Introduction to Problem Solving and Programming

Prof. Deepak Gupta

Department of Computer Science Engineering

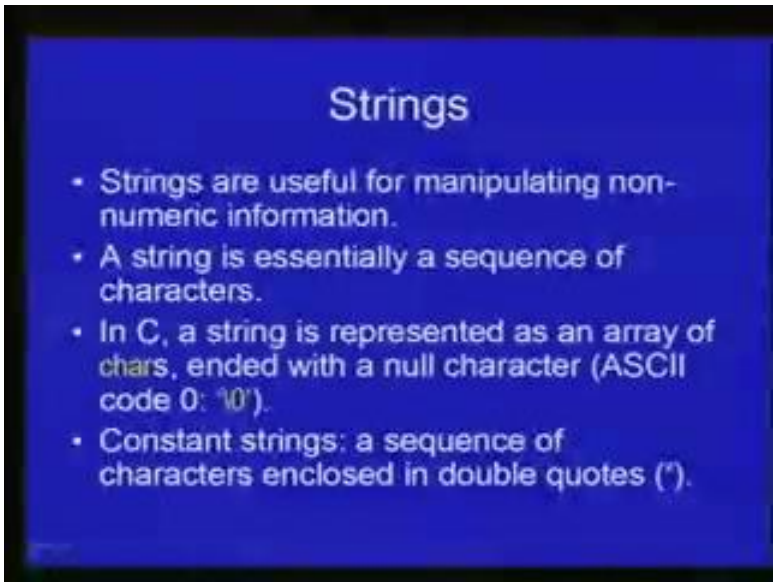
Indian Institute of Technology, Kanpur

Lecture - 13

(Refer Slide Time: 00:36)

In the last lecture, we had talked about arrays and we saw how arrays can be used to store and systematically manipulate large number of data elements. Today we will see another very useful constructive data type that known as string.

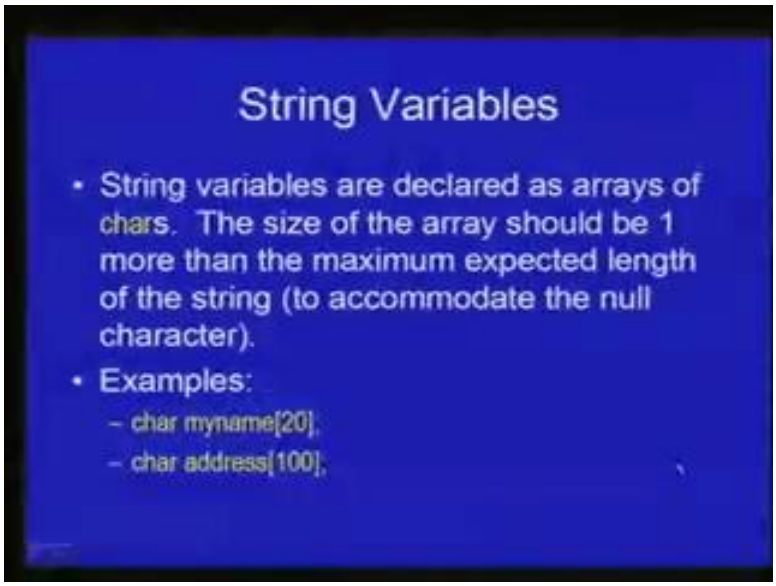
(Refer Slide Time: 00:40)



Strings are useful for manipulating non-numeric information. As we said in one of our earlier lecture one of the major powers of the computer addressed from fact that computers can deal not only with numeric information that is they can not only deal with numbers. But they can also deal with information in a wide variety of other formats, for example, the text characters, and sound and audio and so on. So, today we will look at strings which allow us to write programs that can deal with text characters in the input.

So, essentially a string is a sequence of characters and in C when you want to declare a string variable, we will declare it as an array of chars and the special property of the string as in array of chars is that it ends with a null character. That is after the number of character that we want to store in array the next character that is stored is a null character which has ASCII code zero and is also represented in the programs with this notation which is within codes backslash zero. And constant strings are represented as the sequence of character enclosed in the double quotes character.

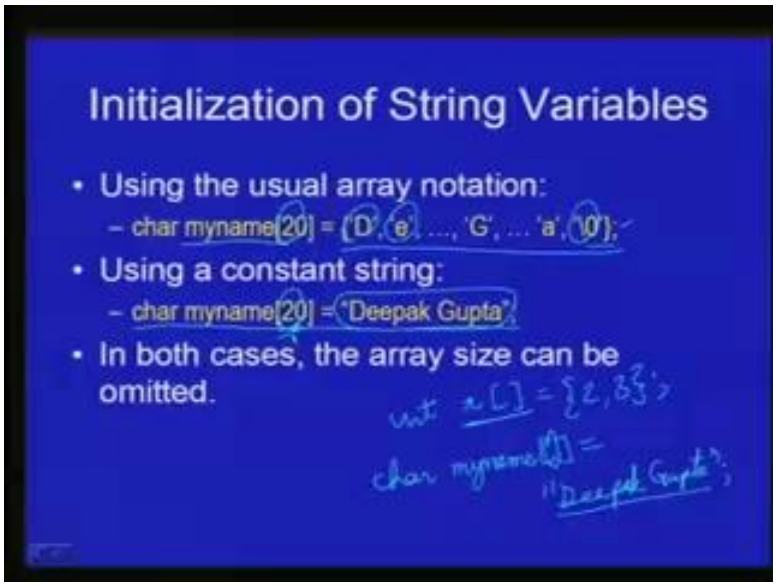
(Refer Slide Time: 02:05)



