

ÖGOR Summer-Workshop for PhD-candidates and Post-Docs

An introduction to Julia and JuMP for Operations Research

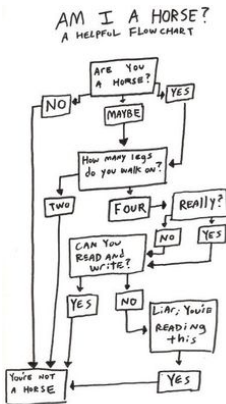
Prof. Dr. Xavier Gandibleux

Nantes Université – France
Département Informatique – Faculté des Sciences et Technique

Topic 4

Control flow

The conditionals



if ... endif

Definition:

```
if condition  
    instruction(s)  
end
```

Example:

```
julia> if zipcode == 4020  
    println("Welcome to Linz")  
end
```

if ... endif

Definition:

```
if condition  
    instruction(s)  
end
```

Example:

```
julia> if zipcode == 4020  
    println("Welcome to Linz")  
end
```

if ... else ... endif (1/2)

```
if condition
    instruction(s) 1
else
    instruction(s) 2
end
```

```
julia> if zipcode == 4020
    println("Welcome to Linz")
else
    println("Welcome to Austria")
end
```

if ... else ... endif (2/2)

ifelse instruction:

```
ifelse( condition, case_true, case_false )
```

```
julia> println("Welcome to ",  
              ifelse(zipcode == 4020, "Linz", "Austria")  
              )
```

Ternary operator:

```
condition ? case_true : case_false
```

```
julia> println("Welcome to ",  
              zipcode == 4020 ? "Linz" : "Austria")  
              )
```

if ... else ... endif (2/2)

ifelse instruction:

```
ifelse( condition, case_true, case_false )
```

```
julia> println("Welcome to ",  
              ifelse(zipcode == 4020, "Linz", "Austria")  
              )
```

Ternary operator:

```
condition ? case_true : case_false
```

```
julia> println("Welcome to ",  
              zipcode == 4020 ? "Linz" : "Austria")  
              )
```

if ... elseif ... [else ...] endif

```
if condition 1
    instruction(s) 1
elseif condition 2
    instruction(s) 2
else
    instruction(s) n+1
end
```

```
julia> if zipcode == 4020
    println("Welcome to Linz Central area")
elseif zipcode == 4030
    println("Welcome to Linz South area")
elseif zipcode == 4040
    println("Welcome to Linz North area")
else
    println("Welcome to Austria")
end
```


Logical operators

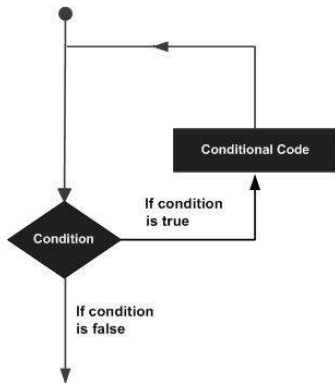
With `a` and `b`, two logical conditions:

<i>Operator</i>	<i>Expression</i>	<i>Signification</i>
<code>!</code>	<code>!a</code>	NOT <code>a</code>
<code>&&</code>	<code>a && b</code>	<code>a</code> AND (then) <code>b</code>
<code> </code>	<code>a b</code>	<code>a</code> OR (else) <code>b</code>

Example:

```
julia> countrycode == "AT" && zipcode == 4020
```

Control flow Loops



while ... endWhile

Definition:

```
while condition  
    instruction(s)  
end
```

Example:

```
julia> zipcode = 4020  
while zipcode <= 4040  
    print(zipcode, " ")  
    zipcode = zipcode + 10  
end
```

while ... endWhile

Definition:

```
while condition  
    instruction(s)  
end
```

Example:

```
julia> zipcode = 4020  
while zipcode <= 4040  
    print(zipcode, " ")  
    zipcode = zipcode + 10  
end
```

for ... endFor (1/3)

Definition:

```
for variable in collection  
    instruction(s)  
end
```

Also:

```
for variable = collection  
for variable ∈ collection
```

Collection:

