Week 2 Module 4 Data Types and Expressions Part 1

Data Types and Expressions 1.2

Hi there today we're going to talk about data types and expressions in quite a detail. Before we do that I want to say that each programming language has three fundamental constructs. There are data expressions and a control flow. In order to master in a programming language you have to know a lot in each one of them. You have to know a lot in data a lot in expressions a lot about control flow and actually takes some time in order to understand so much. So we are going to take step by step and you have to be kind of patient.

Data Types and Expressions 1.3

Let's take a second look at a problem we already implemented the program that reads from the user two numbers and print their sum and see if we can find some of these constructs some data expressions and control flow. So here is the program we implemented you can see that when this program is executed it is executed line after line. First there is cout that prints an instruction to the user please enter two numbers. Then there is the cin that reads from the user these two numbers then we add them store them into sum and eventually we print out the message that whatever the sum is. You can see that the program is executed line by line we call this flow a sequential flow. This is a default flow of a program that lines executed one after the other. You could also see that we have here some data we have three variables numOne numTwo sum they are all integers in C which is a C++ which is a strongly type language. Each data must have a type of its own in this case both or all of our variables are integers int. There are also expressions here in our program there is numOne plus numTwo there is an assignment expressions there is a cout a cin an io expression. So there are all the kinds of these constructs here in this program.

Data Types and Expressions 1.4

There is a sequential flow IO expressions arithmetic expressions and data of a type int. Today we are going to focus mostly on some data types there are several data types that are used in C++. There is the int we already talked about. We are going to talk about float and double which are going to be used to represent real numbers numbers with a fractional part. There is a char that is used for characters there is string for textual information and there bool for Boolean values True and False. We are going to talk in much detail about each and every one of these data types. We are going to start talking about int we are going to do it right away.

The int Data Type 2.1

Ok so let's get into the details of int of the data type for integers. So integers as we said are used to store integer numbers so every time we want an integer number we use the type int. Let's see how this int data is represented inside the memory of our computer. What is the inner representation of an int. So there are basically two options here. Either each data would be of a different size for example smaller numbers would take less space less bits in order to represent them. And larger number would take more space in order to represent them. That would make integers be of a different sizes. Another option would be that every int data would have a fixed amount of bytes in its representation. There are advantages and disadvantages for each one of these options and C++ the data types the int datas are all of the same size. Each int data is four bytes long which is 32 bits each byte is 8 bits so four bytes is 32 bits. So each integer is 32 bits long. For example if we have an integer x then inside our memory there would be four bytes for this x. Let's say starting at address 100 until 104. If we have an integer y after x would come this four bytes for the y from 104 to 108. So each int data uses four bytes. So I think the

greatest disadvantage of a fixed size variables or fixed size data is that we can say that 32 bits is quite a lot we can represent two to the power of 32 different values in 32 bits. Two to the power of 32 by the way is around four billion different numbers so there are a lot of numbers we can represent in an int variable. But then if we were thinking about that integer number domain it is an infinite domain and if our program would like to represent a number that are very big for example astronomical distances then 32 bits won't be enough for that kind of data. So that's the biggest disadvantage of using a fixed size data but then there are also a lot of advantages of using fixed size data and we will talk more about them in more detail when we speak about arrays later on in this course. Ok so inside the memory each integer takes four bytes let's think how the integer is represented inside these four bytes. What bits what zeroes and ones would be used in order to represent an integer data. So when we talked in one of our previous modules about number systems we saw that we can represent numbers using zeroes and ones using the binary number system base two. And then since integer can also be negative numbers we talked about the 2's complement representation method and that's how integers are represented in C++ the numbers are basically represented in 32 bit 2's complement representation method. So for example if we set x to 6 obviously 6 decimal is 110 in base 2 but if we are thinking about the 2's complement representation of 6 basically we are padding it with zeroes so 6 then is this representation in a 32 bit 2's complement. So these 4 bytes these 32 bits are then placed inside the 4 bytes of x. If we set y to -6 then again -6 as a decimal are represented like in a 32 bit 2's complement representation method and these 4 bytes are then taken inside the y's location in the memory. So the numbers are represented in four bytes and using the 2's complement representation method that's a very important thing to know about integers.

The int Data Type 2.2

Let's talk about some more stuff regarding integers. For example what are the literals that are built in inside C++ for integers. Maybe let's take a step aside here and understand what do we mean when we say C++ literals. So when we are talking about data basically data can come into two major forms either variables or constants. Variables are as we can understand by their name can change their values where constants are values that are constants cannot be changed. So if we look at these lines of code here we see that we have x and y which are variables we can see that we can set their values x equals 6 and then set it to 7 or to 8 or whatever they are variables their value can change. And then there are also some constants here in this program. As we said every data in C++ is has a type of its own C++ is a strongly typed language. So 6 -6 even 0 way down there are all of some type in this case they would be of type integers. So we can see that they are variables we can declare them let's say int x later on we can define double y or whatever type we want to see for each variable. And then there are also contants and then for constants we talk about two kinds of constants one that are built in inside our language when we name C++ literals. Basically we don't need to define 6 or -6 or 7.3 or abc we don't need to define what these values are the compiler already recognizes them they are built in inside programming language. But we can also define our own constants we can create a program and define contants. The syntax for that goes something like cons int max = 5 and in this case we are creating a constant of type integer its name is max and the value is set to 5. We will talk more about it later on but the major idea here is once we set max with 5 this value cannot change cannot set max to 7 after a few lines of code. Once we created a constant that value is there to stay. So let's get back to literals of type int what kind of built in data the compiler recognizes as integers. So as we can expect writing numbers in their decimal representation just like 3 4 -6 3954 or whatever number you want the compiler would recognize them as integers. So we can use them and we know that they are considered to be integers.