So, as I said string variable would be declared as array of the characters. The size for the array that we need to declare should be at least one more than the maximum expected length of the string. And that one more is required because the string will store not just the characters that you want to store in the string, but also the terminating null character. And therefore, its length must be at least one more than the length of the maximum string that we expect to store in that array. So, here are a couple of examples of declarations of string. So, the variable my name is a string with size twenty, and so as you can see it is array declared of char of size twenty.

So, this means that I can use this variable to store any string which has at most nineteen characters followed by one null character. And of course, this variables can be manipulated just like other arrays can be, but we will see that there are some special library functions and special notations for manipulating strings as a whole. Similarly, here is another string variable declared address, which has the size hundred; that means that it can have strings of length up to ninety-nine. Strings can be initializes like arrays can be initialized as we saw in the last class one way of initializing a string variable is to initialize it using the notation that we use for initializing arrays that is we enclose the characters in the string within curly braces.

(Refer Slide Time: 03:21)



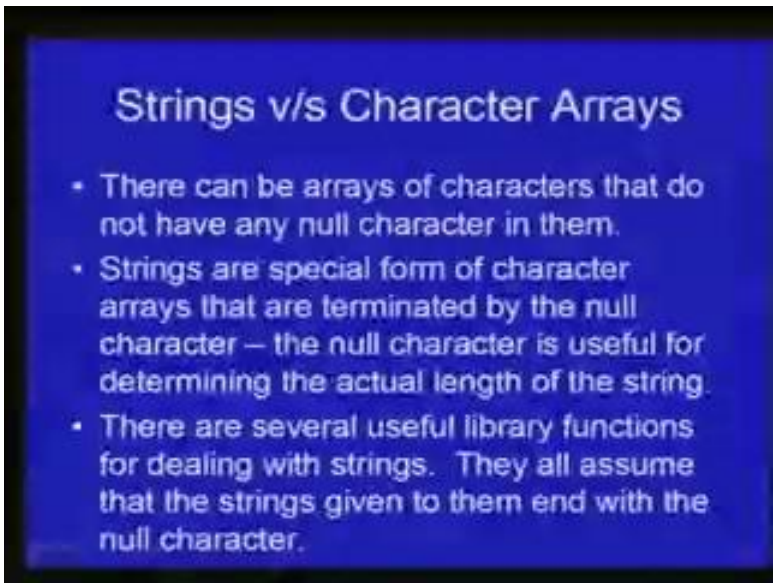
So, in this example what is being done is that this variable myname is being initialized to a string which is myname Deepak Gupta. So, I have created an array in this notation comprising of characters from my name and the last character is the null character. Note that even though I have given the length of the array as twenty, but the number of character that will be specified here is actually much less, the size of myname including the blank space is just twelve characters, and including the null characters that is thirteen characters. So, I have initialized only the first thirteen elements of this array the remaining seven elements are left un initialized.

And here is another way of doing this, this is special to string this cannot be done with other kinds of arrays. I could have initialize this to the string Deepak Gupta by putting the string simply like this within double quotes, and this has the same effect as the earlier initialization. Another thing is that in both these kind of declarations, I can leave out the array size, and the compiler will automatically compute what should be the correct size. In fact, that is a general rule for initializing the arrays in general for example, if I declare an array as the following.

So, in this case, the compiler will automatically assign the size two to the array x; and similarly, in this case, I could have initialized this variable in the following notation without giving any size of the array over here. And compiler would automatically compute that there are twelve characters in the string plus one character is needed for the

null character. So, the array created would have size thirteen character. But note that in this case, the string would have size thirteen and later on we would not be able to store any string in this array which is of size more than twelve characters. Whereas when we have explicitly declared the length as twenty then later on we can change this string and store in this variable a larger string up to the size of nineteen characters.

