**TAS – Software design and development – Part 3 transcript**

## Data structures revision: Data types, variables

(Duration:15 minutes 33 seconds)

Okay, in the last series of videos we looked at programming was Do Plus Store and we looked at how the do was the control structures. We're going to dive a little more deeply into data structures and revise some of the concepts around there that you'll need. Okay, recalling programming with Do Plus Store. Remember that we're looking at storing, we're looking at the datatypes, that is how we represent the zeros and ones to be other types of data, whether they're characters, strings, integers, floating points, et cetera. We're looking at variables that is the storage for those datatypes and being variables they can change and we can put new values into those variables and slightly more complex if you like, arrays and records, you recall that we discussed there are other types of data structures, including trees, stacks, and cues that programmers all use, but for our purposes and for this HSC course in software design development, the key concepts that you need to understand or need to have under your belt are datatypes, variables, and arrays and records and how we use the control structures or the algorithms to manipulate and change that stored data. So ... let's have a quick look at a revision and some quiz questions that you should know. What is a bit? A bit is a binary digit, take the bit of the word by, bit of the word it, combine them, combined binary digit to get a bit, a zero or a one. Eight bits together are called a byte, a conversion of that number there, it's a fairly significant number, 65. The reason it's significant for software design developers is it's such an easy number to remember when you're thinking about ASCII. Remember what's special about 65 in decimal is that it represents ASCII in A, and what's ASCII again? ASCII is the American Standard Code of Information Interchange. What are the limitations of ASCII? You'll recall, it's just used for English, which has 27 alphabetic characters, sorry 26, sorry, letters of the alphabet, and you may recall that there are dozens, hundreds of thousands of various alphabets from different languages and so to be able to cover those and overcome the limitations of ASCII, we've now got Unicode and ASCII forms a subsidy of Unicode. You'll get the study more about Unicode during the course, but I suppose it's important for us to remember that we have different ways of representing data, different types of data that we could represent using the same zeros and ones. So let's have a look at data structures and in particular... A couple of questions to think about around datatypes, why we have them? Well, a couple of examples to think about are how many possible numbers are there? Possible numbers, real numbers, how many numbers are there? So how many integers are there, but also how many real numbers are there? And if you say that an infinite amount of both, you'd be correct, but do computers have an infinite memory? Well, no, they don't, so we've got a problem, how do we represent such a vast range of numbers, a range and resolution, so very big numbers and very small numbers with computers and with binary? The programmer needs to choose the range and resolution, and they do this by choosing an appropriate data type for the purpose, so you might think about, for example, the type of numbers that a microbiologist might be concerned about, very small numbers perhaps, for the scale of the work they're looking at and likewise, an astrophysicist might be looking at very big numbers or an astronomer to look at the distances between galaxies. So the numbers can have a different range and a different resolution and we need to know, for the purposes of programming a solution, what we want the zeros and ones to represent, you could simply put a big string of numbers on the board and those string of numbers could represent anything, they could represent an identification number, social security number, a password, a telephone number, all sorts of things. We don't really know the meaning of what they are representing until we decide what type of data or what datatype we want them to represent. And you'll recall from our last slide that to represent characters we have ASCII code and more recently... Okay, one way of visualizing datatypes is using a cylinder or cups that represent the variables and the memory addresses and the type of data that we put into those cups or that memory could be represented with a different type of lid. Here's a few examples of datatypes in Arduino, which is similar to see and how you might declare those variables using these datatypes, so in the first example we've got int which stands for integer, and we've got a pin and we're putting 13 into that. In the second example, we've got a character and we're storing that character as A, and A, as you know, is 65. In mySense, we're putting in a float and a float is a real number decimal point, and it's called a float because we can move that float decimal point number around, you'll see the size of the number that we can store in there, we can store four bytes or 32 bits. The integer we can store is 16 bits or two bytes, but for float we're doing 32 bits. In the long, if we're storing long, we're storing some large numbers and that might be example of light years and in a boolean we're storing a true or a false very simply there. So there's the key datatypes that we use. So