TAS – Software design and development – Part 1 transcript

## Control Structures and Data Structures Overview

(Duration: 8 minutes 46 seconds)

Hello, welcome to HSC Hub, Software design and development. This short series of videos provides a flyover of the main concepts covered in the programming aspects of the software design course, and then an opportunity to practise the skills using past examination questions. Two key concepts in programming are, control structures and data structures. What are control structures and data structures? A simple statement that provides a useful guideline or rule of thumb for students to easily remember what these two structures are, is do plus store. So programming equals control structures or do, and data structures or store. There are many different types of programming languages, some focus on the doing aspect, and they called imperative or procedural languages, and some focus on the storing part of the statement, and they called declarative languages. Those languages which combine the do and the store into something called an object refer to object-oriented programming languages, and they access the object using methods, and the object has attributes. You can study more about object-oriented programming languages in the paradigms topic in HSC.

Now focus in line with the software design course and its focus on algorithms is on the doing languages, the imperative languages. The programming languages that focus on doing are historically based on the concept behind monuments architecture of computers, that is the ability to store programme instructions in memory along with the data on which the instructions operate. Until Von Neumann proposed this possibility, each computing machine was designed and built for a single predetermined purpose. All programming of the machinery required the manual rewiring of circuits, a tedious and error-prone process. If mistakes were made, they were difficult to detect and hard to correct. If you choose the option topic on the interrelationship between hardware and software, you'll see how logic gates provide ways of doing at the circuit level. Just remember that both the do and the store at the circle level are using binary representing the high and low voltages.

So let's have a look now in more detail at the doing part or the control structures in the programming statement. The doing part refers to algorithms, and algorithms are a series of unambiguous instructions or procedural steps that will result in the solution to a specific problem within a finite time. The programmer needs to know only three main control structures, processes or algorithms if you like. Computer science shows us that these three main control structures can be arranged in many different ways to solve any solvable problem, do refers to the algorithms instructions used to manipulate the stored data. Even at the stage four silvers level now, students are required to design algorithms that use a range of datatypes, branching, selection, and iteration or loops, and represent them diagrammatically with flow charts and in English or pseudo-code.

Programming languages that emphasise the do are code imperative or procedural languages, and our focus in the software design course is on the doing. You'll notice that the software design examination paper is full of algorithms. And these questions that if you can master the three ways of doing the control structures and put them together in different ways to use them to manipulate data structures like erase records, you'll answer any of the questions in the HSC. Let's have a look at the three ways of doing, we have sequence or a series of steps or processes, do this, do that, do the other, a step by step, really easy to do in an unplugged version. We've got selection and selection is the powerful if statement that makes computers appear like they can make a decision or choose between options. So sequence, selection and iteration, and iteration are the loops. There's a number of different types of loops, but basically, the three main control structures or ways of doing within an algorithm are sequence, selection and iteration. Most languages, in fact, nearly all languages have ways of doing a sequence, a selection and a loop or an iteration, the difference may be in the terminology or in the syntax that they use. But certainly these three control structures are available in most languages, and any that you're currently using within the course.

After looking at the do part of the programming equation, that is the control structures, sequence, selection, and iteration, let's now look at the main aspects of the store part, that is, what we need to know about storing the data in programming. The store part of the equation refers to the ways of keeping the data, the zeros and ones in memory, what they represent and how they can be organised. We'll be looking at the different types of storing, we'll be looking at the arrays and records, we'll be looking at datatypes and variables. There are others that programmers use, including trees, sticks and cues to organise and access data, though our focus will be on datatypes, variables and arrays and records. Datatypes, how to use the zeros and ones to represent data. The variables, where to store these datatypes or where to put them. The litter boxes if you like, that can be the contents of which can change, and some more sophisticated datatypes like arrays and records. Remember, programming equals do all control structures plus store all data structures. Remember, some languages take both to do on the store and combine them into something called an object that uses methods and attributes. But for our purposes, for the main aspect of the course, which has to do with algorithms and control structures, will be focused on imperative languages and the doing aspect.

End of transcript.