(Refer Slide Time: 06:23)

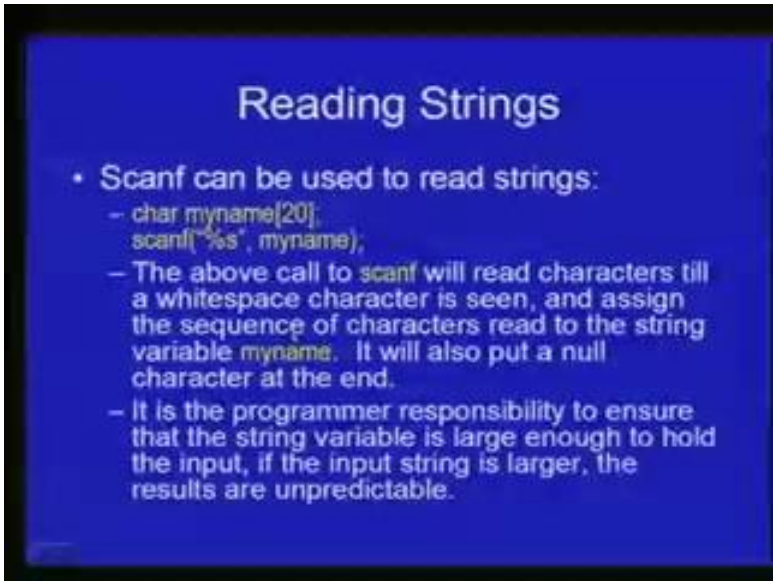


So, strings we have seen are the same as character arrays, but with a difference you can have a array of characters with no null character anywhere in it. Now, the strings have a special property that they are character arrays, but they are terminated by null character that is the last character at the end of the string is a null character. And this property is useful because this allows various library functions as we will see to find out what is the actual length of the string. The actual length of the string should be carefully distinguished from the declared length of the array. So, for example, when we want to print a string we want to print the actual number of characters in the string and not the entire array because some part of the array may not be getting used.

So, this null character at the end of the other characters in the string is helpful in determining the actual length of the string and. So, when we say a string we actually mean a character array which is terminated by the null character. As we will see, there are several useful library functions for dealing with strings, all of them assume that the

strings given to them as input contain the null character and that is the end with null character.

(Refer Slide Time: 07:33)



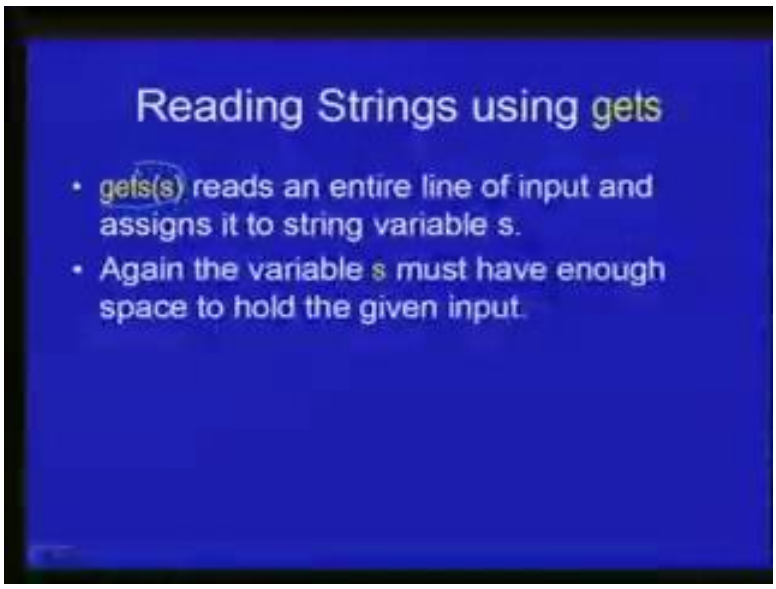
So, let us start seeing some of these useful library functions for dealing with strings here is first function which is the familiar scanf and it we have seen how scanf can be used to read integers and floating point numbers and so on and so forth. Scanf as a matter of fact can also be used to read strings, and all you have to do to read string using scanf is to use the percent s specifier instead of the percent d specifier as for an integer or for percent s for a float and so and so forth.

So, in this example, we have a declare we have declared a string of a size twenty and we are using scanf percent s to read a value into the string. So, first of all, note that there is no ampersand before the variable name myname in the scanf call this is different from the notation used for reading integers and so on. The reason for this will become clear later on, but the only thing you have to remember right now is that when you are reading strings, you do not put the ampersand there in the call to scanf.

So, exactly what this scanf call will do is that it will read characters from the input till a wide space character is seen, we call that wide space character are the blank characters the tab and the new line characters and so on. And whatever characters have been read till the first wide space character was seen, they will be assigned to the string variable. And of course, the scanf will also put a terminating null character after these characters

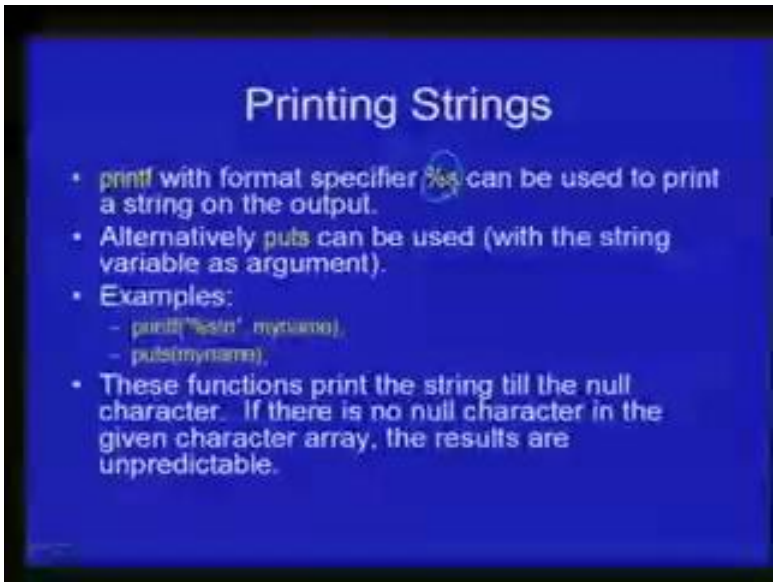
in the string. Now, one question that you might ask is that what happens in this example if the word that is typed once by the user has more than nineteen characters in it. While in that case, the string given will not fit in the string variable myname, but will not the compiler will not be able to detect this happening at compile time or at run time. Therefore, it is really a programmers responsibility to ensure that this never happens and the string variable is large enough to hold the largest expected input that the user might give, because if string is not large enough then the results are really un predictable.

