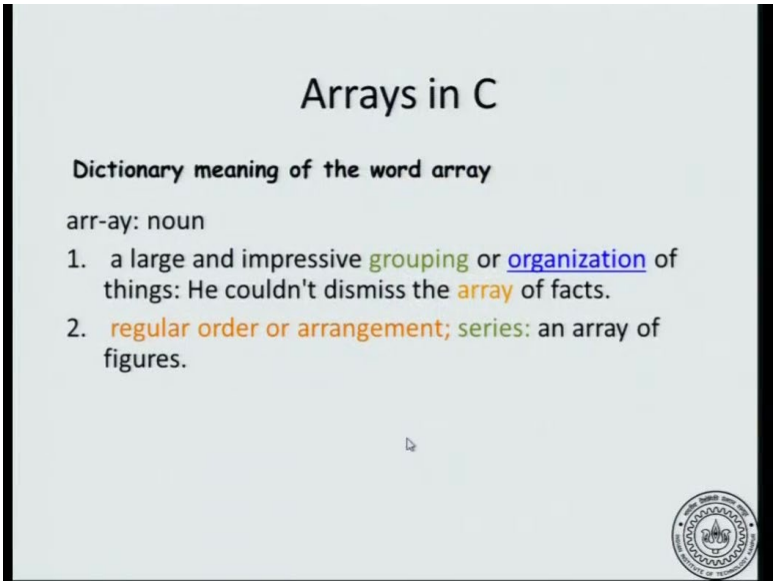
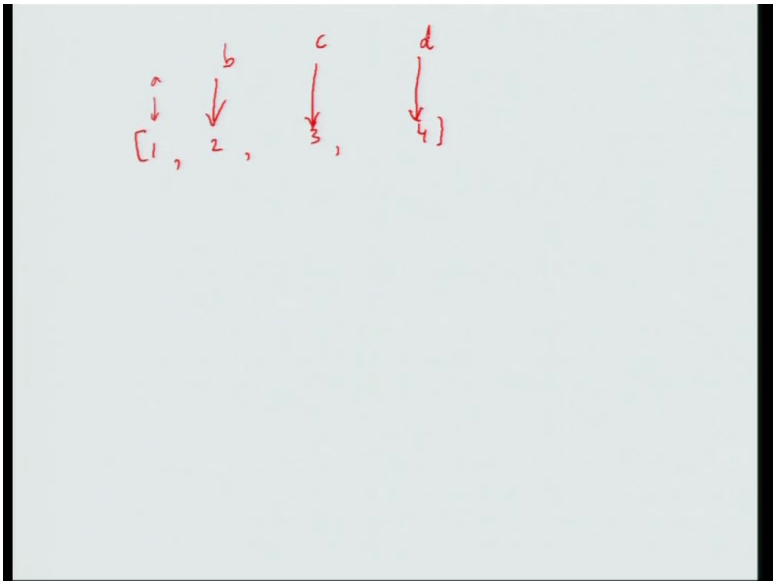
Introduction to Programming in C
Department of Computer Science and Engineering

(Refer Slide Time: 00:08)



This session will learn about arrays in C. Now, what is the word array mean, it means a grouping or a collection of objects. So, for example, you could say that he could not dismiss the array of facts. So, that means, a collection of facts and it also implies a regular order or arrangement that is in the case of a series. So, what do we mean by an array? And why do we need it?

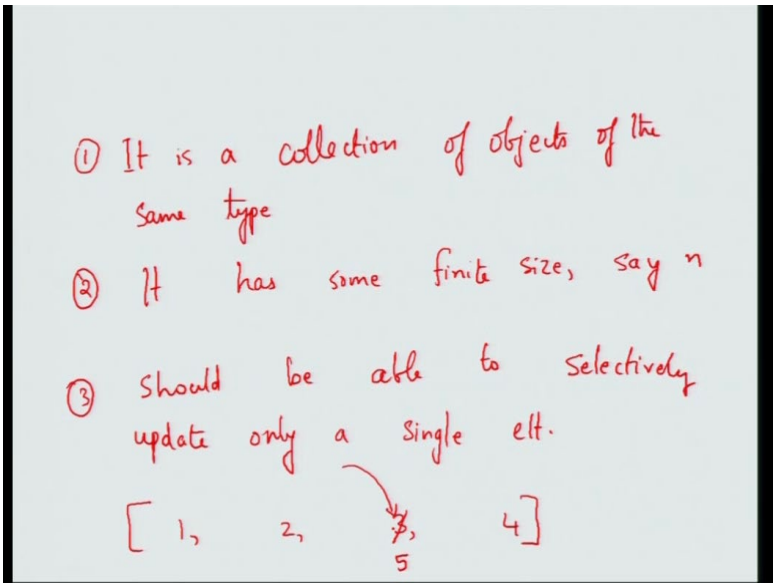
(Refer Slide Time: 00:45)



So, let us consider that I have a bunch of numbers say 1,2,3,4 and I want to consider

them as being part of the sequence. Let us say 1 , 2 , 3 , 4. So, the first element is 1 and so on. Now, I want them to be stored and one way I can do it is that, I can store them in to separate variables. For example, I can say that a is 1, b is 2 and c is 3 and d is 4. But, when I do it in that way, they are separate variables and the relationship between those variables, the fact that b comes after a and things like that is something that the programmer knows, but it is hard for somebody else looking at the code to figure out. Often we need to store sequence as a sequence itself.

