Programming 2

Christian Grévisse (christian.grevisse@uni.lu)

Lab 3 – Arrays

Exercise 9 – Frequency Analysis (based on Exercise 1-14 in *The C Programming Language*) #ArrayType

Write a program that prints a histogram of the frequencies of different characters in its input. You can gather the input character by character with the `getchar()` function until reaching `EOF`. Only consider the 26 characters of the latin alphabet, other characters can be skipped. Implement your program to be case-insensitive, i.e. do not distinguish between uppercase or lowercase letters.

To test your program, you may, e.g., send the content of a file via piping to your executable (cf. Operating Systems 1), or enter letters until hitting ctrl + D.

Exercise 10 – Merging Two Sorted Arrays #ArrayType #PreprocessorMacroDefinition #RandomNumbers

Write a C console application performing the following tasks:

1° Create two arrays (t_1 and t_2) of length l_1 and l_2. l_1 and l_2 are constants that are defined in the program. Make sure that $l_1 \neq l_2$.

2° Write a function `void fill_array(int array[], int size)` that fills the array of given size with random numbers between 0 and 99. Use the function in your main program to fill both t_1 and t_2.

> ⚠ **Say whaaat?**
> You will notice that although you do not return the sorted array explicitly, the changes will be reflected outside the function.
> **Q:** Didn't you tell us in Lab 2 that *the C programming language employs call by value when passing arguments to functions*?
> **A:** Yes, but arrays are different to primitive types. Stay tuned for the explanation in one of the next lectures!

3° Write a function `void sort_array(int array[], int size)` that sorts the given array (you may choose your favorite sorting algorithm).

4° Write a function `void print_array(int array[], int size)` that prints the given array.

5° In the main program, sort both t_1 and t_2 and print them to the console.

6° Merge both arrays and store the result in a new array t_3, which at the end shall also be sorted, but without calling the `sort` function on it. To do so, use a single loop that accesses all elements of both arrays. At each step, the program shall then choose the smallest element that has not yet been inserted into t_3 and insert it.

7° Print t_3 to the console.

Your solution must be able to handle arrays of arbitrary sizes, i.e. it shall be easily possible to change l_1 and l_2 without requiring lots of modifications throughout the code.

Exercise 11 – Mini-Minesweeper #ArrayType #PreprocessorMacroDefinition #RandomNumbers

Write a game similar to Minesweeper with a grid, represented by a two-dimensional array. The size of the grid is a constant value. Fill the grid with mines. The probability of encountering a mine in the grid is another constant value. The size of the grid as well as the probability of encountering a mine should be easily changeable in your code.

The user shall be constantly asked for x and y coordinates. The game goes on until he hits a field with a mine, in which case he loses, or he has uncovered all empty fields, in which case he wins.