Lab 2 – Basic C programming

Lab targets: writing basic C programs, build them with gcc, run and test the programs.

Exercise: *datatype size, type qualifiers*

- The memory size of the built-in datatypes int, float, etc. is system dependent. Write a program that prints the memory size of int, float, double, void, and pointer.
- What happens to the memory sizes when you use the type qualifiers short and long?
- Find out if a char is an unsigned or signed data type on the system you use.

Exercise: *formatted output, compact code*

Implement a program that prints the ASCII table in character, decimal, hexadecimal, and octal format. Remember that some ASCII-values have no printable character code: just print a space in this case. [Hint: check out the 'isprint()' library call.]
Try to write very compact code. (Actually, you can write this program in less than 10 lines of code, blank lines and comments not included.) The output should look like:

```
char  dec   hex   octal |     char  dec   hex   octal
-----------------------------------------------------
      0     0x0   \o0   |           1     0x1   \o1
      2     0x2   \o2   |           3     0x3   \o3
      4     0x4   \o4   |           5     0x5   \o5
      6     0x6   \o6   |           7     0x7   \o7
      8     0x8   \o10  |           9     0x9   \o11
      10    0xa   \o12  |           11    0xb   \o13
      12    0xc   \o14  |           13    0xd   \o15
      14    0xe   \o16  |           15    0xf   \o17
      16    0x10  \o20  |           17    0x11  \o21
      18    0x12  \o22  |           19    0x13  \o23
      20    0x14  \o24  |           21    0x15  \o25
      22    0x16  \o26  |           23    0x17  \o27
      24    0x18  \o30  |           25    0x19  \o31
      26    0x1a  \o32  |           27    0x1b  \o33
      28    0x1c  \o34  |           29    0x1d  \o35
      30    0x1e  \o36  |           31    0x1f  \o37
      32    0x20  \o40  |     !     33    0x21  \o41
"     34    0x22  \o42  |     #     35    0x23  \o43
$     36    0x24  \o44  |     %     37    0x25  \o45
&     38    0x26  \o46  |     '     39    0x27  \o47
(     40    0x28  \o50  |     )     41    0x29  \o51
*     42    0x2a  \o52  |     +     43    0x2b  \o53
,     44    0x2c  \o54  |     -     45    0x2d  \o55
```

etc …

Exercise: *random numbers, the time() function*

Implement a countdown-program. Generate a random number 'count' in the interval [MIN, MAX], with MIN and MAX constant values. Next a loop performs a countdown from 'count' to zero after which a number of beeps are generated.
Hint: use the library function srand() to initialize the pseudo-random generator with the result of time(NULL); use rand() to generate random numbers; finally, print character \a to generate a beep.

Exercise: *system() library call, Linux console commands*

Open a Linux console terminal and use the man-pages to get information on the following Linux commands: clear, ls, sort, more.

Now execute the following command sequence:

```
clear
ls –l > files.txt
sort files.txt | more
```

The symbol > is the ‘redirection’ command, sending the output of a command to a file.

The symbol | is the ‘pipe’ command and uses the output of the left command as input for the right command.

Implement a C program that executes this sequence of commands from within the main-function using the system() library call of stdlib.h.

Use a reference manual or the man-pages on C to find more information on the system() library call.

Exercise: *Tokenizing strings*

Implement a program that can tokenize a text sentence in words and prints every word on a new line. Use the string function ‘strtok’. The following is an example output:

Exercise: ***flow control and functions***

THE SOLUTION OF THIS EXERCISE NEEDS TO BE UPLOADED ON TOLEDO BEFORE THE NEXT LAB.

Implement a coca-cola vending machine simulator. The machine offers the following drinks (i) coca-cola (80 cent), (ii) aquarius (1 euro), (iii) orange-juice (1 euro), (iv) water (65 cent), and (v) lemonade (80 cent). The machine accepts only coins of value 5 cent, 10 cent, 20 cent, 50 cent and 1 euro. The machine doesn’t return any money! The program that simulates the machine indicates (on the command line) to the user:

-if no money is inserted: “waiting for input …”;

-if money is inserted: the amount of money that is inserted so far is displayed;

The simulator is waiting on a new line for money insertion. Money insertion is simulated by entering a number representing the value of the coin in ‘cent’, followed by the ENTER key-stroke. A soda is selected by entering the first letter of the soda (c, a, l, w or o) followed by the ENTER key-stroke.

Extensions to this exercise:

-Sometimes a coca-cola machine refuges an inserted coin. Simulate this behaviour.

-Coin insertion is simulated by entering the value of the coin followed by ‘c’ to indicate cent or followed by ‘E’ tot indicate euro. E.g. 10c or 50C or 1E. The coin-scanner should not be case-sensitive.

-The simulator can return money.