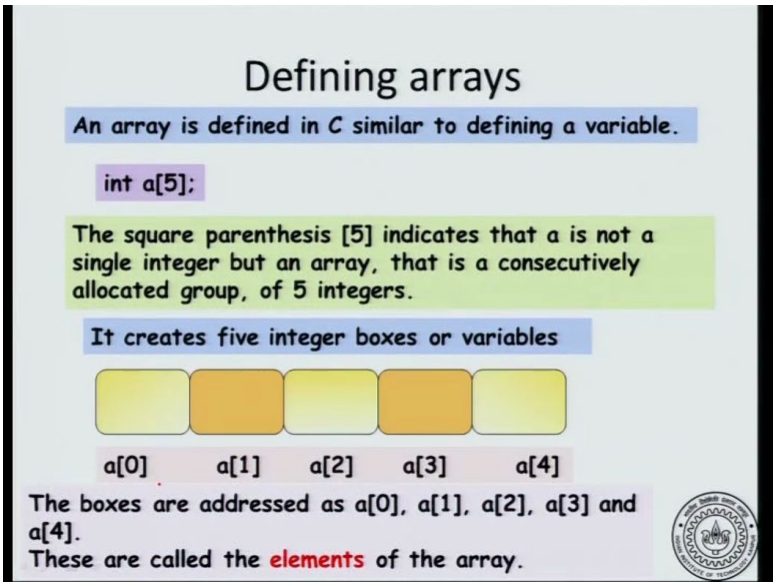
(Refer Slide Time: 01:47)



So, an array has the following properties, one it is a collection of objects of the same type. Second, is that it has some size, some finite size say n there are n elements in the array and the third is that, I should be able to selectively update only a single element in the array. By which I mean, suppose I have four elements in the array I should be able to say that, I want to replace the third element by 5. So, 3 will be replaced by 5 without touching the other elements, that is what the third thing is about

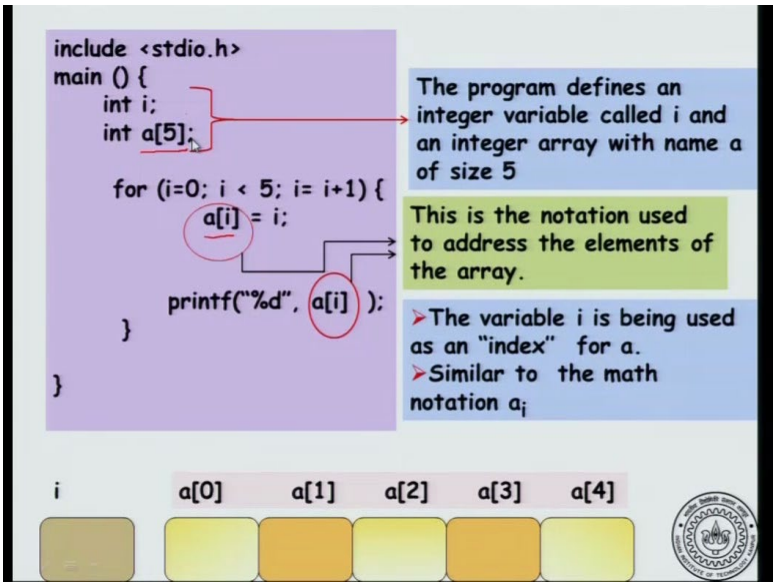
The first thing says that I want a collection such that it is a collection of objects of all of type integer and not of any other type. And the second says that, it has a some finite size so, it is not an infinite collection, so, that is what an array is supposed to do. Now, let us see how we can define arrays, will try to motivate why arrays are needed by introducing certain problems. And I will try to convince you that it is easy to do using arrays and whereas, it was difficult to do without using arrays, using only the facilities in c that we have seen so, far.

(Refer Slide Time: 03:58)



So, an array is defined in c, similar to how we define a variable. If we had an integer variable we would say int a semicolon instead of that when we declare an array we have int a 5. So, this would declare that it is an array containing 5 integers. Now, one thing that is certain about arrays in c is that, the five integers which makeup the array will be allocated consecutively in memories so, they will happen one after the other. Also one think to note is that, arrays in c start with index 0 so, the first element is a 0. So, if we have an array of five elements it will go from a 0 to a 4. So, have we seen arrays and mathematics for example, you can think of vectors similarly matrices these are all arrays and c arrays will have similarities to mathematical vectors and mathematical arrays. But, note that in mathematics, it is customary to start from index 1, here it is from index 0. So, the boxes are addressed as a 0 to a 4 these are called the elements of array. The array is the whole collection of boxes and each box in it will be called an element of array.