(Refer Slide Time: 10:10)



So, as we just saw scanf is the format specifier percent s can be used to read a word from the input. If you want to read an entire line from the input, we can use the gets function instead and using the get(s) function is considerably simple simpler than using scanf. All we need to do is to pass the string variable name as the argument to the get(s) function. So, this function will read an entire line of input and assign it to the string variable s, and as usual it will put the null character at the end and once again the variable s must have enough space to hold the given input. Otherwise the results are likely to be unpredictable.

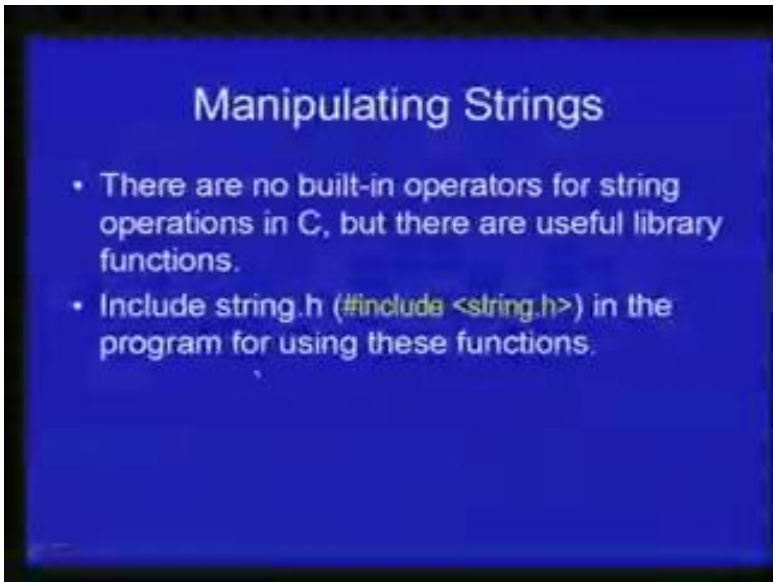
(Refer Slide Time: 10:49)



Printing strings can again be done with the similar function printf with the same format specifier percent s that we used for scanf. So, printf percent s some string variable name will print that string on the output. We will see an example of this in the example program that we take up today. Alternatively puts can be used with string variable s and argument. So, here are a couple of examples, one of printing the same variable printf on the screen one using printf function and the other using the puts function. So, with the printf function we have used the format specifier percent s and the corresponding argument is the name of the string variable and for puts knownly argument is the string variable.

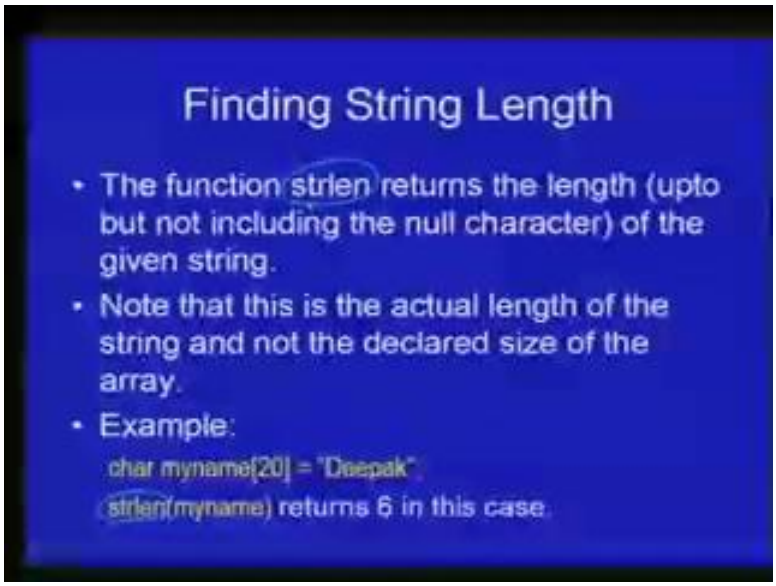
So, these functions it is important to relive print the string till the null character is seen in the string. So, that is how they determine how much data to be print, how many characters are there to be printed. If suppose the character array that is passed as the argument to these function does not contain any null character then again the result are going to be un predictable and that should never be attempted.

