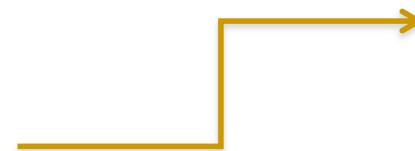


# Unit 4: Control flow (III)

# While loop

## ■ while

```
while expression
    statement1
    statement2
    ...
end
```

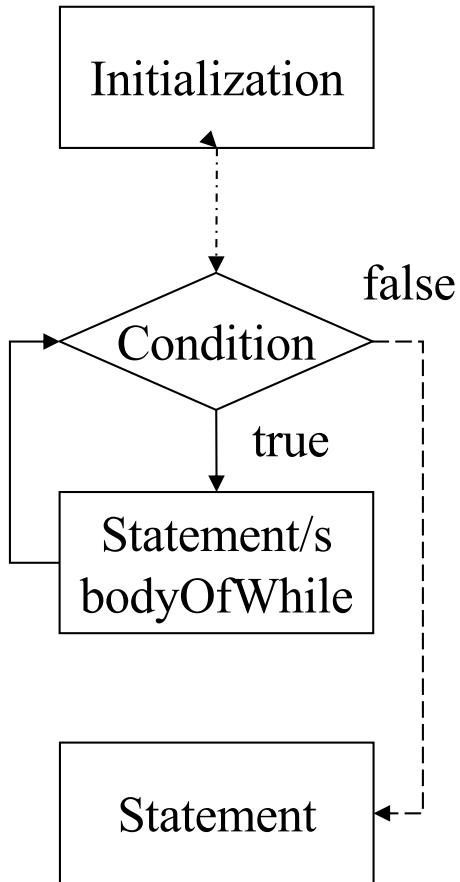


The expression is a *true* or *false* value.

In the same way as the *ifs* it can be a boolean value, a simple condition or a compose condition.

- This loop is used when the number of passes is not specified on beforehand.
- The statements are executed again and again as long as the expression is evaluated to be TRUE
- When the expression is evaluated to false MATLAB stops looping and continues on with the next command after the end

# While loop



- **Initialization:** before the 'while' statement the variables that appear in to the condition should be initialized.
- **Modification:** variables included in the condition should be modified within the body of the 'while' because if the condition goes unchanged, we get an infinite loop.
  - To stop an infinite loop press the keys **crtl + c**

# Example

```
i= 1;  
while (i<=5)  
    fprintf('\n The value is %d', i);  
    i = i +1;  
end
```

# Example

```
i= 1;  
while (i<=5) true           i = 1  
    |   fprintf('\n The value is %d', i);  
    ↓   i = i +1; now i takes the value 2  
end
```

1<sup>st</sup> iteration

Program output  
The value is 1

# Example

```
i= 1;  
----> while (i<=5)  
|       fprintf('\n The value is %d', i);  
|       i = i +1;  
|-----  
end
```

Program output  
**The value is 1**

# Example

```
i= 1;  
while (i<=5) true           i = 2  
2nd iteration    |   fprintf('\n The value is %d', i);  
                   ↓   i = i +1; now i takes the value 3  
end
```

Program output

The value is 1

The value is 2

# Example

```
i= 1;  
----> while (i<=5)  
|       fprintf('\n The value is %d', i);  
|       i = i +1;  
|-----  
end
```

Program output

The value is 1

The value is 2

# Example

```
i= 1;  
while (i<=5) true           i = 3  
3rd iteration   |     fprintf('\n The value is %d', i);  
                  ↓     i = i +1; now i takes the value 4  
end
```

## Program output

The value is 1

The value is 2

The value is 3

# Example

```
i= 1;  
----> while (i<=5)  
|       fprintf('\n The value is %d', i);  
|       i = i +1;  
|-----  
end
```

## Program output

The value is 1

The value is 2

The value is 3

# Example

```
i= 1;  
while (i<=5) true           i = 4  
4th iteration |   fprintf('\n The value is %d', i);  
                  ↓   i = i +1; now i takes the value 5  
end
```

## Program output

The value is 1  
The value is 2  
The value is 3  
The value is 4

# Example

```
i= 1;  
----> while (i<=5)  
|       fprintf('\n The value is %d', i);  
|       i = i +1;  
|-----  
end
```

## Program output

The value is 1  
The value is 2  
The value is 3  
The value is 4

# Example

```
i= 1;  
while (i<=5) true           i = 5  
5th iteration    |   fprintf('\n The value is %d', i);  
                   ↓   i = i +1; now i takes the value 6  
end
```

