## Tartalomjegyzék

- 1 Exercises
- 1.1

Distance

- 1.2

Min-max
-1.3 Sort

- 1.4 C
string
- 1.5

Histogram

- 1.6

Constellations

- 1.7 Size
of a
closed
space


## Exercises

Open a new project for each exercise or a new file if you're in the command line.

## Distance

- Write a function with 2 parameters. The parameters are 2 dimensional coordinates. The function returns their distance.
- Define the appropriate struct type to use.
- Write a main function to test the function.
- The math.h library contains an sqrt function that we can use. First include the library just like stdio.h:
\#include<math.h>
Then the sqrt function can be used as expected:
sqrt(4)


## Min-max

Write a function with 2 int pointer parameters, min and max. The function should make sure that the value stored at min is smaller (or equal) to the one stored at max. If this isn't the case, it should swap their values.

## Sort

Write a function that sorts a given (int) array in ascending order.

## C string

Study this code a bit:
\#include <stdio.h>
int main() \{

```
    char str[] = "puppy";
    printf("The word is %s.\n", str);
    return 0;
}
```

Try to print the elements of the str array one by one as if they were integers (use \%d in printf). The str array has 6 elements actually (even though it only stores 5 characters).

You can also read strings with scanf, but for this we'll need an appropriately large char array:

```
char str[100];
scanf("%s", str);
```

So strings are actually just character arrays with a "http://wiki.math.bme.huterminal null"http://wiki.math.bme.hu character at the end. The character code of this is ' 10 '. Example of a for cycle to go through a string:

```
for(i = 0; str[i] != '\0'; i++) {
    printf("%c", str[i]);
}
```


## Histogram

- Write a function that counts how many times each number appears in a given array.
- The array and its length are given to the function as a parameter. We can assume that the array contains integers with values between 0 and 10 (why do we need this?).
- The result should be an array that should be returned through a parameter.
- Write a main function that reads 10 integers into an array, executes the function on the array and prints the output.


## Constellations

The x an y coordiantes of stars are given (we're working in a 2 dimensional space). Group the stars into constellations. A constellation is defined by each star inside a given contellation having a star to star path to every other star in the contellation that doesn't use steps longer than 10 distance units. (The length of the whole path doesn't matter, only that each star to star step is at most 10 units.)

The coordinates of the stars:

```
float x[] = {28.1,13.9,29.1,25,6.5,16.4,5.9,24.6,4.4,6.3,24,2.4,18.8,12.9,2.5,26.8,22.4,13,21.3,4.
float y[] = {29.2,12.4,10.8,5,25.6,4,17.2,22.4,29.3,28.7,16.3,23,3.3,3.6,4.9,23.9,27.4,1.7,23.8,19
```

Line by line print the indices of the stars in each constellation separated with commas.
Once done modify the code so it doesn't use 2 arrays for the coordinates, instead it should use 1 struct array for the coordinates.

## Size of a closed space

Given the following figure (or something similar) where '.' shows empty spaces and '\#' shows walls. Write a function that calculates the size of the largest "http://wiki.math.bme.huroom"http://wiki.math.bme.hu (how many '.' characters it contains). This is a $37 \times 19$ table.

```
...#######################
. . . # . . . . . . . . . . . . . . . . . . . . . # . . . . . . . . . . .
. . .# . . . . . . . . . . . . . . . . . . . . . # . . . . . . . . . . .
. . .# . . . . . . . . . . . . . . . . . . . # . . . . . . . . . . .
. . . # . . . . . . . . . . . . . . . . . . . . . # . . . . . . . . . . .
. . . # . . . . . . . . . . . . . . . . . . . . . # . . . . . . . . . . .
...#.....................#######
. . . ### . . . . . . . . . . . . . . . . . ## . . . . . . #
...#..##. . . . . . . . . . . . .## . . . . . . . . # . . . . .
. . .# . . . .## . . . . . . . . . ## . . . . . . . . . . #
. . . # . . . ..## . . . . .## . . . . . . . . . . . . # . . . . .
. . . # . . . . . . . ##### . . . . . . . . . . . . . . # . . . . .
. ..# . . . . . . . .# . . . . . . . . . . . . . . . . . . #
. . . # . . . . . . .## . . . . . . . . . . . . . . . . . . # . . . . .
. . . # . . . . .## . . . . . . . . . . . . . . . . . . . . # . . . . .
. . . # . . .## . . . . . . . . . . . . . . . . . . . . . . #
...#############################......
```

