Computer Algebra and Technical Computing (MTH1006)

B. Vorselaars
bvorselaars@lincoln.ac.uk

School of Mathematics and Physics, University of Lincoln

Today

- Reminders:
  - Hand-out 1st assignment was last week
  - Deadline hand-in first coursework: 10/11/2023
- Recap
- More flow control constructs.

Recap - 'for' loop

A `for`-loop loops over the elements of a vector. Example:

```
for n=1:10
    n
    pause
end
```

demo

Recap - 'for' loop

`for` loop to sum the numbers 1 to 5 to S, where you add everything one-by-one to the variable `S`

```
S=0; % initialize

for x=1:5
    S=S+x;
end
```

A `for` loop has a fixed number of iterations: the number of elements in the vector

More complex loop

How to add the square of the numbers ranging from 1 to 20?

$$S = \sum_{n=1}^{20} n^2$$

```
S=0; % initialize
for n=1:20
    S=S+n^2;
end
S
```

More complex loop: continue

It is possible to skip elements, using the keyword `continue`

$$S = \sum_{n=1, n\neq 5}^{10} n$$

```
S=0; % initialize
for n=1:10
    if n==5
        continue
    end
    S=S+n;
end
```

- `continue`: implies go to the next iteration at that point.
- Here: because of the `if` statement, if $n = 5$, the number n will *not* be added to the variable S.
- The result is therefore
  $S = 1 + 2 + 3 + 4 + 6 + 7 + 8 + 9 + 10 = 50$.

Alternative code

In this example the same can be achieved using an extra `if` (but at the expense of an extra indentation).

```
for n=1:10
    if n~=5
        S=S+n;
    end
end
```

Nested 'if' statement

`if`'s can be combined:

- Example: If the number is non-complex, apply an operation. If it is complex, display a message.

```
x=sqrt(3);
if  imag(x)==0
   disp('Number is real')
   % apply operation.
else
   disp('Number is complex')
end
```

Similar to an envelope in an envelope. demo

Nested 'if' statement

`if`s can be combined:

- Example. If the number is non-complex, apply an operation: if the number is negative, square it and multiply it by -1, otherwise just square it. If it is complex, display a message.

```
x=sqrt(3);
if  imag(x)==0
   disp('Number is real')
   if x<0
      x=-x^2
   else
      x=x^2
   end
else
   disp('Number is complex')
end
```

Note: the solution has an `if` within an `if` (nested)

Nested 'if' statement

One can have multiple nesting.

```
x=sqrt(3);
if condition1
  % statements
  if condition1.a
    %statements
    if condition 1.a.i
       %statements
    end
  end
else
  % statements
  if condition 1.b
    % statements
  end
end
```

Nested 'for' loops

Simlar with nested `for` loops. Example

$$S = \sum_{n=1}^{2} \sum_{m=1}^{2} n^m$$

```
S=0;
for n=1:2
   for m=1:2
      S=S+n^m
   end
end
```

Nested 'for' loops

```
S=0;
for n=1:2
   for m=1:2
      S=S+n^m
   end
end
```

This is equivalent to

```
S=0;
n=1;
m=1;
S=S+1^1
m=2;
S=S+1^2
n=2;
m=1;
S=S+2^1
m=2;
S=S+2^2
```

Nested functions

Just as with `for` and `if` one can also have nested functions
There are two types

- Nested function *calls*: call another function from a function

```
function y=f(x)
y=g(x)+10;
end
```

with g given by

```
function y=g(x)
y=x^2;
end
```

- Nested function *definitions*: define another function within a function. This implies the local function is only known within that function. We will be mostly concentrating on nested function calls.

Early termination of function

One can jump out of a function earlier by using the command `return` (similar to `break` in a `for` loop).

```
function c=f2c(f)
%
if imag(f)~=0
    disp('This routine does not work with
        complex numbers')
    c=nan;
    return % End the function here if f is
        complex
end

c=(f-32)*5/9;
end % If f is not complex the function ends
    here.
```

if elseif chain

Chain of `if` – `elseif` statements

```
dim=2;
if dim==0
    disp('point')
else if dim==1
    disp('line')
else if dim==2
    disp('plane')
else if dim==3
    disp('space')
else
    disp('Unknown')
end
```

Can this be more efficient?

