Introduction to Problem Solving and Programming

Prof. Deepak Gupta

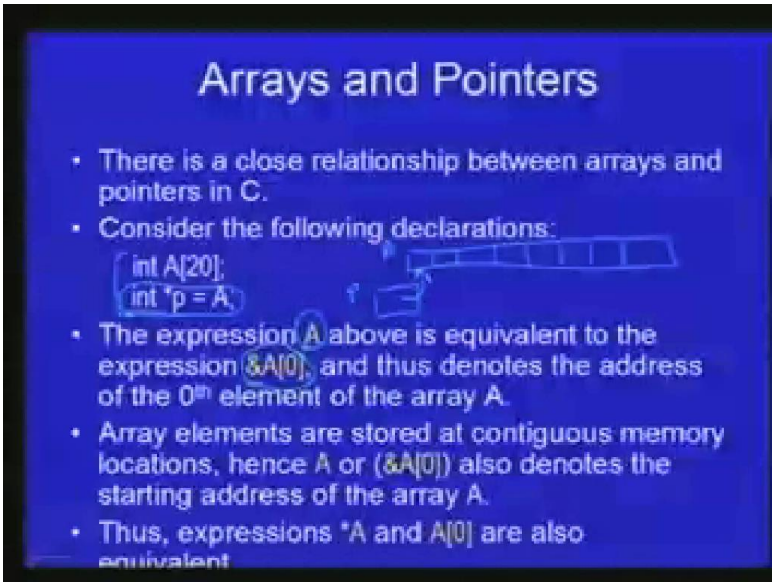
Department of Computer Science Engineering

Indian Institute of Technology, Kanpur

Lecture No. # 18

In the last lecture, we are talked about the notion of pointer and we saw that essentially a pointer is an address of a variable. And we saw the two operator which are related to pointer that is ampersand operator which given the operand as a variable is gives the address of that variable. And star operator are the pointer dereferencing operator which given the pointer value as an operand gives the contents of the variable or the location to which the pointer points. So today, we will continue our discussion on pointer by exploring the relationship between arrays and pointers

(Refer Slide Time: 00:59)



In C there is a very close relationship between arrays and pointes and as you will see in some cases arrays and pointers can be used interchangeably. So, to begin with consider the following two declarations int A 20, so we are declaring an array called A of 20 integer and then we are declaring a pointer variable p which is the type in star and the initial value of p is nothing but A. Now in C the expression A itself, where A is the name

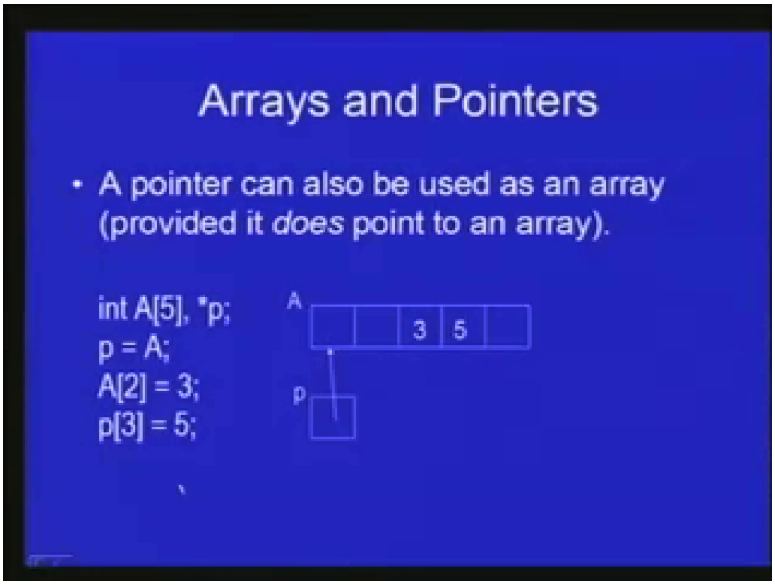
of the array is nothing but the address of the zero eth element of the array and therefore, the expression A and the expression ampersand A 0 are completely equivalent, which means that in this initialization, p is the initialized to point to the zero eth element of the array.

So, in terms of pictures that we saw last time which help us understand pointers better. So, p is pointing to the zero eth element of the array A. So, essentially what are seeing at the expression A itself where A is the name of the array is the same or is equivalent to the address of the zero eth element of the array. Now recall that is not see we said earlier that the difference element of an array are stored at contiguous or adjacent memory location. So, for example, if the first element or the zero eth element of the array is at the memory location thousand and the each elements required four bytes then the first element could be at memory location 1004 and the second element could be at 1008 and so on and so forth. So, essentially viewed in that sense the expression A gives the starting address where the array is stored in the memory.

And similarly you can see that since A and ampersand A 0 are equivalent, so are star A and A 0. This is because of star of ampersand x is always equivalent to right because we are first taking the address of variable x and then you have been dereference in that address. So, this always true and so therefore, star of ampersand A 0 must be same as A 0 and which is nothing but A. So, therefore, what we get is at star A is same as A 0. Note that what does means is that both stars can be applied on the left hand side of an expression of in the assignment expression which will mean that the zero eth element of the array A is being kind some values.

