Компьютерные технологии в научных исследованиях и образовании

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

Инструкции

Инструкция `if`

Инструкция `switch`

Инструкция `while`

Инструкция `do-until`

Инструкция `for`

Инструкция может быть простым выражением или сложной конструкцией вложенных циклов и условных инструкций.

Управляющие инструкции управляют потоком выполнения в программах `octave`. Они начинаются со специальных ключевых слов, например `if`, `while`, чтобы отличать их от простых выражений. Многие управляющие инструкции содержат другие инструкции, например, инструкци `if` содержит инструкци, которые может выполняться или не выполняться.

Каждая инструкция управления имеет соответствующую инструкцию `end`, отмечающую конец текста инструкции. Например, ключевое слово `endif` обозначает конец инструкции `if`, а `endwhile` обозначает конец инструкции `while`. Можно использовать краткую форму — ключевое слово `end` везде где ожидается появление его конкретного варианта (например `endwhile`), но предпочтительнее всегда использовать конкретные варианты ключевых слов, т.к. при этом octave обеспечивает лучшую диагностику несоответствующих или отсутствующих слов `end`. Список инструкций, содержащихся между соответствующими ключевыми словами обозначающими начало и конец текста инструкции (напрмер `if` и `endif`) называется `телом` управляющей инструкции.

Инструкция `if`

Существует три основные формы инструкции `if`.

Первая форма:

```
if (условие)
  then-тело
endif
```

Фрагмент `then-тело` будет выполнен только если выражение `условие` имеет истинное значение. Принимается, что для всех значений не равных нулю `условие` имеет истинное значение, а для значения нуль — ложное значение. Если значение выражения `условие` является массивом (в частном случае вектором или матрицей), оно считается истинным только в том случае, если массив непустой и все его элементы отличны от нуля.

Вторая форма:

```
if (условие)
  then-тело
else
  else-тело
endif
```

Если `условие` истинно, то выполняется `then-тело` в противном случае - `else-тело`.

Третья, наиболее общая форма позволяет проверить несколько условий:

```
if (условие)
  then-тело
elseif (условие)
  elseif-тело
```

```
else
  else-тело
endif
```

Можно указать любое количество ключевыцх слов `elseif`. Каждое условие проверяется по очереди, и если одно из них оказывается истинным, выполняется соответствующее ему тело. Если ни одно из условий не является истинным и слово `else` присутствует, выполняется его тело. Может появиться только одно слово `else`, и оно должно быть последней частью инструкции. Пример:

```
octave:2>
octave:2> x=1
x = 1
octave:3> if (rem (x, 2) == 0)
> printf ("x is even\n");
> elseif (rem (x, 3) == 0)
> printf ("x is odd and divisible by 3\n");
> else
> printf ("x is odd\n");
> endif
x is odd
octave:4> x=2
x = 2
octave:5> if (rem (x, 2) == 0)
> printf ("x is even\n");
> elseif (rem (x, 3) == 0)
> printf ("x is odd and divisible by 3\n");
> else
> printf ("x is odd\n");
> endif
x is even
octave:6> x=3
x = 3
octave:7> if (rem (x, 2) == 0)
> printf ("x is even\n");
> elseif (rem (x, 3) == 0)
> printf ("x is odd and divisible by 3\n");
> else
> printf ("x is odd\n");
> endif
x is odd and divisible by 3
```

```
octave:8>
octave:8> x=1
x = 1
octave:9> if (rem (x, 2) == 0)
> printf ("x is even\n");
> elseif (rem (x, 3) == 0)
> printf ("x is odd and divisible by 3\n");
> endif
octave:10> x=2
x = 2
octave:11> if (rem (x, 2) == 0)
> printf ("x is even\n");
> elseif (rem (x, 3) == 0)
> printf ("x is odd and divisible by 3\n");
> endif
x is even
octave:12> x=3
x = 3
octave:13> if (rem (x, 2) == 0)
> printf ("x is even\n");
> elseif (rem (x, 3) == 0)
> printf ("x is odd and divisible by 3\n");
> endif
x is odd and divisible by 3
octave:14>
```

В следующем примере, если первое условие истинно (то есть значение x делится на 2, то выполняется первая инструкция `printf`. Если оно равно false, то проверяется второе условие , и если оно истинно (то есть значение x делится на 3), то выполняется вторая инструкция `printf`. В противном случае выполняется третья инструкция `printf`. Далее а примере представлен вариант без применения слова `else`.