(Refer Slide Time: 12:01)



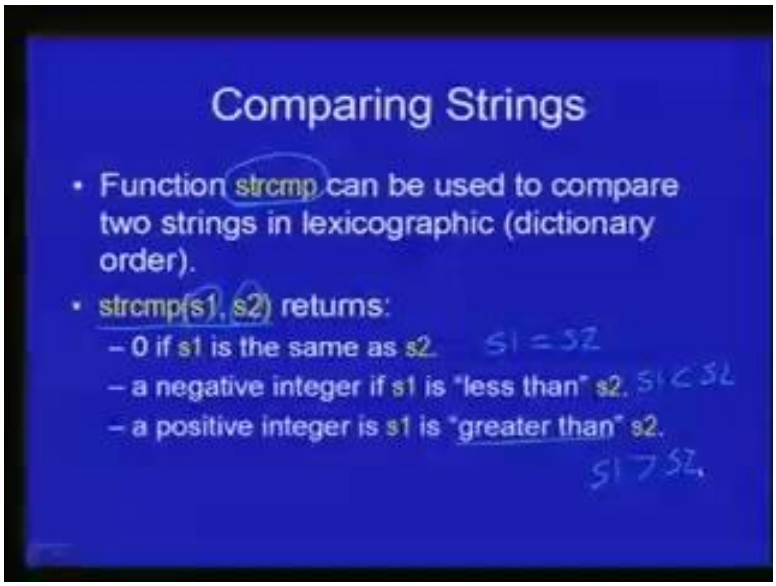
Let us now look at some functions which are useful for manipulating strings there are no built in operators in C for dealing with strings for example, for integers we have the plus minus and comparison operators and even assignment. So, all those kinds of operators are not there for strings. Fortunately, there are number of useful library functions that can be used for performing all these common operations on strings. And we will look at some of these string functions, but first before using any of these string functions, we need to put this statement at the top in our program. Just like we include stdio dot h to use the standard IO library functions we need to include the file string dot h in our program for using the string functions that we are going to look at soon.

(Refer Slide Time: 12:51)



So, the first function that we are going to look at is the str length function. Here it is which can be used to find simply the length of the string, and again ret and again note that it returns the actual length of the string and not the declared size of the array and the actual length of the string is the number of characters before the first null character in the string. So, in this example, we have again declared a variable myname of size twenty and it is initialized to the string Deepak which as you can see has six characters in this. So, if you call str length on the string myname, it will return six. We can also compare strings and the way to do that is use the library function strcmp. Now what does string comparison really mean, when do we say that is a string is larger than another string.

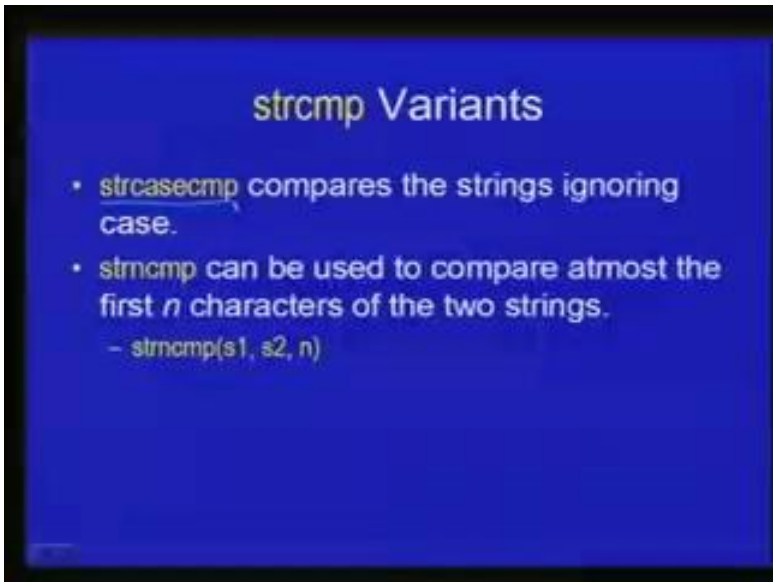
(Refer Slide Time: 13:59)



We say that string is larger than an another string if first string would appear later in a dictionary than as compared to the first string. So, this order is called the dictionary order or the lexicographic order. And the function string compare or strcmp compares two given strings in this order. So, the arguments to strcmp are the two strings s one and s two that we need to compare and this function returns zero if s one is same as the string s two. And it returns negative value if s 1 is less than the string s 2 meaning that s 1 would appear first in the dictionary as compared to s 2. Note that of course, in in the dictionary all the words have only alphabet characters whereas, the strings s 1 and s 2 in general can have other characters as well.

