

## ЛАБ. РАБОТА №4. УСЛОВНЫЕ ВЫРАЖЕНИЯ, ОПЕРАТОРЫ ВЕТВЛЕНИЯ И ЦИКЛЫ

### Условные выражения (conditional expression)

Условные выражения предназначены для ответов на вопросы заданные с помощью операций (инструкций) отношений и логических операций (инструкций).

### Операторы отношений (Relational operations)

- Операторы (инструкции) отношений всегда применяются к двум операндам, которые представлены в виде арифметических значений или отдельных текстовых символов. Результатом операции является булевское значение **true** или **false**.
- Операторы отношений могут также применяться к матрицам одинакового размера и размерности, элементами (ячейками) которых, являются арифметические значения или отдельные текстовые символы. В этом случае результатом операции будет матрица, полученная поэлементным вычислением со значениями true или false в соответствующих элементах.
- Применение **всех** операторов отношений для комплексных чисел возможно только в отношении их реальных или мнимых частей. Для этого используются функции выделения реальной `real(x)` или мнимой `imag(x)` части таких чисел. Непосредственное сравнение двух комплексных чисел допускается только для операторов **==** (равно) и **~=** (не равно).

Синтаксис и семантика операторов отношений для двух элементов (ячеек) представлены в таблице №1.

Таблица 1

Операция	Описание результата операции
$A < B$	Истина (1 или true), если левый операнд A <b>строго меньше</b> правого операнда B, иначе ложь (0 или false).
$A > B$	Истина (1 или true), если левый операнд A <b>строго больше</b> правого операнда B, иначе ложь (0 или false).
$A \leq B$	Истина (1 или true), если левый операнд A <b>меньше или равен</b> правому операнду B, иначе ложь (0 или false).
$A \geq B$	Истина (1 или true), если левый операнд A <b>больше или равен</b> правому операнду B, иначе ложь (0 или false).
$A == B$	Истина (1 или true), если левый операнд A <b>строго равен</b> правому операнду B, иначе ложь (0 или false).
$A ~= B$	Истина (1 или true), если левый операнд A <b>строго не равен</b> правому операнду B, иначе ложь (0 или false).

### Примеры

```
% №1-1. Пример для сравнения предварительно  
% вычисленных арифметических выражений X1 и X2
```

```

X1 = 2.34;
X2 = 2;
(X1 > X2)
           ans = 1
(X1 < X2)
           ans = 0

```

% **№1-2.** Пример для сравнения предварительно  
 % вычисленных комплексных выражений X1 и X2  
 % по их реальной и мнимой части.

```

X1 = 1 + 2.34i;
X2 = 1 + 2i;
real(X1) == real(X2)
                        ans = 1
imag(X1) == imag(X2)
                        ans = 0

```

% **№1-3 (непосредственное сравнение).**  
 % Пример для сравнения предварительно  
 % вычисленных комплексных выражений X1 и X2

```

X1 = 1 + 2.34i;
X2 = 1 + 2i;
X1 == X2
           ans = 0
X1 ~= X2
           ans = 1

```

% **№1-4.** Сравнение двух символов

```

X1='a';
X2='A';
X1 == X2
           ans = 0
X1 < X2
           ans = 0
X1 > X2
           ans = 1

```

% **№1-5.** Сравнение двух арифметических массивов

```

X1=[1 2 3; 4 5 6];
X2=[1 2 8; 4 6 6];
X1 == X2
           ans =

           1         1         0
           1         0         1

```

```
X1 < X2
```

```
ans =
```

```
0    0    1
0    1    0
```

% №1-6. Поэлементное сравнение двух текстовых строк

```
X1='qwerty';
```

```
X2='QwErTy';
```

```
X1 == X2
```

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

% №1-7. Полное сравнение двух текстовых строк

```
X1='qwerty';
```

```
X2='QwErTy';
```

```
strcmp(X1,X2)
```

```
ans = 0
```

```
X1='qwerty';
```

```
X2='qwerty';
```

```
strcmp(X1,X2)
```

```
ans = 1
```

