 1) все подвыражения являются скалярами и 2) размерность индексного выражения M равна размерности массива, к которому оно применяется. Например:

```
ybgv@ybgv-home:~> octave -q -p ~/MyOct
octave:1> newwhos
octave:2>
octave:2> # создадим 3- мерный массив
octave:2> A=(1:8)
A =

   1   2   3   4   5   6   7   8

octave:3> B=reshape(A,2,2,2)
B =

ans(:,:,1) =

   1   3
   2   4

ans(:,:,2) =

   5   7
   6   8

octave:4> # элемент второй строка первого столбца второго слоя
octave:4>
octave:4> B(2,1,2)
ans = 6
octave:5> whos
Variables visible from the current scope:

variables in scope: top scope

Attr   Name        Size              Type               Bytes Class
====   ====        ====              ====               ===== =====
       A           1x8               double_range          24 double
       B           2x2x2             matrix                64 double
       ans         1x1               scalar                 8 double

Total is 17 elements using 96 bytes
octave:6>
```

Количество элементов возвращаемого объекта в конкретном измерении равно количеству элементов в соответствующем подвыражении индексного выражения. Когда все подвыражения являются скалярами это количество равно единице. Однако, если какое-либо подвыражение является вектором или диапазоном, то это количество есть декартово произведение множеств подвыражений в соответствующих измерениях. Например:

```
octave:12>
octave:12> [B(1,1,2);B(2,1,2)]
ans =

   5
   6

octave:13> size([B(1,1,2);B(2,1,2)])
ans =

   2   1

octave:14> # этому индексному выражению тождественно равно:
octave:14> B([1,2],1,2)
ans =

   5
   6

octave:15> size(B([1,2],1,2))
ans =

   2   1

octave:16>
```

Общее количество возвращенных элементов является произведением количества элементов, возвращенных для каждого компонента индекса. В приведенном выше примере общая сумма равна 2*1*1 = 2 элемента.

Обратите внимание, что количество элементов возвращаемого объекта в данном измерении равно количеству элементов в подвыражении индекса для этого измерения. В приведенном выше коде первый компонент индекса ([1, 2]) был указан как вектор строки, но его форма не имеет значения.

Важным фактом является то, что в подвыражении указано два значения, и, следовательно, результат должен иметь размер, равный двум в первом измерении и поскольку первое измерение соответствует строкам, общий результат представляет собой вектор столбец. Еще примеры.

```
octave:16> B (1,[2,1,1],1)
ans =

   3   1   1

octave:17> size (B (1,[2,1,1],1))
ans =

   1   3

octave:18> ones(2,2)
ans =

   1   1
   1   1

octave:19> B(ones(2,2),1,1)
ans =

   1
   1
   1
   1

octave:20>
```

Первый пример показывает, что количество элементов выходных данных в заданном измерении равно количеству элементов в соответствующем компоненте индексации. В этом случае выходные данные содержат три элемента во втором измерении (которое соответствует столбцам), поэтому результат является вектором строкой. В примере также показано, как повторяющиеся записи в индексном выражении могут быть использованы для повторения элементов в выходных данных.

Следующий пример показывает, что форма подвыражения не имеет значения, важно только количество элементов (2x2 = 4).

Приведенные выше правила применяются когда размерность подвыражения больше единицы и M > 1. Однако для одномерных подвыражений применяются специальные правила и форма выходных данных определяется формой индексирующего подвыражения. Например:

```
octave:29> B
B =

ans(:,:,1) =

   1   3
   2   4

ans(:,:,2) =

   5   7
   6   8

octave:30> B([1,2])
ans =

   1   2

octave:31> size(B([1,2]))
ans =

   1   2

octave:32> B([1;2])
ans =

   1
   2

octave:33> size(B([1;2]))
ans =

   2   1
octave:34>
```

Результатами являются вектор строка и вектор-столбец. Отметим что допустимо использовать одномерное подвыражение с многомерным объектом. В этом случае элементы многомерного массива берутся в порядке первого столбца, т. е. предполагается, что столбцы массива «уложены» друг на друга, образуя вектор столбцов.

Символ двоеточия **‘:’** может использоваться в качестве подвыражения для выбора всех элементов в соответствующем измерении.

