Week 8 Exercises : Pointers

Mar 26th 2012

1 Approach

As we move into the second term we want you to look at larger programs that you build up in steps. The first task is to be completed by everyone on the course. The second task is optional for students who want to progress further. Each of the steps inside a task corrosponds to the amount of work in an exercise last term.

Task 1 : 2D sorting

In this task you will write a program to sort data in a 2D array, and show the output for each step of the problem. To solve the sorting problem you will write two separate sort function; a function to sort a single column of the 2D array, and a function to sort a single row of the 2D array. Many parts of this task should be familiar from previous weeks (how to write the sorting functions) and the idea is to gain experience of the new syntax that you will need to manipulate 2D arrays. You need to sub-divide this problem into smaller stages and then tackle them in a sensible order. Here is a suggestion:

1. Write a program that allocates a 2D array and fills it with random data.
2. Write a function that outputs the 2D array to the screen.
3. Write a function that sorts a single column of the array.
4. Alter your program so that it calls the sort on each column, displaying the result of each step.
5. Write a function that sorts a single row of the array.
6. Alter your program so that it sorts a row, displays the result, sorts a column, displays the result etc ...
7. The final stage is to detect when the array is completely sorted, and exit the program.

Task 2* : Game of Life.

Conway's Game of Life is a simple simulation of a cellular automaton. The state of the simulation is a 2D array of cells. Each cell is in one of two states (alive or dead). The simulation proceeds in steps, for each step the new state of the simulation is calculated from the previous state. Each cell has eight neighbours (diagonals are included as well as up, down, left and right). The state of a cell is decided by the state of the neighbouring cells. For cells that were alive in the previous generation the following rules apply:

0 or 1 alive neighbours means the cell becomes dead.

2 or 3 alive neighbours means the cell stays alive.

4-8 alive neighbours means the cell becomes dead.

For cells that are dead in the previous generation the following rules apply:

0-2 alive neighbours means the cell stays dead.

3 alive neighbours means the cell becomes alive.

4-8 alive neighbours means the cell stays dead.

Once a new generation of cells has been calculated the previous generation is no longer needed. This implies that you will need two separate 2D arrays to hold the state. It is not possible to use a single array as you would be updating cells and erasing their old values before they could be used to calculate their neighbours.