Week 5 Module 10 Strings

Introduction 1.2

Today we're going to talk in more detail about the class string. We've already seen the string data type. Today we are going to see some more abilities of this class.

Initializing & Concatenating Strings 1.3

So far we've seen that in order to use the string type we need to include the string library. Then we can declare a string variable in this case string str. And then we can initialize str with some textual data which is basically a line of text or a string enclosed in double quotes. Formally the double quote ABC is not a class string type it is formally a c string type. But C++ has a default an implicit cast from c string to the string class. So for us it would be considered to be just a string so we would have str equals ABC. And then we can also use the plus operator that's also a thing that we've seen about strings. Plus basically let's us concatenate strings one after the other. So we'll do for example str plus DEF since str is ABC str plus DEF basically concatenates ABC and DEF giving us the text ABCDEF. So when we print it we just get ABCDEF.

Reading Strings 1.4

For now we know how to initialize and concatenate strings using the assignment operator and the plus operator. Let's see how we can read a string from the user. So for example let's take a look at these few lines of code. If we print please enter your name ask the user prompt the user to enter their name we can then cin into a string variable. And afterwards we can cout this value this name this str. So when we executed we ask the user please enter your name the user says for example I don't know Donald Duck. And then kind of surprisingly when we print str we get only Donald back. If we take a look what happened here basically when we used cin in order to read the string it only read the first word out of the line that the user basically entered. And that's how cin works. That's the way cin was designed. It is skipping leading white spaces then reads the word and stops at the next white space. In this case cin read Donald into str. If we want to read an entire line and that's a thing we would like to do a lot of times we would need to use some other way of reading a string and that would be by using the get line the get line function. It works something like that we just call get line and get line has two parameters that's kind of surprisingly. The first parameter is cin itself and then the string we want to update the line with. So get line cin str basically reads an entire line basically a sequence of characters until the new line is pressed into the str variable. You'll see in later modules that get line can be given a different first parameter besides cin and for now what I want you to know is that this cin basically tells get line to read this line from the standard input. From the basically the keyboard from whatever the user enters. Later on you will see that there could be other inputs where or other streams that can give us input besides cin or besides the standard input. So if we execute this code here we would prompt the user to enter their name and then when the user enters Donald Duck get line cin str would read the entire Donald Duck into str. Then when we print str we would get Donald Duck printed.

Indexing Strings 1.5

Ok so we already know how to initialize concatenate read a string. Let's see some more stuff we can do with strings. So strings are basically a sequence of characters just like arrays can be a sequence of array of integers is a sequence of integers. Array of double is a sequence of doubles and so on. So it would be natural to wonder if we can use this collection or sequence of characters just as we use arrays. Basically indexing separating single elements from this collection. And apparently we can do that. The syntax for indexing with strings is very similar to what we are doing with arrays. Basically we are just having str and then in a square brackets the index of the character we're we want to access. For example if we have a string str one initialized to ABCDEFG we can then access str one at the index zero that would be A. Str one index one that would be B. Str one index two that would be C. If we print them with a space between them we'd get A space B space C. Basically each element in this string is of type char. So we can for example set ch which is char variable to str one index three. So in this case ch would be D.

Slicing Strings 1.6

So we can basically take separate elements from a string using the indexing system indexing syntax just like arrays. We can do something even a bit more advanced with strings we can take a slice of a string out. That is the slicing syntax and it works something like that. In order to take a slice to take a sub string out of a string we use the sub str method. The way we call this method is str one dot sub str and then we have two parameters basically saying where this sub string starts and how long is this sub string. In this case this sub string would start at index three and it would be two characters long. So index three is a D two characters long the sub string is basically DE in this case. Now I believe you've noticed that the syntax for calling this method is a bit different than what we are used to when we've used functions. Probably you've you probably thought that in order to create the sub string of str one starting at index three and it's two characters long. We should use sub str as a function and pass str one and three and two as three different parameters and that would obviously a right way of thinking. That's how we've used functions so far. But then I've said that sub str is a method and that's a bit different. We'll talk about it in much more detail when we speak about object oriented programming and classes and you will see that methods how exactly or how formally they are defined. For now I just want you to know that sub str is a method it is not a function so we are not calling it sub str with three parameters the string the starting index and the length. We are kind of passing the str one parameter in a different way. Str one is basically the calling object of this method. So sub str is aware of str one it is operating on str on it is taking the sub str of str one without us passing it as a parameter. We've used the dot notation in order to make str one be our calling object. So we use str one dot sub str three and two. So we are calling the method sub str on str one and the additional parameters are three and two. If we'll for example print str one dot sub str three two that would print DE. I hope it makes sense we will use methods in other cases as well. Each time we are going to use a method I will explicitly say that this functionality is not a function it is a method. In this case sub str is a method. If we'll think about the result or the value of whatever sub str returns that is basically in this case it was a DE it was of type string. So we can do something like str two equals str one dot sub str two three. Basically taking the sub string of str one once again our calling object is str one. Starting at index two in this case which is the C and it is three characters long. So str or the result of this method call would be a string of length three CDE. And that would be str two so we are assigning str one dot sub str into a string variable into str two. If we'll print str two as expected it would print CDE which is the three characters string that sub str returned. So the slicing syntax is using the sub str method we have our calling object for example it could be the str variable and then two additional parameters come in parentheses. The starting location and the length of this sub string.