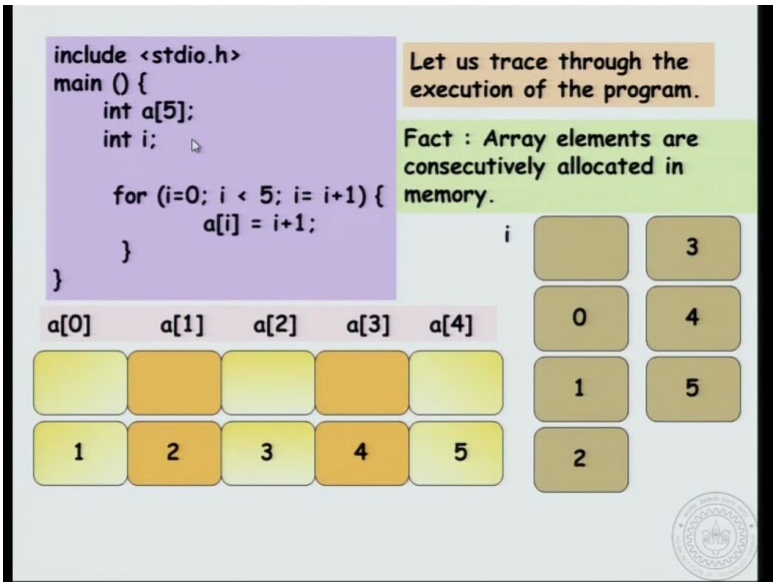
(Refer Slide Time: 05:35)



Now, let us consider a simple program using an array. So, I mentioned that the third requirement that I want for in array is that… So, that first requirement was that all elements of the array are of the same type. Second requirement was that, it has a finite size and that the third requirement is that I should be able to selectively update only one element of the array without touching the other elements. So, let us see a program where we can do all that. So, here is a simple program it declares an integer I and integer array a five and then a for loop. So, let see what the for loop is supposed to do. So, the for loop starts from i qual to 0 and then goes from i = 0 to 5, filling in the elements by sing the statement a[i] equal to i. So, let us see what that is supposed to do.

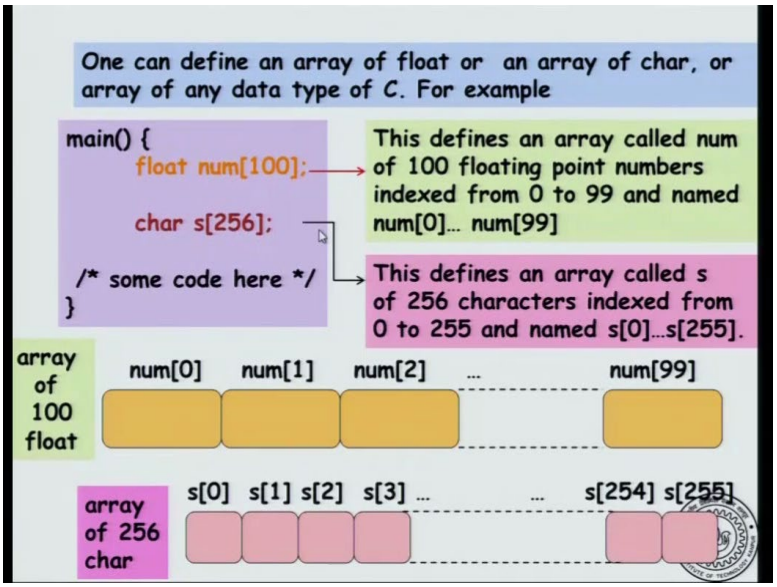
So, this is the notation a i is the notation used to address the elements of the array. So, notice the similarity here a 5 when you declare it say's that it is an array of size 5. a[i] is saying that I want the I th element in the array. So, when i = 0 it will refer to the 0th element in the array until i = 4. It will go on and till the fourth element of the array. So, a of 5 similar to a of 5 the way we row declare the array say's I want the ith element of the array. So, the variable I is being used as an index for a, that means, if I say a[i] will pick the ith cell, in the ith element in the array. Now, this is similar to the mathematical notation a subscript i, which is what we normally use for vector and matrices.

(Refer Slide Time: 07:46)



So, let us run through the program once to see what is doing. So, first we declare a 5, which is five consecutively allocated integers in the memory. And we also have a variable i, i starts with 0 and for this 0th iteration a[i] is allocated let say i + 1, so, a 0 will be 1 then we update i. So, this statement becomes a[1] = 1 + 1which is 2. So, and then execute it a 2 becomes 3, a 3 becomes 4 and a 4 becomes 5. So, notice that because we have indices, and indices can be numbers, they can also be replaced by integer expressions,, this is the trick that we have used here. So, a a[i] goes from a 0 all the way up to a 4.

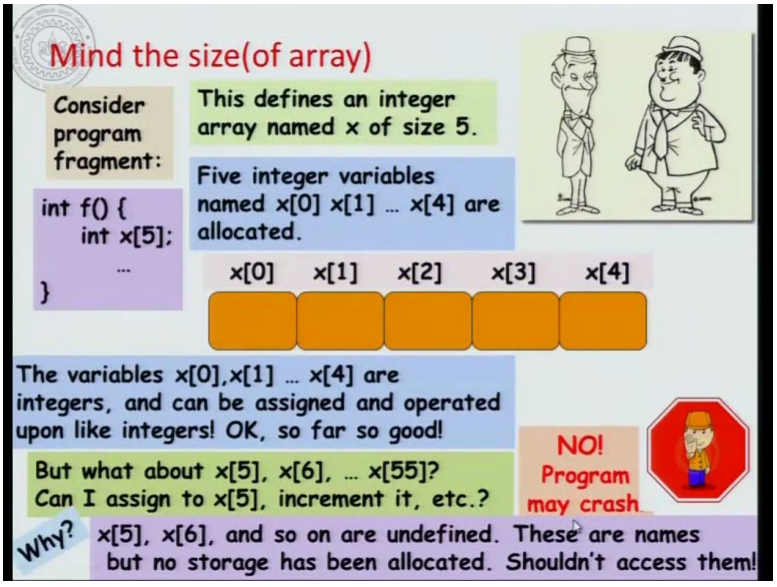
(Refer Slide Time: 09:05)