% №1-8. Полное сравнение двух текстовых строк или матриц

% одинакового размера и размерности с помощью функции **isequal**

```
X1='qwerty';
```

```
X2='QwErTy';
```

```
isequal(X1,X2)
```

```
ans = 0
```

```
X1='qwerty';
```

```
X2='qwerty';
```

```
isequal(X1,X2)
```

```
ans = 1
```

```
X1=[1 2 3; 4 5 6];
```

```
X2=[1 2 8; 4 6 6];
```

```
isequal(X1,X2)
```

```
ans = 0
```

```
X1=[1 2 3; 4 5 6];
```

```
X2=[1 2 3; 4 5 6];
```

```
isequal(X1,X2)
```

```
ans = 1
```

## **Вычисление отношений алгоритмическим путем**

% **№2-1.** Полное алгоритмическое сравнение двух текстовых строк

```
X1='qwerty';
X2='QwErTy';
Y1=(X1 == X2);
% Вычислить произведение всех элементов Y1 посредством
% явного перечисления индекса в цикле
k=1;
for n = 1 : length(Y1)
    k=k * Y1(n);
end;
% Отобразить результат сравнения
if k == 0
    disp('Строки X1 и X2 НЕ равны');
end;
```

% **№2-2.** Полное алгоритмическое сравнение двух текстовых строк

```
X1='qwerty';
X2='QwErTy';
Y1=(X1 == X2);
% Вычислить произведение всех элементов Y1 посредством
% косвенного перечисления индекса в цикле
n =1;
for k = Y1
    n = n * k;
end;
% Отобразить результат сравнения
if n == 0
    disp('Строки X1 и X2 НЕ равны');
end;
```

% **№2-3.** Полное алгоритмическое сравнение двух текстовых строк  
% с использованием функций **prod** (перемножить элементы массива)  
% и **double** (преобразовать элементы массива к арифметическому  
% типу) .

```
X1='qwerty';
X2='QwErTy';
Y1=(X1 == X2);
% Вычислить произведение всех элементов Y1
n = prod(double(Y1));
% Отобразить результат сравнения
if n == 0
    disp('Строки X1 и X2 НЕ равны');
end;
```

## Логические операторы и операции (Logical operations)

Логические операции «И», «ИЛИ», «НЕ» оперируют с **вычисленными отношениями** и (или) **логическими переменными**, которые принимают значения ИСТИНА (**true** или **1**) или ЛОЖЬ (**false** или **0**). Результатом логической операции также является булевское значение **true** или **false**.

### Логическая операция «И».

Логическая операция **И** выполняет **логическое умножение двух переменных**, каждая из которых может содержать только булевское значение **true** или **false**. Варианты результатов операции «И» представлены в следующей таблице:

$\begin{smallmatrix} \text{X} \\ \text{Y} \end{smallmatrix}$	0	1
0	0	0
1	0	1

Другими словами результат операции **И** равен **true** только когда оба операнда X и Y равны **true**.

В **MATLAB** операция **И** обозначается символом **&** для матриц (поэлементное **И**) или **&&** для скалярных величин.

% **№1-1.** Явное вычисление

```
true & true           % матричная форма
ans = 1
true && true          % скалярная форма
ans = 1

true & false
ans = 0
false & true
ans = 0
```

% **№1-2.** Операция **И** для двух логических матриц Y1 и Y2

```
X1=[1 2 3; 4 5 6];
X2=[1 2 8; 4 6 6];
Y1=(X1==X2)

Y1 = 1     1     0
      1     0     1

Y2=(X1~=X2)

Y2 = 0     0     1
      0     1     0
```

```
Z1=(Y1 & Y2)
```

```
      Z1 = 0      0      0
           0      0      0
```

### Логическая операция «ИЛИ».

Логическая операция **ИЛИ** выполняет **логическое сложение двух переменных**, каждая из которых может содержать только булевское значение true или false. Варианты результатов операции «ИЛИ» представлены в следующей таблице:

Y \ X	0	1
0	0	1
1	1	1

Другими словами результат операции **ИЛИ** равен **false** только когда оба операнда X и Y равны **false**.

В **MATLAB** операция **ИЛИ** обозначается символом **|** для матриц (поэлементное **ИЛИ**) или **||** для скалярных величин.

