Условный оператор

Одним из основных операторов реализующим ветвление в большинстве языков программирования является условный оператор `if`. Существует обычная и расширенная формы оператора `if` *Scilab*. в Обычный `if` имеет вид:

```
if условие
    операторы_1
else
    операторы_2
end
```

Здесь `условие` - логическое выражение; `операторы_1`, `операторы_2` - операторы языка *Scilab* или встроенные функции. Оператор `if` работает по следующем алгоритму:

- если условие истинно то выполняются , `операторы_1`
- если ложно - `операторы_2`

В *Scilab* для построения логических выражений могут использоваться условные *Scilab* операторы:

- `&` или `and` (логическое и)
- `|` или `or` (логическое или)
- `~` или `not` (логическое отрицание)
- операторы отношения:
  - `<` - меньше
  - `>` - больше
  - `>=` - больше или равно
  - `<=` - меньше или равно
  - `/==` - равно
  - `<>` - не равно

Зачастую при решении практических задач недостаточно выбора выполнения или невыполнения одного условия. В этом случае нужно воспользоваться расширенной формой оператора `if`:

```
if условие_1
    операторы_1
elseif условие_2
    операторы_2
elseif условие_3
```

```
    операторы_3
...
elseif условие_n
    операторы_n
else
    операторы
end
```

Пример:

```
a = 5;

if a > 0:
    disp(">");
elseif a < 0:
    disp("<");
else
    disp("=");
end
```

Цикл

Для создания циклов используется оператор `for`:

```
for x = xn:hx:xk
    операторы
end
```

Здесь `x` - имя скалярной переменной параметра цикла, `xn` - начальное значение параметра цикла, `xk` - конечное значение параметра цикла, `hx` - шаг цикла. Если шаг цикла равен 1, то `hx` можно опустить, и в этом случае оператор `for` будет таким:

```
for x = xn:xk
    операторы
end
```

Выполнение цикла начинается с присвоения параметру стартового значения `x = xn`. Затем следует проверка: не превосходит ли параметр конечное значение `x > xk`. Если `x > xk`, то цикл считается завершенным и управление передается следующему за телом цикла оператору. Если же `x <= xk`, то выполняются операторы в теле цикла.

Далее параметр цикла увеличивает свое значение на `hx` (`x = x + hx`). После чего снова производится проверка значения параметра цикла и алгоритм повторяется.

Пример:

```
vector = [0 1 2 3]
vector_size = size(vector)
vector_size = vector_size(2)

for i = 1:vector_size
    disp(vector(i))    // вывод элементов вектора
end
```

Функции

Функции - это специальные блоки кода, которые выполняют определенные задачи. Они помогают сделать программу более структурированной и позволяют многократно использовать один и тот же код в разных частях программы. Функция в *Scilab* определяется следующим образом:

```
function [output1, output2, ..., outputN] = functionName(input1, input2,
..., inputM)
    // Тело функции
    // Операции с входными параметрами для получения выходных значений
    output1 = ...;
    output2 = ...;
    ...
    outputN = ...;
endfunction
```

Здесь:

- `functionName` - имя функции
- `input1, input2, ..., inputM` - входные параметры функции
- `output1, output2, ..., outputN` - значения, которые возращает функция
- тело функции содержит операторы для работы с входными данными и вычисления результата

Вызов функции выполняется следующим образом:

```
[output1, output2, ..., outputN] = functionName(nput1, input2, ..., inputM)
```

Пример:

```
function y = line_func(k, b, x)
    y = k * x + b;
endfunction

b = 3;
y = line_func(5, b, 10)
```