again, these datatypes, in this case it's an integer, it's going there, in this case it's a character in the A and that's being stored there, et cetera. So floating point, and you'll see that how many bits are available. Okay, so the, some languages will have various amounts and you can look that up with the documentation of language that you're using and see, well, how many bytes are used to represent, for example, integer, character, float along, et cetera. And just recall that those bits, zeros and ones, one thing you might be able to recall for not just a bit of fun, but also for your own memories, is how to count in binary with your hands and you might recall that you can just simply represent the columns ones, twos, fours, eights, 16 32, 64, 128, 256, 512 and for a little bit of practice, a little bit of homework, a little bit of fun with your family and friends count in binary, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 31, losing my way, you get the idea, a lot of fun to practice with and a nice way of remembering binary and decimal numbers. Okay, let's have a look at variables in a little more detail and revise. So variables of the store, remember that we discussed earlier that the right-hand side of the equation goes into the left-hand side of the equation, okay, very different if you need to program, just suddenly look at this thing that you think says equals, so I might read that little bit of pseudocode there as sum equals zero, I'd be scratching my head. Now what we're saying is we're taking the value zero and we're putting it inside or we're assigning it to the variable sum, okay, so we're putting it into that bucket or that cylinder, or that storage called sum, and likewise for number we're doing the same, we're taking the numbers one through to nine and we're putting them into number. So a very obvious thing to people who have been coding for awhile, but for those new to the game, that's one thing that can sometimes interfere with their understanding of how to read the pseudocode. Okay, here's another example that's a good fun activity to use to help clarify an understanding of variables. We could take some cups, some paper cups, and we could write on one X and we could write on the other Y, we could take a piece of paper and write on that piece of paper five, and we could take another piece of paper and write on that piece of paper eight, we could put the five into the cup labelled X, and we could put the paper with the number eight written on it, we could put that into the cup labelled Y. Okay, and we could ask students or our colleagues or our friends to swap the value, okay, so the question is let's swap the value inside the variables. Okay. But the rules state that each variable can only hold one value at a time. So as soon as you take a number from one variable, it will overwrite the value that's in another, once you put it in there, have a look at one of the issues, some of the problems that you might have in writing code for this. So have a go at writing the code to solve that problem, swap the variables in... swap the values inside the variables and see what you come up with. And there's a little bit of code down the left-hand side and the R code, it says X equals five, put five into the bucket X, if you like, into the cylinder, put eight into Y, so assign the value eight to Y, assign the value eight to X, that's all good. What do we do with Y? Okay, have a think about that process for a moment. There are a couple of ways to solve this, but the main concept we're considering here or thinking about is how variables change and how you overwrite the value of a variable once you put a new, another value in, okay, so here's a common mistake, a common swap mistake that's used, if we do a desk check, we might say in line one, X is five, in line two, we might say, Y is eight, we've assigned eight to Y, if in line three, we assign the value of Y into X, so X becomes what's in Y, which is eight and then in line four, we say, let's assign Y to X it's eight again, so we haven't actually swapped anything, spend a minute now just writing code successfully that we'll swap those values in those variables and how we got that, and yes, there are other solutions, but have a think of a solution that will work in this instance that explains how variables operate. You can pause the video for a moment and think about how you could write or rewrite that code. Okay, here's a solution, we take the same lines of the same, we put five into X, we put eight into Y, aha! We have another variable called temp, here it is, down here, and we can take the value of Y which is eight, put it into X, let's follow it through our desk check, so we're on line three, we're taking X, which was in five and putting it into temp, and then we're taking the value of Y which was eight and putting it into X, but now we take on the final line what's currently in temp and put it into Y. So for students struggling with variables and reading the assignment of variables and mistaking equals values as equals and not actually for what they stand for, which is assign the value of, have a practice with using cups and having cylinders and thinking of letterbox shapes and visualizing the process can, for some students, be of value. So here we are with our variables and we follow the desk check and follow the algorithm, we do exactly what the five lines of code tell us to do and there's the results. A lot of fun to be had with variables in class.

End of transcript.