Now, it is only required that a single array can be objects of same type, so, we have dealt

with integer arrays so, far. Now, we can also deal with floating arrays and float arrays and character arrays and things like that. So, in general you can declare an array of any data type in c for example, you can say float num 100. So, that will declare array of hundred floating point numbers, similarly char s 256 will declare, a character array of size 256. So, you can declare floating point array and you can visualize it as hundred floating point boxes, allocated consecutively that is important part. The consecutive location in the memory one after the other will be allocated for the same array. Now, for a character array similarly hundred boxes are allocated one after the other. Now, depending on the size of the data type involved, obviously, the size of the array will be different. So, the float array will be of size hundred times the size of the single float and the character array will be of size 256 times with the size of a single character and so on.

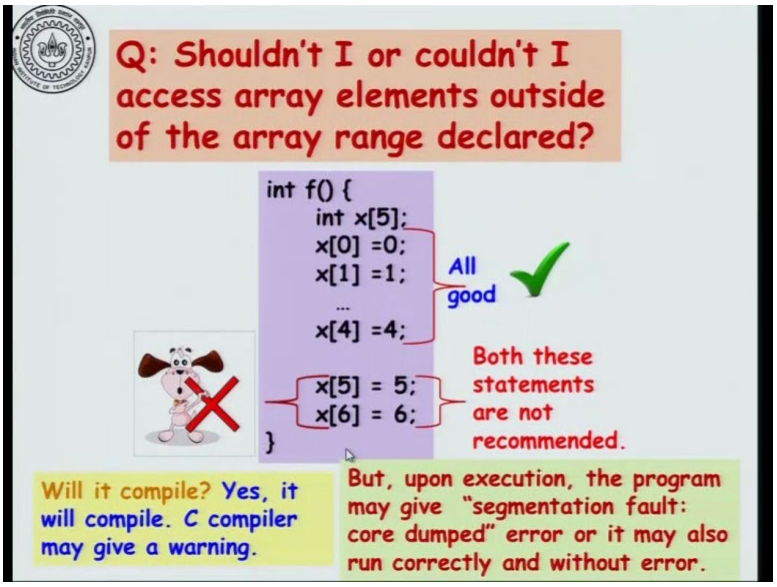
(Refer Slide Time: 10:26)



So, one thing is, we have to take care of the size of the array. For example, if we have an integer array of size 5x this means that 5 integer variables named x0 to x4 are allocated. Now, the variables x0 to x4 are integers and they can be assigned and also they can be operated on, they can be part of other expressions and so, on. Now, what about arbitrary integers, we know that 0 to 4 are valid integers what about 5 and so, on. What happens to x5, x66 something like that. Similarly, what happens what will happen if I right x[-1] what are these valid. So, the answer is no, you cannot in general assume that indices other then 0 to 4 make any sense. Your program may crash and this is the most important thing in c programing when we use the array it is the main part of it, because it is not even guaranteed that a program will crash. So, you may run the program once with x of 5

let us say and the program will work fine. And you will be under the false impression that everything is correct in our program, but the next time you run it, may be your program will crash. So, it is not even guaranteed that it will crash, if it is guaranteed that it will crash, then of course you can know that there is an error, and you can go back to the code. In this case you just you have to be careful when you write the code. So, x 5 x 6 and so, on are undefined, these are names, but there are no storage location that they correspond to, so, you should not access them.

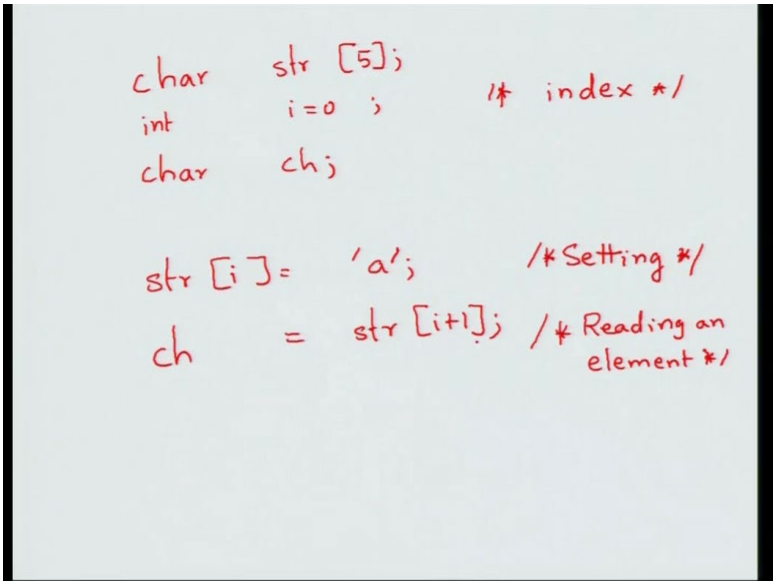
(Refer Slide Time: 12:42)



