**TAS – Software design and development – Part 4 transcript**

## Data structures continued: Arrays and records

(Duration:17 minutes 16 seconds)

Okay, last series of videos, we looked at the control structures, the sequence selection and iteration, and the algorithms that we need to use to process the data that we're storing. Now we're going to have a look at the data structures, we've looked at data types, the types of data that we store into the variables in the arrays. And we're going to look at arrays and records. Okay, so an array is a collection of variables of the same type, they call them homogeneous types. For instance, an array of integers is a collection of integers, an array of strings is a collection of strings. The elements of the array are accessed using an index value, and each element of the array behaves just like an instance of that type. You'll notice that in some languages, array indexes start with zero, certainly in C and Arduino if you're using those. And that's referred to by the index or the subscript. You'll see on this diagram, we have an array of six elements, zero through to five. And we've got a variable, and we're using these cylinders as just a way of visualising, I suppose, the storage of the values. And again, we've got a small shape to lead, in this case, we're referring to integers. If you recall back to the data types in the previous video. Okay, let's have a look at arrays, and a card game is a really good way of thinking about arrays. And let's look at the value of them. Imagine if we had a computer game and we wanted to represent every single card in the deck of 52 cards. We could set it up with simple variables, but that would be a lot of typing to allocate or assign the value to each variable. 52 variables from the ace of clubs through to the king of spades. That's quite a bit of typing. The 52 separate variables would be really difficult to shuffle and deal using code. So there must be a better way. A much better way would be to have an array of variables instead of 52 separate ones. And a variable would specify the index of the array to control how we use the deck. So for example, print deck N would print the value of the card that's in the variable index. We could set up the deck using an array, and would be so much easier if you have a look at the for index=1 to 52 deck index, it was index, next index, so much easier to assign those values. So how then would we represent the different ranks and suits if we were to use this model to represent the cards? Well, we could describe the cards as integers one to 52, and then have some simple calculations to determine the suit of the card. And the rank of the card. So to find the suit of the card, we could put together a function that says, take the card number as an integer. Given the card number, return the suit as a number between one and four, and that would indicate what suit it was. So we'd have, clubs would be one, diamonds two, hearts three, and spades four. And that's a simple formula or calculation to put inside the function. Likewise, if we wanted to find out the rank of the card, we could simply have a function that takes the card number again, and applies a mod, 13 to it. So it takes the remainder of that division, and determines the rank of the card. Take a few minutes now to see how that calculation works, choose a number between one and 52. Apply that number to the functions, to work out the suit of the card. And then to work out the rank of the card. One example might be to choose the number 22. Where would that take us? Well, to find the suit, we'd take the number 22, minus one would be 21, divide it by 13, would leave one, and add one to that would be in two, so we'd be in the diamonds section. The diamond section is in here. We'd be choosing a diamond. Okay, and then to work out the rank or what number 22 represented, we'd simply put it through the function, the rank of card function. So we'd say 22 minus one is 21. We'd then mod 13, we'd leave seven, plus one is eight. We'd be looking at the eight of diamonds. Okay. Have a play with those functions. You may want to think about creating an array of cards, or a card game. Some other questions about the array of cards that could be worth thinking about that might help you along with some projects. And the software leisure gaming industry has always been at the forefront of software development. A lot of students want to be game developers. Let's hope they can be producers rather than just consumers and help develop some better games, but certainly a way to look at the concepts in this course, that is the do and the store, the control in the data structures, is to allow some creation of interesting games using things like arrays and cards. So what are the questions? How would you shuffle the cards? If an array of 52 integers can represent a deck, how could we represent individual hands? And what algorithms would be used to process or manipulate the data in a two-dimensional array? So have a think about those three questions. Take a minute. Okay, to shuffle the cards, we could use a temporary storage variable, as we saw in the swap exercise earlier, in the earlier series of videos. To represent a deck, we've got a single array. But a two-dimensional array could provide us four rows of five columns. So four players in a poker game with five cards each. And to process those hands we could use four loops, or nested loops, loops inside loops to read through the process, the two-dimensional race. We'll be having a look at that shortly. So here is a snippet of code from Arduino, which is very similar to C, that shows how to write values into array and extract the values out of that, so remembering that we're thinking about programming equals do plus store. And when we have a lot of variables of the same data type, we can put them into an array to make this easier. If you're familiar with using Python you would have come across using lists, which