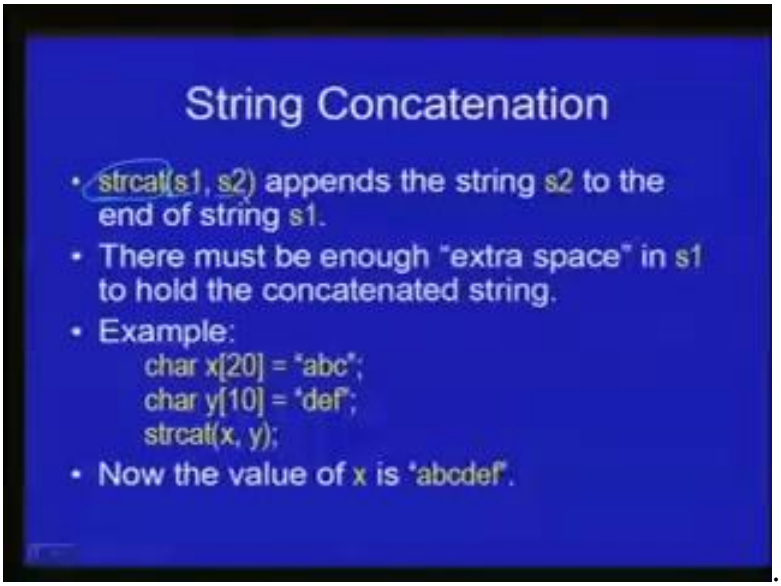
So, how do we compare two individual characters to determine which comes first in the dictionary in the lexicographic order well a character is smaller than another character if its ASCII code is smaller. And as we have already seen that the ASCII codes of small a, small b, small c etcetera, they are one after the other, so this ordering is consistent with the dictionary ordering. And finally, string compare s 1 s 2 returns a positive integer if s one is greater than in the dictionary order or the lexicographic order than the string s two.

(Refer Slide Time: 15:21)



So, this function can be used for comparing strings, there are some variants of this function. For example, strcasecmp compare the string ignoring case that is whenever it is comparing two alphabetical character, it ignores the case that is small a and A are treated as the same. Whereas the string compare function would treat them differently that is they would not be treat it as equal the other useful string comparison function is this strncmp function where you can specify a third argument which is an integer to the function. And this says that compare utmost the first n characters of the string s one and s two.

(Refer Slide Time: 16:01)

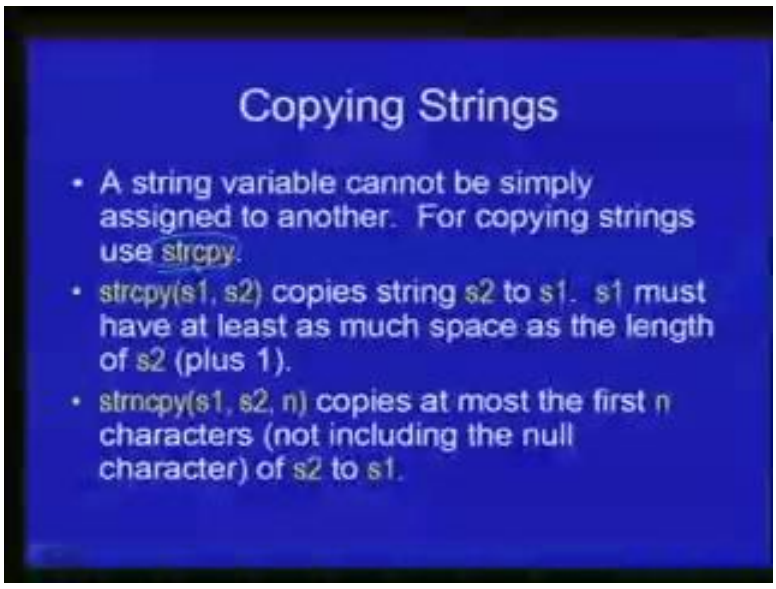


.

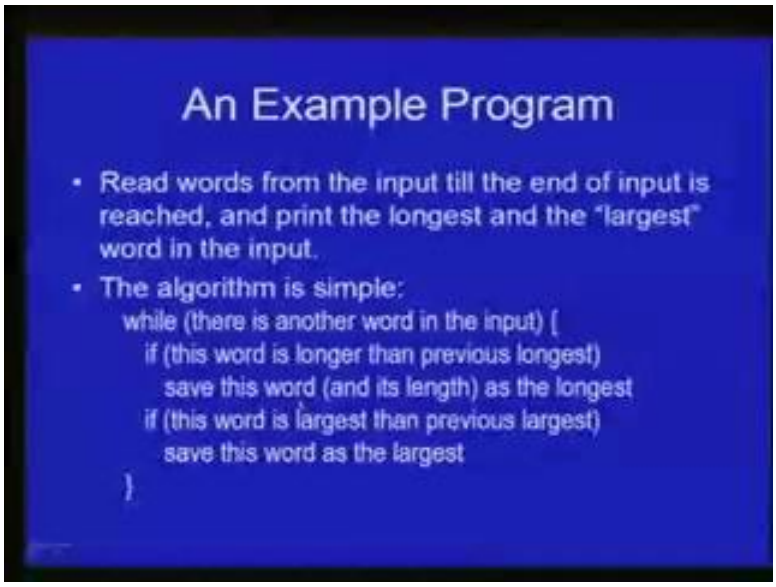
So, here is a function which can be use to concatenate a string to another string that function name is strcat. The arguments are two strings s one and s two and the result of calling this function strcat is that the string s two would be appended at the end of the string s one that is the string s one would change. And at the end of it, the string s two would be appended, and naturally for this operation there must be enough etcetera space in s one to hold the concatenated string, otherwise again the results will be un predictable. Here is a simple example. So, we have the variable x of size twenty containing the string just abc, which is of size length three of course, and here is another variable string variable y of size ten which has a string def.