Пример.

```
octave:1> newwhos
octave:2>
octave:2> A=[20,30;40,50]
A =

   20   30
   40   50

octave:3> A(1,[1,2])
ans =

   20   30

octave:4> size(A(1,[1,2]))
ans =

   1   2

octave:5> A(1,1:2)
ans =

   20   30

octave:6> size(A(1,1:2))
ans =

   1   2

octave:7> A(1,:)
ans =

   20   30

octave:8> size(A(1,:))
```

```
ans =

   1   2

octave:9> whos
Variables visible from the current scope:

variables in scope: top scope

Attr   Name        Size            Type          Bytes Class
====   ====        ====            ====          ===== =====
       A           2x2             matrix           32 double
       ans         1x2             matrix           16 double

Total is 6 elements using 48 bytes

octave:10>
```

Здесь три индексных выражение эквивалентны и выделяют первую строку матрицы.

Когда двоеточие используется в частном случае одномерной индексации, результатом является вектор-столбец построенный из столбцов исходного массива. Создание векторов столбцов с индексом двоеточия - это часто применяемая идиома языка octave.

Пример.

```
octave:1>
octave:1> newwhos
octave:2>
octave:2> A=[20,30;40,50],size(A)
A =

   20   30
   40   50

ans =

   2   2

octave:3> A(1,[1,2]),size(A(1,[1,2]))
```

```
ans =

   20   30

ans =

   1   2

octave:4> A(1,1:2),size(A(1,1:2))
ans =

   20   30

ans =

   1   2

octave:5> A(1,:),size(A(1,:))
ans =

   20   30

ans =

   1   2

octave:6> A(:),size(A(:))
ans =

   20
   40
   30
   50

ans =

   4   1

octave:7> B=(1:8)
B =

   1   2   3   4   5   6   7   8

octave:8> C=reshape(B,2,2,2),size(C)
C =

ans(:,:,1) =
```

```
   1   3
   2   4

ans(:,:,2) =

   5   7
   6   8

ans =

   2   2   2

octave:9> C(:),size(C(:))
ans =

   1
   2
   3
   4
   5
   6
   7
   8

ans =

   8   1

octave:10> whos
Variables visible from the current scope:

variables in scope: top scope

Attr   Name        Size            Type                Bytes Class
====   ====        ====            ====                ===== =====
       A           2x2             matrix                 32 double
       B           1x8             double_range           24 double
       C           2x2x2           matrix                 64 double
       ans         1x2             matrix                 16 double

Total is 22 elements using 136 bytes

octave:11>
```

Дополнительно об индексных выражениях можно прочесть в [Док, с. 155 — 157] и на ресурсе:

Программирование на Octave/Векторы и матрицы [Электронный ресурс] — URL: https://ru.wikibooks.org/wiki/Программирование_на_Octave/Векторы_и_матрицы#Создание_матриц (27.09.2022)

Арифметические операторы

Операндами следующих арифметических операторов могут быть скаляры и массивы (в частности векторы и матрицы).

Поэлеменнтные операторы.

Для этих бинарных операторов возможна любая комбинация операнда скаляра и операнда массива. Если только один операнд скаляр то оператор выполняется для него и каждого элемента массива. Если массивы оба операнда, то они должны иметь одинаковую размерность и оператор выполняется над соответствующими элементами.

`x + y` Сложение.

`x - y` Вычитание.

`x .* y` Умножение.

`x ./ y` Поэлементное правое деление. Делимое — `x`, делитель — `y`.

`x .\ y` Поэлементное левое деление. Делимое — `y`, делитель — `x`.

Матричные операторы.

`x * y` Умножение матриц. Количество столбцов `x` должно совпадать с количеством строк `y`.

`x / y` Правое деление. Результат эквивалентен выражению $((y^T)^{-1} \cdot x^T)^T$, или, в обозначениях `octave` – `(inv (y') * x')`, где символ ' обозначает операцию транспонирования матрицы. Отметим, что при вычислении матрица $(y^T)^{-1}$ не формируется. Если матрица `y` является не квадратной или сингулярной, вычисляется минимальное нормальное решение.

`x \ y` Левое деление. Результат эквивалентен выражению $X^{-1} \cdot y$ или, в обозначениях `octave`, `inv (x) * y`. Отметим, что при вычислении матрица X^{-1} не формируется. Если матрица `X` является не квадратной или сингулярной, вычисляется минимальное нормальное решение.