## Program output

The value is 1  
The value is 2  
The value is 3  
The value is 4  
The value is 5

# Example

```
i= 1;  
----> while (i<=5)  
|       fprintf('\n The value is %d', i);  
|       i = i +1;  
|-----  
end
```

## Program output

The value is 1  
The value is 2  
The value is 3  
The value is 4  
The value is 5

# Example

```
i= 1;  
while (i<=5)  
    fprintf ('\n The value is %d', i);  
    i = i - 1;  
end
```

# Example

What would be the output of  
the program this time?

```
i= 1;  
while (i<=5)  
    i = i +1;  
    fprintf('\n The value is %d', i);  
end
```

# Example

```
i= 1;  
while (i<=5) true  
    i = i +1; now i takes the value 2  
    fprintf('\n The value is %d', i); i = 2  
end
```

1<sup>st</sup> iteration

Program output  
The value is 2

# Example

```
i= 1;  
----> while (i<=5)  
| i = i +1;  
|----- fprintf('\n The value is %d', i);  
|----- end
```

Program output  
**The value is 2**

# Example

```
i= 1;  
while (i<=5) true  
    i = i +1; now i takes the value 3  
    fprintf('\n The value is %d', i); i = 3  
2nd iteration  
    ↓  
    end
```

Program output

The value is 2

The value is 3

# Example

```
i= 1;  
----> while (i<=5)  
| i = i +1;  
|----- fprintf('\n The value is %d', i);  
|----- end
```

Program output

The value is 2

The value is 3

# Example

```
i= 1;  
while (i<=5) true  
    i = i +1; now i takes the value 4  
    ↴      fprintf('\n The value is %d', i); i = 4  
3rd iteration  
    end
```

Program output

The value is 2

The value is 3

The value is 4

# Example

```
i= 1;  
----> while (i<=5)  
| i = i +1;  
|----- fprintf('\n The value is %d', i);  
|----- end
```

Program output

The value is 2

The value is 3

The value is 4

# Example

```
i= 1;  
while (i<=5) true  
    i = i +1; now i takes the value 5  
    fprintf('\n The value is %d', i); i = 5  
end
```

4<sup>th</sup> iteration

Program output

The value is 2  
The value is 3  
The value is 4  
The value is 5

# Example

```
i= 1;  
----> while (i<=5)  
| i = i +1;  
|----- fprintf('\n The value is %d', i);  
|----- end
```

## Program output

The value is 2  
The value is 3  
The value is 4  
The value is 5

# Example

```
i= 1;  
while (i<=5) true  
    i = i +1; now i takes the value 6  
    fprintf('\n The value is %d', i); i = 6  
end
```

5<sup>th</sup> iteration

## Program output

The value is 2  
The value is 3  
The value is 4  
The value is 5  
The value is 6

# Example

```
i= 1;  
----> while (i<=5)  
| i = i +1;  
|----- fprintf('\n The value is %d', i);  
|----- end
```

## Program output

The value is 2  
The value is 3  
The value is 4  
The value is 5  
The value is 6

# Example

```
i= 1;  
----- while (i<=5) false  
| i = i +1;  
| fprintf('\n The value is %d', i);  
| end  
----- ^
```

As 6 is not smaller or equal than 5 the condition is false this time, so we don't execute the loop anymore and continue on after the *end* of the while

Program output

The value is 2  
The value is 3  
The value is 4  
The value is 5  
The value is 6

# Example

```
i= 1;  
while (i<=5)  
    fprintf('\n The value is %d', i);  
end
```

And what would be the output of the program **this time?**

## Example

```
i= 1;  
-----> while (i<=5) true  
|           |           fprintf('\n The value is %d', i); i = 1  
|           |           ↓end  
|           |-----
```

And what would be the output of the program **this time**?

## Program output

The value is 1

# The value is 1

## The value is 1

### The value is 1

THIS PROGRAM WILL CAUSE AN **INFINITE LOOP**.