NBNBNB. Для получение корректного результата слово `elseif` должно быть написано без пробела между словами `else` и `if`.

Инструкция `switch`

Если нужно выполнить много различных действий в зависимости от значение одного выражения то использование конструкции `elseif`

приведет к громоздкому, плохо читаемому тексту программы. В таких случаях следует использовать инструкцю switch, задаваемую в виде:

```
switch (expression)
  case label
       command_list
  case label
       command_list
...
  otherwise
       command_list
endswitch
```

Проверяется совпадает ли текущее значение выражения expression с одним из значений выражений (произвольных) label. При совпадении выполняется соответствующий список команд — command_list. Претворяющиеся значения выражений label не обнаруживаются, будет выполнен только список команд, соответствующий первому совпадению. **NBNBNB**. Должна присутствовать по крайней мере одно конструкция:

```
  case label
       command_list
```

в тоже время конструкция:

```
  otherwise
       command_list
```

является необязательной.

Мы рассмотрим массив ячеек, задаваемый с помощью символо „{ }“ позже, здесь приведем полезный пример использования его в инструкции switch. Соответствующий список команд выполняется, если какой-либо из элементов массива такого массива совпадет с выражение label. Пример:

```
octave:1>
octave:1> A=6
A = 6
octave:2> switch (A)
> case { 6, 7 }
> printf ("variable is either 6 or 7\n");
> otherwise
> printf ("variable is neither 6 nor 7\n");
> endswitch
variable is either 6 or 7
octave:3>
octave:3> A=7
A = 7
octave:4> switch (A)
> case { 6, 7 }
> printf ("variable is either 6 or 7\n");
> otherwise
> printf ("variable is neither 6 nor 7\n");
> endswitch
variable is either 6 or 7
octave:5>
octave:5> A=5
A = 5
octave:6> switch (A)
> case { 6, 7 }
> printf ("variable is either 6 or 7\n");
> otherwise
> printf ("variable is neither 6 nor 7\n");
> endswitch
variable is neither 6 nor 7
octave:7>
```

Инструкция while

Инструкция задается в виде:

```
while (условие)
  тело
endwhile
```

Здесь `тело` - это инструкция или список инструкций, которые мы называем телом цикла, оно повторно выполняется до тех пор, пока `условие` истинно. Точно также, как и для инструкции `if` принимается, что для всех значений не равных нулю выражение `условие` имеет истинное значение, а для значения нуль — ложное значение. Если значение выражения `условие` является массивом (в частном случае вектором или матрицей), оно считается истинным только в том случае, если массив непустой и все его элементы отличны от нуля.

Работа инструкции `while` начинается с проверки значения выражения `условие` и если оно истинно, то выполняется список `тело`. После того, как тело было выполнено, значение выражения `условие` проверяется снова, и если оно оно по-прежнему истинно, список `тело` выполняется снова. Этот процесс повторяется до тех пор, пока условие станет ложным. Если же значение выражения `условие` изначально было ложным, то список `тело` не выполнится ни одного раза.

В примере ниже создается вектор `fib`, содержащий значения первых десять элементов последовательности Фибоначчи.

```
octave:3> i=3
i = 3
octave:4> fib = ones (1, 10)
fib =
```

```
   1   1   1   1   1   1   1   1   1   1

octave:5> while (i <= 10)
> fib (i) = fib (i-1) + fib (i-2)
> i++
> endwhile
fib =

   1   1   2   1   1   1   1   1   1   1

ans = 3
fib =

   1   1   2   3   1   1   1   1   1   1

ans = 4
fib =

   1   1   2   3   5   1   1   1   1   1

ans = 5
fib =

   1   1   2   3   5   8   1   1   1   1

ans = 6
fib =

    1    1    2    3    5    8   13    1    1    1

ans = 7
fib =

    1    1    2    3    5    8   13   21    1    1

ans = 8
fib =

    1    1    2    3    5    8   13   21   34    1

ans = 9
fib =

    1    1    2    3    5    8   13   21   34   55

ans = 10
```

Отметим, что цикл завершается, когда переменная `i` получает значение 11.