% **№1-1.** Явное вычисление

```
true | false      % матричная форма
ans = 1
true || false     % скалярная форма
ans = 1

true | true
ans = 1
false | false
ans = 0
```

% **№1-2.** Операция **ИЛИ** для двух логических матриц Y1 и Y2

```
X1=[1 2 3; 4 5 6];
X2=[1 2 8; 4 6 6];
Y1=(X1==X2)
      Y1 = 1      1      0
           1      0      1

Y2=(X1~=X2)
      Y2 = 0      0      1
           0      1      0

Z1=(Y1 | Y2)
      Z1 = 1      1      1
           1      1      1
```

### Логическая операция «НЕ».

Логическая операция **НЕ** выполняет **инвертирование значения переменной**, которая может содержать только булевское значение true или false. Варианты результатов операции «НЕ» представлены в следующей таблице:

X	НЕ (X)
0	1
1	0

Другими словами результат операции **НЕ** равен **false** только когда операнд X равен **true**.

В **MATLAB** операция **НЕ** обозначается символом **~** как для матриц (поэлементное **НЕ**) так и для скалярных величин.

% **№1-1.** Явное вычисление

~true

ans = 0

~false

ans = 1

% **№1-2.** Операция **НЕ** для логической матрицы Y1

X1=[1 2 3; 4 5 6];

X2=[1 2 8; 4 6 6];

Y1=(X1==X2)

Y1 = 1      1      0  
     1      0      1

Z1=~Y1

Z1 = 0      0      1  
     0      1      0

### Логическая операция «ИСКЛЮЧАЮЩЕЕ ИЛИ».

Логическая операция **ИСКЛЮЧАЮЩЕЕ ИЛИ** выполняет **двоичное сложение без переноса для двух переменных**, каждая из которых может содержать только булевское значение true или false. Варианты результатов операции **ИСКЛЮЧАЮЩЕЕ ИЛИ** представлены в следующей таблице:

X \ Y	0	1
0	0	1
1	1	0

Другими словами результат операции **ИСКЛЮЧАЮЩЕЕ ИЛИ** равен **false** только когда оба операнда X и Y равны **false** или **true**.

В **MATLAB** операция **ИСКЛЮЧАЮЩЕЕ ИЛИ** выполняется только с помощью функции **Z = xor(X,Y)** как для матриц (поэлементное **ИСКЛЮЧАЮЩЕЕ ИЛИ**), так и для скалярных величин.

% **№1-1.** Явное вычисление

```
xor(true,false)
ans = 1
xor(true,true)
ans = 0
xor(false,false)
ans = 0
```

% **№1-2.** Операция **ИСКЛЮЧАЮЩЕЕ ИЛИ** для двух логических  
% матриц Y1 и Y2

```
X1=[1 2 3; 4 5 6];
X2=[1 2 8; 4 6 6];
Y1=(X1==X2)
Y1 = 1      1      0
      1      0      1
Y2=(X1~=X2)
Y2 = 0      0      1
      0      1      0
Z1=xor(Y1, Y2)
Z1 = 1      1      1
      1      1      1
```

## **Некоторые законы для логических операций**

### **Законы поглощения**

**X И (X ИЛИ Y) = X**  
**X ИЛИ (X И Y) = X**

**Штрих Шеффера** – бинарная логическая операция, булева функция над двумя переменными. Введена Генри Шеффером в 1913 г.

Штрих Шеффера, эквивалентен операции **И-НЕ**

**Стрелка Пирса** – бинарная логическая операция, булева функция над двумя переменными. Введена Чарльзом Пирсом в 1880–1881 г.г.

Стрелка Пирса, эквивалентна операции **ИЛИ-НЕ**

### **Законы де Моргана**

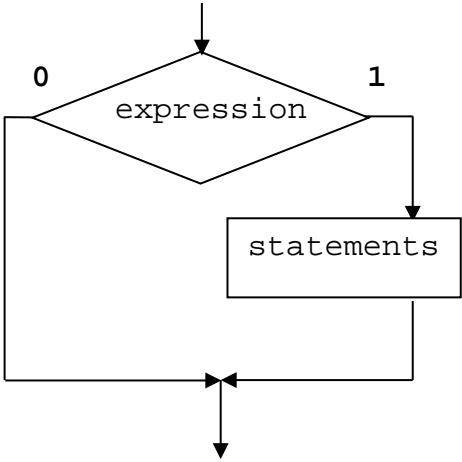
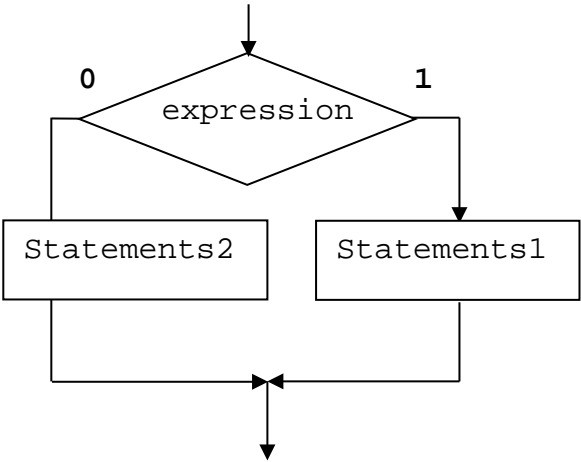
**НЕ(X ИЛИ Y) = НЕ(X) И НЕ(Y)**  
**НЕ(X И Y) = НЕ(X) ИЛИ НЕ(Y)**