The expression A itself cannot appear on the left hand side of an assignment because that could been that the starting address of the array A is being changed which is not possible of course, because as you saw last time the address of the variable are fixed and they are assigned by the complier we cannot change them, but when A at define an address is stored in the another variable then the contents of that variable can be changed

(Refer Slide Time: 04:31)

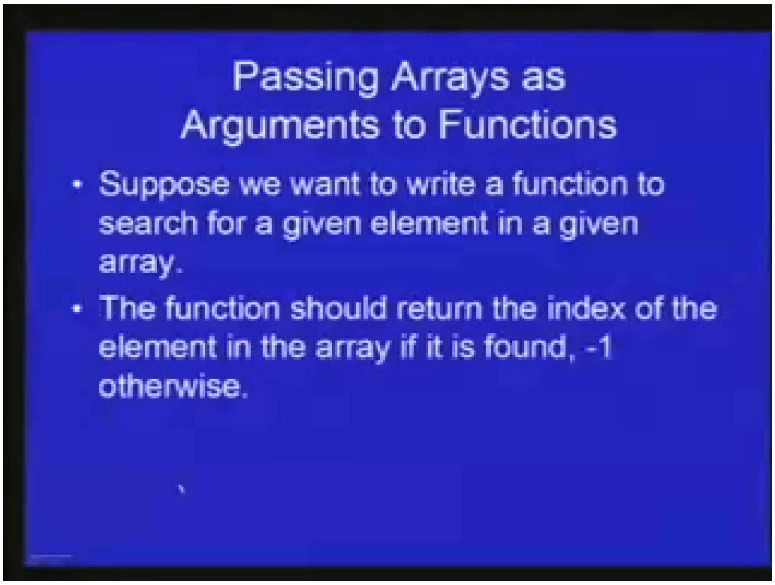


Now the next important thing to understand, if that even a pointer can be used using the array notation that is dereferencing using the pointer can be done, even using the array notation - provided that the pointer does points to an array. So, let us take a simple example. So, we have two declarations here again A is n array of five elements. So, that is how it is in picture and p is the pointer of type in star which means that this variable p will hold, the hold the pointer to some integer from the first assignment that is here is the assigned A. So, what happens to in this assignment executes the address stored in that variable p is nothing, but the.

Address of the zero eth element of the array now when A 2 is assigned 3; you know that familiar to us what should happen is at the second element of the array A will get the value 3, but now what is the interesting is that you could move to the variable p as if it is an array and that is because it is pointing to an array element. So, when p 3 is done that could be the same as a matter of A 3 assigned 5 because p and A are phenomenon (( )) at that value of A of the value of the variable p is nothing but, the starting address of the array A.

So, when p 3 is assigned 5 is executed; the third element of the array A is a find the value 5. This is become hopefully more clear using more example that you see as selected progressive.

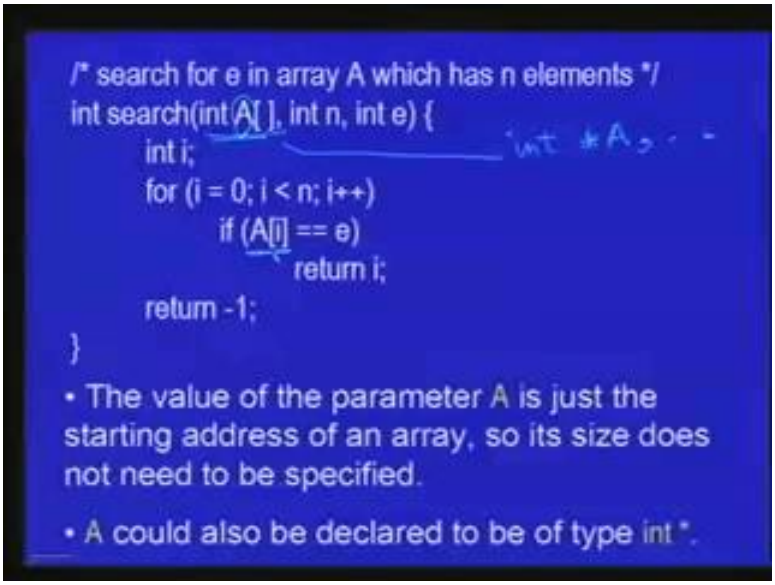
(Refer Slide Time: 06:22)



Now you can also pass array as an argument to functions and interest you have been doing that in the case of some library function for example, we saw the string function in the C, standard library for example, function is to find the length of the string and so on. So, there when we passed a string as an argument as we knew string is nothing but an array of character is the special property that it is terminated by another character. So, what we are passing to the string length function is really an array. So, what you want to next explore is what is happen when arrays are passed to a function, and how do we define such function and how do we call such function and so on.

So, let us take a very simple example suppose you want to write a function to search for a given element in a given array. So, that is why we given an array of integer and we are given the number of elements in number of elements in that integer and we are given the particular value and we would not to search in the array for that value. And if the array contains that value then we should return the function suppose to written the index at which in the array that value used further, and if that value is not present anywhere in the array than the function should return minus 1.

