Exercises on arrays

Exercise 08.01: There are 12 months in a year

- Create an array of the 12 months of the year.
- Set the data correct.
- Show in the console every value, by looping thru the elements.

Exercise 08.02: C# arrays

- https://www.w3schools.com/cs/exercise.asp?filename=exercise_arrays1
  - Exercise 1.
  - Exercise 2.
  - Exercise 3.
  - Exercise 4.
  - Exercise 5.

Exercise 08.03: Bubble Sort

See for explanation “Leren programmeren – Deel 08 – Algoritmes, slides 5 till 12 of the first module in learning to program.

We are going to sort an array. Normally you use the method .Sort() for doing that. Here we do some style exercises to do it by ourselves. The goal of the exercise is to sort the array by yourself.

Please, do not download the solution from internet. Try it by yourself, if you want to learn something.

- You declare and assign an array (1 dimension, several values).
- You program a method to sort an array.
  - Input is an array, output is a sorted array.
- You are programming this algorithm below.
- Loop thru the elements of the array.
  - indexOfLastNotSorted = Length of the list minus index of current element minus 1. Current element is the loop position.
  - Loop from 0 to indexOfLastNotSorted minus 1.
    - If element on current position of inner loop is bigger than element next position of inner loop.
      - Remember value of element on current position.
      - Element on current position becomes element of next position.
      - Element on next position becomes the remembered value.
  - Continue with loop (inner loop).
- Continue with loop (outer loop, current element).

Variant 1

- You stop with the first loop, when no items are switched in the second loop. This means that they are already in the correct order.

Exercise 08.04: Insertion Sort

See for explanation “Leren programmeren – Deel 08 – Algoritmes, slides 13 till 17 of the first module in learning to program.

This is just an exercise to getting used to work with arrays.

Please, do not download the solution from internet. Try it by yourself, if you want to learn something.

- You declare and assign an array (1 dimension, several values).
- You program a method to sort an array.
  - Input is an array, output is a sorted array.
- You are programming this algorithm.
- Loop thru the elements of the array, except the first one.
  - Remember the value of the current element.
  - Loop from current position towards the second position.
    - If element on the position before the current position is bigger than remembered element.
      - Replace element on current position with the one on the position before the current position.
      - Continue with loop (inner loop).
    - If Not.
      - Exit the loop (inner loop).
  - Place the remembered value on the position where you ended the loop.
- Continue with loop (outer loop).

Variant 1

- Make the array a bit bigger, and put randomly values in the array. Try it out with the size of 50, 100, 200, 10.000, 100.000.
- Look at the time it takes to execute.

Exercise 08.05: Merge sort

See for explanation “Leren programmeren – Deel 08 – Algoritmes, slides 13 till 21 of the first module in learning to program.

This is just an exercise to getting used to work with arrays.

Please, do not download the solution from internet. Try it by yourself, if you want to learn something.

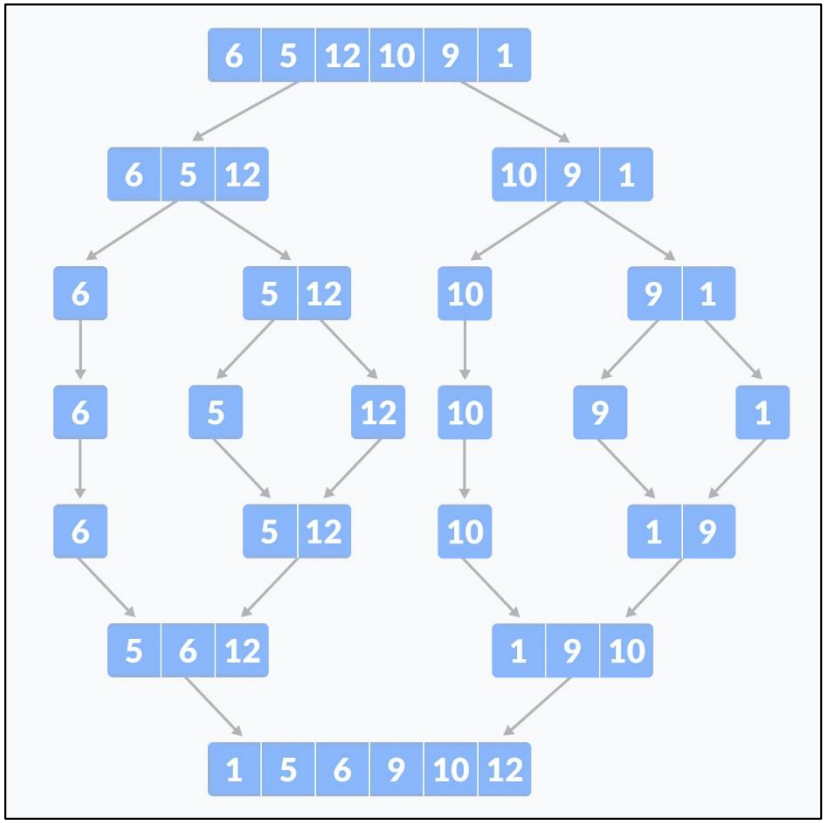
- You declare and assign an array (1 dimension, several values).
- You program a method to sort an array.
  - Input is an array, output is a sorted array.
- You are programming this algorithm.
- You divide the array in 2 parts when the size is bigger than 1. When the size is one, it is sorted.
  - 2 possibilities for arrays bigger than 1.
    - The sizes are equal (when number of elements is even).
    - The first one is 1 smaller than the other (when number of elements is odd).
    - Restart the same routine with the 2 smaller arrays you now have.
  - Put the 2 results back together, by creating an array and putting the elements of 2 smaller arrays in the correct order.
    - By taking the smallest element of the not yet taken elements of both smaller arrays.
- You end up with a sorted array.

