Week 2 Module 4 Data Types and Expressions Part 3

The char Data Type 1.2

Ok so we've talked about ints that represent integers. Float and double that represent real numbers. Let's talk about the char type. Chars are used to represent characters each char data represents a single character. Let's see how we represent internally a character in the memory. So for example if we want to represent the letter A which is a character inside the memory obviously we won't just write a curly A inside the memory because the memory contains only zeroes and ones. So we should figure out a way to represent textual information in this case a character with only zeroes and ones. A common way to do that is to map a letter to a number that corresponds to it. So there would be a number mapped to the letter A a number mapped to the letter B a number mapped to actually lower case letters upper case letter digits some symbols the dollar sign question mark all the other punctuation marks and so on. Let's see how many different numbers we would need to in order to represent these characters that would give us how big should be each character data. So it seems like one byte that contains 8 bits actually contains 2 to the 8 different values 2 to the 8 is 256. And 256 seems more than enough to represent all the lower case letters upper case letters digits punctuations and stuff like that. So a character data each is takes one byte in the memory. Now the mapping the mapping function that gives each character its number its value is called the ASCII table you can see here that upper case a is mapped to 65 upper case b is mapped to 66 lower case a is mapped to 97 lower case b is mapped to 98. The dollar sign is mapped to 36 you can see that each symbol has its own value that is mapped to it. So for example if we want to write the data A inside the memory using the ASCII table the ASCII value corresponds to A is 97 and then we write 97 in a binary representation in this case would look like 011 four zeroes and a 1. And these 8 bits are now stored in the memory representing the letter A. So characters are mapped to numbers by the ASCII table which are then represented in binary and stored in the memory.

What's my ASCII Value? 1.3

Let's try to write a program that reads from the user a single character and then prints what's the ASCII value that corresponds to that character. For example the program would first ask the user please enter a character the user would say upper case T and then the program would respond by saying the ASCII value of upper case T is 84.

What's my ASCII Value? Implementation 1.4

Ok so let's go ahead and implement this program. First we need to ask the user for a character so cout please enter a character break the line. Then we should read this character for that let's declare a char variable. Input char and let's cin into this character this variable. Ok now we should basically convert this character to its ASCII value. For that let's have an integer to store this ASCII value and declare a variable name ASCII value. Basically I should assign this variable ASCII value with a ASCII of the input char. But if we think about it the input char variable basically contains the ASCII value in the memory so when the user enter a character in the keyboard it was translated by the compiler from its letter representation to its ASCII value. So basically we should say take the number stored in the character variable and copy it and translate it into an int variable. This way the since it will be inside an int variable it would be considered as a number and not as a character. So we should say something like ASCII value equals input char. Basically saying take the data from this variable which physically is a number and store it in this variable here which is an integer. So we can make this assignment just as is this assignment would create an implicit cast that would translate the char into an int and this cast is basically an implicit cast because it doesn't lose any accuracy. In a char there is only 256 different values and all of them can be

store with exactly the same value in an int variable. But just more formal we can say cast it into int. So take this input char cast it to an int and this int value this entire int value is stored in the ASCII value character. After we do that what we have left to do is announce it to the user so cout the ASCII value of and then let's have the input char printed is and then let's write the ASCII value. So when we print let's break the line. When we print character even though inside the memory there is a number the compiler knows to print the textual representation of this number. The letter that corresponds to this number by the ASCII table so this cout would print the letter or the character that the user entered and here since ASCII value is int cout would print the integer value. It would print it as a number in this case. So this should be fine let's try to execute it and make sure it all works as we expect it to. So ok please enter a character let's put upper case T and then it says the ASCII value of upper case T is 84 just as we expected.