- ▶ range: *start:stop* or *start:step:stop*
- ▶ string: "*characters*"
- ▶ tuple: (*val*₁, *val*₂, ..., *val*_{*n*})
- ▶ array: [*val*₁, *val*₂, ..., *val*_{*n*}]
- ▶ set: Set([*val*₁, *val*₂, ..., *val*_{*n*}])
- ▶ dict: Dict(*key*₁=>*val*₁, *key*₂=>*val*₂, ..., *key*_{*n*}=>*val*_{*n*})

for ... endFor (1/3)

Definition:

```
for variable in collection  
    instruction(s)  
end
```

Also:

```
for variable = collection  
for variable ∈ collection
```

Collection:

- ▶ range: *start:stop* or *start:step:stop*
- ▶ string: "*characters*"
- ▶ tuple: (*val*₁, *val*₂, ..., *val*_{*n*})
- ▶ array: [*val*₁, *val*₂, ..., *val*_{*n*}]
- ▶ set: Set([*val*₁, *val*₂, ..., *val*_{*n*}])
- ▶ dict: Dict(*key*₁=>*val*₁, *key*₂=>*val*₂, ..., *key*_{*n*}=>*val*_{*n*})

for ... endFor (1/3)

Definition:

```
for variable in collection  
    instruction(s)  
end
```

Also:

```
for variable = collection  
for variable ∈ collection
```

Collection:

- ▶ range: *start:stop* or *start:step:stop*
- ▶ string: "*characters*"
- ▶ tuple: (*val*₁, *val*₂, ..., *val*_{*n*})
- ▶ array: [*val*₁, *val*₂, ..., *val*_{*n*}]
- ▶ set: Set([*val*₁, *val*₂, ..., *val*_{*n*}])
- ▶ dict: Dict(*key*₁=>*val*₁, *key*₂=>*val*₂, ..., *key*_{*n*}=>*val*_{*n*})

for ... endFor (1/3)

Definition:

```
for variable in collection  
    instruction(s)  
end
```

Also:

```
for variable = collection  
for variable ∈ collection
```

Collection:

- ▶ range: *start:stop* or *start:step:stop*
- ▶ string: "*characters*"
- ▶ tuple: (*val*₁, *val*₂, ..., *val*_{*n*})
- ▶ array: [*val*₁, *val*₂, ..., *val*_{*n*}]
- ▶ set: Set([*val*₁, *val*₂, ..., *val*_{*n*}])
- ▶ dict: Dict(*key*₁=>*val*₁, *key*₂=>*val*₂, ..., *key*_{*n*}=>*val*_{*n*})

for ... endFor (2/3)

Examples:

```
julia> for i in 1:10
        print(i , " ")
      end
```

```
julia> for i in 1:2:10
        print(i , " ")
      end
```

```
julia> for i in "Linz"
        print(i , " ")
      end
```

for ... endFor (3/3)

```
julia> for i in (4020,4030,4040)
        print(i , " ")
    end
```

```
julia> for i in [4020,4030,4040]
        print(i , " ")
    end
```

```
julia> for i in Set([4020,4030,4040])
        print(i , " ")
    end
```

```
julia> for i in Dict("Center"=>4020,"South"=>4030,
                    "North"=>4040)
        print(i , " ")
    end
```

Multiple for ... endFor

```
for var1 in collection1  
    for var2 in collection2  
        instruction(s)  
    end  
end
```

```
for var1 in collection1, var2 in collection2  
    instruction(s)  
end
```

Example:

```
julia> for i in 1:3, j in "hello"  
        println(i, " ", j)  
    end
```

Multiple for ... endFor

```
for var1 in collection1  
    for var2 in collection2  
        instruction(s)  
    end  
end
```

```
for var1 in collection1, var2 in collection2  
    instruction(s)  
end
```

Example:

```
julia> for i in 1:3, j in "hello"  
        println(i, " ", j)  
    end
```

Review and exercises

(notebook)