See the picture below for what is exactly asked.

Ask questions to colleagues if things are not clear.

First work out a battle plan (User Story and Tasks) on the things that you want to do. And estimate them.

An example on how the routine works



Variant 1

- Compare the speed of all the algorithms with random generated arrays of size:
  - 100
  - 1.000
  - 10.000
  - 100.000
  - 1.000.000

COPY PASTE)

Exercises that will take a bit of time

The goal of this exercises is to think about on how you get to your solution and to keep your solution as maintainable as possible.

Write down tasks. Think about their size (volume of work). Try to execute. Be ware, your code will change a lot in the variants.

Exercise 08.06: Multiply small numbers (tricky exercise)

Be careful here. Think before you program.

You must do that, always, …

The goal is to have a correct solution, that you can change very quickly. It will change in the variants. And it will make you mad.

Think in methods that are reusable. Read all the variant before starting to code.

Having some fun with basic calculations and small numbers.

- The goal is to have on the console every number between 1 and 15, to be multiplied with every number between 1 and 15.
- First, 1 multiply with 1 till 15.
- Then, 2 multiply with 1 till 15.
- …
- As last 15 multiply with 1 till 15.

Use in the text you write to the console the escape symbol \t to have a tab between the numbers.

See documentation Leren Programmeren – Deel 03 – C# Data Types, slide 9 and 10.

Example result

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	4	6	8	10	12	14	16	18	20	22	24	26	28	30
3	6	9	12	15	18	21	24	27	30	33	36	39	42	45
4	8	12	16	20	24	28	32	36	40	44	48	52	56	60
5	10	15	20	25	30	35	40	45	50	55	60	65	70	75
6	12	18	24	30	36	42	48	54	60	66	72	78	84	90
7	14	21	28	35	42	49	56	63	70	77	84	91	98	105
8	16	24	32	40	48	56	64	72	80	88	96	104	112	120
9	18	27	36	45	54	63	72	81	90	99	108	117	126	135
10	20	30	40	50	60	70	80	90	100	110	120	130	140	150
11	22	33	44	55	66	77	88	99	110	121	132	143	154	165
12	24	36	48	60	72	84	96	108	120	132	144	156	168	180
13	26	39	52	65	78	91	104	117	130	143	156	169	182	195
14	28	42	56	70	84	98	112	126	140	154	168	182	196	210
15	30	45	60	75	90	105	120	135	150	165	180	195	210	225

Variant 1

- Ask the maximum number, so you can create the multiply table as big as you want.

In the console, the layout will be put wrongly in the console, because you have only 80 characters.

But that is not a problem for this exercise. The goal is to try-out loops inside loops.

Variant 2

- You add titles on top and on the left side of the table.
- Use a "-" for placing a line horizontal, and "|" to place a line vertically.
- Make sure that the horizontal line has the correct length ("--------").