So, if you ask a very specific question, shouldn't I access them are can't I access them. So, what will happen if I write a code like this where I declare an integer array of size 5 then I know that x0 to x4 are valid location they are the first five locations. But the problem comes with statements like x[5] = 5 x of 6 equal 6 5 and 6 do not refer to valid locations in the array so, what will happen. So, the initial statements up to x[4] are all fine, but the last two statements x[5] = 5 x[6] = 6 lead to arrays will it compile? Yes, if you just give the source code, with these erroneous locations it will compile, but c compiler does not check that the indices are within the proper range. So, it will compile and the compile will not tell you that there is anything wrong with there, but when you run a program the program will give something called a segmentation fault it may are may not give that. So, this is one of the most notorious errors when you program in c. So, we will see this error in greater detail when we understand something called pointers. But in general when you exceed the bounce of the array, when you go beyond the locations permissible in the array, your code may crash and the code will crash usually

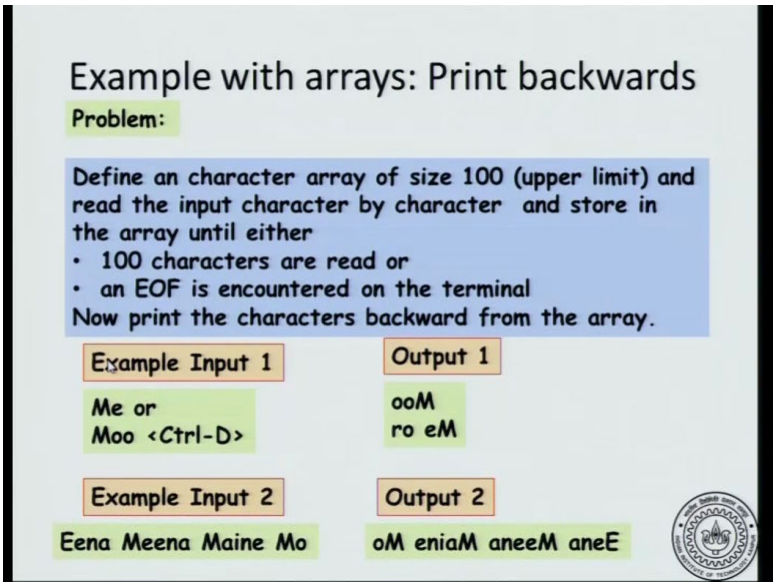
with the error segmentation fault. So, if you run the program and if you see a segmentation fault this is a good indication that may be you are referring to locations in your array that do not exist. So, you should go back and rectify the code, but the danger is that, it may not always crash. So, the only way to be really sure is to go through your source code and examine it. Your program may crash, so, we have seen certain aspects of arrays in c so far.

(Refer Slide Time: 15:04)



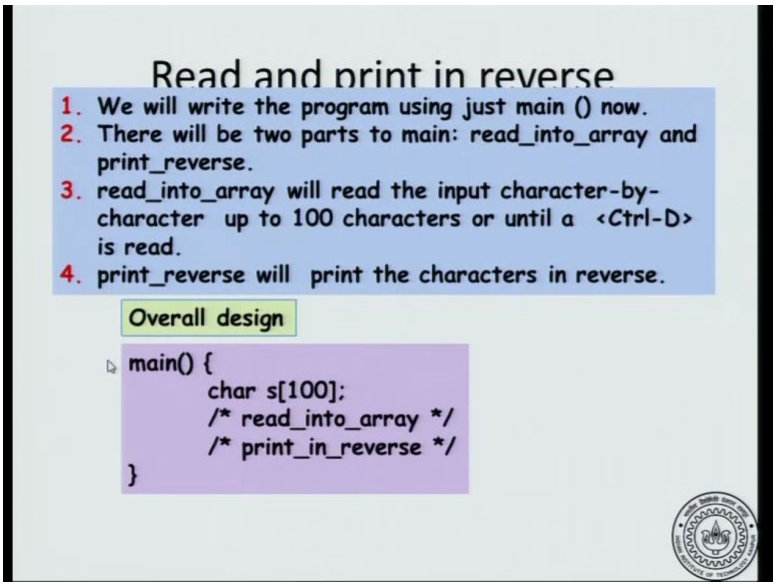
So, for example, let say that I declare a character array str of size 5, so, it has five characters inside it. And let us say that I use the variable I as an index into the array. So, str[0] to str[4] can be addressed using the index i. So, if I have the index, I know that I can set particular values as str[i] = 'a'. Since, i is 0 this will set the 0th element in the array to character a. Similarly, if I say ch equal to str[i + 1] it will take whatever is in the first cell in str[1] and assign it to the variable ch. So, we can set particular element in an array like this. Similarly, we can also read the value in an element and then assign it to something else. So, these are possible with the help of an array now let us consider a particular example, which is the problem is as follows. We want a character array let say a size 100 and then we have to read input which is from the key board and store them in the array, After we have stored it we should stop, once at least hundred character have been written, because that is array size or when the user first end a file. Remember that you can press <Ctrl-D> to enter the end of file.