## Условные операторы (инструкции)

### Операторы условного ветвления

#### Простой условный оператор

Простой условный оператор с одной ветвью по <b>true</b>	Простой условный оператор с двумя ветвями по <b>true</b> и <b>false</b>
<b>Синтаксис:</b>  if expression statements end	<b>Синтаксис:</b>  if expression statements1 else statements2 end
	

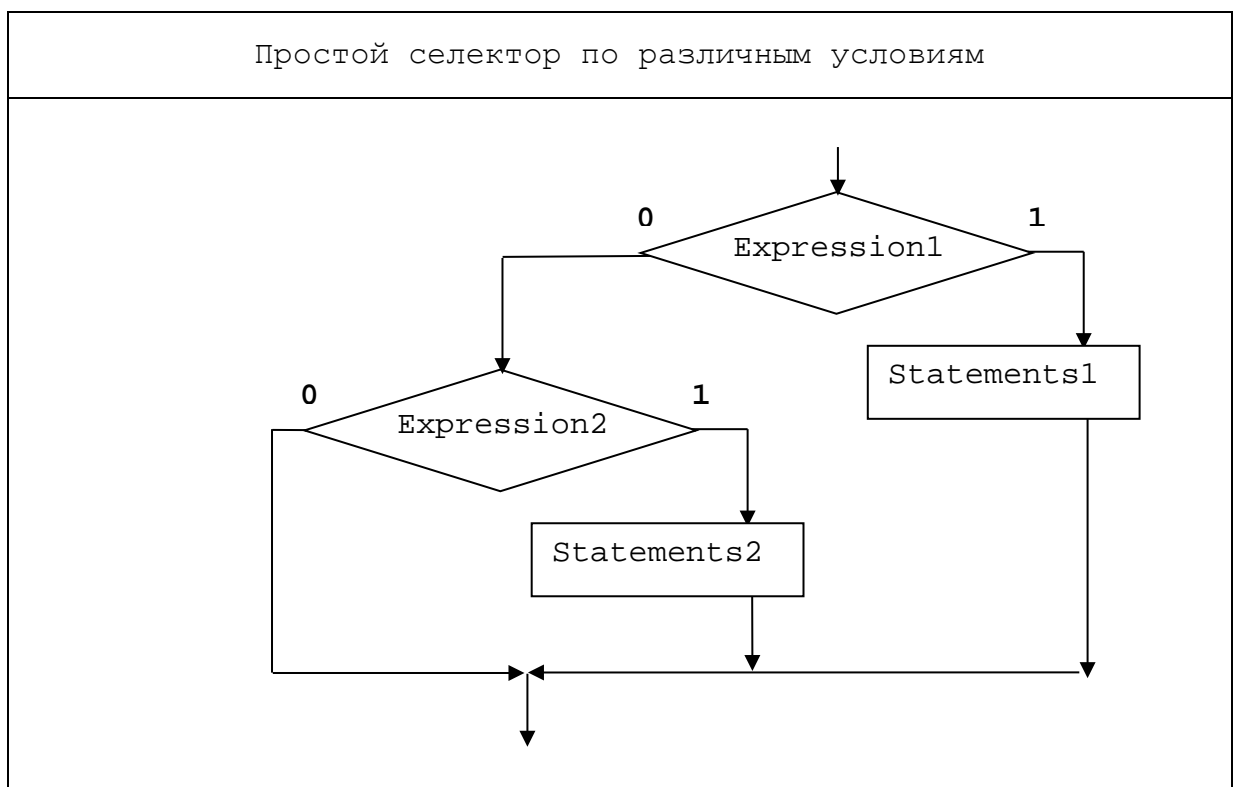
#### Примеры:

```
% №1-1. Условный оператор с одной ветвью по true
x=pi/7;
if sin(x) < cos(x)
    disp('Для заданного x верно sin(x) < cos(x)');
end;
```

```
% №1-2. Условный оператор с ветвями true и false
x=pi/7;
if sin(x) >= cos(x)
    disp('Для заданного x верно sin(x) >= cos(x)');
else
    disp('Для заданного x верно sin(x) < cos(x)');
end;
```