- The problem is that we never change the value of the variable i. It will always have the value 1, which is smaller than 5, and therefore the condition of the while will always be **true**.... so the loop will never ends

This is a very common mistake. If the execution of your program produces an infinite loop you can only stop it pressing the keys: CTRL and C at the same time

# Example

```
i= 7;  
while (i<=5)  
    fprintf('\n The value is %d', i);  
end
```

Finally, what will be the output of the program **this time?**

# Example

```
i= 7;  
----- while (i<=5) false  
|           fprintf('\n The value is %d', i);  
|           end  
----- ▾
```

Finally, what will be the output of the program **this time?**

The program will print nothing on screen, as it will never execute the statements inside the loop

# While loop

- Steps to follow when solving a ‘while loop’ exercise
  - 1. Write down the statements you want to repeat
  - 2. Write down the condition that (when evaluated to **TRUE**) keep them repeating
  - 3. **Give an initial value to** the variables of the conditions **before** the while
  - 4. Make sure that (at least one of ) the variables of the condition change its value **inside** the loop, so at some point the condition is evaluated to false and the looping stops

# Top 4 problems when working with while loops

1. Not modifying the value of the variable of the condition inside the while.
2. Not knowing where to modify the value of the condition inside the while
3. Not knowing which value use to initialize the variable of the condition
4. The condition of the while is incorrect

As a results of these issues you might end up with an infinite loop, not executing the while, missing some cases or, in general, obtaining incorrect results

# Top 4 problems when working with while loops

1. Not modifying the value of the variable of the condition inside the while.
2. Not knowing where to modify the value of the condition inside the while
3. Not knowing which value use to initialize the variable of the condition
4. The condition of the while is incorrect

To avoid these problems try to replicate the execution path of the program for your solution (trace the execution, as in the first example of these slides)

# Top 4 problems when working with while loops

**Remember:** there is not a general “correct template” for writing while loops that you can learn by heart. Depending on the problem we initialize the variables with one value or another, we modify them at different parts of the while, or we write different types of conditions. The correct solution might be different for each problem.

# Exercise

Exercise: Write a program which asks the user to introduce numbers until he/she introduces a 0.

Example of execution:

Introduce a value: 7

Introduce a value: 5

Introduce a value: 3

Introduce a value: 9

Introduce a value: 2

Introduce a value: 0

# Exercise

**Exercise:** Write a program which asks the user to introduce numbers until he/she introduces a 0.

```
numberRead = input('Introduce a number');

while (numberRead ~= 0)
    numberRead = input('Introduce a number');
end
```

# Exercise

**Exercise:** Write a program which asks the user to introduce numbers until he/she introduces a 0.

```
numberRead = -1;  
  
while (numberRead ~= 0)  
    numberRead = input('Introduce a number');  
end
```

Another possibility is to initialize the var numRead ourselves, with a value different from 0 so that it gets inside the while at least once.

# Exercise

Exercise: Modify the previous program so that it also prints on screen the numbers the user introduces.

Introduce a value: 7

The number is 7

Introduce a value: 5

The number is 5

Introduce a value: 3

The number is 3

Introduce a value: 0

# Exercise

**Exercise:** Modify the previous program so that it also prints on screen the numbers the user introduces.

```
numberRead = input('Introduce a number');

while (numberRead ~= 0)
    fprintf('\n The number is :%d', numberRead);
    numberRead = input('Introduce a number');
end
```

# Exercise

**Exercise:** Modify the previous program so that it also prints on screen the numbers the user introduces.

Is this solution correct?

```
numberRead = -1;

while (numberRead ~= 0)
    numberRead = input('Introduce a number');
    fprintf('\n The number is :%d', numberRead);
end
```

# Exercise

**Exercise:** Modify the previous program so that it also prints on screen the numbers the user introduces.

Is this solution correct?

```
numberRead = -1;

while (numberRead ~= 0)
    numberRead = input('Introduce a number');
    fprintf('\n The number is :%d', numberRead);
end
```

It is almost correct. The problem is that in this case the program prints the value 0, so the output will not be exactly the same as in the example provided

# Exercise

**Exercise:** Write a program which asks the user to introduce numbers until he/she introduces a 0, and then it prints the sum

Introduce a value: 7

Introduce a value: 5

Introduce a value: 3

Introduce a value: 9

Introduce a value: 2

Introduce a value: 0

The sum is 26

# Exercise

**Exercise:** Write a program which asks the user to introduce numbers until he/she introduces a 0, and then it prints the sum

```
varsum = 0;  
numberRead = input('Introduce a number');  
while (numberRead ~= 0)  
    varsum = varsum + numberRead;  
    numberRead = input('Introduce a number');  
end  
fprintf('\n The sum is %d ', varsum);
```

# Exercise

**Exercise:** Write a program which asks the user to introduce numbers until the sum of the numbers introduced is greater than 50, and then prints the sum.