(Refer Slide Time: 16:49)



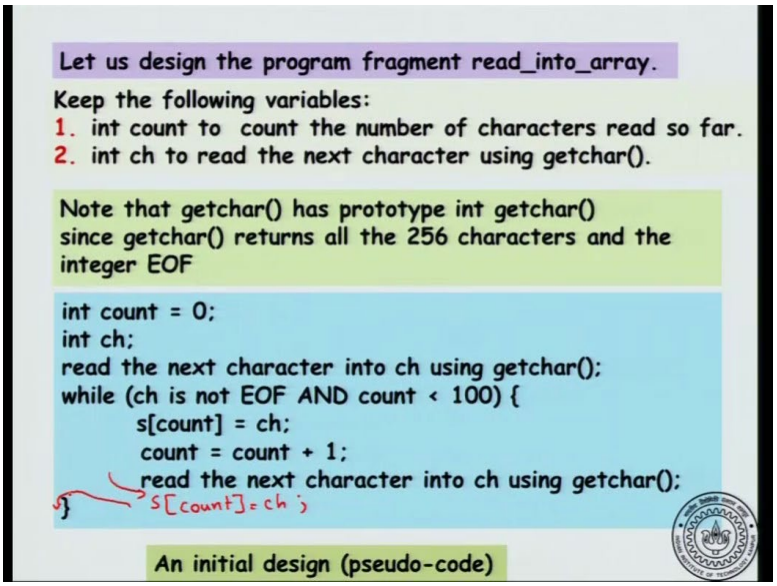
Now, what we have to do is take the error, take the array and print it in the reverse order. Now, if you think for a little bit you can see that it is difficult to do this without an array. Instead of an array, if you are storing it in a single character, there is no way to store hundred characters in one variable and then print them in the reverse order right. Because the first character has to be printed at the end and last character entered has to be printed first. So, you need to remember all the characters, this is an intuitive reason why arrays are important for this problem. So, what is an example problem let say that we have m e or then new line then Moo <Ctrl-D>. So, when you reverse it, you will have oom then the new line then or emn and so, on. So, you have to reverse everything input. Similarly, if you have a string what you have to output is the exact reverse of the string including the spaces.

(Refer Slide Time: 18:09)



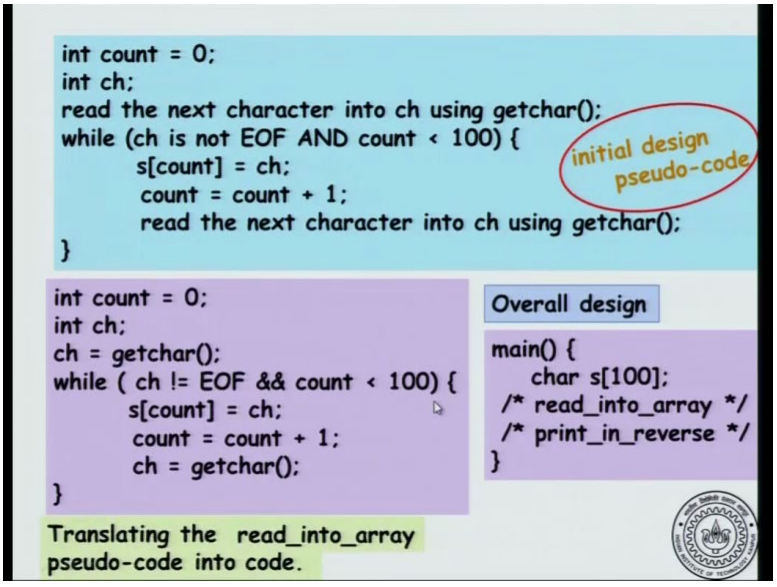
So, let us design the program we will just try to write the program using may. Now, there are two parts in this program the first is just to read what has been input into the array and the second part is to print the array in reverse. So, the read into array that part of the program will read the input character by character, until one of two events happen. The first is hundred characters have been input because you have declared the array of size only hundred. So, you can read only hundred characters, so, once you reach that you should stop. Otherwise, before you reach hundred characters may be the use of first a <Ctrl-D> to say that I am done with input ok. In that also, you have to say that I have done by reading the input. Now, print reverse we print the characters in reverse ok.

(Refer Slide Time: 19:07)



So, let us design the program for reading into the array. So, keep the fallowing variables, one is to keep the count of how many characters I have read so far. And then I will keep a variable to store the currently read character. Now, we have touched upon the topic once, I am going to declare it int ch instead of char ch. So, I am not going to do this, this is because to getchar will give you whatever character has been read just now from the input. So, in particular they are character that can be entered can be an ASCII value which is from 0 to 255 or something. And then it can also enter the eof character the end of file character which is actually -1. So, -1 does not correspond to ASCII character. So, getchar can also read an end of file character, this is the reason why, if you are reading character through getchar and we are doing this because, the user can also enter the -1 then in order to hold at value you need an int ch rather than a character ch.

(Refer Slide Time: 22:50)