The int Data Type 2.3

One more thing I want to talk about are our operators that can be applied on integers. Using our arithmetical operators basically we create expressions so we have the literals or the variables which are atomic expression but combining them using arithmetic operators we create more compound expressions. For example plus we can add two integers and then create a greater or a bigger expression. Expression basically is a thing that has a value in this case for example if we have x and y and then x is set to 5 we create an expression x plus 5 basically 5 plus 2 or 7. If we cout x plus 2 the value of this expression would be printed in this case 7 would be printed. But plus is an arithmetical operator that can be applied on integers x is an integer 2 is an integer it is a C++ literal of type int and plus can come in between two integers and create a compound expression in this case x plus 2. So we can print the value of an expression we can set a variable to have the value of an expression. Y equals x plus 2 basically is an assignment when an assignment is evaluated first the write inside is evaluated x plus 2 and then this value which is in this case 7 is set into the variable y. So given an expression we can print it we can assign it to a variable. Actually we can also just type it X plus 2 obviously it is a useless line of code here because no one does nothing with the value x plus 2 but the x plus 2 on its own is a legal expression in C++ and we can write just like that. So plus is arithmetic expression arithmetic operator sorry but then we can have some more arithmetic operators. For example minus and star for times so we can subtract integers x minus 2 we can print it and it could print 3. We can multiply x times 2 5 times 2 which is 10 we can assign it to y y would be 10. So we can apply different arithmetic operators on ints. Plus minus times. We can also apply the divider operator but here it is a little bit tricky. What do you think would happen if we for example cout x divided by 2 5 divided by 2. I would say that it would probably print 2.5 because 5 divided by 2 is two and a half. But then it behaves a little bit differently.

The int Data Type 2.4

Let's stop for a minute and roll back to I don't know second grade. And recall what our math teacher used to say when we divide 13 by 5. So when we divide 13 by 5 we get something like 2 with a remainder of 3. Basically means 5 fits into 13 two full times and there is still a remainder after these two full times of 3. There are two designated operators that give us these 2 and 3 they are called div and mod. For example if we do 13 div 5 we get 2. Which basically div means how many full times times 5 fits in 13. So 13 div 5 results to 2. And 13 mod 5 results to 3. Basically meaning what's the remainder when we are dividing 13 by 5. In C++ we don't use div and mod in the textual writing there are specific symbols to do div and mod. For div we have the slash operator so if we have int slash int that would do a div operator in this case that would be 2. And for mod we will use the percent symbol so 13 percent 5 would be 3 for the remainder when we divide 13 by 5.

The int Data Type 2.5

If we print or cout x divided by 2 since x and 2 are both integer the slash here is in the context of div. So it would print how many full times 2 fits into 5 which is 2 in this case and if we cout x mod 2 that would print the remainder when we are dividing 13 by 5 or 5 by 2 sorry and that is one. Because this is the remainder when we are 5 by 2. So when we are speaking about arithmetic operators in the context of int we have the plus minus multiplication div and mod. One last operator I want to talk about here is the assignment. Formally an assignment is considered to be an arithmetic operator. Basically by using it we create an arithmetic expression. And when we set x to 6 there is a side effect that basically sets the value of x to 6. But this expression basically also has a value. The value of this expression is the value

that is assigned to the variable. So for example if we cout x equals 7 so first the side effect of x getting the value of 7 happens but then the value of this expression 7 is also in this case printed. Later on if we assign y to be x equals 8 once again x equals 8 when it is evaluated sets x to the value of 8 but the value of this expression 8 goes into y. So using assignment as arithmetical operator we can for example create multiple assignments in a single line of code. So we can do y equals x equals 8. Actually we don't need the parentheses we can just do y equals x equals whatever in this case 9. So these are the major or the fundamental arithmetic operators when we are using integers. And if we recap what we said about integers about the int data type so they are representing they are used represent integer numbers. Each one has a fixed size of 4 bytes. The data in these 4 bytes is represented using a 32 bit 2's complement method. C++ has built in datas that are considered to be integers basically the C++ literals. We programmers just write integer numbers in their decimal representation and C++ would recognize them consider them to be integers. And we can create arithmetic operators using integers with arithmetic operators in between them.

