ENCE360 Lab1: C revision

1. Objectives

The overall goal of this lab is to revise your experience with simple C programs. The subsequent labs and assignments build upon this knowledge and assume you have a strong familiarity with the topics here. It is an opportunity to assess your knowledge in C and put some time to brush up on the basics.

Note that quiz questions for this week are based on the topics covered in the lab.

The aims of this lab are for you to revisit:

a. Using pointers
b. Structures and typedef
c. Memory allocation
d. Strings
e. Function pointers
f. Basic data structures

2. Preparation

Download and extract your Lab 1 files from "*Lab 1 C revision.zip*" on Learn.

NOTE. The files used in this lab:

a. vector.c
b. buffer.c
c. linked_list.c

Compiling programs:

compile with: gcc <program>.c -o <program> -std=c99 -Wall -Werror
run with ./<program>

Where <program> is replaced by the C file you're compiling (one of vector, buffer or linked list)

3. Program vector.c

An exercise in memory allocation and pointers. Your goal here is to implement two functions, new_vector(), which allocates space for a vector, and add_vector() which builds on new_vector (in order to allocate the result) adds two vectors together.

4. Program buffer.c

An exercise in memory management and handling strings. String handling can be awkward in the C language, because memory needs to be manually allocated and free'd. Handling binary data is more tricky, and normal use of string functions (which expect a null terminator) becomes problematic. The goal here is to implement copy_buffer() which copies a buffer of binary data.

5. Program linked_list.c

An exercise in data structures and function pointers and memory management. Your task is to write a function map_list() which takes a linked list and a function pointer and returns a new linked list in the same order, with the values in the new list transformed by the function passed in.

The second task is to write a function to clean up, to free a linked list. Memory management can be a pain in C, but it has to be done. Write a function to free a linked list and any of the memory allocated in it's creation by implementing free_list().

Test your modified program for memory leaks using Valgrind.

Run Valgrind using the following command: valgrind --leak-check=yes ./myprogram

Where myprogram is the name of the program you are testing.