`x ^ y` Возведение в степень. Если `x` и `y` оба скалярами, то выполняется стандартное возведение в степень. Если `x` — скаляр, а `y` — квадратная матрица, результат вычисляется с использованием разложения по собственным значениям. Если `x` — квадратная матрица, то вычисляется результа путем повторного умножения, если `y` является целым числом, и путем разложения по собственным значениям, если `y` не является целым числом. Ошибка возникает, если и `x`, и `y` являются матрицами.

`x .^ y` Поэлементное возведение степень. Если оба операнда являются матрицами, количество строк и столбцов должно совпадать, или они должны транслироваться в одну и ту же форму. Если возможно несколько комплексных результатов, берется тот, у которого наименьший неотрицательный аргумент (угол). Это правило может возвращать комплексный корень даже когда возможен и вещественный корень. Рекомендуется использовать фукции `realpow`, `realsqrt`, `cbrt` или `nthroot`, если предпочтителен вещественный результат.

`-x` Унарный минус, поэлементно меняет знак операнда.

`+x` Унарный плюс. Ничего не делает.

`x'` Комплексно сопряженное транспонирование. Для вещественных аргументов этот оператор совпадает с оператором `x.'` транспонирования матриц. Для сложных аргументов этот оператор эквивалентен выражению `conj (x.')`.

`x.'` Транспонирование матрицы.

NBNB. Поэлементные операторы `octave` начинаются с символа точка `'.'`, т. е. возможна двусмысленность для таких утверждений, как `1./m` т.к. этот символ может быть интерпретирован либо как часть константы, либо как часть оператора. Для разрешения конфликт, Octave обрабатывает выражение так, как если бы вы ввели `(1)./ m`, а не `(1.)/ m`.

В [Док, с. 162— 164] представлены функции также выполняющие арифметические операции.

Операторы сравнения

Оператор сравнивает числовые значения с помощью операций отношения. Результат получает значение `true` или `false`, принадлежат классу `logical` и имеет тип `bool`.

Если оба операнда массивы, то они должны иметь одинаковую размерность, сравнение происходит поэлементно и результат имеет ту же размерность. Если один операнд скаляр, а второй — массив, то скаляр сравнивается с каждым его элементом и результат имеет размерность массива. Оператор получает значение `true/false` если справедливо/не справедливо отношение:

`x < y` x меньше y.

`x <= y` x меньше или равно `y.`

`x == y` x равно `y`.

`x >= y` x больше или равно `y`.

`X > y` больше, чем y.

`x != y или x ~= y` не равно `y`.

Для комплексных чисел определен следующий порядок:

`z1 < z2` тогда и только тогда, когда

```
abs (z1) < abs (z2)|| (abs (z1) == abs (z2) && arg (z1) < arg (z2))
```

Это согласовано с порядком, используемым функциями `max, min` и `sort`, но не согласуется с принятым в пакете `matlab`, который сравнивает только вещественные части.

Сравнение строк также может выполняться с помощью функции `strcmp`.

Также в [Док, с. 166 — 168] описаны поэлементные булевские операторы и булевские операторы короткого замыкания, которые мы не рассматриваем.

Выражения присваивания

Присваивание - это выражение, дающее переменной новое значение, например, `z = 1`. Символ ‘=’ называется оператором присваивания.

Левый операнд присваивания не обязательно должен быть переменной . Он может быть элементом матрицы или списком возвращаемых значений функции (рассмотрим позже)]. Для возможных видов левых операндов существует обобщенное название — lvalues. Правым операндом может быть любое выражение, вычисляющее новое значение левого операнда.

Подчеркнем, что в `octave` тип переменных на фиксируется, а определяется типом последнего присвоенного ей значения. Например в

следующем фрагменте переменная foo сначала имеет числовое значение, а затем строковое:

```
octave:13> foo = 1
foo = 1
octave:14> foo = "bar"
foo = bar
```

Если слева элементы массива, задаваемые индексным выражением а справа скалярное значение, то оно присваивается всем, определенным слева, элементам массива. Использование пустой матрицы ‘[]’ позволяет удалять строки или столбцы массивов.