Variant 3

- You do only the half of the work, because 10 * 15 is the same as 15 * 10.
- Pay attention. Horizontal till x and vertically till x-1.

Example result variant 2 and 3

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	2	4	6	8	10	12	14	16	18	20	22	24	26	28	
3	3	6	9	12	15	18	21	24	27	30	33	36	39		
4	4	8	12	16	20	24	28	32	36	40	44	48			
5	5	10	15	20	25	30	35	40	45	50	55				
6	6	12	18	24	30	36	42	48	54	60					
7	7	14	21	28	35	42	49	56	63						
8	8	16	24	32	40	48	56	64							
9	9	18	27	36	45	54	63								
10	10	20	30	40	50	60									
11	11	22	33	44	55										
12	12	24	36	48											
13	13	26	39												
14	14	28													

Variant 4

- Do you see something strange in the result of exercise variant 3?
- How can you correct it?

Variant 5 (First do variant 4)

- Repeat this table for other operators.
  - First the “+”.
  - Then the “-“.
  - Then the “*”.
  - Then the whole division. Operator “\”.
  - Then the rest after division. Operator “%”.

Do not repeat your code, but create a method that accepts one parameter.

The operator that must be used to calculate is the given variable for the routine.

Exercise 08.07: Repeat a calculation a lot (looping)

- Loop all the numbers between 100 and 1234.
- Within the loop these are the steps you must do.
  - Example is given with number 471.
  - Make from the number 471 a text.
    - You have a text “471”.
  - Put 2 times that text after each other.
    - You have a text “471471”.
  - Make from that text a number.
    - You have the number 471471.

There are other ways to do this.

*e.g., You can do this: Number * 1000 + Number, but when you have 1000 or more, the way of calculating changes.*

Just do it like described above.

  - Divide that number with 7.
  - Take the result and divide this by 11.
  - Take the result and divide this by 13.
  - You have an end result.
  - If that end result is different than the number you have started with (471), exit the loop.
- Show on the console, the number that causes the exit of the loop.

Exercise 08.08: The persistence of a number

Having some fun with basic calculations and big numbers.

This exercise will force you to work in small steps.

Test every time your small step is a good one.

Check also the speed of your solution.

Part 1: Calculating the persistence of a number in a method (not Main)

- Take a number.
  - Example: 88.
- Multiply all the digits of that number.
  - 8 * 8 → 64.
- Loop this routine, till the result is smaller than 10.
  - 88 → 64 → 24 → 8.
- The persistence of a number is the number of times you did the loop routine.
  - In the case of 88 it is 3.
  - Doing 3 times the loop, gets me a number smaller than 10.

Another example

- 679 → 378 → 168 → 48 → 32 → 6.
- So, 679 has a persistence of 5.

We write a routine that calculates the persistence of a given number.

So input is a number, the output is the persistence of that number.

Part 2: Using this routine to play around

- Loop from 10 till you have found the first number with persistence 1, 2, 3, 4, 5, 6 and 7.

Your routine stops when you have found the first number with persistence 7.

Work in small steps. Start with 1, and then 2, to see if you looping routine works.

- During the loop, you show messages.
  - Show in the console the first number for persistence 1.
    - This should be 10.
  - Show in the console the first number for persistence 2.
    - This should be 25.
  - Show in the console the first number for persistence 3.
    - This should be 39.
  - Show in the console the first number for persistence 4.
    - Ask your colleagues what the number should be.
  - Show in the console the first number for persistence 5.
    - Ask your colleagues what the number should be.
  - Show in the console the first number for persistence 6.
    - Ask your colleagues what the number should be.
  - Show in the console the first number for persistence 7.
    - Ask your colleagues what the number should be.

Part 3: What is the first number for persistence 8?

Can you optimize your routine to make it faster?

- Example. If there is a 0 in the number, you are sure that the multiplication will be 0.
- And so on (you can invent some more solutions to make your routine faster).

Part 4: What is the first number for persistence 9?

- What with a number that contains a “2” and a “5”?

Part 5: What is the first number for persistence 10?

Can you optimize your routine to make it faster?

- If you have number 101. What will be the next number that does not contain a 0?
- Optimize your routine so the loop goes from 101 to the next number you are sure of that does not contain a 0.

	*Test your routine to make sure it works correctly, before you are actually using it in your loop.*

Part 6: What is the first number for persistence 11?

