Week 1 Module 3 Hello World Script

Process of Executing a Program 1.2

Hi there. Today we are going to implement our first C++ program. Typically a first program in a programming language is a Hello World application that reaches out to the user saying 'Hey I am here' or printing Hello World. We are going to write a slightly different first program but before we do that let's take a closer look at the process at how an application is executed inside a computer. Let's focus on three main components inside the computer we've talked about a few. Let's focus on these three the memory the main memory the ram and the secondary memory whether it is a hard disk or a solid disk drive. And obviously the CPU which is the central processing unit that does all of the job inside a computer. So both memories ether it is a second memory or the RAM are basically contains a lot of zeroes and ones. Everything inside a computer is zeroes and ones as you all know. And these bits of zeroes and ones encode a lot of information. The bits are collected into collections of eight bits each collection of eight bits is called a byte and they are stored inside the memory. For example the first byte that was eight bits located in a physical address of zero the second one would be one two and three and all. So there is a very long sequence of bits each one containing zeroes and ones. This is used to represent all types of data with zeroes and ones we saw how we could represent numbers but using numbers we can represent audio we can represent video we can represent documents spreadsheet. All types of data can be represented with zeroes and ones. One other kind of data that is represented is a program or an application. An application is basically a set of instructions. For example these 95 bytes here are a sequence of instructions for a program named prog.exe. Typically that's the file extension for an application it is executable .exe. So assuming these bytes here are a program prog.exe we can execute it and make the magic of this application happen. What happens when we double click an application? When we want to start executing a program? So there are a few steps that are happening first thing that happens is obviously the program is stored inside our secondary memory so the first thing that happens is this program is copied into our main memory in order for the computer to have fast access to this set of instructions. And then these instructions are starting to execute one after the other. Obviously the CPU is responsible for that. The CPU has a program counter register that says where or what's the next instruction that would be executed. It is initialized to 100 which is the first instruction in this program and then the fetch decode executes cycle is starting to happen. Each cycle the instruction that the program counter points to is fetched from the main memory into the CPU. Then the CPU decodes it understand what the meaning what it has to do with this instruction and then it executes it. And over and over so the program counter is increased and the next instruction is fetched decoded and executed. Program counter is increased and the instruction is fetched decoded and executed. This happens literally million times each second and each the instructions are just executed one after the other. Whatever the program is intended to do basically happens.

Compilation 1.3

Ok so we have a program that can be executed by the computer. It looks as a sequence of zeroes and ones that represent instructions for the CPU to execute. These instructions are very low level they can add things multiply move data from one place to another jump to another position. Very simple set of instructions. We are not going to program with zeroes and ones. This kind of programming language is called machine language that is obviously understood by the machine by the CPU by a computer. We are going to program and write and implement our algorithms using a high level language. There are a lot of

high level languages we are going to write in C++ but there are also Java Python C# and C and Pascal and Scheme and a lot of high level languages. A program let's say in C++ would not look as a sequence of zeroes and ones it would look more like that. Which is a text in English contains the English alphabet you can see we don't understand exactly what it does but you can see it has a text Hello World it has some English words you can probably recognize like main and include and return and stuff like that. So we write in something that is much closer to our natural language not zeroes and ones. But then the computer doesn't understand this type of language. So we need to translate this code here into a machine language code. This kind of translation or this process of translating a high level language into language the machine the computer can understand is called compilation or a build process and we lucky for us it is an automated process since even high level languages are very strict and we have to follow the rules accurately. And automated process can just take the code in the programming language and translate it into machine language. So we just have to start this process and then our computer will be able to execute the program we write in the high level language. I will show you how we do that as we write our first program.

First C++ Program 1.4

Ok so let's start to write our first C++ program. Let's make it easy let's write a program that reads from the user to numbers and just prints what the sum of these numbers are. So we want to interact with the user something like that we would first ask the user please enter two numbers separated by a space. The user would enter for example 7 and 5 and then the program would respond 7 plus 5 equals 12. Let's see how we write it in C++.

First C++ Program Screencast 1.5

Ok so here at the screen you can see the implementation of the program that reads two numbers and prints what their sum is. Let's take a closer look at this implementation let's first look at the file here. We see that there is text in English plain English this program reads two integers from the user and prints their sum. Will hold the first input will hold the second input. That's English and that obviously is not C++ and you can see that we can write text in English for documentation purposes. So other programmers can understand what we are trying to say. These are called comments the syntax for writing a comment basically saying for the compiler to ignore that's for only humans to read is starting the comment by slash star and ending the comment by slash stair slash. And then in between you can write whatever text you want the compiler would just ignore it. You can write in how many lines that you wish you can write your text. For this kind of a comment that starts with slash star and ends with a star slash. A different kind of a comment uses two slashes basically telling the compiler to ignore whatever is written from this point to the end of the line. You can't break the lines for this type of comment. So we typically write some information describing what the purpose of the program is what the purpose of some lines of code if they have like a complicated goal that they are trying to do. So they are commenting it using one of these syntaxes. So let's ignore the comments they are not compiled they are not translated to machine language they are not understood by compiler at all. If you see we have int main with a empty parentheses and then you have an open curly brace and the closed curly brace basically saying that a program is everything that is enclosed in these two curly braces. We always start our program with int main and empty parentheses and then inside a block of curly braces we write our program. We also have these two lines at the beginning include iostream using namespace std. Make sure you write exactly as it appears here with that hashtag and the less than greater than symbols and the semi colon over here. It makes a lot of difference for the compiler. I am not getting into the details

