**TAS – Software design and development – Part 5 transcript**

## Bringing these two concepts together to investigate a HSC question

(Duration:15 minutes 33 seconds)

Let's now look at using these control and data structures or applying the knowledge to a past HSC examination question. This examination question came from the 2020 HSC Software Design Development paper, it's question 27 and 27 c. It was a five-mark question, which should take around about 10 minutes, but we'll have a look at a couple of ways of approaching a question like this. So here's the question. An audience is to be seated in 14 rows of 10 seats, so we have the 14 rows and they are here. The seating plan can be represented as a two-dimensional array, so one of these rows could be represented as one dimension, but given that we've got rows and columns in two dimension, it's a two-dimensional array. Column 0 contains the letters A to N, and row 0 contains the numbers 1 to 10. An X indicates that the seat has already been allocated, so once we sell a ticket, the seat gets allocated. For example, if four tickets are ordered, the seats B6, here we are, a little bit like a spreadsheet, B7, B8, B9 will be allocated. So that looks like a fairly straightforward question, and theatre booking examples are a very good opportunity for looking at control and data structures, and particularly in this case of a two-dimensional array. You may recall from studying IPO charts that one way of thinking about solving a problem or any problem is to start with the outputs, in this case, the ticket allocation, look at the inputs, in this case, the number of tickets that wish to be purchased, and then the processes needed to turn those inputs into the outputs. Now that might be a lengthy process during a HSC examination to try and do but it's certainly one worthwhile looking at, if you were to, say, use this example for a small project. You could determine an IPO chart, write up the outputs, look at the inputs, and then determine the processes or the algorithms required to turn those inputs into outputs. And sometimes, well, actually very often, there's an HSC scenario or examination question that can readily be turned into a small project in class, which works of benefit in a number of ways, one, it gives you some experience with HSC questions, and two, it gives you some experience with coding up solutions. So let's have a look at this question. Another way you might approach it is to rewrite the requirements of the question and then determine, take a few minutes to determine the control and data structures that you might need, that is, the dos, the sequence selection and iterations that you might need and where you might need them, but also the data structures, the variables you're going to need, in this case, the two-dimensional array that you'll need, and how you're going to process those, you're going to combine those two things together to answer the question. So let's have a look at this question. We need to ask the user for the number of tickets required. If only one ticket is required, we need to allocate any unallocated seat. If more than one ticket's required, we need to allocate seats next to each other in the same row. And if the required seats cannot be found together, an appropriate message should be displayed and the booking process restarted. And finally, we output or print the allocated seat numbers. So spend a few minutes now rewriting or thinking about the data structures and the control structures you're going to need to solve this problem. You can either do that by highlighting aspects of the question, and/or you could rewrite it on a blank page, but the questions or the requirements for this module are the number of seats, in this case, we might call that variable n. We need to inspect every seat so we can call the seat s, which is the column, and every row, which we can have as a variable r, to see if it's vacant. We'll also need to refer to these within the grid as an index, and you've seen that process earlier in previous videos. We'll need to keep a running total, t, of the number of empty seats and whether there is enough in a row, we'll need to ask that question, is there enough of that total within the row to meet the ticket number requirements? And then we'll need to display the ones that are sold. We'll also need to update the array and make sure that we don't try and resell a sold ticket, so we'll need to make sure that we get those X’s on those allocated seats and sold. And if there are not the required number of empty seats, we'll need to provide a message to start again, start the process again, okay, so we can already see that there might be the opportunity of using control structures like an if statement, in a number of places, but also loops and certainly nested loops. Okay, imagine that you are working in this theatre as a way of physically understanding the processes involved, and what you'd have to do if you were to inspect each and every seat to see if it's vacant, well, you might walk along and see if there's a person on it, and imagine doing that, every seat, check this seat, empty, go to the next seat, empty, go to the next seat, is it empty, go to the next seat, is it empty, go to the next seat, that goes on and on and on and you might recall from our earlier videos that the sort of coding of that process would take an inordinate amount of time to finish. So we need to think of other ways of doing it, and the way that we came up with before, you might recall, is to use a loop to process that array. Because we've got an array of arrays, or a two-dimensional array here, we have a loop inside a loop. Okay, so this is a repetitive action, a perfect opportunity to use a loop inside a loop, and also an opportunity to put in the if statement within the loop to see and address those conditions that are asked for in the module requirements. So have a go in the next few minutes, you can pause the video, of writing your algorithm and obviously a really good place to give you confidence to start is to just simply write the word begin, write the word begin, and that might give you the confidence to think about how you're going to write this particular algorithm. Okay, have a go at the algorithm and pause the video