8 Unit 8: Program Organization

Having covered functions in the previous Unit, we are ready to attack more challenging issues of program organization. A program may consist of arbitrary many functions, that may make use of each other through function calls. A function is always wise to be introduced, when the same function (and its implementation) is used more than once in the program. A function may also be introduced for the sole purpose of structuring your program even if it is only called once. The use of internal and external variables, static and auto variables, and scopes (of variables) are addressed here as well. The interplay of those elements and functions allow to divide a program and building blocks of suitable size.
The first of the following programming projects makes use of functions that can be found in the corresponding Book Chapter 10. The last programming project is a simple form of artificial intelligence, where a cute little robot, the so called "Array Walker" finds his way - but he is not the only one, who is trying to do so, as there are two. The function that defines the behaviour allows through parameters to set a specific "behaviour". We do not want to push this too far, but one can easily imagine that this could be extended to a more complex simulation, for example a traffic simulation, or the behaviour of ants - in its present form it is most similar to Tron. As for now, the purpose of this last programming project is to combine the many different kinds of idioms and language constructs we have gone through in this lecture - and enjoy programming and running the program with different parameters. Can you guess what the results will be?

8.1 Project 17: Automatic Array-Walker

Modify programming project "Array Walker" to put a computer controlled "automatic array-walker" on your field (grid). Instead of keyboard-inputs it should operate independently as follows: It has 3 parameters, s, d, and t, which allow to configure the robot.

- Parameter s: The maximum number of steps it goes into the same direction before reconsidering its walking direction. It continues in another direction, updating parameter d according to parameter t. It does this until it finds a free way or comes to a halt. If it comes to a halt, the program prints the final board and terminates.
- Parameter d: This parameter determines into which direction the walker starts walking. For d values 0-3 are allowed, corresponding to north, east, south, west. It continues with this parameter until either the maximum number of d consecutive steps is reached or if the way is blocked - by the border of the board or by some marked field. In this case p is increased or decreased depending on parameter t. The walker comes to a halt if no open way can be found.
- Parameter t: Determines whether d is increased ($t = 1$) or decreased ($t = 0$) if the way is blocked.

The program allows to enter a "walker" on the field. The path of the walker is marked by using letters A-Z (multiple times if more than 26 steps can be performed). The size of the board is 10x10. The program prompts the user to enter the walker's parameters and

computes the walker's path until it cannot continue to move. Note that a walker cannot go to the same location twice (i.e. it cannot visit a marked location).
The program prints only the final board. The final board is the one where the walker came to a halt. The whole trace is shown on the board.
Example:

```
% u8_autowalker
Max consecutive steps: 4
Direction initializer: 0
Turn direction: 1
Start Pos X: 2
Start Pos Y: 2

Final board:
..CDEFG...
..BWVUH...
..AXYTI...
..PQRSJ...
..ONMLK...
..........
..........
..........
..........
..........

%
```