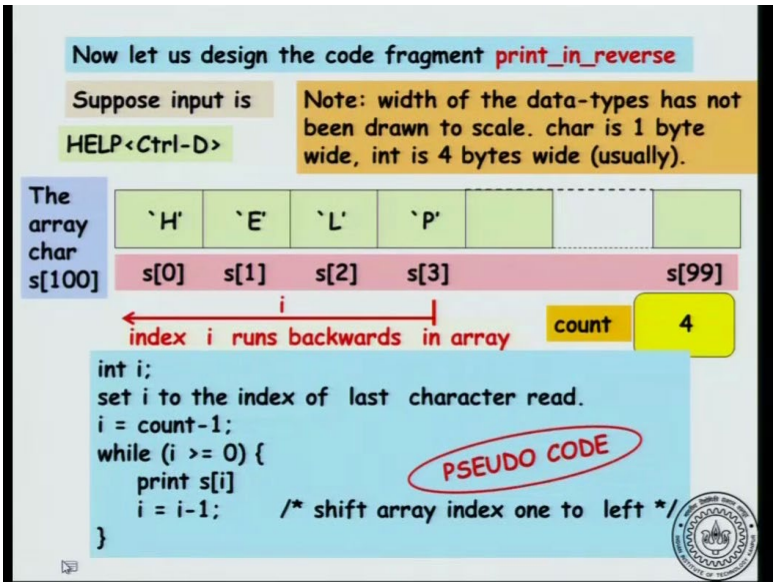


So, this is what we just mentioned, the end of file character is usually -1. So, it is not a valid as key value. So, the code at the top level looks like this, we have the logic to read a next character into the ch using getchar, and then we have a let us say while loop which says that, while the character is not the end of file ant the number of characters read count is less than 100. You store the character into the array increment count and then read the next character. So, please look at the structure of the loop very carefully the s is a character array. So, technically it cannot hold end of file, but then if you think about it little bit you will see that we will never encounter the situation where, you will store end of file into the s array, because suppose first character is end of file then we will not even enter the loop. Now, at any point when we enter end of file, it will be at this point right

we will read the character only here, before storing it into the array we will actually check whether it is end of file. So, we will not accidentally set the array to -1 at any point, so, character array suffices. So, think carefully about the way this loop has been interpreted.

In particular, if I had just done this as the last line before the loop ended then, you would run into problems because you could store the end of files character into the s array by mistake so, just think about that issue. Now, here is an initial design and so, the overall design is that first you have to read into the array and then you have print it in reverse. So, let us make the read into array little bit more precise. So, we have ch = getchar() and because you are using the getchar function we have int ch, because it could also be an end of file. Now, the while loop says that while the ch is not end of file and the number of characters read is strictly less than hundred increment. So, you first set s[count] equal to the character read,, the increment count and then get the next character. So, this loop keeps on filing characters into the character array until you see either end of file or you have enter 100 characters.

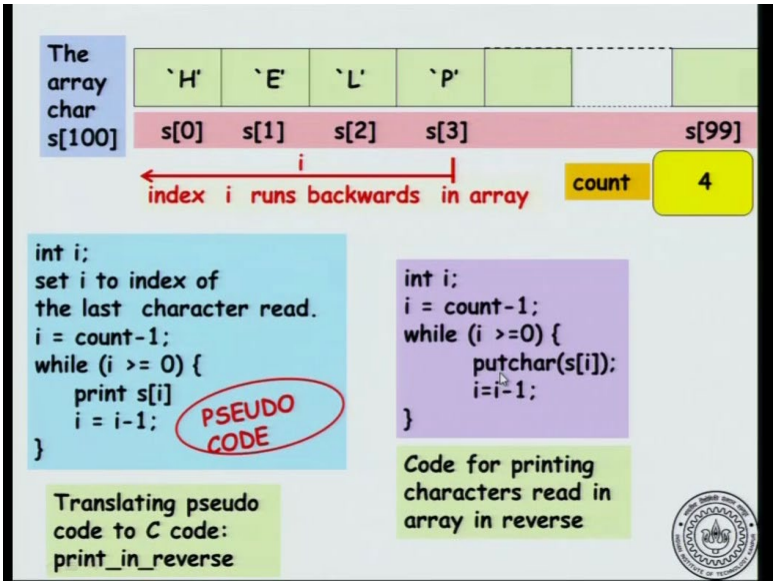
(Refer Slide Time: 23:54)



Now, let us design the remaining thing which is print in reverse. So, suppose the input is let us say to be concrete HELP and then <Ctrl-D> the end of file character. So, once you run the reading to array part, it will lead to the array looking like this s 0 will be the character h, s 1 will be e, s 2 will be l, s 3 will be 3, p at this point you will read end of file and you will not store the end of file in the array right. So, s 0 to s 3 are valid characters and at this point, if you go back to the code count always keep tracks of how

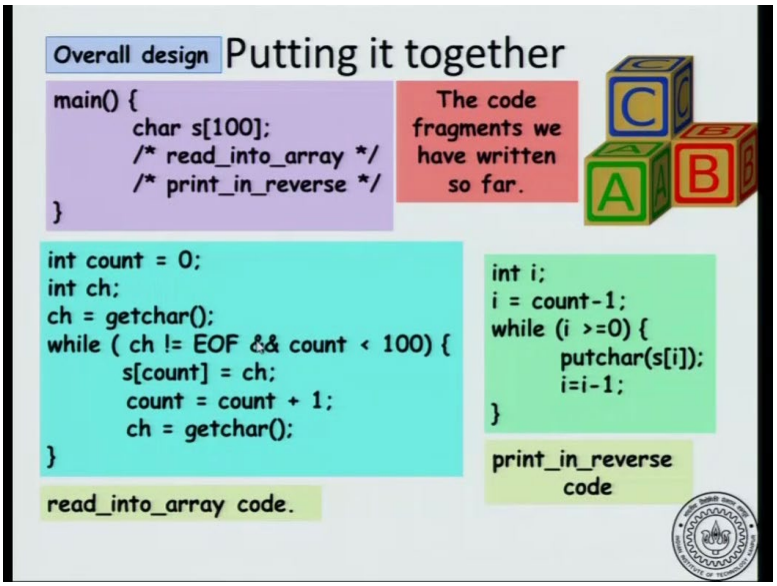
many characters have been read. So, in particular count will be 4 when you exit the array. Now, to print it reverse, all you have to do is to start printing from s 3, then s 2, then s 1, then s 0 right. So, you read in this direction and you print in the reverse direction. So, we should be somewhat careful at this point suppose you have read the array before you enter this part, then you declare i which is the array index that we are going to use now i should be set be the index of the last character read. So, here is the tricky part notice that count is 4. So, four characters have been read therefore, the last character read is at index count -1. So, it is not at count index, if you say s of four that is an invalid index whereas s of three is where we should start from. So, start from i = count - 1 that we will start at this character, now, while i >= 0. So, we will start from s[3] then print s[5]. So, we will print s 3 then, decrement I because you have to go back to the next to the previous character. So, i becomes 2, i >= 0, so, you will print s 2 decrement I, so, i becomes 1. So, you will print e, then decrement i you will come to i become 0 and you will print h you decrement once again i becomes -1. So, you will exit the while loop ok.

