Fall 2019
Sébastien Combéfis

E301B C Programming

Competencies List

This document provides the list of basic and advanced competencies, with a precise description, that can be acquired through the *E301B C Programming* activity.

Basic Competencies

Basic competencies are specific to a teaching unit or activity and a 100% mastery level for all of them is required to succeed the teaching unit or activity (10/20).

Code	The learner is able to...
Programming	
CP001	declare, initialise and use a variable in a program.
CP002	write constant values with the relevant literal forms for integer, floating-point and character.
CP003	build a formatted output and print it to the standard output with the `printf` function.
CP004	use correctly the arithmetic, comparison and logical operators.
CP005	use correctly the integer division and modulo operators.
CP006	use `if-else` and `switch` conditional statements, and “translate” from one to the other.
CP007	use `while`, `do-while` and `for` iterative statements, and “translate” from one to the other.
CP008	perform conversion between data types, implicitly or with the cast operator.
CP019	declare and use one-dimensional arrays and use the access operator.
CP010	use simple “direct” pointers and the address (`&`) and dereference (`*`) operators.
CP011	use pointer to pass parameters to a procedure or function by reference.
CP012	use pointer arithmetic to manipulate pointers.
CP013	use the `NULL` pointer.
CP014	define and use a structure on the stack and on the heap.
CP015	declare and use two-dimensional rectangular arrays and use the access operator.
CP016	use procedures and functions from the stdlib given their specification.
CP017	read and write text files, and manage errors.
Using GCC	
CP101	write, compile and execute a single source file C program with the command line.
CP102	create an object file from a source code file.
CP103	understand the compilation chain converting one source file to one executable file.
CP104	understand and use the `#include` and `#define` (for constants) preprocessor directives.
Memory representation and structure	
CP201	understand how to talk about information quantities and use the correct units.
CP202	understand the positional notation and write and convert a number between different bases.
CP203	understand and compare different ways to represent integer.
CP204	write an integer and perform fundamental arithmetic operation with two’s complement.
CP205	write a real number and perform fundamental operation with floating-point arithmetic.
CP206	understand how implicit and explicit conversion are performed between data types.
CP207	understand how variables and their values are stored in memory.
CP208	understand how procedure and function calls are managed with environments stack.
CP209	understand the concept of addressing and pointer.

Code	The learner is able to...
CP210	understand and manipulate dynamic memory with `malloc` and `free` functions.
CP211	understand the differences between the stack and heap memory areas.
Code architecture and quality	
CP301	deal with the eight basic data types and choose the most adapted one to store a given data.
CP302	define a procedure or a function with or without parameters given a specification.
CP303	define prototypes for procedure and function and structure the code to use them.
CP304	handle rigorously the errors when calling procedure and function.
Debugging	
CP401	understand and explain a given basic source code.
CP402	understand basic compiler errors and warnings and fix the code accordingly.
CP403	find a logical error by printing and logging useful and relevant information.
CP404	understand what is stack overflow and identify and fix such bug.
CP405	understand what is buffer overflow and identify and fix such bug.

Advanced Competencies

Advanced competencies could be transversal to several teaching units or activities and increasing the mastery level of any of them is global to all the teaching units and activities where it is declared.

Code	The learner is able to...
Programming	
CP018	use pointer of pointer and define non rectangular two-dimensional arrays.
CP019	define and use complex data structure with arrays, structures and pointers.
CP020	define and use `enum` and `union`.
CP021	define a new type with `typedef`.
CP022	declare and manipulate strings with procedures and functions from the stdlib.
CP023	read and write binary files, and manage errors.
GP001	write a readable program consisting of a single source file.
GP002	produce a nice, beautiful and well-formatted output for a command-line program.
GP003	use adequately syntactic sugars (shortened operator, ternary operator, etc.).
GP004	simplify boolean conditions with logical equivalences.
GP005	use adequately the short-circuit property of logical operators.
GP006	use correctly bit manipulation operators.
GP007	define and use “flags” variables thanks to bit manipulation operators.
GP008	use pointers to procedures or functions.
Using GCC	
CP105	generate the `.i` and `.s` intermediate files and explain the relation with the source code.
CP106	define and use macros with the `#define` preprocessor directive.
CP107	compile several object files and link them together to get an executable file.
CP108	understand the full compilation chain.
Memory representation and structure	
CP212	understand the relation between arrays and pointers concepts and notation equivalence.
CP213	write a code that is free of memory leaks and check it with `valgrind`.

Code	The learner is able to...
GP201	understand and explain overflow situations for operations on integers.
GP202	understand and explain the limitations of floating-point numbers.
GP203	understand what is endianness and what are the impacts of this concept.
GP204	define and use a list implemented by a chained structure.
Code architecture and quality	
CP305	define header files and implement them.
CP306	define a structure with associated procedures and functions (POO-like approach).
CP307	write a code without any warnings when compiling with `-Wall`.
GP301	write robust code with good error management.
Debugging	
CP406	debug a simple program with `gdb`.