Now, if you perform the operation strcat x comma y then it will place the contents of the string y, which is the string def at the end of the string contained in x, which is abc. So, at this end of this operation, the final value of x will be the string abcdef. So, if you note what is happening here is that the string x contain a followed by b followed by c followed by a null character and the string y contain the character def followed by the null character. And the string concatenation operator would replace this null character by the first character in the string y and then this will be followed by e, it will followed by f and then the null character will come at the end. So, finally, the value of x will be this string abcdef.

(Refer Slide Time: 17:45)



We can also copy strings from one variable to another variable. We cannot simply use assignment to do that in the case of strings as we do it for integers and so on. We cannot say x assigned y where x and y are string variables. In fact, if you do that the compiler will give an error. So, if you copy a string from one variable to another we have to use the library function strcpy which stands for string copy. So, strcpy s one s two copies the string s two to s one and whatever the whatever string the variable s one initially contained will be overwritten. And s one length or s one size must be at least as much as the length of s two plus one plus one, because the null character also has to be stored. And similarly, we have the function strncpy and this has an additional argument n which is an integer this says copy at most n characters from the string s two to the string s one. (Refer Slide Time: 19:06)

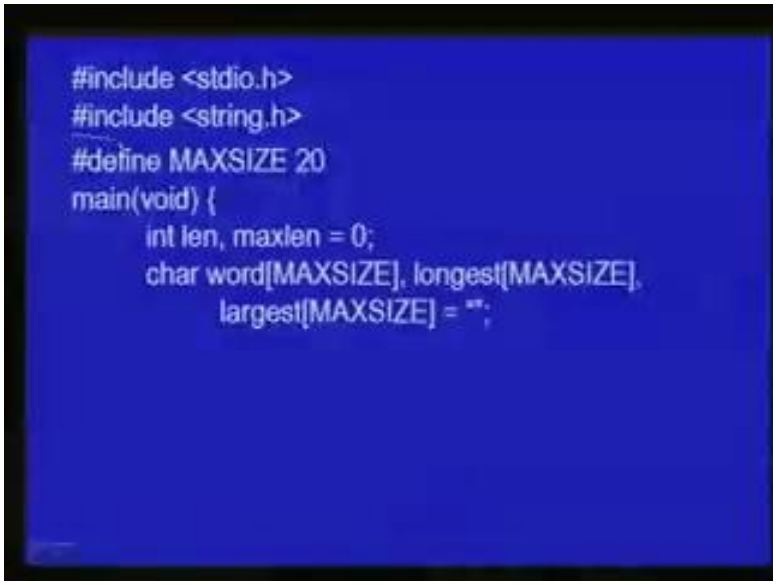


So, at the end of this lecture, let us take a simple develop a simple program to understand and reinforce the concept that we have learnt in this lecture and this program is is very simple What we are supposed to do is to read words from the input till en end of the input is reached. And then we are supposed to the program is supposed to print the longest and the largest word in the input. The longest word is the one, which has the largest length and the largest word is the one, which is lexicographically largest among all the words in the input.

So, the algorithm is reasonably straight forward as you can imagine as long as we keep getting more words in the input we read a word, and we compute its length, and we compare it compare the length of this word with the length of the longest word that we

have seen so far. And if the length of this word is larger or long if this word is longer than the previous longest word then we replace the previous longest word with this word and we save its length also. So, we save this word and its length as the longest seen so far. And similarly we maintain the largest word seen so far. And if this word that we just saw is larger than the previous largest then we save this word as the largest.

(Refer Slide Time: 20:26)



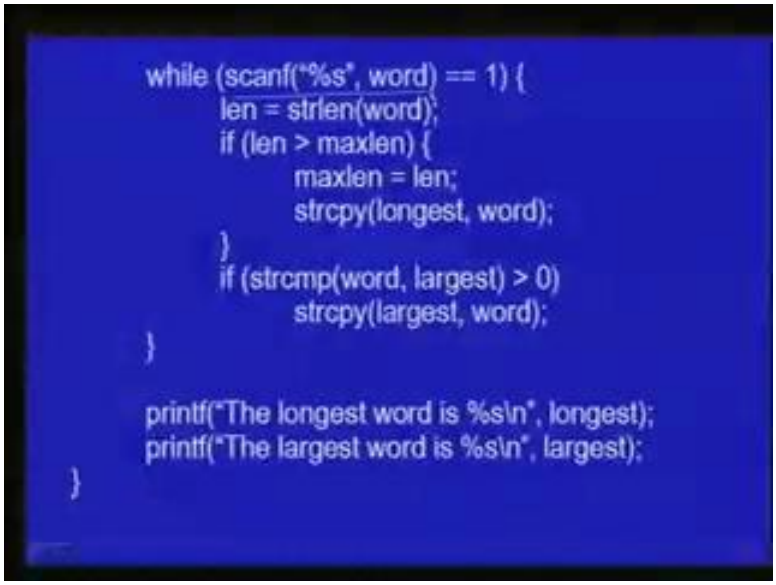
So, let us now put this algorithm into practice by writing a C program for this. So, here is the initial part of the program as you can see we have included the file string dot h as well, because we are going to use some functions from the string library. We have defined a constant max size which is equal to twenty which we have expect to be the longest word size. Actually the longest word size we expect is to be nineteen because one character as you know is needed extra for storing the null character. These are the variable declaration this variable length we are going to use to store the length of the word that we have most recently seen the variable max length is the length of the longest word that we have seen so far and this is initialized to zero.