	*This will take a while.* *Add to your routine a check that your PC is still running.* *E.g., after every 250.000.000th loop, you write a “.” to the console.*

Part 7: What is the first number for persistence 12?

	*This will take you a lifetime, probably because it does not exist.*

Exercise 08.09: Encryption and decryption

You have the alphabet of 26 letters. Define a encryption and decryption key.

- This means:
  - You change every letter by another letter.
    - A → S.
    - B → C.
    - And so on.
  - Make sure you don't have 2 different letters that are change by the same letter.
- You have to make the encryption and decryption key at run time.
  - You can use a random technique to make it.
  - You can decide how.

Then you ask a text and you encrypt it according to your encryption key.

- Show the result on the screen.
- Symbols like comma's, dots, are not encrypted.
- This technique is called substitution.

And then you decrypt the result.

- Show the result on the screen.
- This must be exactly the same as your original text.

The moment this works, you can update your program by changing the encryption and decryption key or the routine to encrypt and decrypt (the substitution technique)

Whatever you do, you must be able to encrypt and decrypt your message.

Have fun with it.

Exercise 08.10: Try to work in a structured manner

For this exercise, I want you to do several things. I've you want to look this up on the internet, it is called "Kaprekar's Constant", but try to create this exercise by yourself, before you are looking at video's or using ChatGPT to receive a solution.

Part 1

- Divide the big exercise in smaller parts (on paper).
- Estimate how long you think you will work on every subtask.
- You program it.
  - Every subtask separately.
  - Test all the methods you created.
- Compare your estimations on the subtask with the real time you've needed to have the stuff working.
- Can you find where were the differences?

Part 2

The asked routine:

- You start with a number of 4 digits → e.g. 1284.
  - Attention. The number 0001 is possible. So leading zeros should be possible.
- Put all the digits into order, from high to low.
  - So in our example you will have 8421.
- Put all the digits into ordedr, from low to high.
  - So in our example you will have 1248.
- Subtract the lowest number from the biggest number.
  - So in our example → 8421 – 1284 → 7137.
- Repeat the procedure, till you are in a loop.

Example (I made some mistakes on purpose, not to spoil the result)

Pay attention. On purpose is the example below wrong. There are several mistakes in it.

Compare your programming result of this example with the example shown here. Your solution will probably more correct than my example.

Better, find the mistakes (plural) in what I show here as example.

- 1284 → 8421 – 1284 → 7137.
- 7137 → 7731 – 7137 → 0594.
- 0594 → 9540 – 0594 → 8946.
- 8946 → 9864 – 8946 → 0918.
- 0918 → 9810 – 0918 → 8892.
- 8892 → 9882 – 8892 → 0990.
- 0990 → 9900 – 0990 → 8910.
- 8910 → 9810 – 8910 → 0900.
- 0900 → 9000 – 0900 → 8100.
- 8100 → 8100 – 8100 → 0000.
- 0000 → 0000 – 0000 → 0000 (Same result as previous result).
- And you are in a loop.

Pay attention.

It is possible that you have a loop of more than one step.

Meaning: From A → B → C → A (again).

Part 3

I want to know for all numbers from “0000” till “9999”, how many steps you have before you are looping yourself.

For the example of 1284, you have 11 steps before you are in a loop. (Wrong).

Optimize your code on speed, when you have a working solution.

Exercise 08.11: Russian Peasant Multiplication

The following exercise is to calculate the multiplication of two numbers, but not by multiplying them, but by using another technique. When you follow the steps, you have a nice algorithm.

It is called the "Russian Peasant Multiplication", but it is sure that the old Greeks already knew this method.

Part 1

Ask two numbers (integers) you want to multiply.

An example.
46 and 3

Part 2

You take a first number, and you divide that number every time by 2 until you reach 1.

When the number is not even, your division will result in something and a half. Drop the half (you want to do an integer division)

An example.
46 → 23 → 11 → 5 → 2 → 1

Part 3

You take the second number, and you multiply that number every time by 2 until you have done that as many times as you needed to divide the first number.

An example.
So you needed 5 steps to go from 46 till 1. So you do the multiplication by 2 five times for the second number.

3 → 6 → 12 → 24 → 48 → 96

Part 4

For every odd number in the result of part 2, you take the corresponding numbers of the result of part 3 and add them together.

An example.

The second number 23, the third number 11, the fourth number 5 and the last number 1 are odd.

