Electrical Engineering and Information Technology, B.Eng.
Introduction to the C Programming Language: Exercises

Exercise Sheet 5

---

1. Exercise. a) Understand and test the following program.

b) Write a variant of the program, using now the switch-statement.

```
#include <stdio.h>

int main(void)
/*******************************************************/
/*                                                     */
/* Counting digits, white spaces and other characters. */
/*                                                     */
/*******************************************************/
{
char    c;
int     i, index;
int     no_white, no_other; /* Number of white spaces....*/
int     no_of_digit[10];    /* Number of digit           */

        /* Initialization */
        no_white = 0;
        no_other = 0;
        for (i = 0; i < 10; i++) {
                no_of_digit[i] = 0;
        }

        /* Read and count */
        c = getchar();
        while (c != '$') {
                if ('0'<= c && c <= '9') {
                        index = c - '0';
                        no_of_digit[index]++;
                }
                else if ((c==' ') || (c=='\n') || (c=='\t')) {
                        no_white++;
                }
                else {
                        no_other++;
                }
                c = getchar();
        }

        /* Ausgabe */
        printf("digits =");
        for (i=0; i < 10; i++) {
                printf(" %d, ", no_of_digit[i]);
        }
        printf("\n");
        printf("white space = %d, other = %d\n",
               no_white, no_other                  );
        return 0;
} /* END_main() */
```

2. Exercise. a) Understand and test the program.

b) Change the program to allow float numbers as base. The exponent can now be in the hole range of int, that is it can be negativ.

```
/*** This file contains ***/
int  main(void);
int  power(int base, int expo);

/* Implementation */

int main(void)
/*********************************************/
/*                                           */
/* Demonstration of the power-function       */
/*                                           */
/*********************************************/
{
int     i;
int     pow_of_2, pow_of_3;

        for (i = 0; i < 5; i++) {

                pow_of_2 = power( 2, i);
                pow_of_3 = power(-3, i);
                printf("Power of 2=%d, Power of -3=%d\n",
                        pow_of_2, pow_of_3               );
        }
        return 0;
} /* END_main() */

int power(int base,
          int expo)
/*********************************************/
/*                                           */
/* power: base to the power of expo; expo >= 0.*/
/* expo < 0 not caught.                      */
/*********************************************/
{
int     i, p;

        p = 1;
        for (i = 1; i <= expo; i++) {
                p = p * base;
        }
        return p;
} /* END_power() */
```

3. Exercise. Our program from the 2. exercise shall be distributed on two sourcefiles. source1.c shall contain main() , source2.c shall contain power(). The prototype of power() is now declared in source1.c as

```
extern int  power(int base, int expo);
```

Find out, how to compile and to link source1.c and source2.c in your development environment.

4. Exercise. a) For a positive integer n, the sum of the numbers from 1 to n has to be computed. E.g. if n == 3, the requested sum is 6 . Please develop a *recursive function* sum(n), which returns the requested sum. Test your function in a main().

b) Find a *nonrecursive* solution for example E.4-1.

c) I assume, the recursive Fibonacci function is known to you from the foundation year. If not, you can program it and try the function with different parameters.

$$fib(n) = \left\{ \begin{array}{c} 0\, if\ n = 0 \\ 1\, if\ n = 1 \\ fib(n-1) + fib(n-2) \end{array} \right\}$$

d) Develop a recursive version power_recursive(int base, int expo) of our function power(int base, int expo) from the 2. Exercise.