(Refer Slide Time: 26:44)



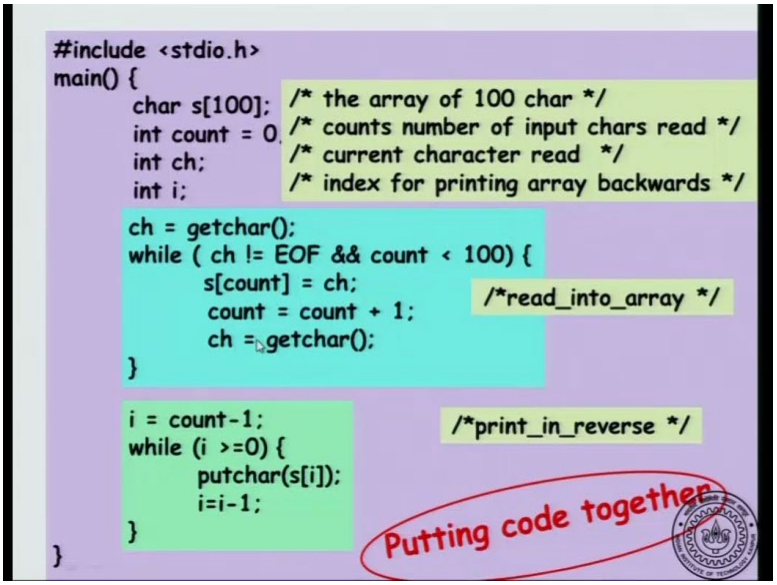
So, this is the array that we were doing and so, here is the code for printing the characters in reverse. So, here is the pseudo code where we said print s of I instead of that in c we have a particular function which will print the character which is put char. So, due to this the dual function of getchar. So, put char takes an character as an argument and prints it on to the standard output. So, you have int i i is set to be count -1 because that way we will get the last index of the character in the array and then you start counting down until use print the first character and till the end of the array ok.

(Refer Slide Time: 27:37)



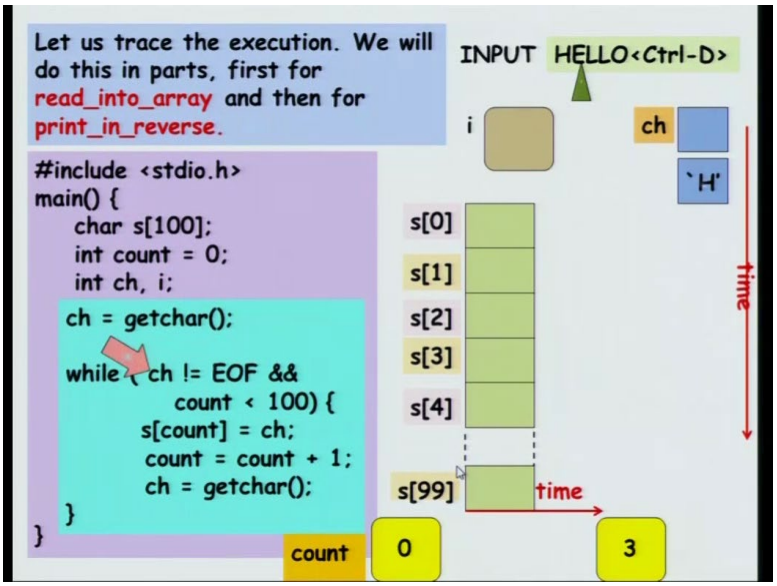
So, putting these two together, you have the read into array part and then you have the reverse part print in reverse part. So, when you put these two together the first thing you do is, bring all the declarations together.

(Refer Slide Time: 27:51)



So, this is the declarations for read into array as well as print to put together. Similarly, first you have to print, you have to put the code for the read into array part and then the code for the print in reverse part ok.

(Refer Slide Time: 28:13)



So, let us trace the execution for a small sample input. So, then we have the input is hello and then the user presses <Ctrl-D> for end of file, let see what will happen. So, you start reading into the array. So, s[count] with count equal to 0 starts setting the array. So, s 0 will be h and then h s 1 will be e and so, on. So, once ch becomes <Ctrl-D> the end of file character you will exit the loop. So, the character array is hello.