## Селектор по различным условиям

Простой селектор по различным условиям (синтаксис)	Селектор по различным условиям с блоком else (синтаксис)
<pre> if expression1     statements1 elseif expression2     statements2 . . . end         </pre>	<pre> if expression1     statements1 elseif expression2     statements2 . . . else     statements3 end         </pre>

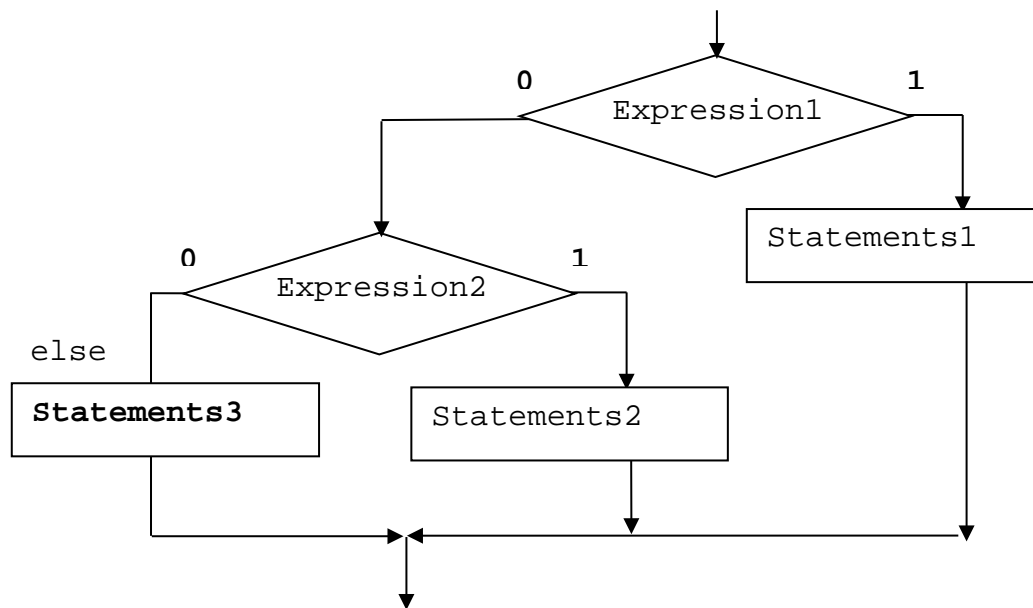


### Примеры:

```

% №2-1. Простой селектор по различным условиям
x1='A';
x2='B';
if (x1 == 'A') & (x2 == 'B')
    disp('Заданы символы A и B');
elseif (x1 ~= 'A') & (x2 == 'B')
    disp('Задан только символ A');
elseif (x1 == 'A') & (x2 ~= 'B')
    disp('Задан только символ B');
end;
    
```

## Селектор по различным условиям с блоком else



**% №2-2.** Селектор по различным условиям с блоком else

```

x1='C';
x2='D';
if (x1 == 'A') & (x2 == 'B')
    disp('Заданы символы A и B');
elseif (x1 ~= 'A') & (x2 == 'B')
    disp('Задан только символ A');
elseif (x1 == 'A') & (x2 ~= 'B')
    disp('Задан только символ B');
else
    disp('Символы A и B НЕ заданы');
end;
    
```

Пример с if	Пример с if и elseif
<pre> if A     x = a else     if B         x = b     else         if C             x = c         else             x = d         end     end end end             </pre>	<pre> if A     x = a elseif B     x = b elseif C     x = c else     x = d end             </pre>

## Селектор по различным значениям

### селектор по различным значениям (синтаксис)

```
switch switch_expr
  case case_expr
    statement,...,statement
  case {case_expr1,case_expr2,case_expr3,...}
    statement,...,statement
  ...
  otherwise
    statement,...,statement
end
```