Length 1.7

One last thing I want to say in this context here is to tell you about a new or another method that is defined in the string class. And that's the length method. The syntax is str dot length and then empty parenthesis. Once again it is a method so in order to calculate the length of a string you need to give the string as the calling object. So it would be I don't know str one or str two or whatever the variable name is dot length and then empty parenthesis and that would result as the length of the string. The number of characters in that string. So that is also useful if we want to figure out how many characters are in a string. If we want to iterate over the string or for any other reason.

Printing Backwards Introduction 1.8

Ok let's use this syntax in order to implement the problem. So this program would read the user's name and print this name in a reverse order. So for example we would prompt the user please enter your name the user would say Donald Duck and then it would print whatever so Donald Duck in a reverse order. So let's go ahead and implement it.

Printing Backwards Implementation 1.9

Let’s go ahead and implement this program. So first let’s prompt the user cout please enter your name and break the line. And then let’s read the user’s name so for that let’s first include the string library and let’s create a string variable. Let’s name it username. And let’s read whatever the user enters into the user name variable. Since the user name would be first and last name it would probably contain spaces so let’s use the get line syntax for that. So get line first parameter as we said it’s cin basically saying that that input comes from the keyboard from the user. And the user name that is where we want it to read it to. So now we have the user name string. What we need to do now is what I was thinking is to create a new string with the reverse name in it. Let’s create another string variable or actually since we’ll just need to print this name we don’t need to create a string we can just iterate and print in each iterator. But then we would need to go in a reverse order we need to go from the enter to the beginning. I think it is a good practice let’s use a for loop. I will have an index variable and in this case we should initialize index to the end of the string and each iterator instead of incrementing index we will decrease index. We’ll do something like index minus minus. So let’s think of what value the index should be set to initially. That should be the index of the last character in the string. We can definitely use the length method for that so it would be user name dot length that’s basically the number of characters in this string. But then if we think about it let’s say we have a string ABC the length is three but the last character is in the index two right. Because like since it is a zero based index system it starts at zero and ends at basically length minus one. So let’s set the initial index to username length minus one. That’s the index that’s the position of the last character in the username string. So initially index is the length minus one. Each iterator index is decreased and we keep on going as long as our index is greater or equal to zero. Basically it is still a valid index we can access. Each iterator we’ll just print the current character we are at. So let’s cout username at the index position so first iteration would print the last character. Then we’ll decrease index and we’ll print the character that comes before that. We’ll decrease index and print the character that comes before that and so on until index equals zero in the last iteration where we’d print the first character. At the end let’s just break the line so cout end l here. And seems to be fine let’s try to execute it make sure we didn’t make any silly mistakes. So please enter your name Donald Duck and then it just prints it in a reverse order. Great another option here would be as I started before to create a string variable with a reverse name in it it would be a great practice for you to try to implement this version on your own.

Comparing Strings 1.10

Ok so we know how to initialize how to concatenate how to read strings from the user. We can access specific characters using the indexing system syntax we can slice out a substring we can figure out the length. Let's see how we can compare strings. So obviously we can use a double equal signs in order to figure out if two strings are equal to one another or not equal to one another. But how about the less than. Can we also try and find whether one string is less than the other? For example if we have str one is A B C and str two is D E. Can we do something like if str one is less than str two? For example in this case we'll print that str one is smaller than str two otherwise we'll print that str one is not smaller than str two. What do you think it would happen if we try to execute these few lines of code? So one option it would figure out or let's say first start in the second option. So one way is that that could be a compilation error. The compiler won't approve comparing strings with less than operator. Another option is that that would be legal expression in C++ but then if so what do you think it would happen? Does str one is smaller is less than str two? Is ABC smaller than DE? So let me first tell you that it is not a compilation error. It is a legal expression in C++ it is a legal a Boolean expression actually in C++. But then when we talked about the less than operator we compared integers and doubles. I didn't explain too much it was kind of intuitive if four is less than seven or five point five is less than six point seven. But in case of strings it is not very straight forward. Is ABC less than DE? How do we compare strings? By their length? ABC is a three length string or DE is a two lengths strings so in this case ABC is not less than DE. Three is not less than two. But then if we try to execute this code we'll see that it would print that ABC is smaller than DE. In other words ABC is less than DE. So it is not comparing the strings by their length it has other criteria in order to determine whether one string is less than the other. And the criteria C++ chooses and it is common in other languages as well not only C++. The criteria by which we compare strings is by a lexicographical order. Other words in alphabetical order what string comes before that in let's say in a dictionary. So obviously ABC is before DE therefore ABC is less than DE. So in this case str one was less than str two that was true therefore it was it printed that str one is smaller than str two. So when comparing strings we can use the less than the greater than less or equal to greater or equal to operators. And it would do some kind of lexicographical ordering or in alphabetical ordering. In more detail the way we compare the way C++ compares strings let's say we have two strings ABCDEFG and ABCDXYZ. So when C++ tries to compare these strings it kind of iterates over the strings simultaneously and figures out the first location where they differ from one another. So first A and A are equal it goes forward to B and B C and C D and D and then the first position where they differ from one another is E and X. That's how or that's the point where C++ determines whether one of them is less than the other and the criteria is basically comparing the ASCII value of E and X. In this case E is less than X so the entire string ABCDEFG is less than the entire string ABCDXYZ.

