School of Information and Physical Sciences

Inft2012 Application Programming – Notes for week 7

Pair programming

Remember, programming in pairs – and taking turns at the keyboard – is known to be beneficial when learning to program and to enhance productivity in actual programming. You are strongly encouraged to work with a partner on the exercises each week. The assignment is set this week: you and your partner should start working on it together. If you don't have a partner, the lab tutor might help you to find one.

Lab exercises

Remembering that learning to program takes lots of practice, you are strongly advised to finish the first four exercises before next week's lab class, bringing to the class any problems you encounter.

1. Assuming that a program has an array of integers called iNumber, consider the following code:

   ```
   bool bValid = true;
   for (int i = 0; i <= iNumber.Length – 2; i++)
       if (iNumber[i] > iNumber[i + 1])
           bValid = false;
   ```

   (a) What is the purpose of the variable bValid?

   (b) Why does the loop only go as far as iNumber.Length – 2?

   (c) What does the code do? That is, what is its purpose?

2. Write a program that accepts a number of words as input, in turn, from a single textbox – then displays those words in the opposite order, ie starting with the last one that was input and finishing with the first. Use an array to do this. You can assume that no more than 10 words will be entered.

3. Use the Random class to create a sequence of integers in the range 1-20 inclusive. You do not need to store the integers, just to display them as they are generated. The following lines of code might be of some help, but they might need adjusting.

   ```
   Random rand = new Random();
   int iValue;
   iValue = rand.Next(1,20);
   ```

   Now alter the program so that instead of displaying the numbers it produces 100 numbers and counts the number of times each integer in the range appears. You will need to decide how best to display the result. How many 1s would you expect to see? How many 13s? How many 20s? If you're not sure, check C#'s Help, or revise lecture 3.

4. A beachside block of flats has 8 floors, numbered 1 to 8. On each floor are 1-bedroom flats, 2-bedroom flats, and 3-bedroom flats. The weekly rental for a flat is calculated by the formula $(250 + 30f)(1 + n/4)$, where f is the floor number and n is the number of bedrooms in the flat.

   Create a 2-dimensional array of rental prices, with floor number on one dimension and number of bedrooms on the other. Fill the array with the rentals calculated by the formula.

(a) Permit the user to nominate a floor number and a number of bedrooms, and display the weekly rental for the flat in question. Do not recalculate this value – take it from the filled array.

(b) Find a way of displaying the full table of rentals on your form, with properly labelled axes.

5. There is no exercise 5.

6. *Challenge question*: create a 25x25 array of integers and a screen representation of the array (perhaps something like the Game of Life demonstration form), and use them to implement the Snake game.

   The snake starts off as a sequence of, say, 5 adjacent cells, one of which represents its head. It moves headfirst in a direction specified by the user. You might set up four arrow buttons to move the snake's head up, down, left and right. When the snake turns, the first tail character follows the head, and each successive tail character follows the one before it, so each character of the tail changes direction exactly where the head changed direction.

   There will be some random single-digit numbers scattered around the board. When the snake's head hits a number, it eats it, and its length increases by that number. So if a snake 13 characters long hits a 5, it becomes 18 characters long. The tail doesn't grow instantly, but for the next 5 moves the end of the tail stays where it is as the snake becomes longer. Then the end of the tail starts moving once more as the head moves.

   If the snake hits a wall or itself, it dies. The player's goal is to make the snake as long as possible without it dying.

7. If you got that far, try modifying the game so that the snake moves once for every tick of a timer, and instead of four arrow buttons there are only two, one to turn the snake to its left and one to turn it to its right. These directions will, of course, be different from the player's perspective, depending on which way the snake is moving when the button is clicked.