#### Примеры:

% **№3-1.** селектор по различным значениям строк

```
str1 = 'Bilinear';
switch lower(str1)
  case {'linear','bilinear'}
    disp('Заявлено имя linear методов');
  case 'cubic'
    disp('Заявлено имя cubic метода');
  case 'nearest'
    disp('Заявлено имя nearest метода');
  otherwise
    disp(' Заявлено имя Unknown метода.');
```

end;

% **№3-2.** селектор по различным значениям диапазонов

```
x1 = 12;
switch x1
  case 1
    disp('Обнаружено x1 = 1');
  case {2,3,4,5,6,7,8}
    disp('Обнаружено x1 в диапазоне [2:1:8]');
  case {9,10,11,12,13,14,15,16}
    disp('Обнаружено x1 в диапазоне [9:1:16]');
  otherwise
    disp(' x1 ВНЕ областей обнаружения.');
```

end;

## ЦИКЛ FOR

### Синтаксис

```
for variable = expression
    statements
end
```

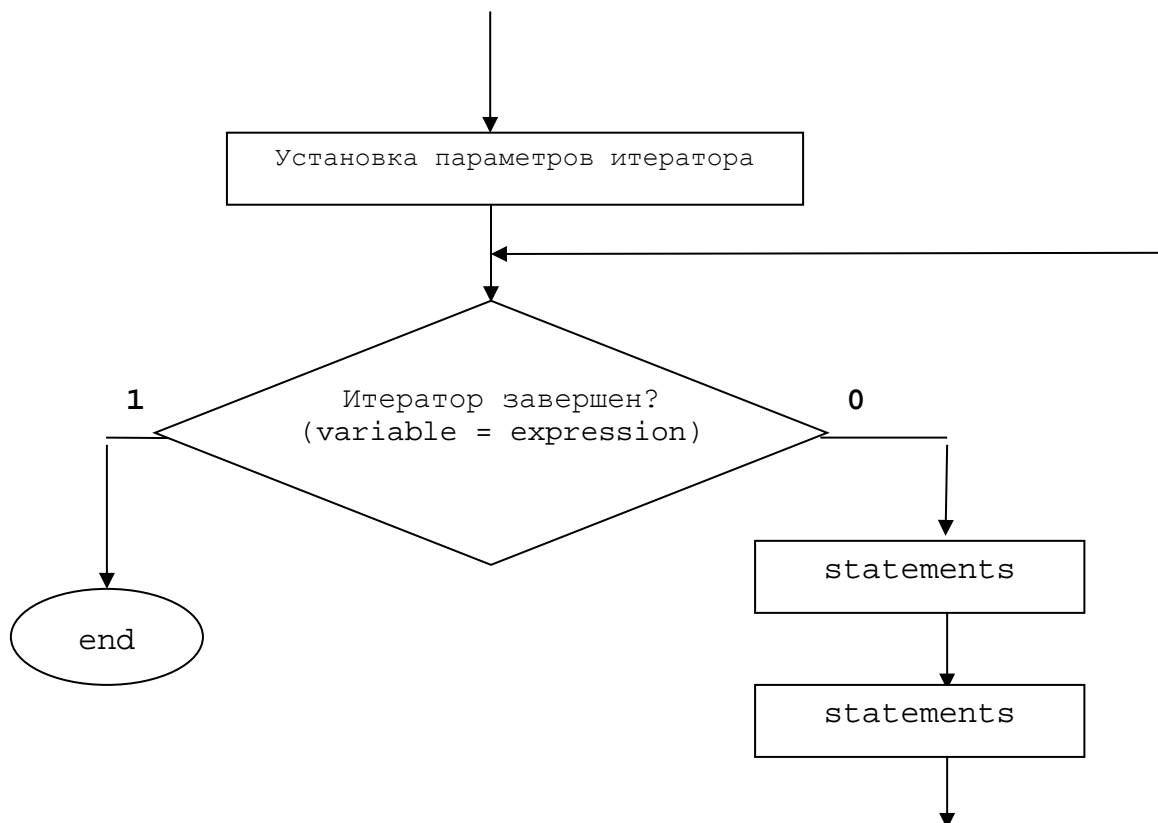
### Общий формат

```
for variable = expression
    statement
    ...
    statement
end
```

Цикл предназначен повторять инструкции (statements) определенное количество раз. Для этого в заголовке цикла используется конструкция, которая в современных языках программирования обобщена термином - итератор:

```
variable = expression
```

В MATLAB эта конструкция определяет переменную (variable), которая может использоваться в теле цикла, а также определяет собой количество повторений тела цикла.