First Word Introduction 1.11

Let's try to write a program that reads from the user three words and then prints the one that comes first in an alphabetical order. For example let's prompt the user please enter three words separated by space. The user can then enter I don't know dolphin cat tiger. And then the program should respond with word that comes first in a lexicographical order. In this case it is cat. Let's go ahead and implement it.

First Word Implementation 1.12

Let's start by prompting the user to enter three words separated by space. So please enter three words separated by space and let's break the line. And then let's read these three words. Let's first include the string library and then let's declare three string variables. So let's have word one word two and word three. And now we need to read three words into these variables. Since each of them is supposed to be a single word actually cin is a good way to read these words into these variables because cin separated by space. So we can just do cin into word one into word two and into word three. So now we have the three words in these variables. Now let's try to determine which one is the smallest. So I was thinking of having a multiway if statement here three cases. Checking if word one is the smallest if word two is the smallest or otherwise it is obviously word three. So let's have an if statement and let's figure out a condition that determines whether word one is the smallest of the three. So we can do something like if word one is less or equal to word two and also word one is less or equal to word three. If this is true basically it means that word one is both less than word two and less than word three. It is thus the smallest of them all in this case we'll just cout word one let's break the line. Otherwise let's do the same to check if word two is the smallest so if word two is less or equal to word one and word two is less or equal to word three. That means word two is the smallest of the three so we should print word two break the line. And otherwise I will just comment that word three is the smallest in this caser we'll just print word three. That's supposed to be good enough let's test and make sure we didn't do anything wrong here. So please enter three words separated by space so we have dolphin cat and tiger. And that it just prints cat and yea that is the smallest word of these three.

Searching in a String 1.13

Ok so we know a lot of things about the string class. One more thing I want to talk about now is searching in a string. So assuming you have a string and you want to search if you have some substring contained in it. The syntax to do that is by using the find method. You call the find method on an original str string so you do str dot find. And then you pass as a parameter the string you are searching for in this case s. Let me show you how it works for example if we have str the original string ABCDEFG whatever you can call str dot find of DE. Basically searching for DE inside the original string the original str. In this case you can see that DE is a substring of the original string the find method would just return the starting index of DE. So A is zero B is one C and so on DE starts I think in index six so if we print whatever str dot find returns in this case it would just print six. Let's take a look at another call for the find method for example if we'll call str dot find on a string XYZ. In this case XYZ is not a substring of the real original str so find needs to tell us needs to return some value that says that XYZ is not found in str. Find returns in this case a constant named npos that is stored in the string namespace. In order to access the string namespace basically we use the syntax string double colons and then npos that's the constant name. So we can do something like if str dot find XYZ equals to the npos constant we can just print not found otherwise if it doesn't equal to npos basically it returns a valid index it is found. In this case when we are searching for XYZ obviously it would return that it didn't find the XYZ in the original str and therefore it would print not found. Let's take another look here. What do you think it would happen if we'll call the find method to search for CD in the original str? I believe you noticed that CD appears more than once in this original string. It appears ABCD and then B and another CD so we have two occurrences of CD. Any guesses on what find would return when we call it to search for CD? So maybe it would return arbitrarily one of the positions maybe it would return a collection of the entire indices. I'll just tell you that find the find method just returns where the first occurrence of the string we are searching for appears in the original string. In this case CD first appears at index two therefore it would print two.

Starting Index 1.14

So this is the syntax for the find method. We have our calling object str dot find and then an additional string we are searching for passed a parameter to the method. I want to say or tell you about an extension of the find method that in addition to the string we are searching for we also pass a starting index. Let me show you how it works. So let's go back to the example we've talked before. If we'll search for CD and give a starting index of three. In this case the first occurrence of CD after index three would be actually the second occurrence of CD in the original string which is I think index five. Yeah that's what it basically prints. So the extended version searches for s but starts not at the beginning but at whatever starting index we give it. And gives us the first occurrence of the s after that position.