Char Literals 1.5

Ok so we have the char data type for representing a character. Each character takes one byte of memory and we use the ASCII table in order to map letters into numbers. Let's see what built in literals there are in C++ for characters. So let's say we have a char variable ch and we want to assign A into this character. So obviously we cannot say something like ch equals A because the compiler would then try to assign would consider A to be a variable name and to assign the value of the variable A into the variable ch. In this case there is no variable A and that would result with a compilation error but there is another way to have a constant a literal of a value A in C++. We don't just write the letter A. For that we have the letter or each symbol enclose in a single quote. So we have single quote A single quote that is the constant or the literal for the letter A. Single quote upper case B single quote that would be a constant for the letter upper case B. And for the digits and for other symbols as well. So if we want to assign ch with a letter A we have it by ch equals and then A inside single quote. We can then cout this A and we can cout the literal itself cout quote B. We cannot by the way do ch equals double quotes A. The double quotes is for strings not for characters and that kind of assignment is totally illegal. So you can't do ch equal double quote A. There is a very big difference between character and string we will talk about the string data type in a few minutes. Ok so we have these literals what should we do if we want to have a character contain a value that cannot be printed or doesn't have a key on the keyboard. Or even if it does have a key on the keyboard we don't want to type it because it is like used for something else. For example if I want the character to be new line obviously I cannot do quote press enter and then another quote because pressing enter would just break the line in our code. So there should be another way to say to the compiler we want to represent a new line character. For that we use a special syntax named escape character which is back slash symbol. So we do it like that we have a single quote and then in the these two single quotes write back slash n. Basically telling the compiler it is not the letter N we don't want to represent the letter N now. The back slash and the n come together to represent a single character in the case of back slash n it represents the new line character. So we can have a character ch and then assign or in this case we can just print cout and back slash N. That would break the line. Equivalent by the way to cout endl same thing they both do. So we can assign ch with back slash n and then print this ch. We can also cout the string ABC and continue printing back slash n just like cout ABC and breaking the line. By the way a cool thing we can also do cout and then inside a string inside double quotes we can have ABC back slash N that would also print ABC and break the line. So the compiler knows that A is the first character in the string B is the second character in the string C is the third character in the string and then back slash since it starts with a back slash they come together and they represent new line character. So that would also print ABC and break the line. A few other back slashes a few other characters that have this escape character that uses escape character syntax. There is also a back slash T for tab and if we want to print or to use the back slash symbol itself we just do back

slash back slash. There are a few more but I think that's enough for the start. Let's talk about the arithmetic operators in context of char. Now typically arithmetic operators are used for numeric types but then characters are considered to be numeric types in this sense. And there are arithmetic operators for characters as well. For example we can add in context of characters. If we think about it it doesn't have a lot of sense to add I don't know A and W or to add C and the dollar sign. But taking a look at the ASCII table we can see that the lower case letters are all sequential and the upper case letters all come one after the other and the digit symbols also come after the other. So it would make sense to for example if we have a character A to add 1 to add an integer to the character A. Probably trying to say give the next ASCII value that comes after A so if we assign ch to be A plus 1 it would assign ch two in this case to be a B. If we print ch too it would print a B. Actually and that is kind of surprising if we cout A plus 1 I would expect it would also print a B but then if you try it out you will see that it doesn't print a B. It would print the ASCII value of B and this case it would be 98. And if we take a closer look here we will see that when we have A plus 1 A is a character 1 is an integer and when we add datas of different types if you recall in one of our previous talk we said that when we have mixed types expression one of the types is then casted to the other type in order to match the types. In this case an integer cannot be casted to a character without losing accuracy but a character can be casted to an int without losing accuracy. So A is basically casted to an int to 97 and then this 97 is added 1 and it comes to be 98. So A plus 1 is an int expression that's why it prints 98. You can take a look back at our assignment A plus 1 into ch two so A plus 1 is an int ch is a char is a character so this assignment basically has an implicit cast in it. Basically transforming translating casting the integer back to a character. If we want to make it more explicit we can for the cout for example say cout char casting of A plus 1. Basically saying transform cast translate this A plus 1 integer 98 back to a character that would print the character representation of 98 that would print a B. A few other arithmetical operators since we can add integers to characters moving in our ASCII table we can also subtract an integer from an ASCII value from a character. Basically again moving inside our ASCII table we can assign obviously characters as we have seen.

