Part III: More Flow Control

1. switch
2. while

SWITCH:
a more convenient if

If you know in advance SEVERAL OUTCOMES ARE POSSIBLE, use SWITCH.

For example: ask user to choose a number from 1 to 10, each number prints an associated joke.

SWITCH: syntax

Something like:

choice = input(‘number from 1 to 10? ‘)

switch choice

  case 1 …

  case 2 …

…

 otherwise …

end.

Choice is the name of the variable that can take the values listed after the “case” word.

SWITCH: syntax

More generally, if testing the variable X that can take values {x1,x2,x3 or x4}

```
switch X
    case x1
     …
    case x2
          …
    otherwise
          …
end
```

{x1, x2, …} is the list of possible values that X can take}

SWITCH example :
edit MHarmony.m

```
myage='34';
status='single';
avail='immediate';
info=input('what information do you want to know?  Enter 1 for age, 2 for status, 3 for
    availability or 4 to quit.')
switch info
        case 1
            txt=['The age is ' num2str(myage)];     %help num2str
                                %  vectors within brackets are
                                %  concatenated too.

        case 2
            txt=['The status is ' status];
        case 3
            txt=['The availability is ' avail];
otherwise
            txt=['Quitting so soon?'];
end
txt
```

Switch exercise

"Guess the mood of the user (Happy, Surprised, Sad, Angry, Depressed, Overworked)"

Ok, only first four options.

Switch exercise

Mood.m

```
%here are your comments for help
name = input('What''s your name? ');
    %Note double single quotes…
    %if you don't specify 's', you can still input
    %a string with the ' command

randmood = floor(rand * 4) +1;   %rand is between 0 and 1
switch randmood
    case 1
        message = [name ' is happy!']
    case 2
        message = [name ' is surprised!']
    case 3
        message = [name ' is sad.']
    case 4
        message = [name ' is angry!']
end;
```

WHILE

A loop for an a priori unknown number of iterations.

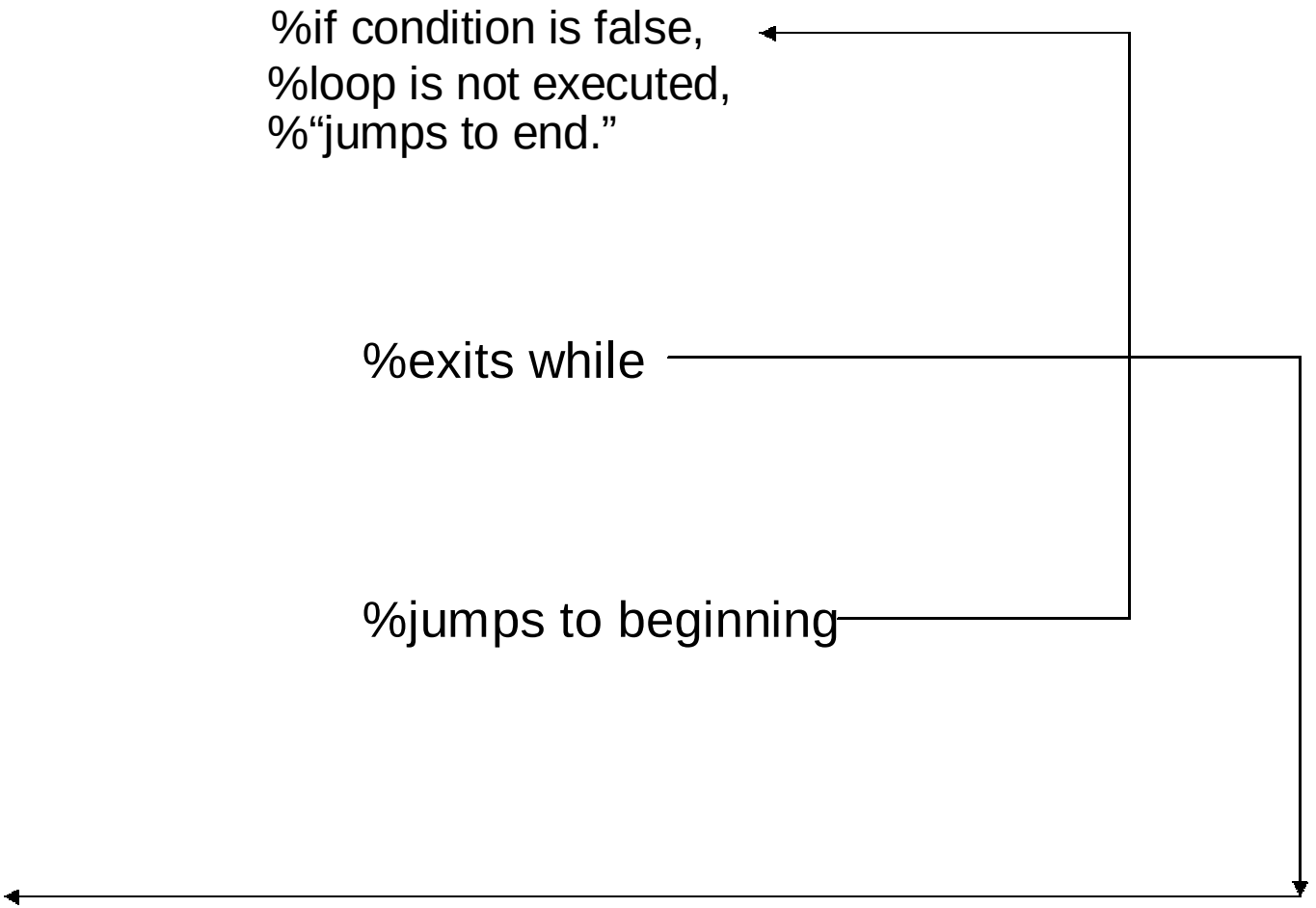
WHILE (condition is satisfied)