act in a similar way to arrays. So we have an array here called myArray. And that's indexed at zero. And it has five elements through to myArray four, the fourth element, the zero-based array. And inside those values, we're putting these numbers, two, four, -8, three, and two. So to retrieve a value for an array, there's an example there, we could take my value, indexed at two, take the value of that and put it into a variable called X, for example. And we'll see how arrays can be very useful. Okay, a record is a data structure that contains a number of fields. Each field is contained in a record can be of a different data type. So unlike arrays, which are homogeneous data types, records are heterogeneous data types. You'll notice that we've used in this graphic different types of lids to represent, again, different data types. So this could be an integers, for example. This one here might be a character, this one here could represent a floating number, a floating point number, this one here could be a Boolean, for example, and this one might be a long. So these are simple examples just to visualise, if you like, the fact that a record will contain different data types, different data types. And if you have an array of records, you'll have something like a database or a table that you would have come across. And certainly you may have seen, or a non-computing example of a record, which would be a telephone book, which might have integers as well as characters and strings. So records will have different data types. So two dimensional arrays are capable of storing in both rows and columns, and each row in a two dimensional row is associated with the number of columns that are defined for the array. So in a one-dimensional array, it would be one single row, which we would then loop through to process. This might be a single dimensional array here. But given that we've got a two-dimensional array, we've got a matrix, if you like, or rows after rows after rows, we'd need to have another way of processing that, and the way we process that is to put a loop inside a loop. We can also have multidimensional arrays. Some languages have many, many multidimensions arrays that you can use, which would be very hard to visualise. But for our purposes, the two-dimensional array in this suite of videos is fairly straightforward to follow, and we'll be applying it to our HSC examination question. It's probably a good time now to pause the video and ask students to process the array, according to the algorithm, by following exactly what the code says. So this code says, and we can do this simply with a finger. This code says, for row zero to three, zero to three. Column zero to five, so we start here. We're outs on the outer loop, zero to three. Do this, go zero to five, so we're going into zero to five, so in each zero to three, we're going zero to five. And we're adding one to the column to move to the next column. We're ending the loop there and then going to the next row, so we're pressing that array. It's a nice thing to point through and work on. A fascinating and fun activity to do in class is to ask students, or consider how you might process the array in different ways. How could we, for example, decide to, instead of going row by row, in our processing of the array, how would you rewrite the pseudocode to go down column by column? What would you need to change, it's an interesting thing to think about. Further to that, you might want to consider how we could change, again, the code to work in reverse, to go up. You might want to say, start from the bottom, and go up. Or we might start from the bottom, and go across. So think about the types of code you might write to process this two-dimensional array in different ways, and then have a go at coding that in a language that you're familiar with. Let's have a go now at writing some pseudocode to input and output numbers in a two-dimensional array, that has four rows and six columns. Often within the HSC examination question, you may not need to declare the data type that you're using. But certainly for this process, we might be able to declare a two-dimensional array of four rows and six columns. And we'll use some pseudocode that's not dissimilar to some of the languages you may have used, or come across. Always we're going to start in Begin. So take a few moments to think about how you would write this pseudocode, to input and output numbers in a two dimensional array of four rows and six columns. Start with the word Begin. Okay, we can declare the array. And here we've got an array cord number. With four columns and six rows, and you'll notice that they are inside square brackets, which you may be familiar with if you've used a number of different programming languages. And now we're going to process the array with the four loop. For I, zero to three, so we need to have an index, in this case, we've chosen a variable called I. And we want to process the columns, we've chosen a variable called K. So we're moving through each element in the row. And within each element in each row, we're moving through each column of that row. So K, zero to five, And we're getting an array number. We're adding one to the column as we move through the array, in each column. And we're adding through, and we're adding to each row. So you can see, we have a nested loop. To get the array numbers, to load this array. Have a go now at writing some pseudocode using this type of format to output numbers from a two-dimensional array. Take a couple of minutes to write your solution or answer. Again, for zero the three. For zero to five. Okay, and recall that our Is and Ks are really standing for rows and columns. And if you'd like to draw that matrix or that grid up on a page, and that grid of four rows and six columns, you can follow along with the code of the algorithm to process that two-dimensional array, and make sure you're clear on how that works.

End of transcript.