The corresponding numbers are 6, 12, 24 and 96 when you add them together you get 138.

138 is the multiplication of your 2 asked numbers from Part 1. Always.

Can you find out why this is correct?

Think about how numbers are stored in a computer program.

Exercise 08.12: Morse Code

Part 1

You have a text with the possible 26 letters of the alphabet. We don't take into account other symbols. You can ask for a text, you can hard code the text, …

- Every letter from A till Z must be converted into it's Morse code and a space.
- Every other symbol is converted into itself. So a space becomes a space, a question mark becomes a question mark, and so on.

There are 2 exceptions.

A dot (.) will be removed, a dash (-) will also be removed.

The best way is first remove the dots and the dashes from the given text.

Below you find the list of characters and its corresponding Morse code.

A ●−	J ●−−−	S ●●●
B −●●●	K −●−	T −
C −●−●	L ●−●●	U ●●−
D −●●	M −−	V ●●●−
E ●	N −●	W ●−−
F ●●−●	O −−−	X −●●−
G −−●	P ●−−●	Y −●−−
H ●●●●	Q −−●−	Z −−●●
I ●●	R ●−●	

Create a routine that converts an English text into Morse code.

Part 2

You have a text containing dots (.) and dashes (-) and some other characters like spaces, questions marks and so on, …

- Every series of dots and dashes till the first space, must be converted to the corresponding letter.
- Every other symbol is converted into itself. So a space becomes a space, a question mark becomes a question mark, and so on.

Create a routine that converts a Morse code into English text.

Part 3

Test your routine carefully and in parts. Every functionality must be tested separately, and working correctly before continuing with another functionality.

Pay attention.

I want that your testing code is in a different program than your end result. So you have a program that is used for testing, you have a program that can be executed to convert text in morse and vice versa.

Meaning.

- Be sure that dots and dashes are removed from the given text.
- Be sure that letters are correctly converted into Morse code.
- Be sure that all Morse codes are correctly converted into letters.
- You can for example start with a text.
    - Remove the dots and the dashes. (Result 1)
    - Convert it to morse. (Result 2)
    - Convert it back to the text. (Result 3)
    - Compare both results. Result1 and Result 3 should be equal.

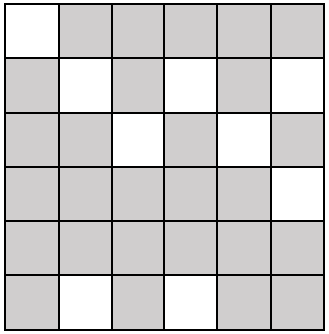
Exercise 08.13: The Cardan Grille (Part 01)

Suppose you have square. In this example it will be 6 by 6. And it is filled with letters.

D	A	T	N	I	N
I	A		N	Q	
O	E	G	U	R	N
E	N	C			I
T	R	E	C	C	Y
P	L	A	L	R	H

This is the message that travels from point A to B. It is an encrypted text.

The sender, the person who writes the message, and the addressee, the person who receives the message has another grill. This grill do not travel with the message. That grill contains holes where you can look thru.



The white parts of the grill are the holes.

When you put the grille on the message, you can read the first part of the message. Then rotate the grille 90 degrees counter clock wise. When you put the grille on the message, you can read the second part of the message.

This routine, rotate and read, is repeated another 2 times. So you have 4 parts of the message.

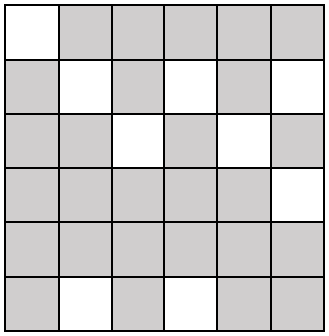
Your task

- Write a program that asks a text of maximum 36 letters.
  - Check the input.
  - Re-ask if not correct.
- Make according to the given grille on the previous page your encrypted text.
  - Make an array of Booleans (true or false).
    - True you can read.
    - False is hidden.
- Your result after looking, rotating, looking, rotating, looking, rotating and a final look will be an encrypted text.
- Show the grid on the screen
- Try to write also a routine to decrypt your result.
- You should have back your original.

Variant 1

- On the console I must be able to choose an option.
  - Do you want to encrypt.
  - Do you want to decrypt.
  - Type an encrypted or decrypted text, and the result of encryption or decryption will be shown.

