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

Wintersemester 21/22

Exercises for the Class

Elements of Computer Science: Programming

Assignment 04

Submission of solutions until 3:00 p.m. at 01.12.2021
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 Moon Landing (Evaluation: Numbers)

From the early days of programming originates the following small console game to simulate a moon landing:

A manned Lunar Module is to land on the moon. Important are the height $H(i)$ at the time i and the velocity $V(i)$ at the same time. i is the number of seconds since the beginning of the landing attempt.

At the beginning of the landing phase (time: $i = 0$ seconds) the Lunar Module has a height of $H(0) = 200\,\mathrm{m}$ above the surface and a starting speed of $V(0) = 0\,\frac{\mathrm{m}}{\mathrm{s}}$, i.e. it stands still at $200\,\mathrm{m}$ height above the ground. Due to the gravity of the moon it starts to fall (the velocity increases by $5\,\frac{\mathrm{m}}{\mathrm{s}}$ every second). Without countermeasures, it would hit the surface and be destroyed.

For the simplification we will not give the units m, s and $\frac{\mathrm{m}}{\mathrm{s}}$ in the following, moreover only integers are possible values. This allows you to use the already known number type `int`.

The pilot of course wants to gently touch down on the surface, so $H(j) = 0$ at a time j which is still unknown. The passengers should not be injured, therefore the speed $V(j)$ must not be more than 10 at the time of landing.

To achieve this, the pilot can give the Lunar Module limited counter thrust ($0 \leq T \leq 10$). To simplify the procedure, we assume that this thrust can be re-adjusted exactly once per second. In every flight second there is a thrust $T(i)$ entered by the pilot. If the pilot enters a thrust of $T(i) > 10$, the system reduces this to the maximum value of $T(i) = 10$. On the other hand a negative thrust $T(i) < 0$ is not possible, then $T(i) = 0$ is set.

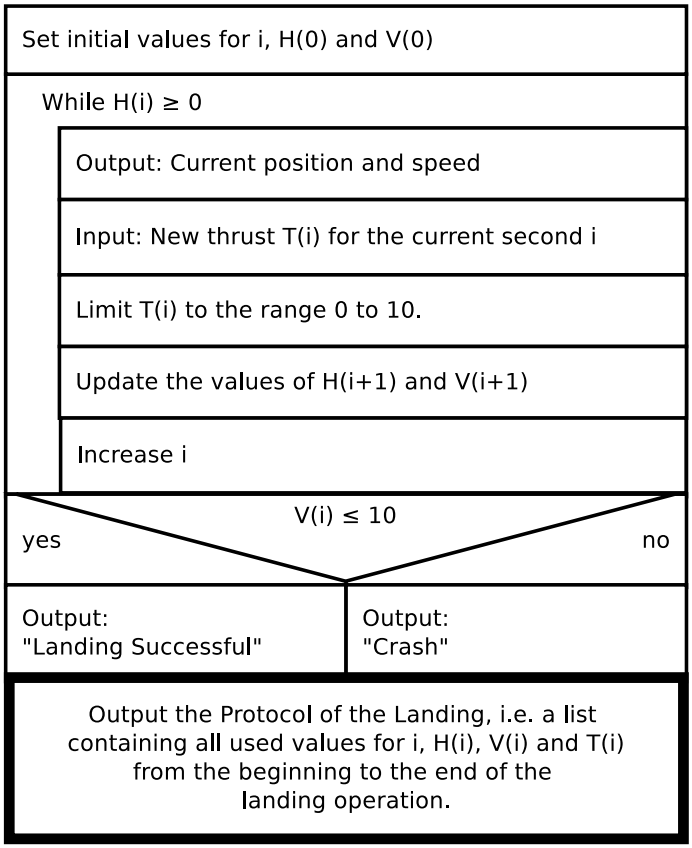
This results in new values for altitude and speed in every second, which are given by the following equations:

$$H(0) = 200, \qquad V(0) = 0$$

and the change of the values from each second i to the next second $i+1$:

$$H(i+1) = H(i) - V(i)$$
$$V(i+1) = V(i) + 5 - T(i)$$

Now implement the algorithm for interactive moon landing given in the following structure chart in a Java program:



Notes:

- In the given program there are already 3 fields H, V, T (each with 1000 components) defined, in which you can store the corresponding values. You can assume that the landing does not take $1000s$ and that the fields are large enough.
- To enter the values for the thrust, please use the Scanner.
- The time elapsed in the game is of course completely independent of the real time, the value of i is determined by the number of entries you make during the game.
- How you implement the last part of the structogram ("output of a protocol") is up to you.

