Prof. Dr.-Ing. Ralf Schenkel
Tobias Zeimetz
Trier University

Wintersemester 21/22

Exercises for the Class

Elements of Computer Science: Programming

Assignment 03

Submission of solutions until 3:00 p.m. at 24.11.2021
at `moodle.uni-trier.de`

- Every task needs to be edited in a meaningful way!
- Please comment your solutions, so that we can easy understand your ideas!
- If you have questions about programming or the homeworks, just ask you teachers!

Exercise 1 Evaluation: Numbers

In the given program a size and then a corresponding sequence of values are inputted into a field. (The corresponding part of the program is already preset in the program and should not be changed.)

Very often it is necessary to search for data in an array. The structure of the corresponding program parts then usually consists of three components: (a) the preparation of the search, (b) the actual search itself, and (c) an evaluation of the search.

As an example, you should now find and output the largest value among the numbers stored in an array. In this case, (a), (b) and (c) result as follows:

(a) Use a variable in which the maximum value can be stored and assign it a sufficiently small value (here, for example, any of the entered numbers (in the array) as a candidate for the maximum).

(b) Compare all saved values one after the other to see whether they are larger than the candidate found for the maximum (then: update the candidate if necessary).

(c) At the end of the loop: Output the maximum value found in this way.

- The Input **5 0 1 2 3 4** (i.e. 5 values from 0 to 4) returns the output **4**.
- The Input **9 4 6 4 2 0 2 8 6 2** returns the output **8**.
- The Input **1 10** returns the output **10**.

- The Input **1 0** returns the output **0**.
- The Input **2 10 5** returns the output **10**.

Exercise 2 Evaluation: Numbers

Again, first a size (≥ 0) and then a corresponding number of values are read into a field (which is already contained in the program).

For many purposes it is important that the data is sorted in an array (or other datatypes you will learn), i.e. `array[index] <= array[index+1]` always applies. You will learn many sorting algorithms later. But first we just want to know how unsorted the data is.

Now determine the number of "simple inversions" in the array of the data you read, i.e. the number of positions where `array[index] > array[index+1]` applies. First think about what the three components (a)-(c) of the previous exercise should look like. You can assume that the array will contain at least two numbers, i.e. the size of the array is at least 2.

- For the input **5 0 1 2 3 4** (i.e. 5 values from 0 to 4, all sorted in this case) there are 0 simple inversions; Therefore the output should be **0**.
- For the input **5 4 3 2 1 0** there are 4 positions with simple inversions, because $4 > 3$, $3 > 2$, $2 > 1$ and $1 > 0$; Therefore the output is **4**.
- The input **6 0 1 0 1 0 1** contains three simple inversions; The output is **2**.
- The input **6 0 0 4 4 2 2** contains only one simple inversion at $4 > 2$; The output is **1**.
- The input **2 42 42** contains no simple inversions; The output is **0**.

Exercise 3 Evaluation: Numbers

Again, first a size (≥ 0) and then a corresponding number of values are read into a field (which is already contained in the program).

As in the previous exercise, you should again search for the maximum value in the array, but you should now additionally specify how *often* this maximum value is contained in the array.

Examples:

- The input **5 0 1 2 3 4** (i.e. 5 values from 0 bis 4) returns the output **4 1**, because the greatest value 4 only occurs once.
- The input **9 4 6 4 2 0 2 4 6 2** returns the output **6 2**.
- The input **7 10 10 10 10 1 10 10** returns the output **10 6**.

- The input **1 0** returns the output **0 1**.
- The input **2 10 5** returns the output **10 1**.

Exercise 4 Evaluation: Numbers

Again, first a size (≥ 0) and then a corresponding number of values are read into a field (which is already contained in the program).

As with the above exercise, you should again search for the maximum value in the field, but you should now additionally specify *which indexes* contain this maximum value in the field.

Hint: Use two loops that are executed one after the other.

Examples:

- The input **5 0 1 2 3 4** (i.e. 5 values from 0 to 4) returns the output **4 4**, because the greatest value 4 is found on position 4 in the array.
- The input **9 4 6 4 2 0 2 4 6 2** returns the output **6 1 7**, because 6 is found on position 1 and 7 in the array.
- The input **7 10 10 10 10 1 10 10** returns the output **10 0 1 2 3 5 6**.
- The input **1 0** returns the output **0 0**.
- The input **2 10 5** returns the output **10 0**.