### Описание

Столбцы выражения (expression), даже если они состоят всего из одного элемента, сохраняются по одному в переменной (variable) и доступны для каждой инструкции (statements) в теле цикла. Заметим, что на практике выражение (expression) почти всегда имеет вид scalar, и в этом случае его столбцы являются просто скалярами. Область действия оператора for всегда заканчивается соответствующим end.

### Примеры

Пример (1)

```
% Вложенные циклы с единичным шагом. (Матрица Гильберта):  
k = 8;  
a = zeros(k,k); % матрица предварительного выделения  
for m = 1:k  
    for n = 1:k  
        a(m,n) = 1/(m+n -1);  
    end;  
end;  
plot(a);
```

Пример (2)

```
% Цикл for с заданным диапазоном 1.0 .. 0.0 и шагом -0.1  
a=[];  
ind = 1;  
for S = 1.0: -0.1: 0.0  
    a(ind) = S;  
    ind = ind + 1;  
end  
plot(a);
```

Пример (3)

```
% Последовательная выборка в тело цикла элементов из вектора строки  
B = [3, 4, 5];  
for S = B  
    S  
end;
```

Пример (4)

```
% Выборка в тело цикла вектора столбца (полностью)  
B = [3; 4; 5];  
for S = B  
    S  
end;
```

Пример (5)

```
% Выборка в тело цикла очередного вектора столбца из  
двухмерной матрицы  
A=[[1,2,3]; [4,5,6]; [7,8,9]]  
for S = A  
    S  
end;
```

% Эквивалентная форма примера (5)

```
A=[[1,2,3]; [4,5,6]; [7,8,9]]
```

```
[d1,d2]=size(A)
```

```
for k = 1:d2
```

```
    V = A(:,k)
```

```
end
```

% Эквивалентная форма примера (5)

```
A=[[1,2,3]; [4,5,6]; [7,8,9]]
```

```
[d1,d2]=size(A)
```

```
for k = 1:d2, V = A(:,k)
```

```
    V
```

```
end
```

Пример (6)

% Последовательная установка значений sqrt(m) вместо единиц в  
единичных n-векторах:

```
n=5;
```

```
m = 2;
```

```
for B = eye (n)
```

```
    B.*sqrt(m)
```

```
end;
```

Пример (7)

```
A=rand(2,2,2)
```

```
for S = A
```

```
    S
```

```
end;
```

**Смотрите также**

**end, while, break, continue, return, if, switch**

## ***ЦИКЛ FOR. ПРИМЕРЫ***

**% 1. Создание и инициализация вектора строки**

**% 1.1 Вариант 1**

```
>> for m=1:10
```

```
    x (m)=m;
```

```
end
```

**% 1.2 Вариант 2**

```
>> for m=1:10
```

```
    x (1,m)=m;
```

```
end
```

**% 2. Создание и инициализация вектора столбца**

```
>> for m=1:10
```

```
    y (m,1)=m;
```

```
end
```

**% 3. Инициализация вектора строки**

```
>> x=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
```

```
>> for m=1:length(x)
```

```
    x(m)=m;
```

```
end
```

**% 4. Суммирование элементов вектора строки**

```
>>sm = 0; ...  
for m=1:length(x)  
sm=sm+x(m);  
end
```

**% 5. Суммирование элементов вектора столбца**

**% 5.1. Вариант 1**

```
>> y=x'; sm = 0; ...  
for m=1:length(y)  
sm=sm+y(m);  
end
```

**% 5.2. Вариант 2**

```
y=x'; sm = 0; ...  
for m=1:length(y)  
sm=sm+y(m,1);  
end
```

**% 6. help input  
help isempty**

**% 7. Создание и инициализация двухмерного массива**

```
r = input('Input ROW count: '); ...  
if isempty(r)  
r = 10;  
end; ...  
c = input('Input COL count: '); ...  
if isempty(c)  
c = 10;  
end; ...  
for n=1:r  
for m=1:c  
M1(n,m) = 0;  
end  
end
```

**% 8. help rand  
help size**

**% 9. Создание матрицы случайных чисел**

```
Y=rand(5,8)
```

**% 10. Поиск максимального элемента в массиве**

```
WM=input('Input array name (example: Y): '); ...  
mx =WM(1,1); ir=1; ic=1;  
for r=1:size(WM,1)  
for c=1:size(WM,2)  
if WM(r,c) > mx  
mx =WM(r,c); ir=r; ic=c;  
end  
end  
end;  
r, c, mx % Отобразить результаты
```