and we'll come back shortly. Okay, when you've written your algorithm, that is, you've had a go at writing the nested for loops that inspect each seat and determine whether or not they're empty, they take the number of tickets, swap your algorithm with a peer and desk check your peer's algorithm. So write up the variables on the top and desk check to see and ask the questions, does the algorithm accept an input, does it ask for number of tickets? Does the algorithm search the array, is there nested for loops involved or they could be while loops as well? Does it update the array, does it have a total and does it indicate where the seats are sold? Does it display the allocated seats? Does it provide suitable messages? And certainly, does it allow for the process to start again? So swap that attempted algorithm, the first attempt, and do a quick desk check on it and see whether it fulfils those criteria. You should have available for download a sample algorithm from NESA that provides a solution to this problem. One thing you're encouraged to do is to, again, for practising your use of desk checking but also to find out and understand the logic, is draw up a desk check for each of the variables. There's an incomplete one here, which is just a way of setting up the structure for example, so have a go at writing up a desk check with the number of variables, and simply filling in the first few values as you're processing the algorithm, the sample algorithm answer that you have, to understand how it works, and also to compare it with the sample response that you provided, so your answer to the problem, compare that with that issued from NESA, the sample answer, and do a quick desk check of this sample answer to see the logic involved and whether or not you've missed anything in your response. Another useful tool while you're doing this process is to have the full grid, and I've got another sample grid on this image, have the full grid available so that you can actually see the processing while you're working through the algorithm to determine whether or not you've met all the criteria and that you will receive the full marks. So there's a number of ways of answering the question, there's a number of possible solutions for this, but certainly the sample answer response that you have accessed through the downloads with this video is a good way of better understanding the logic involved and getting more practise in solving these HSC responses. Another really engaging way to work with the algorithm is, as mentioned earlier, to include it as a part of a project, so as a small project. You could, for example, decide that in looking at end-user or rapid application development that you're going to use an Excel spreadsheet, and you're going to get into the back of that spreadsheet through the VBA processes and code up your sample response to this question, so, in actual fact, code up a theatre ticketing system. So a theatre ticketing system, and here's an example of VBA code that will allow you to have rows and columns and cells, so this is a sample aspect of the code, you could use VBA to do a demonstration, not just of this particular response but you could also create it to see how end-user and rapid application development or prototyping works in the implementation of projects, so the other development techniques within software design. Likewise, you could use some other examples here, we have some examples here. If you're familiar with using C, you might have a look at C and have a look at an online, this example, I'll choose a better colour there, this example is an online compiler and allows you to work in C, so you could work using the C language to write up and here in this language it shows you how to set up an array, a two-dimensional array, 14 rows and 10 columns, in C. Likewise, you might be working at Python at the moment, and here's an example of how to set up an array with columns and rows in Python. So you've got a number of choices you can work with, but certainly it would be worth establishing the array, the two-dimensional array, to represent the theatre, and then thinking about how you could code your own solution to that, and that can be a very worthwhile process, not just to understand the coding languages but to see how they relate to the HSC examination in algorithms and pseudocode. One of the problems that I've certainly come across within the course is that students will very frequently respond to an examination question in a language that they are familiar with, so they'll be using syntax and terminology and language features that they are familiar with from a language they've learned, but the question will be asking for their answer in pseudocode, so you must understand the pseudocode to answer the HSC examination, okay? You can then apply the pseudocode in the HSC examination questions from past papers to code up your own solution in whatever language you like. But it's a worthwhile process of working both ways, of taking past examination papers, the HSC examination papers, looking at the scenarios and the algorithms involved, coding them up in a language you're familiar with, either Python or C or Java, whatever language you're using, code up the solutions to the algorithm questions. And likewise, whatever programmes that you're working on, whatever languages you're writing in and whatever you're coding within the classroom, think about then turning it back into pseudocode, how you would write it in pseudocode, and how then would you make a HSC examination question out of the work that you've been looking at? So there's a couple of ideas that you can work with, I hope you found this useful in your study for the HSC. Good luck with the work, and just remember that if you can have a thorough and good understanding of all the control structures and all the data structures used in this course, you should have no problem at answering any of the algorithm questions in the Software Design Development paper. And recall as well that we discussed that the Software Design and Development paper is full of algorithm questions that require you to demonstrate your computational thinking and your logic. Good luck, I hope it goes well, thank you for tuning in to these videos.

End of transcript.