(Refer Slide Time: 07:39)



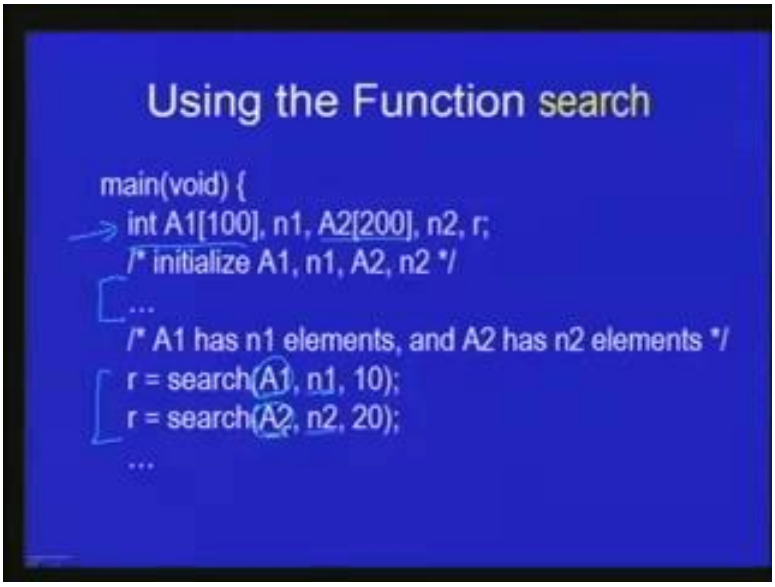
So, here is what the function would look like most that n and e are integer parameter as usual. So, there is nothing special about them; n is the actual number of element in the array which is the past and e is the value that we need to search for. Now this is the array itself which is been passed, so what should happen in C is that when an array is passed as an arguments. What is actually passed to the function is nothing but the starting address of the array. So, the argument A here actually denotes nothing but the starting address of the array which is been passed as an arguments.

So, the function itself is quite simple. So, we have a for loop and we examine all elements of the array if from 0 to n minus 1 and if for a particular value of i A i happens to happens to be equal e then e immediately return i because at the array index i the array A contains the value e note that the return statement would not just break out of the loop, but would immediately call the function to return. So, the next statement would not executed in that case. So that means, that if the array element that the value e was found somewhere in the array then this loop will not finishes as a matter of if and the return statement will call the function to return before the loop finishes, and so, if the loop does finish that implies that the array did not contain the value e at any place or at any index, and therefore, we should return the value minus 1.

Two important points to note this is at the value of the parameter A in this function is nothing but the starting address of the array which is being passed as an argument. The copy of that array is not been passed and that also means that when we declare this array you do not need to specify the size of the array, the reason is that when the memory is

allocated for this parameter A, the amount of memory required is the amount of space needed to store just one pointer value or just one address, and so, therefore, that is not depend on how many elements in the array that is being passed as an arguments have or there. And because of this reason, in fact, we could have return this function slightly differently instead of declaring A in this fashion e could have declared A in this fashioned (( )) and even then the rest of the code of the function would not made to change, because as we just saw even a pointer value can be used as an array right.

(Refer Slide Time: 10:49)

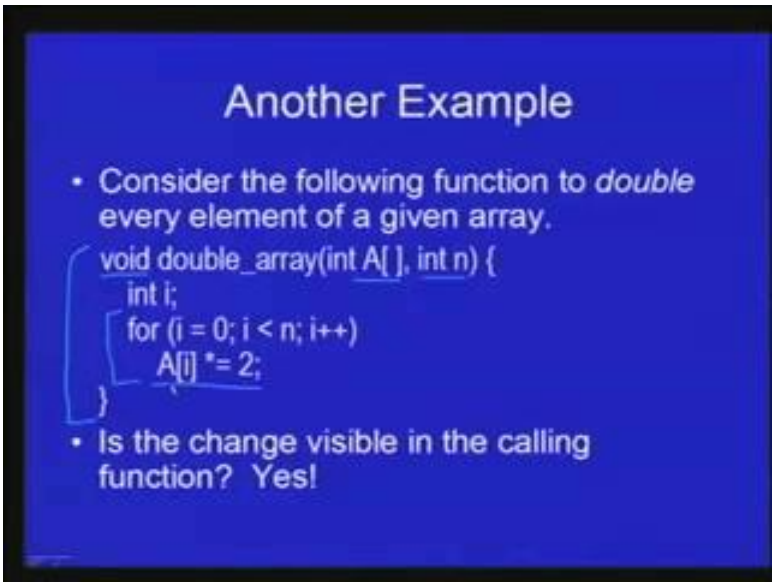


So, let see how this function could be called from some other function, let say the main function do that is not a important could be call from any function really. So, in this examples we have of two array A1 and A2 of size 100 and 200 respectively and from code initializes the value of the element of the array even in need to and also the values of the variable n1 and n2. So, let us assume that A1 has n1 elements and A2 has n2 elements and we could use this same function search to search for some value in both of this array. The fact that the sizes of these two arrays is different does not really matter, because the actual size of the array or the actual number of elements in the array is been passed as an arguments and when the array itself is passed - that what is been passed really is only the starting address and that should be clear also from the rule about that we talk about earlier about how arguments are passed to a function.

So, the function argument A1 would be evaluated in that case and we just saw that an expression denoting just an array name evaluate nothing but the starting address of the array. So, the expression A1 evaluates just the starting address of the array A1 and that is passed as the argument to the function search, and similarly in this case the starting address of the array A2 is passed as an argument.

Let us consider now another example which modifies the array in which the function which is given an array as a parameter actually modifies some of the element in that array. So, let us consider a very simple example again in which we are writing a function which is intended to double the value of every element of an array.

