**Trent University: Operating Systems (COIS3320)**

**Dr. Bin Guo**

**Lab 2: Cache-Friendly Programming**

Cache-friendly C Programs

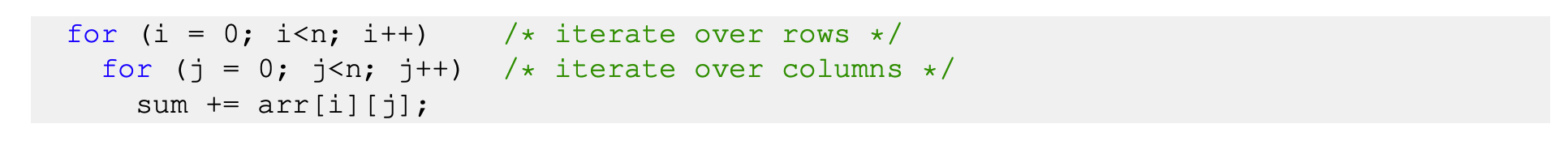
**Outline**

This lab sheet covers C programming, and in particular how to improve the performance of a C program by improving its cache behaviour.

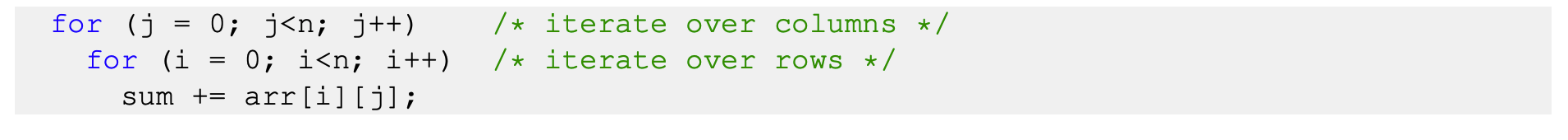
It is impossible to explicitly control the cache when executing the program. However, we can write cache-friendly program to potentially leverage the cache for high performance.

**Samples of cache-friendly C programs**

In the lecture on the Memory Hierarchy we discussed two versions of computing the sum over all elements in a matrix arr of size n×n. The first one uses row-wise iteration like this:



and the second uses column-wise iteration like this:



**Task: Compare the cache performance of these programs**

Write a C program, implementing each of the above functions to compute the sum over all matrix elements. You will need to write a function to generate a random matrix of a given size as test input. Then complete the two functions for doing the sum, one row-wise, one column-wise, based on the core code above. Finally, add timing commands to measure the time for executing the sum operation (use the function **clock()**, imported from **time.h** to get a time-stamp).

Measure and compare both versions of the program. Explain why you see a big difference in performance for both versions.

**Note**: your matrix should be large enough, e.g. 100,000,000 items, to measure the running time (not overflow the memory). Otherwise, the whole matrix can be loaded into L3 cache (always be cache-friendly). Typically, over 100ms is easy to measure. Less than 1ms is impossible to measure.

If you are not familiar with C programming, just ask help for our AA or ask ChatGPT.