An example of an attempted landing could look like the following:

```
H = 200, V = 0, Thrust: 0
H = 200, V = 5, Thrust: -1
H = 195, V = 10, Thrust: 0
H = 185, V = 15, Thrust: 0
H = 170, V = 20, Thrust: 0
H = 150, V = 25, Thrust: 0
H = 125, V = 30, Thrust: 0
H = 95, V = 35, Thrust: 0
H = 60, V = 40, Thrust: 20
H = 20, V = 35, Thrust: 20
Crash
Protocol of the values:
i = 0, H = 200, V = 0, T = 0
i = 1, H = 200, V = 5, T = 0
i = 2, H = 195, V = 10, T = 0
i = 3, H = 185, V = 15, T = 0
i = 4, H = 170, V = 20, T = 0
i = 5, H = 150, V = 25, T = 0
i = 6, H = 125, V = 30, T = 0
i = 7, H = 95, V = 35, T = 0
i = 8, H = 60, V = 40, T = 10
i = 9, H = 20, V = 35, T = 10
i = 10, H = -15, V = 30, T = 0
```

(For the evaluation using `moodle` only the sequence of the values in the above example is important, the exact formatting is not important!)

Exercise 2 (Evaluation: Numbers)

Write a Java program for processing fields whose values were given randomly (more precisely: pseudo-randomly), i.e. you don't know what values are stored inside the array.

First the (given) program takes three integers `size`, `limit` and `seed` (which are all > 0) as input.

Then an array `int[] array` of size `size` is created and filled with numbers from the range `0` to `limit`, which are determined randomly. A random number generator is used, which is initialized with `seed`. If you use always the same values for `seed`, the same random numbers will be used. But if you change the value of `seed` also the random values will change completly.

Up to this point, the program has been predefined. You should now find the smallest and largest value in the field and then output it, for example:

```
Array Size:  5
Limit: 20
Seed: 6
Minimum: 1
```

```
Maximum: 18
```

A limit of 20 and a random start value `seed`= 6 lead to the field values `11 16 6 18 1`. If, however, you take 9 as `seed`, the field values `9 16 8 15 19` result, and the output will look as follows:

```
Array Size:  5
Limit: 20
Seed: 9
Minimum: 8
Maximum: 19
```

Note: First write a loop in which the field values are displayed to the console, so that you can see which numbers you are working with! This output must of course be removed afterwards.

Exercise 3 (Evaluation: Numbers)

Modify the solution to the previous task so that it shows at which position the minimum was first seen (smallest corresponding index) and when the maximum was last seen (i.e. largest corresponding index). For the formatting of the output, please refer to the following examples:

```
Array Size:  5
Limit: 20
Seed: 6
Minimum 1 for the first time at position 4
Maximum 18 for the last time at position 3
```

respectively

```
Array Size:  5
Limit: 20
Seed: 9
Minimum 8 for the first time at position 2
Maximum 19 for the last time at position 4
```

Exercise 4 (Evaluation: Numbers)

Modify your solution from the last task. Instead of printing only the minimum and maximum values and their corresponding positions, you have to do the following:

- In case the index of the first appearing minimum is less than the index of the last appearing maximum: Print all numbers stored in the array that occur after the position of the minimum until you reach the maximum (excluding) of the array.

- In case the index of the first appearing minimum is greater than the index of the last appearing maximum: Print all numbers stored in the array that occur after the position of the minimum until you reach the end of the array.

You can assume, that minimum and maximum are not equal and the array is filled with multiple numbers.

For the formatting of the output, please refer to the following examples: A limit of 20 and a seed of 9 lead to the field values `9 16 8 15 19` and the output will look as follows:

```
Array Size:  5
Limit: 20
Seed: 9
Output: 15
```

A limit of 10 and a seed of42 lead to the field values `0 3 8 4 0 5 5 8 9 3` and the output will look as follows:

```
Array Size:  10
Limit: 10
Seed: 42
Output: 3 8 4 0 5 5
```