(Refer Slide Time: 12:51)

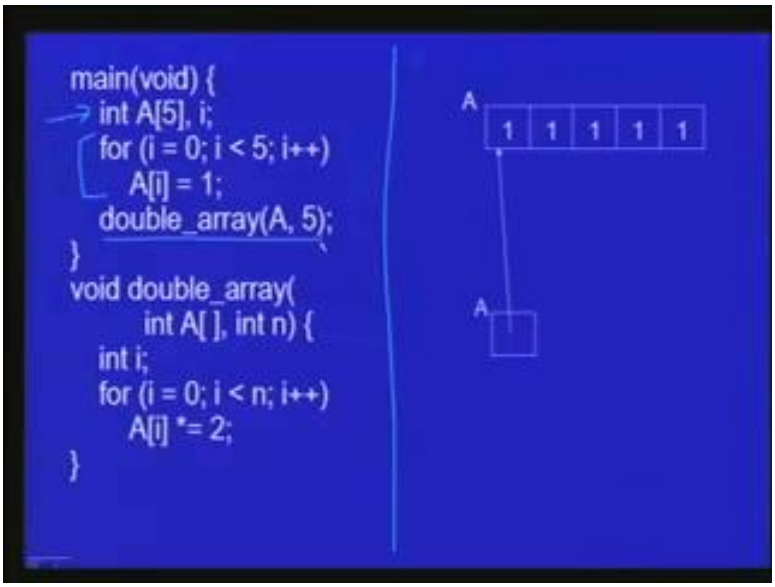


So, here is what the function might look like and double array is the name of the function there is no return value and so the return type is void. The argument is an integer that is define how many elements are actually there are in the array A, which is the passed as the first arguments and the code is very simple. For all elements from 0 to minus 1 is the element of the array, we just double the array element using the star is equal to operator, but the question is whether the change is visible in the calling function that the values of the array that we have changed here in this function in this various elements of the array is at change visible in the calling function.

We call that when we passed, when we modified lets the n integer parameter within the function then if the actual argument happen to be variable in the calling function then the

modification to the parameter does not calls a modification in the variable which was passed as an argument to it because as we know copy of that variable of the value of that variable is made available to the parameter, but in the case of arrays no recall that if it is starting address of the array which is being passed as an arguments and not copy of the values of the element of the array. So, therefore, when this change is made to an array element what is actually happening is at the array element of the original array which was passed as an arguments that is being access by using its address and therefore, the change could be visible even when the function return to the calling function.

(Refer Slide Time: 14:44)



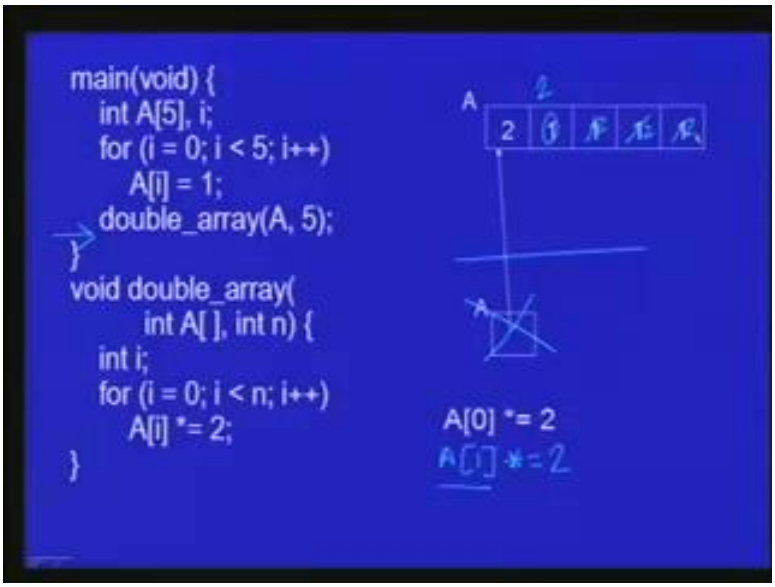
So, let us try to clarify the situation with the help of this example, where I have put the same function in the context of the calling function. So, let us assume that the function double array is being called from the function mean which has an array A of 5 elements and an integer variable i and this loop initializes all array elements of A from index zero to 4 to 4 and then it calls double array with A and 5 with the number of elements in the array has the arguments.

Now, when the function call happens to this variable A is the same variable A of the main function and which represent this array of five element each of which has the value one and when this is pass as arguments to the function in double array. So, there the parameter their also calls A, but what this parameter will contain will not be this, this

parameter a will not refer to an array itself, but a pointers to the zero eth element of the original array which was passed as arguments.

Now, when this function executes let say the when this statements is executed for the value of i equal to 0, which means that A 0 star is equal to 2 is executed. Now what as A 0 really mean? A 0 mean that the zero eth element of the array to which the pointer A is pointing note that in the context of the function. This A is not really an array, but it is a pointer. So, A 0 would refer to the zero eth element of the array to which the variable A is pointing and show.