**% 11. Поиск максимального элемента в массиве с входным контролем**

```
WM=input('Input array name (example: Y): '); ...
```



```

if isempty(WM) || (size(WM,2)< 1)
    disp('< array is incorrect> : end')
else
    mx=WM(1,1); ir=1; ic=1;
    for r=1:size(WM,1)
        for c=1:size(WM,2)
            if WM(r,c) > mx
                mx =WM(r,c); ir=r; ic=c;
            end
        end
    end;
    r, c, mx % Отобразить результаты
end;

```

## ЦИКЛ WHILE

Выполняет тело цикла пока условное выражение в его заголовке (expression) вычисляет истинное значение

### Синтаксис

```

while expression
    statements
end

```

### Описание

Цикл while повторяет инструкции (statements) своего тела неопределенное количество раз. Инструкции выполняются, если действительная часть условного выражения в его заголовке (expression) имеет все ненулевые элементы, то есть, true (истина).

Простые условные выражения в его заголовке (expression) обычно имеют вид:

(условное выражение) rel\_op (условное выражение)

где rel\_op ::= = | == | < | > | <= | >= | ~ =

Сфера действия оператора while всегда заканчивается соответствующим end.

### Аргументы

#### expression

expression - выражение MATLAB, обычно состоящее из переменных или меньших выражений, соединенных реляционными операторами (например, count < limit) или логическими функциями (например, isreal (A)).

Простые выражения могут быть объединены логическими операторами (&, |, ~) в составные выражения. MATLAB оценивает составные выражения слева направо, придерживаясь правил приоритета оператора.

```
(count < limit) & ((height - offset) >= 0)
```

### statements

statements - это одно или несколько инструкций MATLAB, которые должны выполняться только в том случае, если условное выражение в его заголовке (expression) истинно или отличное от нуля.

### Замечания

#### Нескалярные выражения

Если вычисленное условное выражение в его заголовке (expression) дает не скалярное значение, то каждый элемент этого значения должен быть истинным или ненулевым, чтобы все выражение считалось истинным. Например, оператор, тогда как (A < B) является истинным, только если каждый элемент матрицы A меньше его соответствующего элемента в матрице B. См. Пример 2 ниже.

### Примеры

Пример 1 - Простая инструкция

Переменная eps - это толерантность, используемая для определения таких вещей, как близость сингулярности и ранга. Его вычисленное значение является epsilon машины, или расстоянием от 1.0 до следующего по величине числа с плавающей запятой на вашем компьютере.

```
eps = 1;  
while (1+eps) > 1  
    eps = eps/2;  
end  
eps = eps*2
```

Пример 2 - Нескалярные матрицы A и B

A =		B =	
1	0	1	1
2	3	3	4

Выражение оценивается как:

Expression	Вычислено	Пояснение
A < B	False	Если A (1,1) больше B (1,1), то далее уже можно не вычислять.
A < (B + 1)	True	Каждый элемент A меньше, чем тот же элемент B с добавленным 1.
A & B	False	A (1,2) & B (1,2) равно false.
B < 5	True	Каждый элемент B меньше 5.

**Смотрите также:**

**end, for, break, continue, return, all, any, if, switch**

### **ЦИКЛ *While*. Примеры**

```
% Пример №1
% Вычисление суммы элементов вектора строки
vs = rand(1,8)
k=1;
y=0;
while k <= length(vs)
    y=y+vs(k);
    k=k+1;
end;
y

% Пример №2
% Вычисление интеграла функции f, заданной с равномерным
% шагом h по оси X методом трапеций
f = [0 1 2 3 4 5]
h = 1
k=1;
y=0;
while k <= length(f)-1
    y=y+(f(k)+f(k+1))/2;
    k=k+1;
end;
y=y*h

% Пример №3
% Вычисление основания натурального логарифма (e)
x=1;
y=0;
delta = 1;
k=1;
p=1;
f=1;
while (abs(delta) > 1e-10) && (k < 32)
    delta = p/f;
    y=y+ delta;
    f=f*k;
    p=p*x;
    k=k+1;
end;
% Результат и значение ошибки
y
abserror=y-exp(1)
```

1. Выполните все примеры из теоретической части лабораторной работы.

---

Финальная редакция материала 23.08.2018г.  
Воронов С.И.