Prerequisites:

- Basic knowledge of using computer and text editor
- Knowledge how to create a program
- Knowledge how to create functions
- Knowledge of different types
- Knowledge about conditional statements
- Knowledge about loops (3 types)
- Knowledge about 1D arrays

Aims:

In this laboratory student will learn how to:

- use 2-dimensional (2D) arrays
- sort some values

1 Theoretical background

1.1 1D Arrays

To define and initialize some 5-elements array the following source code should be provided:

```
float Marks[5] = {2.0f, 3.0f, 5.0f, 4.5f, 3.0f};
```

It should be noted that in this case you can give in curly braces less elements than you declared in square braces

```
float Marks[5] = {2.0f, 4.5f, 3.0f};
```

and not given elements will be set to `0`. However, it is permitted to give more such elements in curly braces – compiler through an error that range of array was exceeded. Nevertheless, it is possible to let compiler to decide how many elements will be in a newly created array:

```
float Marks[] = {2.0f, 3.0f, 5.0f, 4.5f, 3.0f, 4.0f};
```

In this case we can skip exact number of elements given previously in the square braces and compiler counts these elements itself (in this case there will be created 6-elements array).

If you want quickly define a new array and set each element to `0` you can use the following source code:

```
float Marks[5] = {0};
```

In this case number of elements given in square braces is required.

1.2 Size of the Array

To check how big in bytes is some array the `sizeof` can be used.

```
float Marks[] = {2.0f, 3.0f, 5.0f, 4.5f, 3.0f, 4.0f};
printf("%u", sizeof(Marks));
```

The `sizeof` can be used to check how much memory (in byes) is used by different types of variables as well.

1.3 2D Arrays

To define a new array some name of this array has to be used and a sizes of each dimension of this array have to be given as well. The size is given in `[` and `]` braces separately for each dimension, similarly as for 1D array, and the minimum value is 2.

To define a new 2D array of name `My_array` which has 20 rows and 2 columns (total there are 40 positions in this array) the following instruction has to be provided into a source code:

```
char My_array[20][2];
```

1.4 Extra

The following items will be explained by teacher on some examples:

- sorting elements of some array
- `My_array + 2`
- `memset`

2 Exercises

1. Draw in the terminal a triangle of `*`, where as a top it will be only one `*` and as a bottom there will be the number of `*` given by user.
2. Ask user in the terminal to give size of a new 1D array. Create relevant array and put random `float` numbers as elements. Display then this array in the terminal separating each element by Tab.
3. Now modify your program – random numbers (different on each program's start) will be not greater than `100` and not smaller than `0`. Display this array in the terminal.
4. In a new program remove now all elements which belongs to set {50-100}. Display once again this array in the terminal and see differences.
5. Now try to push all elements to the beginning side of this array (let's say the left side) and display only non-removed elements. Removed elements could be then denoted for instance as `-1`.
6. Draw algorithm of this programs.
7. Write a new program where user at first will give the number of all elements and then will be providing these elements one by one in the terminal. Put these elements into two different arrays. In the first array there will be stored even elements, in the second array there will be stored odd elements. Display in the terminal the result of such a splitting.
8. Modify this program (it is suggest to write a new one) which will be sorting elements of both arrays from task no. 7 from smallest one to the biggest one.
9. Right now modify program from task no. 8 to sort these elements from biggest one to the smallest one.
10. Repeat task no. 7, however, this time collect all odd elements in the first row and all even elements in the second row of the array. Display after all this array in the terminal.
11. Are you able to draw a "snake" board and a snake itself? Let's say right now the game will not depend on time but only will be handled keys pressed by the user. Randomly chose places (some places only, not every places!) on the board where values (let's say from `1` to `6`) will be displayed. Each time the game starts these numbers will be placed in different places.