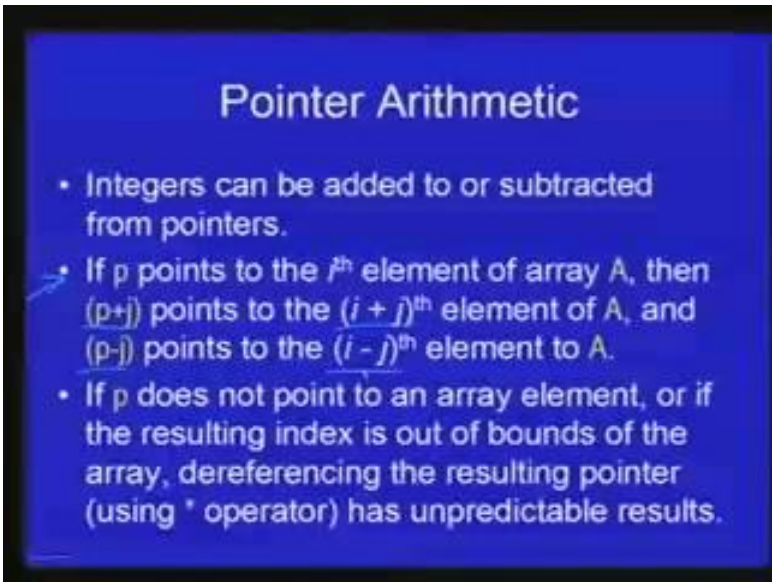
(Refer Slide Time: 16:33)



When this assignment is executed the zero eth element of the original array A is modifies to 2, and similarly in the next iteration of the loop this statement will be executed and A 1 will refer to the first element of the array to which the variable A is pointing which is different, and so, this will become 2. And so, finally, when the function double array return, this variable A will be destroyed and the control will go back to main, but in the main function note that the value of the array elements in a has actually changed. So, essentially what is happening here that when an array is passed as a parameter to a function is the starting address of the array that that gets passed, and in the call function it from the array element it is modified then because this value is modified using the address of the memory location written in this value, the change is visible, even when the function return to the calling function.

Now this is different from when a normal integer variable is passed, because when a simple integer variable is passed the parameter gets it copied of the value of the argument and if the parameter is changed will be in the body of the function and that change is not visible in the calling function.

(Refer Slide Time: 17:59)



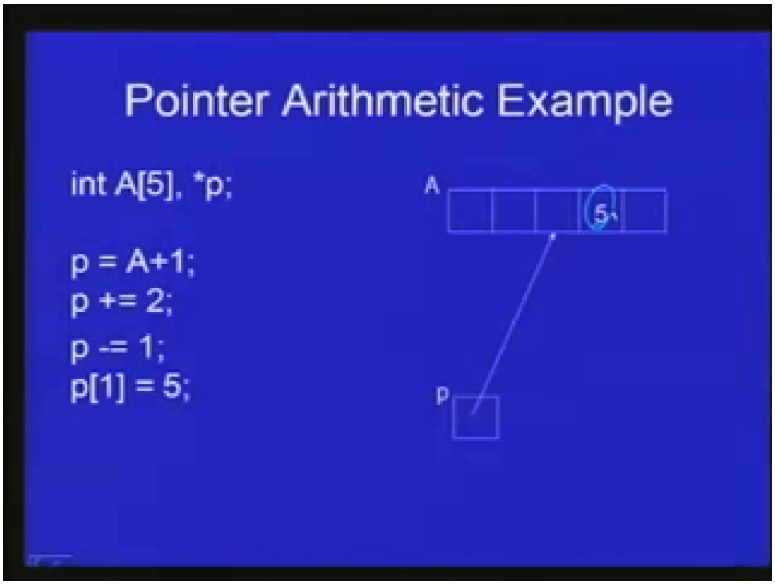
Let us now come to another relative notion and that notion of pointer arithmetic. You might be surprised to move in that in C, you can actually do some arithmetic with pointer. The arithmetic will allowed with pointer is very limited, you can add or subtract integers to or from pointer that is you can add an integer to a pointer of or you can subtract an integer from pointer. Now this makes sense only when the pointer actually points to some array element of some array. So, for example, if p points to the highest element of from array A then p plus j where j is an integer points to the i plus j eth element of the same array A, and similarly p minus j points to the i minus j eth element of the array A.

Note that it is possible that is not pointing to an array at all or maybe then p plus j is an array actually does not have i plus j element or may be p minus j itself, the value of i minus j eth happens to be negative which means again the array does not have an element at that particular location. So, the computation of the pointer resulting from the pointer arithmetic will work, but when you try to dereference such a pointer which as a

resulted from this arithmetic that will call an unexpected result and that will be similar to an array element being accessed with an index which is out of bound.

So, for an example, if an array has n elements and if you try to access the n eth or the n plus 1 eth element or the minus 1 eth element in that say that is an error end result in an unpredictable behavior. And similarly if you try to obtain a pointer to the n eth or n plus 1 eth element of the array or the minus first element of the array you can an obtain a pointer, but when you try to dereference using the star operator then the result will be completely unpredictable, and therefore, it is our responsibility to make sure that this is never happens that whenever we do the pointer arithmetic we always make sure that our pointer is pointing to a valid element of the array.