Convert to UPPER CASE 1.6

Ok so let's take all the syntax we've talked about characters and implement the following program. Let's write a program that reads from the user a lower case letter and convert it to its upper case equivalent and prints the upper case letter of the lower case that was read. For example the program would start by asking the user please enter a lower case letter the user can for example enter lower case T and the program would respond by saying upper case of lower case T is upper case T. Let's try to think how we can implement this kind of behavior. So we already noticed that the lower case letters are sequential the upper case letters are sequential and the digits are all sequential. Using this property we can see that the distance from lower case T to lower case A is the same this offset is the same as the offset from upper case T to upper case A. We can then maybe calculate what is the offset of lower case T from lower case A and add the same distance the same offset to upper case A that would get us to upper case T. Let's use this observation in order to implement the program.

Convert to UPPER CASE Implementation 1.7

Ok so let's implement this program. First let's ask the user for the lower case letter so cout please enter a lower case letter. Let's break the line. Let's read it into a variable let's first declare it lower case letter and let's read it into this variable so cin into this variable here. And now we should convert lower case letter to upper case letter. So let's have an upper case upper case letter. Eventually the upper case letter variable would store the equivalent of the lower case letter value. But we said that we first have to calculate the offset of the lower case from lower case A. So let's have an integer variable named offset

that would store this offset and offset would then be whatever the value of lower case letter is from lower case A. So I have lower case letter minutes lower case A. Now this is a character and this is a character and I want the integer value of this expression here. So subtract this two ASCII values the ASCII of let's say lower case T and ASCII of lower case A and then you would have your offset value. So for example in lower case T the ASCII value is 116 and the ASCII value of A is 97 so you subtract 116 from 97 and that value would be our offset. Now we need to add this offset to upper case A and that would be our upper case letter. So upper case letter would then just be upper case A plus this offset. That would add whatever the offset was to 65 and would get us to 84 to upper case T. Again we don't really have to do all the casting but let's cast to char this expression here which is of type int. Once again when we add a char to an int it would cast the char to an int first and it would add it as two integers. So let's cast it back into char and that would be our upper case letter. Let's now print the output message for some reason I spelled it uppet case letter so let's change both of the declaration and where we used it. Now that it is consisted let's cout upper case of and now we need to have the value of the lower case letter since it is a char it would print it as a letter. And then is and then the uppercase letter and break the line. I can endl or back slash n that would both work. Let's try to execute it see if we don't have any silly mistakes here. Once again please enter the lower case letter we have lower case T and then it says that the upper case of lower case T is upper case T. Very well then

The string Class 2.1

Ok so we have integers we have float and doubles we have char for a single character. The next type I want to talk about is called string. Actually the string type is not a built in type in C++. In order to use the string type we would need to include to extend our language with the string library. So before using the string type we have to pound include and then string. Just as we've extended our language for the c math and for the iostream things that are not built in in C++ we use external libraries to use this kind of syntax. Same thing with string here since it is a very useful type it is important for me to show it to you right here in the beginning of your experience with C++. So after including this string class the data that is represented using this type is a string or a text. In a representation we won't get too much into the details maybe later modules we will talk about strings in more detail. For now all I want you to know is that strings are basically sequences of characters just characters coming up in the memory one after the other. Literals built in C++ strings again not C++ but the string library strings are quotes and then a sequence of characters. So quote double quotes ABC or double quotes this is a string back slash each of them is of type string or could be considered of type string though formally they are not. Arithmetic operators in this case there are plus and assignment. Plus is a very useful operator that basically concatenates string. Let's see a few lines of code to get the hang of this syntax. So for example if we have an int variable x a double variable y obviously we can assign x with 5 and y with 7.3. In order to create a string variable as we said first we need to include string. And then we can declare a string s variable and we can set it with a literal for example hello. So s equals double quote hello. We can then cout s that would print hello and break the line endl. We can also use the concatenation operator the plus operator to for example cout s plus the string world. So s is a string double quotes word is a string and we can add a string to a string that would concatenate the text. So we would print hello world back slash or in this case enter. We can also assign s with s plus world that would turn s instead of having text hello the string hello to have the string hello world. Then we can cout s that would print hello world. So it is a very easy syntax to use but then it is very useful because there is a lot of programs would need to communicate with humans and strings and texts are the way to do that.