keep doing this.

End

WHILE: syntax

```
while (condition)          %if condition is false,
                           %loop is not executed,
                           %"jumps to end."
   ...
   if (condition)
      break                %exits while
   end
   ....
   if (condition)
      continue             %jumps to beginning
   end
   ...
end
more code
```



Exercise

Write a program that asks user for PIN number, until user gets it right.

Exercise

```
pin=0;
While (pin~=1234)
        pin=input('What''s your pin number? ');
end;
```

Exercise

##Modify script to end if user has three bad #entries.

```
pin=0;
counter=0;
while (pin~=1234)
    pin=input('What''s your pin number? ');
    counter=counter+1;
    if (counter==3)
            break
    end
end;
```

Exercise

Write a program that asks user for a password, until user gets it right.

use “strcmp”

strcmp: string comparison

Strings are not Matrices, nor are they mathematical expressions so we cannot do:

if (name==‘alejo’)

we use strcmp to compare to matrices.

strcmp(name,’alejo’)

which returns 1 if true and zero if false.

pwdverify.m

```
pwd='secret';
attempt=input('what''s the pwd?','s');
while ~strcmp(pwd,attempt)
   clear attempt;
   attempt=input('invalid pwd, please try again.
   ','s');
end;
```

the "find" command

A very useful command!

It can save you some time wasted on for loops…

Allows you to efficiently look for a value inside a matrix.
(or scan the inside)

For instance, you want to find all the RTs in your data file that are below 150 ms

find

FINDS a specific value in an array (or matrix) and returns the indeces in which it is located

SYNTAX:

indeces = find(expression);

returns indeces for which expression is true.

the "find" command

RT might be a 10000 rows variable with RT values.
Let's imagine:
> RT = rand(20,1).*500; %20 values between 0 and 500
then
> badRTIndeces = find(RT <150)

and

> badRTs=RT(find(RT < 150)); % or badRTs=RT(badRTIndeces)

How would you change this code to find RTs<150 and RTs>400?
find((RT < 150) | (RT > 400))

the "find" command

How would you change this code to find RTs<150 and RTs>400?

1 . find((RT < 150) | (RT > 400))
2. find((RT < 150) & (RT > 400))

% | is logical OR
% & is logical AND

the “isempty" command

When FIND does not find ANY indeces, it returns an empty matrix.

TRY:

```
A=[1 2 3;1 4 8];
B=find(A==1)
C=find(A==1000)
```

ISEMPTY(X) returns TRUE if the matrix X is empty. TRY:

```
if isempty(C)
    ‘it worked’
else
    ‘im screwed’
end
```

Exercise

Write a program that

(a) finds all RTs smaller than 150 and larger than 1000

(b) calculates the average RT of the “valid” RTs

(c) reports the proportion of rejected RTs

For the data, create the following vector:

data=rand(1000,1) .* 1100;

Exercise

With a for loop:

```
sz=size(data);
counter=0;
mean=0;
for i=1:sz(1)
        if data(i) > 150
            if data (i)<1000
                        mean=mean+data(i);
            else
                        counter=counter+1;
           end;
        else
            counter=counter+1;
        end
end;
mean=mean/(sz(1)-counter)
rejected=counter/sz(1)
```

Exercise

With FIND:

```
clear C;
data=rand(1000,1) .* 1100;

B=find((data>150) & (data<1000));

rejected=1 - length(B)./length(data)
C=sum(data(B))./length(B)
```

Be careful of undefined cases!!
Possible division by 0!!