(Refer Slide Time: 20:23)



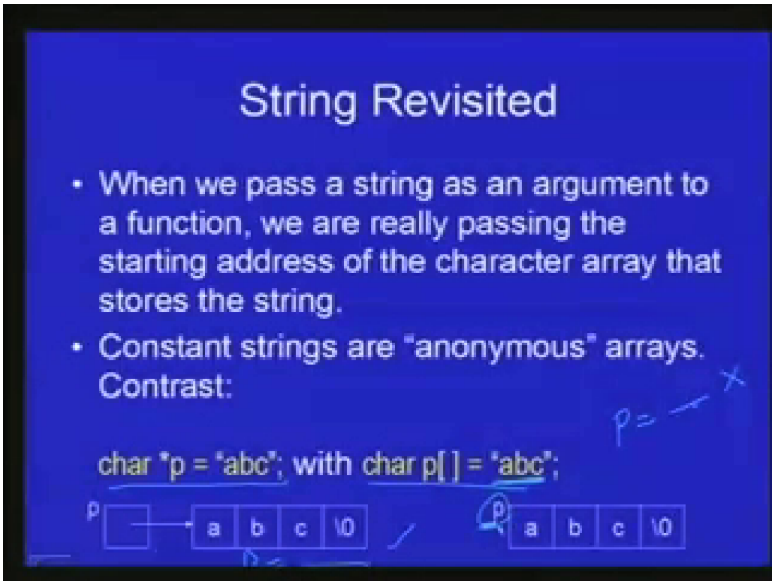
So, let see an example of pointer arithmetic now. So, in this example you have two again a variable. A is an array of 5 element and p is of type int star which means p will contain a pointer to an integer. Now the first assignment that happen is p assigned A plus 1. Now what do think will happen note that the expression A denotes the starting address of the array A or in other words, the address of the zero eth element of A. And now if you add one to it; one is an integer. So, add one to the address of the zero eth element of A, what you get is the address of the first element of A. So, once this assignment executes p will points to the first element of the array A.

Now in the next step, we have the statement p plus equal to 2 which of course, as you know is same as p assigned p plus 2. So, in this case again what we are doing is identify two to a pointer value and assigning the result back to the pointer variable p. Now p is the currently pointing to the first element of the array A. So, p plus 2 will point to the third element of the array A and that pointer value is assigned back into the variable p, and so, after execution of this assignment p will point to the third element of the array A; and in the next step we subtracting something from the pointer, we are subtracting one from the pointer which means that the pointer will move one step back in to the array. So, we will now start pointing to the second element of the array A.

Now we doing something interesting we are p as an array and we are dereferencing from indeed from some pointer using the array notation, but note that p is not actually pointing to the zero eth element of the array. It is pointing to the some element of the array, but that does not really matter when we say p 1 is assigned 5 (( )) to happen is think of an array which starts here. So, think of p section of the array. So, when you are using p as a an array where p is the pointing to and from to the i eth element of the array, then p 1 is the same as p i plus 1 is p is same as sorry is p i plus 1 if p is the same as which is of course, (( )) A plus i.

If p points to the i eth element then p 1 would refer to the i plus 1 eth element of that array. So, therefore, now p 1 is assigned 5 what will happen is that the first element of the array segment will become 1, and so, p 1 would be this particular cell which has you can see which is the third element of the array A and that follows from this kind of rule that we get so, if p is pointing to the second element of the array then the p 1 is same as the 2 plus 1 that is the third element of the array. So, p 1 is assigned 5 and the result in this element of the array A getting the value 5.

(Refer Slide Time: 24:18)



Let us now again talk a little bit about strings and see how strings relate to array, we already know that strings are actually arrays of character with the special property that the useful data in the string is terminated by a null character. And as a matter of that strings are very commonly used as pointers, because as you already seen errors can be used as a pointers so. Whenever in a functions like for an example print f we passed its format string as an arguments or in a function like str ln, we passes a string as an argument in returns the length of the string then we passes a string as an argument to a function what we are really passing the starting address of the character array that stores the string, and this follows just from the fact that this string really nothing but an array. And then we pass an array as an argument to a function, what is really pass is the starting address of the array.

Now strings are also different from other kinds of array instances that it is possible to have constant string, for example, the constant string abc here. Now what kind of an array is that, now what happens to the constant string is that the constant strings are treated by the complier as anonymous array, anonymous means without a name. So, when you use a constant string in the program what happens is that the complier creates some space, somewhere in the memory which contains that string and the value of the constant string expression is nothing but the starting address of that array.

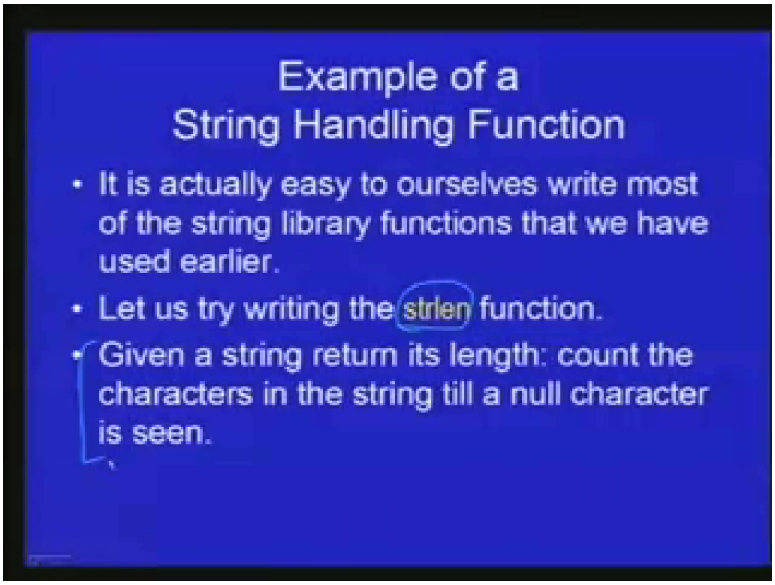
And we should be careful here that different declaration different ways of declaration of a string in actually in a different string. So, now, that we know that the pointer and array can be used interchangeably, we could have this two times of the declaration for a string

variable, but the behavior is going to be slightly different in these two cases. So, in the first case we are declaring a pointer variable p, which is initialized to the starting address of the constant string abc. In the second case, we are declaring an array of character called p which is been initialized to the string abc. So, in the first case which means progress in terms of pictures what is the look likes the calling; that abc is an anonymous array is created in the memory somewhere from the compiler which contain the string abc.