Инструкция do-until

Инструкция задается в виде:

```
do
  тело
until (условие)
```

Как и ранее `тело` - это выполняемые в цикле инструкция или список инструкций. В отличие от инструкции `while` инструкции из `тело` повторно выполняются до тех пор пока `условие` **не станет истинным**, а его проверка выполняется **после** выполнения инструкций из `тело`, которые всегда выполнятся **по крайней мере один раз**.

Точно также, как и для инструкции `if` принимается, что для всех значений не равных нулю выражение `условие` имеет истинное значение, а для значения нуль — ложное значение. Если значение выражения `условие` является массивом (в частном случае вектором или матрицей), оно считается истинным только в том случае, если массив непустой и все его элементы отличны от нуля.

Вектор `fib` из предыдущего примера получается по следующей программе.

```
octave:5>
octave:5> fib = ones (1, 10)
fib =

   1   1   1   1   1   1   1   1   1   1

octave:6> i=2
i = 2
octave:7> do
> ++i
```

```
> fib (i) = fib (i-1) + fib (i-2)
> until (i == 10)
ans = 3
fib =

   1   1   2   1   1   1   1   1   1   1

ans = 4
fib =

   1   1   2   3   1   1   1   1   1   1

ans = 5
fib =

   1   1   2   3   5   1   1   1   1   1

ans = 6
fib =

   1   1   2   3   5   8   1   1   1   1

ans = 7
fib =

    1    1    2    3    5    8   13    1    1    1

ans = 8
fib =

    1    1    2    3    5    8   13   21    1    1

ans = 9
fib =

    1    1    2    3    5    8   13   21   34    1

ans = 10
fib =

    1    1    2    3    5    8   13   21   34   55

octave:8>
```

Инструкция for

Инструкция задается в виде:

```
for var = выражение
  тело
endfor
```

Эта инструкция позволяет более ясно задать итераций цикла. Здесь также тело — это выполняемые в цикле инструкция или список инструкций, а выражение — любое, допустимое в octave выражение.

Выражение присваивания var = выражение задает некоторый набор значений, для котрыз выполняется тело, его левая часть — var может принимать несколько форм. Обычно это имя скалярой переменной или массива (этот случай мы не расматриваем).

Выражение присваивания var = выражение в инструкции for работает не так как в обычном выражение присваивания octave. Левая часть - var не получает полный результат правой части, а получает значение ее элементов последовательно, по очереди. Если выражение есть диапазон, вектор строки или скаляр, значение var будет скаляром при каждом выполнении тела цикла. Если var есть вектор столбец или матрица, то var будет вектором столбцом каждый раз, когда выполняется тело цикла.

Для этой инструкции пример о последовательности Фибоначчи дается следующей программой.

```
octave:13> fib = ones (1, 10)
fib =

   1   1   1   1   1   1   1   1   1   1

octave:14> for i = 3:10
> i
> fib(i) = fib(i-1) + fib(i-2)
```

```
> endfor
i = 3
fib =

   1   1   2   1   1   1   1   1   1   1

i = 4
fib =

   1   1   2   3   1   1   1   1   1   1

i = 5
fib =

   1   1   2   3   5   1   1   1   1   1

i = 6
fib =

   1   1   2   3   5   8   1   1   1   1

i = 7
fib =

    1    1    2    3    5    8   13    1    1    1

i = 8
fib =

    1    1    2    3    5    8   13   21    1    1

i = 9
```

```
fib =

    1    1    2    3    5    8   13   21   34    1

i = 10
fib =

    1    1    2    3    5    8   13   21   34   55

octave:15>
```