Exercise

With FIND:

```
clear C;
data=rand(1000,1) .* 1100;

B=find((data>150) & (data<1000));

if ~isempty(B)
        rejected=1 - length(B)./length(data)
        C=sum(data(B))./length(B)
else
        rejected=1
        mesge=['no valid RTs found. Impossible to calculate mean.']
end;
```

Try at Home

Write a program that asks a user for a new password which must abide by the following rules:

1. Length must be at least 6 characters.
2. Must have at least one digit.
3. Must have at least one Upper case and one Lower case letters.

If user enters valid password, say "Ok, password valid"

If user enters a password without a digit, say "you forgot to include at least one digit"

If passwords does not have at least one Upper and one lower case letter, say "you forgot to include at least one upper case and one lower case letter“

Repeat until a valid password is entered or the user wants to stop trying.

Try at Home

What flow control statements should we use and for which instructions?

- Since we don't know how many times it will take for the user to come up with a good password :

  "while“

- Since the user has the option of stopping his/her attempts at creating a valid password

  “break”

- To check on the appropriateness of a possible pwd:

  "if"

- Since there are three types of feedback:

  "switch“ (maybe)

Try at Home

UTpwd.m

```
%This program asks for a new password from the user
%the password must have at least one digit, one lower caps
%and one upper caps letter and be at least 6 characters long.
%Known bugs: program crashes if input is not
%Programmed by Alejandro Lleras
%Last updated: September 7, 2006

invalid=true;
while (invalid)




end;
'Success. Thanks!'
```

Try at Home

UTpwd.m

```
%This program asks for a new password from the user
%the password must have at least one digit, one lower caps
%and one upper caps letter and be at least 6 characters long.
%Known bugs: program crashes if input is not
%Programmed by Alejandro Lleras
%Last updated: September 7, 2006

invalid=true;
while (invalid)
    pwd = input('Please enter an uptight password, or enter STOP to leave this program‘,’s’);
    pwd_sz = length(pwd);
    if strcmp(‘pwd’,’STOP’)
        break
    end;
    if (pwd_sz > 6)   %if it passes the size test then we check for…
        if ("pwd has a number") %if it also passes the number test then…
            if ("pwd has a lowercase")
                if ("pwd has an uppercase ")
                    invalid=false;          %break would work too
                end;
            end;
        end;
    end;
end;
'Success. Thanks!'
```

Solution

```
%This program asks for a password that is at least 6 characters long
%and has at least 1 digit, 1 uppercase letter, and 1 lowercase letter each.
%Created by Kristin Divis and Alejandro Lleras.
%Last updated 9/9/10
%
%
clear all
invalid=1;
while(invalid~=0)                                                    %Creates loop until invalid is
    false
    password=input('What is your password? ','s');                   %Inputs password
    ascii_pw=double(password);                                       %Switches characters to ascii code
    size=length(ascii_pw);                                           %Finds size of password
    count_flag=0;
    if size<6                                                        %Checks if less than 6 characters
    and changes length_valid to true or false
        length_valid=0;
    else
        length_valid=1;
    end
    for i=1:size
        if ascii_pw(i)>47;                                           %Checks if ascii codes from 48-57
    (the digits)
            if ascii_pw(i)<58;
                count_flag=1;                                %If do have digits, adds to count
            end
        end
    end

    upper_flag=0;
    for i=1:size
        if ascii_pw(i)>64;                                           %Checks if ascii codes from 65-90
    (uppercase)
            if ascii_pw(i)<91;
                upper_flag=1;                                  %If have uppercase, adds to count
            end
        end
    end
```

Solution

```
low_flag=0;
    for i=1:size
        if ascii_pw(i)>96;                                  %Checks if ascii codes from 95-122
    (lowercase)
            if ascii_pw(i)<123;
                low_flag=1;                             %If have lowercase, adds to count
            end
        end
    end

    if length_valid~=0                                  %Checks to make sure all the
    individual "valids" are true
        if count_flag~=0                                 %If they are true, changes invalid
    to false, if not, stays true
            if upper_flag~=0
                if low_flag~=0
                    invalid=0;
                else
                    invalid=1;

                end
            else
                invalid=1;
            end
        else
            invalid=1;
        end
    else
        invalid=1;
    end
    invalid;
    msg=[];
```

Solution

```
switch length_valid
        case 0                                                    %All these
   switch functions print out responses for each time a rule
            msg=[msg ' You must have at least 6 characters.'];
   % is broken, if no rules are broken, nothing is printed
        case 1
    end
    switch count_flag
        case 0
            msg=[msg ' You forgot to include at least 1 digit.'];
        case 1
    end

switch upper_flag
        case 0
            msg=[msg ' You must have at least one uppercase letter.'];
        case 1
    end
    switch low_flag
        case 0
            msg=[msg ' You must have at least one lowercase letter.'];
        case 1
    end
    if ~isempty(msg)
        msg
    end;
end
msg=['Ok, password valid.']
```

GO SLEEP!!

And have sweet Matlab dreams...

Back tomorrow at 8:30 a.m.