The character abc follow by of course, a null character and this variable p a box is created for the variable p or a memory location is assigned for a variable p and that is initialized to point to the starting address of the anonymous array. Note that in case we can change the value of the variable p, because p is the normal variable of type char star and we can reassign to point to some other string. Now if you contrast it with the other kind of declaration what is happening in this case is at we are asking the compiler to create an array called p if no longer and anonymous arrays which is initialized which we will have four elements in it because size of the initializing string is four including the null character. And so, p is the main element of the array whose elements of the character a b c and the null character note that now p is the name of the array and therefore, p cannot appear on the left side of the assignment.

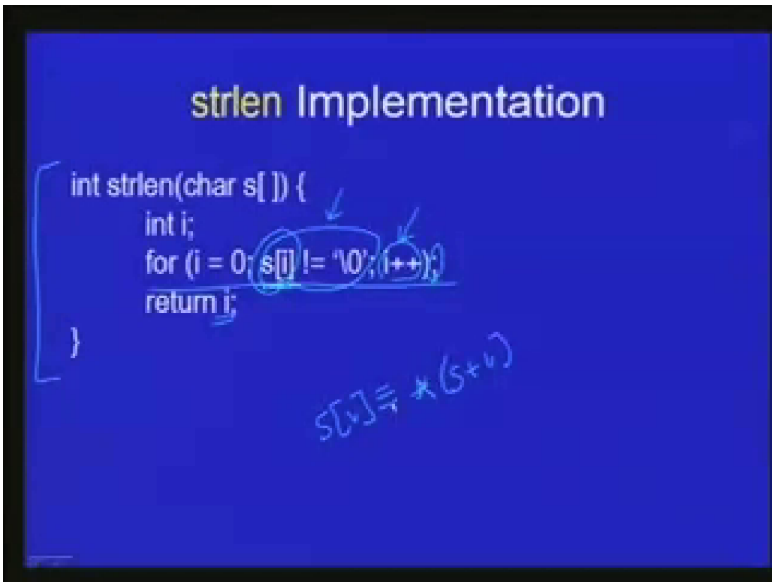
So, p assign something would be wrong here in, which declaration where as p assigned some other things would be correct in this kind of the declaration because here p is associated with the points p is the name of the pointer variable, whereas here p is the name of an array itself now let us (( )) some of the string handling function that we talk about some time there. And we will see that t is very easy to write the string and in the function that we use from the library. So, as an example let try writing the string length.

(Refer Slide Time: 28:50)



Function our self and it will look straight forward isn't to actually right function that is manipulated string and so on. As we know the functionality of the string length function is simple given a string you have to written it plan and we do that always do it, we have to count the character from the string till we find the null character and this count is the length of the string note that we length that we have to return should not include the null character itself.

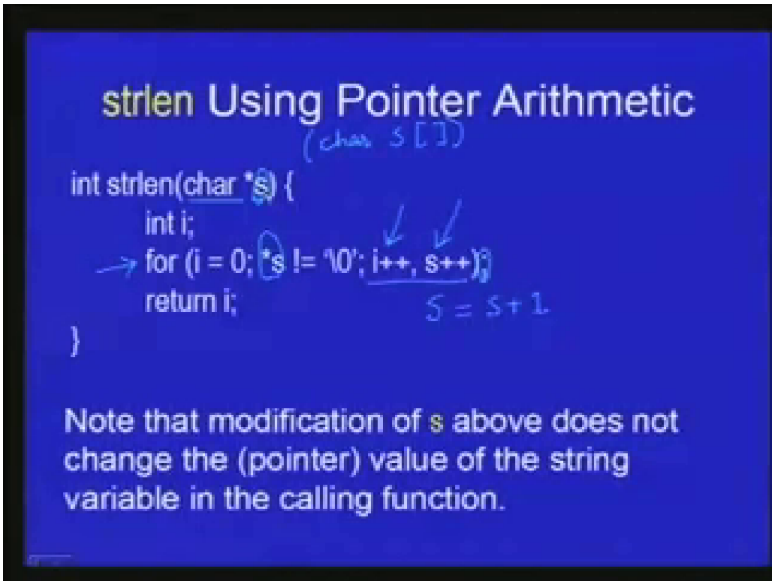
(Refer Slide Time: 29:24)



So, here is the first version of our string length implementation. So, we are actually using the pointer which is past to the function as an array using the array notation. So, this loop initializes i to 0, and then as long as s i is not equal to null character which increment the

i note that the loop body is empty. So, all that is happening is in this i plus plus and in the loop condition. So, essentially this loop will terminate as soon as we find s i is equal to the null character and then i will be nothing but the length of the array. Because note that we are implementing i one for every non null character that we find in that string. Now we could have written the same string handling function the same string length function using pointer arithmetic as well that is instead using s i to s is s element of the string higher character in that string we could have use something like star of s plus i which would be same as the s of i. So, we could have done that and even more interestingly we could have modified the value of the variable s itself in the loop.