Weeks and Days 2.6

Ok so let's try to use all the syntax we talked about integers to implement a simple problem. We name it weeks and days problem. So let's write a program that reads from the user number of days they traveled. And then the program would print what's the travelling time in the format of how many full weeks they traveled and how many additional days are there. For example if we executed this program the program would ask the user please enter the number of days you traveled. The user would say for example 19 and then the program would respond 19 days or two weeks and five days. Because 19 days are two full weeks and extra five days.

Weeks and Days 2.7

Before we get into implementing it let's stop a second and think how we can figure out how many full weeks and extra days are there in a some amount of days. So if we divide let's say 19 by 7 we get two with a remainder of 5. Two would be how many full times 7 fits into 19 which is the number of full weeks. And 5 is basically the days remaining after these two full weeks. So once again 19 div 7 is the full weeks and 19 mod 5 would be the days remaining. After we have that figured out it will be quite easy implementing it using C++. Let's go ahead and do that.

Weeks and Days Implementation 2.8

Let's implement this program so first let's ask the user to enter the number of days they travel so cout. And then please enter number of days you traveled. Let's break the line end line. And then let's read whatever the user enters into a variable. Since it is number of days int is the type we want to use here. We first have to declare this variable so int. And then we have to decide what the name of this variable would be. In my previous examples I just used x y z whatever because they had no meaning just to show you the syntax of integers. But here in this program this variable would be used to store some significant kind of data. It is going to store the days user traveled. So let's give it meaningful name in this case I would name it days traveled. But then I can't have a variable name days traveled made of two words. So I need to somehow create a single word that would be made of more than one word in it. So there are a few common conventions on how to do that. One way is to separate the word something like underscore symbol something like that into days traveled. Another convention could be something like to capitalize the beginning of each word days traveled and then easily you can see what this long sequence of characters basically mean. Third convention is to start capitalizing only from the second word so the first d is lower case then the t for the travel is uppercase. For variable names the common

C++ convention is something like that. Capitalizing each word starting with the second word so int days traveled this is the variable name we will choose here. And then cin days traveled that's the variable we are going to read into. Now that we have the input from the user so we ask the user to enter the number of days they traveled. We read these days into this variable now we should calculate how many full weeks and what's the remaining days. So just as I spoke it the variable names should be probably full weeks and remaining days. So let's declare two additional variables one I would name full weeks. Note that I am using this convention lower case f upper case w. And remaining days once again lower case r upper case d. You can also note that when you declare more than one variable of the same type you can do it in a single line of code and separate the variables with a comma. In this case I declared two variables in this line both of them are of type int. One named full weeks and the other remaining days. So right here after we have the number of days traveled in order to figure out the number of full weeks we said we should div it by 7. So we do full weeks equals days traveled div 7 right. That's the number of full weeks and the remaining days would be numbers of days traveled mod 7. So remaining days equals day traveled mod 7. So I am using the percent sign here. So I have these values so now that I have calculated the amount of full days and how many days are remaining I can output the message to the user cout. And I want it to be in the format of in the case of the input of 19 19 days or two weeks and five days. So let's start with days traveled which would be our 19 and then days are which would be full weeks as a string. And then the remaining days days. After that we want to break the line something like that. So we have in the case of the input of 19 19 days are two weeks and five days. This should be fine let's execute it and see if we need to make some minor changes. So I am compiling it I am just pressing the play button here. And please enter the number of days you traveled. I entered 19 and then it says 19 days are two weeks and five days. There is a spacing issue here and actually it makes sense because we printed days traveled and then the text days are with no spaces. So if we want to space 19 and days we have to add a space over here. We should also add a space here before the two so there would be 19 space days space are space and then two and then again we need to add a space here and a space here and also space here. Now it should look better let's execute it. Once again let's enter the input 19 and then it would say 19 days are two weeks and five days. Seems to work perfectly fine. One last thing that I want to update here is the use of 7 here. So as you can see we have the variables days traveled full weeks remaining days. They are all variables with type int they are all data of type int they are all integers. And then we use 7 here and 7 here are also integers because they are C++ literals but this 7 is not an arbitrary numbers. We use 7 because 7 is the number of days in a week. A lot of times we prefer to define our own constant to represent this value. To make our code more clear. We typically define constants above the main. We can define it just next to the variables as well. Maybe later on we will discuss what's the difference of the location where we define our variables and constants. But for now let's define constants before the main. So up here I will do constant int and now I want to define a constant that the value would be 7. I am trying to think what I should name this integer constant and typically I choose the name by what it represents by describe what it represents. So 7 represents the number of days in a week. For constants the common convention for constant name is using all upper case and separating t he words with underscore. So it would be days in a week and then I would set it to 7. So not like variables where we define the variable all lower case and separate the words starting with the second one with upper case. In constants we have a different naming convention where we type the entire text as uppercase and separates the words with underscore. So days in a week equals 7 and then instead of just typing 7 here I would just divide it by days in a week and I would mod it by days in a week. So when someone reads our code now they can see that full weeks are basically days traveled div the numbers of days in a week. And the remaining days are days traveled mod the number of days in a week. There is no supposed to be any effect on the execution just instead of writing the literal 7 we define our own constant with the value of 7. So here when we type 19 we just get once again two weeks then five days.