Selection Statements

by Ahmet Sacan

selection statements, branching statements, condition, relational expression, Boolean expression, logical expression, relational operators, logical operators, truth table, action, temporary variable, error-checking, nesting statements, cascading if-else, "is" functions

Definitions

- Selection (aka Conditional or Branching) Statements: statements that conditionally execute a block of code.
  - **if** somethingAistrue, dosomethingX; **end**
  - **if** somethingAistrue, dosomethingX; **else** dosomethingY; end
  - **if** somethingAistrue, dosomethingX; **else if** anotherthingBistrue, dosomething Y; **else** dosomethingZ; **end**
- Conditions in if statements are relational (aka Boolean or logical) expression that evaluate to either true or false.

Examples

- if true; fprintf('this always prints\n'); end
- if 5; fprintf('this always prints\n'); end
- a=3; if a+1; fprintf('this always prints\n'); end

- if false; fprintf('this never prints\n'); end
- if 0; fprintf('this never prints\n'); end
- a=3; if a-3; fprintf('this never prints\n'); end

Relational Operators

- Relational operators take two operands and return true/false.

Operator	Meaning
>	greater than
<	less than
>=	greater than or equals
<=	less than or equals
==	equality
~=	inequality

Examples

- 3 < 5

ans = 1 ← true

- 2 > 9

ans = 0 ← false

- 'a' < 'c'

ans = 1

- a = 3<5; a+10

ans = 11

Logical Operators

- Logical Operators take one/two logical operands and return a true/false

```
3<5 && 15>10   %true
3<5 && 15<10   %false
~(3<5)         %false
```

Operator	Meaning
||	or (for scalars)
&&	and (for scalars)
~	not

- Additionally xor(x,y) function returns true if one and only one of its arguments is true.

```
xor(3<5, 'a'<'c')   %false
xor(3<5, false)     %true
```

Truth Tables

- Truth Table lists the output of a function/operator for all values of its inputs.

Table 3.1 Truth Table for Logical Operators

x	y	~x	x || y	x && y	xor(x,y)
true	true	false	true	true	false
true	false	false	true	false	true
false	false	true	false	false	false

Operator Precedence Rules

- Arithmetic, Relational, Logical

Table 3.2 Operator Precedence Rules

Operators	Precedence
parentheses: ( )	highest
transpose and power ', ^	
unary: negation (–), not (~)	
multiplication, division *,/,\	
addition, subtraction +, –	
colon operator :	
relational <, <=, >, >=, ==, ~=	
and &&	
or ||	
assignment =	lowest

Exercise 3.1

- 4>3 +1
- 'e'=='d' +1
- 3 < 9 - 2
- (3 < 9) - 2
- 4 == 3 + 1 && 'd' > 'c'
- 3 >= 2 || 'x' == 'y'
- xor(3 >= 2, 'x' == 'y')
- xor(3 >= 2, 'x' ~= 'y')
- x=0; 3 < x < 5
- choice='n'; choice == 'y' || 'Y'

if statement

```
if condition
    action
end
```

- condition is any expression that can be evaluated to true/false.
- action is any number of statements that get executed if and only if the condition evaluates to true.

Examples

```
if num < 0
        num = num + 1;
end
```

```
num = -5
if num < 0
        num = num + 1;
end
disp(num)
```

```
num = 5
if num < 0
        num = num + 1;
end
disp(num)
```

Exercise

- Ask the user whether we should greet him/her or not, if the user enters one of 'y' or 'Y' then print 'Hello'. Otherwise, don't print anything.

- Note the common pitfall:
  - choice == 'y' || 'Y'

Exercise

- Write a matlab function **divisibility.m** that will take an integer number, and display 'yes' if this number is divisible by 2 and separately by 3; 'no' otherwise.
- Sample Run:

```
>> divisibility( 16 )
divisible by 2: yes
divisible by 3: no
```

Exercise

- Write a function sqrtif.m that takes a number x, and
  - returns its square root if x is positive.
  - if x is negative, notifies the user that taking square root of negative numbers is not allowed and returns NaN.

if-else statement

```
if condition
    action1
else
    action2
end
```

- if condition is true, action1 is performed. Otherwise, action2 is performed.

Exercise

- Write a function sqrtif.m that takes a number x, and
  - returns its square root if x is positive.
  - if x is negative, notifies the user that taking square root of negative numbers is not allowed and returns NaN.

Nested if-else statements

- Example: calculate y using the following:

$$y = \begin{cases} 1 & if\ x < -1 \\ x^2 & if\ -1 \le x \le 2 \\ 4 & if\ x > 2 \end{cases}$$

```
if x < -1
   y=1;
end
if x >= -1 && x <= 2
   y = x^2;
end
if x > 2
   y = 4;
end
```

```
if x < -1
   y=1;
else
   if x <= 2
      y = x^2;
   else
      y = 4;
   end
end
```

```
if x < -1
   y=1;
elseif x <= 2
   y = x^2;
else
   y = 4;
end
```

Exercise

- Write a function getargtype(x) that return whether x is a "scalar", "vector", or "matrix".

Exercise

- Write a function getdayname(x) that takes an integer x from 1 to 7, and returns the corresponding day name.
- Sample Run:

```
>> getdayname(3)
ans =
Tuesday
```

switch statement

```
switch switchexpression
        case caseexpression1
                action1
        case caseexpression2
                action2
        …
        otherwise
                defaultaction
end
```

Exercise

- Write a function printnumber(n) that prints 'good' if n is 0, 1, or 2; prints 'not good' if n is 3 or 4, prints nothing if n is 5, and prints 'cannot decide' otherwise.

```
switch(n)
  case 0
    disp('good')
  case 1
    disp('good')
  case 2
    disp('good')
  case 3
    disp('not good')
  case 4
    disp('not good')
  case 5
  otherwise
    disp('cannot decide')
  end
```

```
switch(n)
  case { 0, 1, 2 }
    disp('good')
  case { 3, 4 }
    disp('not good')
  case 5
  otherwise
    disp('cannot decide')
  end
```

Exercise

- Write a function getdaynameswitch(x) that takes an integer x from 1 to 7, and returns the corresponding day name.
- Sample Run:

```
>> getdaynameswitch(3)
ans =
Tuesday
```

"is" functions

- "is" functions return true if the argument is of a given type.

```
>> isletter('h')
ans =
1
>> isletter('hi !# hi !')
ans =
    1     1     0     0     0     0     1     1     0     0

>> ischar('hello')
ans =
1
>> ischar([4 5 6])
ans =
0
```

Exercise

- Assume x is a scalar value. Write a function myisletter(x) that returns true if x is a letter, and false otherwise. If x is not a character, return NaN. Do not use the built-in isletter function.

Common Pitfalls

- Using = instead of == to test for equality
- Not using quotes when comparing a string variable to a string, such as:

  choice == y
- Not spelling out an entire logical expression:

  if radius || height <= 0; disp('Need positive values'); end
- Unnecessarily testing if something is 0 or 1:
  - if (x < 5) == 1
  - if (x < 5)
- Unnecessary elseif statements

  if x==5; action1

  else~~if x~=5~~; action2; end