(Refer Slide Time: 30:44)



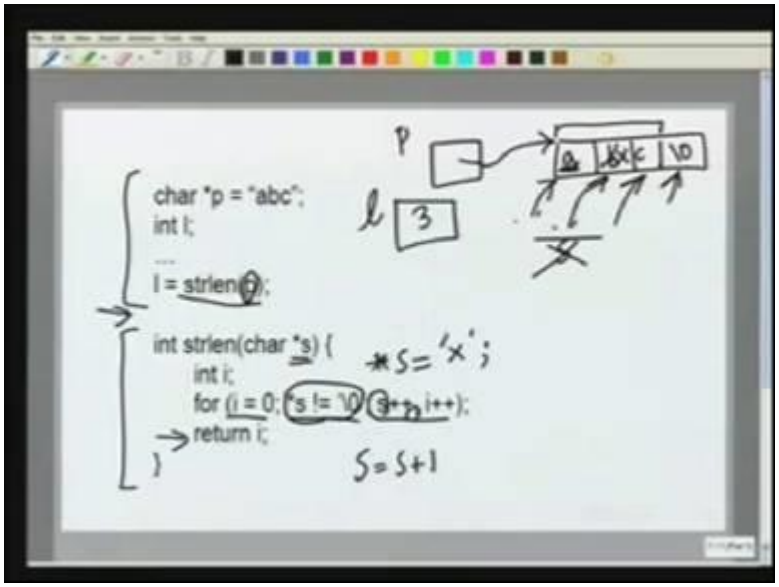
So, lets you get this second implementation of the same function. So, this time I have chosen to declare s as a char star which is supposed to be same as the declare in this an array within the function parameter, but I am going to use the s as the pointer does not use the array notation. So, in this version of the function note is being done use it again I to be zero that will be count and we are checking, whether s is pointing to the null character.

Now star s is nothing but the value of the character to which as point and as long as that is not null we keep running the loop. The loop body is again empty, but after every iteration it will be low we increment i by one of course, to account or found for the non null character that we are just seen and we also instrument the value of s note that this is

again s assigned s plus 1. Now this is not normal arithmetic this is pointer arithmetic because p is a pointer. So, what we are doing here interestingly is that we are modifying the pointer arguments that has been passed to us, but again note that this is the copy of the address that we have been given we are not actually modifying and element of the string whose address is given to us, but the address of the string itself and the address of the string that we have been given is actually copy of the original address from the calling function. So, it change to us will not get reflected in the calling program.

Now this may not be obviously, but once we trace this function with the calling function and with the help of the notation that we have already seen using the box and the row to represent the variable and pointer this will become very clear. So, let us do and do that.

(Refer Slide Time: 32:51)



So, here is the same function again the strings length function that we had the string and let say this is the calling function. So, we use calling function you have a pointer t take this point p to the constant string abc. So, let us draw or familiar boxes and arrow. So, p is the pointing to the zero element of the anonymous array containing the character a, b, c followed by the null character and of course l is the normal integer variable. And when we are calling length with the argument p and the result is (( )) to l, so this two are the variables of the calling function now when this function gets called space is created for the argument for the parameter the parameter is s which is just A.

Pointer and this will be initialized with the value of the argument which is passed. So, the value of the argument is the value of the variable p which we pointer to the zero eth element of the string. So, the value of s will also be big now when this (( )) i is the initialize to zero i is the local variable of the string length function. So, i gets initialized here and s is the already pointing to the zero eth element of the string now the check is made whether the star s is equal to null correct or.

Not star s is nothing, but the character a which is not the null character. So, the. So, the loop body executes the loop body is empty at the end of the loop body this this expression is evaluated note that it is using the comma operator which means both the expression will get evaluated. So, the first expression could be evaluated is s plus plus which will mean which is of course, we saw is same as the s assigned s plus one which will mean that s will not start pointing to the first.

Element of the character array and i will come 1, now the next time the loop condition is attested or s is not be still not be null character. So, s gets incremented again s starts pointing here and i is incremented i becomes 2 and then w are go back to the loop test star s is still not equal to the null character. So, the value of s is change to point in the next element of the character array and the value of i becomes three and then the loop test is executed again this time star s is the.

Null character and which is. So, this test fail to the loop terminates and when the loop terminates the it come to the return state which returns the value of i which is three this is the correct length of the string of course, the value three is written, but now the (( )) of string length function will get deallocated and the control will go back to the calling function. So, one will obtain the return value which was the of course, three and the important point to note is that the modification of that has not resulted in the modification of the that of the variable of the value of the variable p the variable p is still pointing to the zero eth element of the original array.

Now on the other hand, if you had modify an element of the array itself. So, for example, if we had executed a statement like this let us say x and let us say x that pointing time s happen to be pointing time to the first element of the string then this would have been modified and even when the function written and S for deallocated this string this change in the string would still get reflected, but here what is been modified if just the address

which is been passed to us and since we have got an copy of the address this local copy within the sting length function is being modified and the original coy of this address does not get modified. So, that is the end of the lecture today and before we talk I would likely to utilize the technique that we have discussed today to try and implement some of. The other string library function that we have already used in terms of programs, for example, you could implement the string cat strcat function is to concatenate two string or you could implement this string comparison function strcmp which compares two string in the (( )) order.