Session 1 Introduction To C

1.1 Exercises

Exercise 1 Write a C program that asks the user for an integer and iteratively computes the factorial of this integer.

Exercise 2 Write a C program that prints the array 1, 2, 3, 4, 5 on a single line separated by spaces. Ask the user for an integer, multiply all elements of the original array with this integer and print the array again. Now define a function to print the array to avoid duplicating your code. Don't hard-code the size of the array in the code. Hint: If you encounter problems calculating the size of the array inside of a function, try to understand the warning message given by gcc and think of an alternative solution.

Exercise 3 Write a C program that asks the user for an integer. Print the address, the value and the size in bytes of this integer. Now store the address of this integer in a pointer. Then print the address, the value and the size in bytes of this pointer.

Exercise 4 Define a struct containing a float, double, long, string and a single char. Print the size of the struct. Create a new instance of this struct and print the addresses of each field. Draw the memory lay-out chosen by the compiler. Did the compiler introduce padding[1]? If so, where and how much?

Exercise 5 Write a C program that asks the user for a string. You can use the function *fgets* to read a whole string from the console. Print the hexadecimal representation of each character in the string in order. Verify the output using an ASCII table (`https://ascii.cl/`). Hint: use the format string "%02x". The last printed character will be the line feed (hex 0x0a).

Exercise 6 Write a C program that asks the user for a string and outputs the length of the string. First write a version using the function *strlen* declared in the header *string.h*. Then create a second version where *strlen* is not used. Note that the last character of the string will be the line feed (hex 0x0a).

[1]Padding in this context means the introduction of free unused space in between variables stored in memory, usually for low-level performance reasons