of what these two lines are doing just now maybe I will talk about it later or in later modules. I am not also explaining too much of the int main basically does but each program would start with these two lines for now that's what we need to know and with a header of the int main curly braces. One more thing though and that with the return zero. That is something that is common for each program. Another important thing you can see here is that each line of code here ends with a semi colon. That basically says for the compiler that the statement the expression the instruction ends here. So you see that each line here ends with a semi colon you have one two three four five six seven eight instructions. We write them in eight separate lines but for the compiler all that matters are the semi colons. If it was up to the compiler we can write all our program in one single line just separating with semi colons. So for the compiler writing the code something like that in one long line is perfectly fine. The compiler would know that the first instruction ends here the second instruction ends here and the third instruction ends here. We write it in a different lines in separate lines because it is easier for us humans to understand and see the structure of the code. But for the compiler the semi colon is what ends an instruction that's why we have to make sure we end each one of our instructions with a semi colon the compiler won't tolerate an instruction that doesn't end with a semi colon. Another thing I want to say before we start here is that in C++ we have to declare we have to state ahead what are the variables we are going to use. What are the datas that we need inside our memory. So in this case we said we need num1 num2 and sum. We also have to say what is the type of each of them. So basically we say num1 would be a variable that is an int instance for integer meaning that num1 would hold an integer value. Num2 is a variable of type int that would hold another integer value and sum is a third variable again of type int that would hold another integer value. We also said for other programmers that num1 is used to hold the first input num2 is used to hold the second input and sum would be used to hold the sum. But that's only for us to understand. For the compiler we are saying that this program would use three variables num1 num2 and sum all are type int. You can see here we have cout leave the cout but you can see that we have out basically saying we want to output it and we basically say we want to output we want to print out the text please enter two numbers separated by a space. We want to continue printing end l basically saying basically stands for end line. We want to break the line so we want to print this text and break the line. Then cin basically saying getting an input from the user into num1 and into num2. So after printing the announcement please enter two numbers separated by space we are reading whatever the user enter into these two variables num1 and num2. After we have the user's input inside our num1 and num2 we are saying put into the sum variable the value of num1 plus num2. This equal sign here stands for an assignment we want to assign this value here of num1 plus num2 into the variable sum. So our program would first print the instruction for the user to enter two numbers separated by a space. Then it would read num1 and num2 it would read the two inputs into these two variables and then sum would be assigned with the value of the sum of num1 and num2. Eventually we will just cout we will print out to the screen once again. Num1 the value of num1 the text plus the value of num2 the text equals and the value of sum followed by a break line line break. That's basically what we are doing here in order to see that it really behaves as we expected we need to compile it we need to build this program and then to run it. So by now I hope you have your IDE already installed if you are using Windows you are probably using Visual Studio if you are using MAC probably you are using Xcode but other IDEs are also fine. If you don't have an IDE installed please do it right after this module. For Xcode just press this play button it builds a program. Here build succeeded and here we have our program starting to execute first it says please enter two numbers separated by a space I will just type five space seven space five. When I press enter I get the response seven plus five equals twelve and it

also says that program ended with exit code zero that's what the return zero basically does. So it works as we expected which is good I want us to note a few things here now that we get the structure of the code. For example I can do cin num1 semi colon cin num2 semi colon in two separate lines still the execution would be the same. So cin doesn't really mind if we do it in two different lines or in one line it would still behave the same. I prefer doing it this way. We'll talk in more detail about the behavior of cin in one of the future module. Another thing regarding cout I can have multiple cout for example please enter two numbers let's end it and cout separated by a space in a different line. Try to guess if you think that the output would be in one line or in two lines here in the execution. Hope you are guessing it I am building it in the mean time. And you can see that even though the cout is broken into two separate instructions it is still printed in one line and that's because we didn't break the line we didn't say print kind of an enter. If I would say please enter two numbers endl and then cout separated by a space then it would be just one second. Then it would be printed in two lines but if I am just splitting the cout into two lines without breaking the line in the output it would still be in one single line of output. Again let's just bring it back to how it was before. So that's one more thing I wanted to point out here. One last thing is that let's take a look at this output statement here. You see that we didn't do something like cout num1 plus num2 equals sum. If we would do that it would just send a text num1 plus num2 equals sum. But we want to print the value inside num1 variable and inside num2 variable and inside the sum variable. That's why we are not printing num1 num2 and sum as texts inside quotes we are printing num1 on its own that would print the value of num1 and then we are printing the text plus and then we are printing the value of the variable num2 followed by the text equal and then we are printing the value of the variable sum. One last thing to note here if we want to concatenate print outs we always use the double less than symbols so cout start with num1 continue printing this text continuing printing num2 continuing printing the text equal continuing printing sum and continuing printing line break line. Yeah I think this gives us an idea how C++ program is structured and the basics of interacting with the user using cout and cin.