Introduce a value: 10

Introduce a value: 14

Introduce a value: 9

Introduce a value: 20

The sum is 53

# Exercise

**Exercise:** Write a program which asks the user to introduce numbers until the sum of the numbers introduced is greater than 50, and then prints the sum.

```
varsum = 0;  
while (varsum <= 50)  
    numberRead = input('Introduce a number');  
    varsum = varsum + numberRead;  
end  
fprintf('\n The sum is %d ', varsum);
```

# Exercise

**Exercise:** Write a program which asks the user to introduce numbers until he/she introduces 0 or the sum of the numbers introduced is greater than 50

# While loop

- Sometimes the conditions to break the loop are double (or triple...)
  - Example: "...until the user introduces a 0 or the sum is greater than 50"

1    `while ((numberRead ~= 0) && (mySum <= 50))`

...

2    `while ((numberRead ~= 0) || (mySum <= 50))`

...

Which one of the two is the correct?

# While loop

- Sometimes the conditions to break the loop are double (or triple...)
  - Example: "...until the user introduces a 0 or the sum is greater than 50"

1    `while ((numberRead ~= 0) && (mySum <= 50))`

...

2    ~~`while ((numberRead ~= 0) || (mySum <= 50))`~~

...

In this case the right one is **the first one**. Loops in which you iterate until one of two conditions is not satisfied are very frequent

# While loop

- Sometimes the conditions to break the loop are double (or triple...)
  - Example: "...until the user introduces a 0 or the sum is greater than 50"

1

while ((numberRead ~= 0) && (mySum <= 50))

...

2

while ((numberRead ~= 0) || (mySum <= 50))

...

Be careful and examine with precision the conditions in each case

# Exercise

```
varsum = 0;  
numberRead = -1;  
while ((varsum <= 50) && (numberRead ~= 0))  
    numberRead = input('Introduce a number');  
    varsum = varsum + numberRead;  
end  
fprintf('\n The sum is %d ', varsum);
```

# Exercise

- Exercise: Write a program which asks the user to introduce numbers and stores them in a vector one after the other. After introducing a value the program asks the user if he/she wants to introduce more (Y/N), and when the user finishes the program prints the content of the vector.

Introduce a number: 3

Do you want to introduce more values (Y/N)? Y

Introduce a number: 7

Do you want to introduce more values (Y/N)? Y

Introduce a number: 2

Do you want to introduce more values (Y/N)? N

The numbers in the vector are:

3 7 2

# Exercise

```
vector = [];
index = 0;
cContinue = 'Y';
while (cContinue == 'Y')
    number = input('Introduce a number: ');
    index = index + 1;
    vector(index) = number;
    cContinue = input('Do you want to introduce more
values (Y/N) ? ','s');
end

disp ('The numbers in the vector are:');
for value = vector
    fprintf(' %d ', value);
end
```

The problem with this program is that if the user introduces something different from Y or N, it will stop asking for numbers. Next week we will see how to include an inner loop to make sure that the user entry is always Y or N

# Exercise

- Exercise: Modify the previous exercise, so that after the user finishes introducing numbers the program asks him/her to introduce a value. Then the program says if the number is in the vector or not.

Introduce a number: 3

Do you want to introduce more values (Y/N)? Y

Introduce a number: 7

Do you want to introduce more values (Y/N)? Y

Introduce a number: 2

Do you want to introduce more values (Y/N)? N

Introduce a value to search: 7

The value 7 is in the vector

# Search

*vect*

8	7	1	2	9	3
---	---	---	---	---	---

Example: Does the vector *vect* contains the number 7?

# Search

positions	1	2	3	4	5	6
<i>vect</i>	8	7	1	2	9	3

START

Index = 0

found = 0



This will be the result of the search. If we found it, we will put in this variable 1 (true) otherwise it will be 0 (false). Initially we put 0 as we don't know if we will found it

# Search

positions	1	2	3	4	5	6
<i>vect</i>	8	7	1	2	9	3
Index = 1						

is  $\text{vect}(\text{index}) == 7$  ?

found = 0

No

# Search

positions	1	2	3	4	5	6
<i>vect</i>	8	7	1	2	9	3
		↑				

Index = 2

is  $\text{vect}(\text{index}) == 7$  ?

found = 1

Yes, set *found* to 1 (TRUE)

# Search

*vect*

8	7	1	2	9	3
---	---	---	---	---	---

Example: Does the vector *vect* contains the number 4?