So, essentially what we what we are going to do is that we are going to compute the length of the word seen just now and compare it with max length, and if length happens to be greater than max length then max length will become length and longest word will become the one that we have seen so far. And there are three string variables that we are declaring, these are all of the size max size. The first variable word will be used to store

the word that we have most recently read in the loop. The string longest will be used to store the longest word that we have seen so far and largest will be used to store the largest word that we have seen so far.

Recall how we are going to maintain the largest word we have seen so far, we are going to compare every word with the previous largest word, and if the current word is larger than the previous largest then the largest will become the current word. So, therefore, we need to initialize this to some word which is smaller than the any other word. So, that the very first word that we see becomes the largest at the end of the first iteration. Now, the smallest possible string that you can think of is the null string, which has no characters inside it. So, it just contains the two double quotes without anything between them.

(Refer Slide Time: 22:44)



Here is the rest of the program as you can see it surely straight forward, all the where using some new technique in this program that I will explain. So, we are using scanf percent s to read a word as you have already seen it reads the word from the input. Now the question is that this loop must terminate when the end of input has been reached. Recall that the end of the input is typically indicated by the user to the program by typing a control d, but how would our program know that there are no words no more words to be read from the input.

If we are using a function like getchar that would have returned EOF on seeing the end of the input, but with scanf we are going to do something different. In fact, so far

whenever we have been using scanf, we are assuming that scanf only has side effects and scanf expression has no value. In another words, the scanf function does not return any value that is not actually true. Scanf function actually does return a value, which we have been discarding so far in our program and the value returned by scanf is nothing, but the number of items successfully assigned to some variable. So, of the numbers of values successfully assigned to some variable.

So, if you look at this particular use of scanf what we expect to do is to assign some value to the string variable word. And therefore, if scanf succeed, they expect that it should return the value one. Now if it does not return the value one that could only mean in this case that the input really end it. Scanf actually returns the EOF when the end of the input is reached. So, we could have terminated this loop also by checking that the return value of scanf is not EOF which is actually minus one. Note that when you are using scanf with some other format specifiers, for example, suppose you do scanf percent d ampersand x then scanf code returns a value other than one. In fact, it could return zero also which is neither one nor EOF and that would have happened in the case when the input characters that were read actually did not form an integer that is they did not the input characters did not comprise of only digits and the plus or minus sign.

So, in that case also it would return something less than the number of items that we expect to be assigned and that in fact, is a very useful feature which we haven't been using really in our program so far. Whenever we so far in our program read an integer using scanf we assume that some integer was always read, but that is not actually true; suppose instead of typing some integer the user types some arbitrary characters like xyz what happens in that case what is the integer value that is assigned to the variable. In fact, no variable is assigned by scanf in that case and the scanf just indicates that the input was not appropriate for the format conversion specifier by returning a value by returning a value, which could be less than the number of extra arguments for scanf.

And that is something that we should actually make habit of checking that when we read an input integer integer from the input we actually successfully do that read it. So, in the programs that we will write from now on we will make a habit to put that in to practice. Anyway coming back to this program, if you have read a word successfully then we use a string length function to find its length and store that in the integer variable length. And

if length is more than max length then as I said max length is assigned to length so that the next word whenever it is read its length can be compared to the new max length.

And the longest word is now the word that you just read and. So, string copy is used to assign or to copy the current word into the longest word. So, that is takes you the longest word and we also have to compare the current word with the largest word so far. So we used the string compare function to check that. So, if strcmp word comma largest is greater than zero gives the value which is greater than zero; that means, that word is greater than largest, and in that case we copy the current word into the largest word. And at the end of this loop, we simply print the longest word and the largest word using the familiar printf function. Note that we are using percent s to print these two string variables.

So, that is the end of this lecture; in the next lecture, we will start looking at a very powerful facility in C, and in fact, most of the programming languages and that is the ability to define our own function. When program gets considerably more complex than the simple ones that we have writ being writing so far then we need certain amount of discipline in how we design the program, and how we implement the program. And we will see that it then becomes useful to break down the problems into smaller sub problems and so on. And then define individual program units for each of those smaller sub problems these smaller programs units are called functions, and in fact, we have been using library functions and see how we can define our own function in the next lecture.