Switch blocks

For many options: use switch

Whereas if statements test boolean expressions, switch blocks test if a variable or expression is equal to a given value. The general syntax is

```
switch expression  % scalar or string
    case value1
        statements 1 % Executes if expression
            is value1
    case value2
        statements 2 % Executes if expression
            is value2
      .
    otherwise
        statements n % Executes if expression
            does not match any case
end
```

Switch blocks

Repeat previous example

```
switch dim
    case 0
        disp('point')
    case 1
        disp('line')
    case 2
        disp('plane')
    case 3
        disp('space')
    otherwise
        disp('unknown')
end
```

Switch blocks

Example

```
switch x
    case 0
        disp('Number is zero')
    case 13
        disp('That is an unlucky number')
    otherwise
        disp('Number is unequal to 0 or 13')
end
```

Switch blocks with strings

Example

```
switch day_str
    case 'Monday'
        day_num=1;
    case 'Tuesday'
        day_num=2;
        ...
    otherwise
        disp('Unknown day?!')
        day_num=nan;
end
```

Switch blocks with multiple equivalent options

Example

```
switch day_str
    case {'Monday', 'Tuesday', 'Wednesday',
        'Thursday', 'Friday'}
         weekend=false;
    case {'Saturday', 'Sunday'}
         weekend=true;
    otherwise
         disp('Unknown day?!')
         weekend='unknown';
end
```

The multiple options are enclosed in curly brackets: {...}.

For loop

`for` loop to sum the numbers 1 to 5 to S, where you add everything one-by-one to the variable `S`

```
S=0; % initialize

for x=1:5
    S=S+x;
end
```

A `for` loop has a fixed number of iterations: the number of elements in the vector

Variable number of iterations

- How to loop a number of times, where the number is not known a priori?
- Why to have such a loop? Example: user has to choose the right option. A priori unknown how many times it takes for the user to do this.

While loop

A while loop exectutes statements while a certain condition remains satisfied. The general syntax is

```
while condition
  action(s)
end
```

Question dialog

How to ask the user for a certain option?

- `result=questdlg('Question text')` Will ask for Yes, No or Cancel. The variable `result` will contain one of these strings.
- `res=questdlg('Question text', 'Question title', 'option 1', 'option 2', 'option 3', 'option 2')` This will present three options, where the default option is 'option 2'.

demo

While loop examples

- Enter correct option

```
result=questdlg('Is it the case that
    x^2=-1 for x=-i?');
while ~strcmp(result, 'Yes')  % 'strcmp'
    compares the string, true for identical
     result=questdlg('This is wrong, try
         again')
end
disp('Answer is correct.')
```

While loop examples

Add numbers 1 till 5 together

- `for` loop

```
S=0;
for x=1:5
    S=S+x;
end
```

- `while` loop

```
S=0;
x=1;
while x<=5
    S=S+x;
    x=x+1;
end
```

demo

While loop: extra constructs

Similar to a `for` loop:

- Use `continue` to go to the next iteration.
- Use `break` to end the loop prematurely

While loop with 'continue' example

- `while` loop adding numbers 1 till 5 together

```
S=0; x=0; % initialise
while x<5
    x=x+1;
    S=S+x;
end
```

- `while` loop adding numbers 1 till 5 together, skipping 3

```
S=0; x=0; % initialise
while x<5
    x=x+1;
    if x==3
        continue % this will jump to the
            'while x<5' statement
    end
    S=S+x;
end
```

While loop with 'break' example

- `while` loop adding numbers 1 till 5 together

```
S=0; x=0; % initialise
while x<5
    x=x+1;
    S=S+x;
end
```

- `while` loop adding numbers 1 till 5 together, but stop when the sum becomes larger than 8

```
S=0; x=0; % initialise
while x<5
    x=x+1;
    S=S+x;
    if S>8
        break % this stops the loop
    end
end
```