# Search

positions	1	2	3	4	5	6
<i>vect</i>	8	7	1	2	9	3

START

Index = 0

found = 0

# Search

positions	1	2	3	4	5	6
<i>vect</i>	8	7	1	2	9	3
Index = 1						

found = 0

is  $\text{vect}(\text{index}) == 4$ ?

No

# Search

positions	1	2	3	4	5	6
<i>vect</i>	8	7	1	2	9	3
		↑				

Index = 2

is  $\text{vect}(\text{index}) == 4$ ?

found = 0

No

# Search

positions	1	2	3	4	5	6
<i>vect</i>	8	7	1	2	9	3



Index = 3

is  $\text{vect}(\text{index}) == 4?$

found = 0

No

# Search

positions	1	2	3	4	5	6
<i>vect</i>	8	7	1	2	9	3



Index = 4

is  $\text{vect}(\text{index}) == 4$ ?

found = 0

No

# Search

positions	1	2	3	4	5	6
<i>vect</i>	8	7	1	2	9	3



Index = 5

is  $\text{vect}(\text{index}) == 4?$

found = 0

No

# Search

positions	1	2	3	4	5	6
<i>vect</i>	8	7	1	2	9	3



Index = 3

is  $\text{vect}(\text{index}) == 4?$

found = 0

No

# Search

positions	1	2	3	4	5	6
<i>vect</i>	8	7	1	2	9	3

END

Index = 7

found = 0

# Exercise

```
index = 1;
bFound = 0;
number = input ('Introduce a number: ');
while ((index <= length(vect)) & (bFound == 0))
    if (number == vect(index))
        bFound = 1;
    else
        index = index +1;
    end;
end

if (bFound == 1)
    disp ('The number is in the vector');
else
    disp ('The number is NOT in the vector');
end
```

For the sake of simplifying the solution we assume that the vector vect has already been fill with numbers previously

# Exercise

- **Exercise:** Write a program which asks the user to introduce numbers. If the number is divisible by 2, 3 or 5 it prints the result of the correspondent division. Otherwise stops the execution.

Introduce a number: 30

30 divided by 2 is 15

30 divided by 3 is 10

30 divided by 5 is 6

Introduce a number: 15

15 divided by 3 is 5

15 divided by 5 is 3

Introduce a number: 7

Not divisible by 2, 3 or 5

# Exercise

```
number = input('Introduce a number: ');

while ((rem(number,2) == 0) || (rem(number,3) == 0) || (rem(number,5) == 0))

    if (rem(number,2) == 0)
        fprintf('\n %d divided by 2 is %d', number, number/2);
    end;
    if (rem(number,3) == 0)
        fprintf('\n %d divided by 3 is %d', number, number/3);
    end;
    if (rem(number,5) == 0)
        fprintf('\n %d divided by 5 is %d', number, number/5);
    end;

number = input('Introduce a number: ');

end
· disp('Not divisible by 2, 3 or 5');
```

# Exercise

```
bContinue = 1;  
while (bContinue == 1)
```

```
    number = input('Introduce a number: ');\n    if ((rem(number,2) ~= 0) & (rem(number,3) ~= 0) & (rem(number,5) ~= 0))\n        disp('Not divisible by 2, 3 or 5');\n    bContinue = 0;\nelse\n    if (rem(number,2) == 0)\n        fprintf('\n %d divided by 2 is %d', number, number/2);\n    end;\n    if (rem(number,3) == 0)\n        fprintf('\n %d divided by 3 is %d', number, number/3);\n    end;\n    if (rem(number,5) == 0)\n        fprintf('\n %d divided by 5 is %d', number, number/5);\n    end;\nend;
```

We can use a variable containing a boolean value as the condition of the loop. Initially we set the value to true (1) and when we want to stop the loop we set it to false (0)

ANOTHER  
POSSIBLE  
SOLUTION

# Exercise

```
bContinue = 1;  
while (bContinue == 1)  
    bContinue = 0;  
    number = input('Introduce a number: ');  
    if (rem(number,2) == 0)  
        fprintf('\n %d divided by 2 is %d', number, number/2);  
        bContinue = 1;  
    end;  
    if (rem(number,3) == 0)  
        fprintf('\n %d divided by 3 is %d', number, number/3);  
        bContinue = 1;  
    end;  
    if (rem(number,5) == 0)  
        fprintf('\n %d divided by 5 is %d', number, number/5);  
        bContinue = 1;  
    end;  
end  
· disp('Not divisible by 2, 3 or 5');
```

ANOTHER  
SOLUTION