Exercise 7 – Arrays of Arrays

Objective

The objectives of this session are to consolidate your understanding of arrays of arrays in C and to appreciate the efficiency of code produced by the compiler in the presence and absence of an optimiser.

Reference Material

This chapter is based on material in the chapter entitled "Arrays of Arrays". You might also like to refer back to the Arrays chapter and the Pointers chapter for background information. This practical session is located in the following directory:

	Microsoft Windows	*Linux*
Directory:	**c:\qacadv\multi**	**~/qacadv/multi**
Solution directory:	**c:\qacadv\multi\Solution**	**~/qacadv/multi/Solution**

Overview

This exercise is very similar to that completed in the practicals for the Arrays chapter. You may have noticed only small differences in the timings for the different methods of initialising an array. When these same methods are applied to an array of arrays the timing discrepancies become much more apparent.

Practical Outline

On Microsoft Windows open **arraytim.sln, on Linux** change your current working directory.

Look at the code fragments and comments already written in **arraytim.c**. Four functions are called, `withIndex()`, `withBasePointer()`, `withArrayPointer()` and `withMemset()`. To avoid both nightmares and excessive hair-ripping the `withArrayPointer()` function, which uses array pointers to initialise the array, has already been written.

The `withIndex()` function should be written to use array/index notation (i.e. `a[i][j][k] = 0`).

`withBasePointer()` should be written to use pointer/offset notation (i.e. `*p = 0`). You will need to initialise the pointer to the very first element of the array of arrays of arrays.

`withMemset()` should be written to invoke the `memset()` function. You will need to calculate the total size of the array before the invocation. Don't forget that `sizeof(a)` will not work.

There is a `checkArray` function which will ensure the array is initialised properly.

Do you notice a larger disparity between the timings for the different functions? As before, when you have the code working, enable and disable the optimiser and make a note of the timing differences to each routine.

On Linux the supplied makefile compiles five versions of arraytim:

arraytim	no-optimisation
arraytim.1	Level 1 optimisation
arraytim.2	Level 2 optimisation
arraytim.3	Level 3 optimisation
arraytim.s	Optimised for size