Пример.

```
octave:1>
octave:1> newwhos
octave:2>
octave:2> A=[5 10 15 20 25;30 35 40 45 50;55 60 65 70 75]
A =

    5   10   15   20   25
   30   35   40   45   50
   55   60   65   70   75

octave:3> A(:,2)=5
A =

    5    5   15   20   25
   30    5   40   45   50
   55    5   65   70   75

octave:4> A(3,:)=[]
A =

    5    5   15   20   25
   30    5   40   45   50

octave:5> A(:,1:2:5)=[]
A =

    5   20
```

```
    5   45

octave:6>
```

Будучи выражением присваивание имеет значение — это значение присвоенное левой части, что позволяет задавать присваивания вида:

```
octave:30> a=b=c=5
a = 5
octave:31> b
b = 5
octave:32> c
c = 5
octave:33>
```

В в [Док, с. 170] даны описания вариантов выражений присваивания с выполнением операций: `+=`, `-=`, `*=`, `/=`, которые мы не рекомендуем использовать из-за того, что они могут вести к написанию плохо читаемых текстов программ.

Операторы приращения (increment/decrement)

Эти операторы увеличивают или уменьшают значение операнда на `1` и задаются в префиксной: `++x`, `--x` или постфиксной: `x++`, `x--` формах. В обоих формах операнд `x` увеличивается или уменьшается на `1`. В префиксной форме само выражение `++x` или `--x` также увеличивается или уменьшается на 1, в постфиксный форме значение выражение `x++` или `x--` остается равным значению `x` как это показано на примере:

```
octave:37>
octave:37> x=5
x = 5
```

```
octave:38> ++x, x # префиксный инкремент
ans = 6
x = 6
octave:39> x=5
x = 5
octave:40> x++, x # постфиксный инкремент
ans = 5
x = 6
octave:41> x=5
x = 5
octave:42> --x, x # префиксный декремент
ans = 4
x = 4
octave:43> x=5
x = 5
octave:44> x--, x # постфиксный декремент
ans = 5
x = 4
octave:45>
```

Для операндов массивов изменение выполняется для каждого их элемента.

Приоритет операторов

Приоритет определяет последовательность выполнения операторов в выражении. Выражения в круглых скобках имеют приоритет, поэтому рекомендуется использовать их для ясного указания желаемых приоритетов выполнения операторов.

Операторы одного приоритета выполняются слева направо за исключением опрераторов присваивания, которые выполняются справа налево. Т.е. выражение `a - b + c` выполнится как `(a - b) + c`, а выражение `a = b = c` — как `a = (b = c)`.

Приоритет префиксных унарных операторов важен, когда за операндом следует другой оператор. Например, `-x ^ 2` означает `-(x^ 2)`, потому что `'-'` имеет более низкий приоритет, чем `'^'`.

Таблица приоритета операторов в порядке его убывания. Если не указано иное, операторы в одной группе выполняются слева направо.

‘()’ ‘{}’ ‘.’	Вызов функции и индексация массива, индексация массива ячеек и индексация элементов структуры
‘++’ ‘––’	Постфиксное приращение и постфиксное уменьшение. Эти операторы группируются справа налево.
‘’’ ‘.’’.’ ‘^’ ‘.^’	Транспонирование и возведение в степень
‘+’ ‘–’ ‘++’ ‘––’ ‘~’ ‘!’	Унарный плюс, унарный минус, префиксное приращение и префиксное уменьшение, логическое "Отрицание"
‘*’ ‘/’ ‘\’ ‘.\’ ‘.*’ ‘./’	Умножение и деление
‘+’ ‘–’	Сложение и вычитание
‘:’	Двоеточие
‘<’ ‘<=’ ‘==’ ‘>=’ ‘>’ ‘!=’ ‘~=’	Операци отношения
‘&’	Поэлементное логичское "И"
‘|’	Поэлементное логичское "Или"
‘&&’	Логичское "И"
‘||’	Логичское "Или"

‘=’ ‘+=’ ‘−=’ ‘*=’ ‘/=’ ‘\=’ ‘^=’ ‘.*=’ ‘./=’ ‘.\=’ ‘.^=’ ‘|=’ ‘&=’	Присваивание. Эти операторы группируются справа налево.