Exercise 08.14: The Cardan Grille (Part 02)



Your task

The example of the grille above is a good one. Every time I rotate the grille, a different group of letters is shown. You can check it.

- You must write a check that a given grill is correct or not.
  - So you start with an array of 36 Booleans (true or false), and your application must give me this result.
    - Yes: the grille can be used for encryption or decryption.
    - No: there is an error in the grille.

Variant 1

- Show the coordinates in the grille that are causing a wrong grille.

Variant 2

- Generate randomly a correct grille with 7 white spaces (for a message of 28 characters).
- Generate randomly a correct grille with 8 white spaces (for a message of 32 characters).
- Generate randomly a correct grille with 9 white spaces (for a message of 36 characters).

Exercise 08.15: The eight queens chess puzzle (Part 01)

You have a chessboard of 8 by 8.

A queen can take another piece if it stands in the same row, column or diagonal.

- Put randomly 8 queens somewhere on the board.
  - One in every row.
  - An example.

Your task

- Put randomly 8 queens on the board.
- Show this in a console. You will find a way.
- Count the queens, that you put randomly on the board, does see another queen in the same column.
- Count the queens, that you put randomly on the board, does see another queen in the same diagonals.
- Count how many queens can take another queen and show result.
  - In the given example the answer is 0.

Tip: The answer can never be 1.

Exercise 08.16: The eight queens chess puzzle (Part 02)

You have a chessboard of 8 by 8.

A queen can take another piece if it stands in the same row, column or diagonal.

- Put 8 queens on the bord, and none of them can take another one.
  - A queen looks in a column, she does not see another queen.
  - A queen looks in a diagonal, she does not see another queen.
    - Attention: There are two diagonals.
  - A queen looks in a row, she does not see another queen.
- Below you have a solution.

Your task

- Find all solutions.

	*There are 92 solutions possible.* *Have fun with this one.* *Don't look at internet. You will have the solutions (even in C#) in a second.* *Try to find a battleplan. First the battleplan, than the implementation of it.*

Exercise 08.17: The Hangman game

I will describe how you can play it, how you solve it, is completely your decision.

The game starts with a word that is visualize on the screen with dashes.

- E.g. the word is “PROGRAMMER”.
  - You can make an array of existing words, where the program picks randomly one word form.
- The word in the example exists of 10 letters.
- It is visualized as _ _ _ _ _ _ _ _ _ _
  - Every _ replaces a letter.
- The player must try to guess the complete word, letter by letter, before the picture of the hangman is complete.
- The player starts to guess with a letter.
  - There a two options.
    - The letter is in the word (possible multiple times).
    - The letter is not in the word.
- If the letter is in the word, every dash is replaced at the correct place with that letter.
  - E.g. The letter M is guessed.
  - Result: _ _ _ _ _ _ M M _ _ is shown.
- If the letter is not in the word, the first item of the hangman is drawn.
  - There are 10 items to draw the hangman.
- The player wins, when the complete word is guessed before the hangman is completely drawn.
- The player loses, when the picture of the hangman is drawn, before the word is guessed.
- Make sure that the game is playable.

Extra info

Here you can find the picture of the hangman.

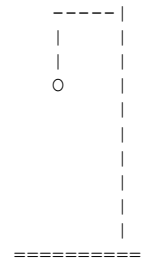
Step 1: The floor

Step 3: The pole

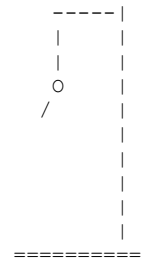
Step 3: The bar

Step 4: The rope

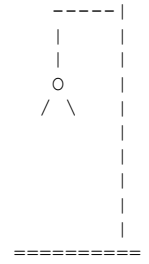
Step 5: The head



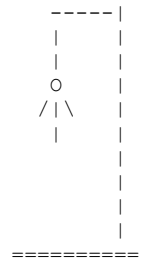
Step 6: The first arm



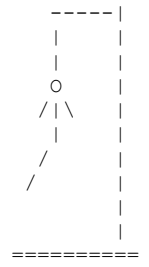
Step 7: The second arm



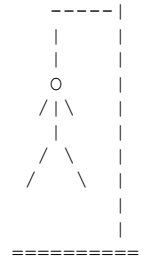
Step 8: The body



Step 9: The first leg



Step 10: The second leg



The hangman is complete.

Have fun with playing the game.