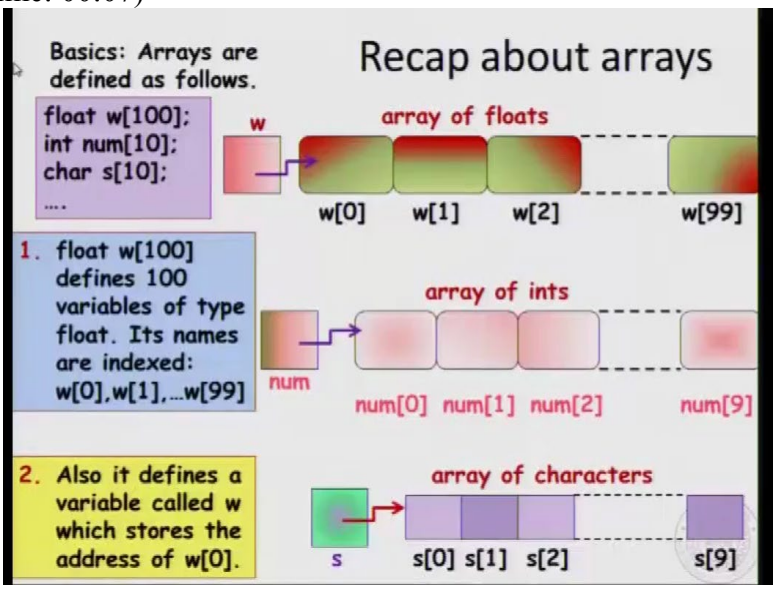
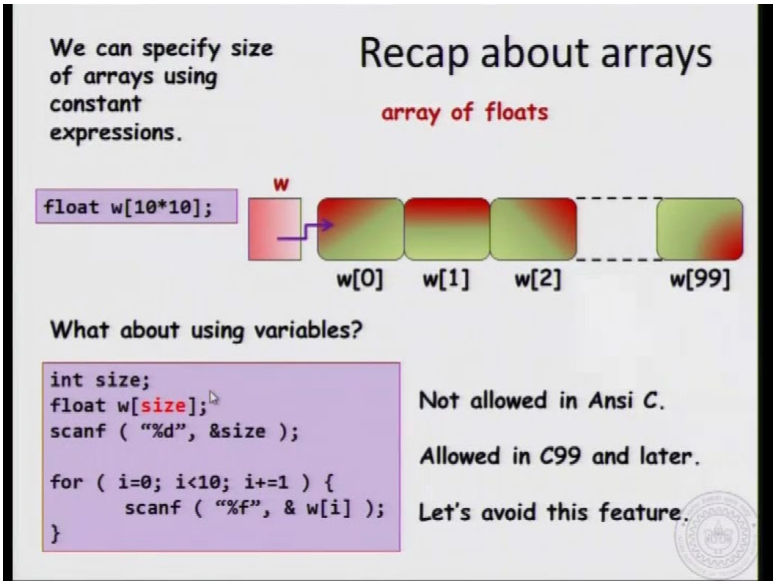
Introduction to Programming in C
Department of Computer Science and Engineering

(Refer Slide Time: 00:07)



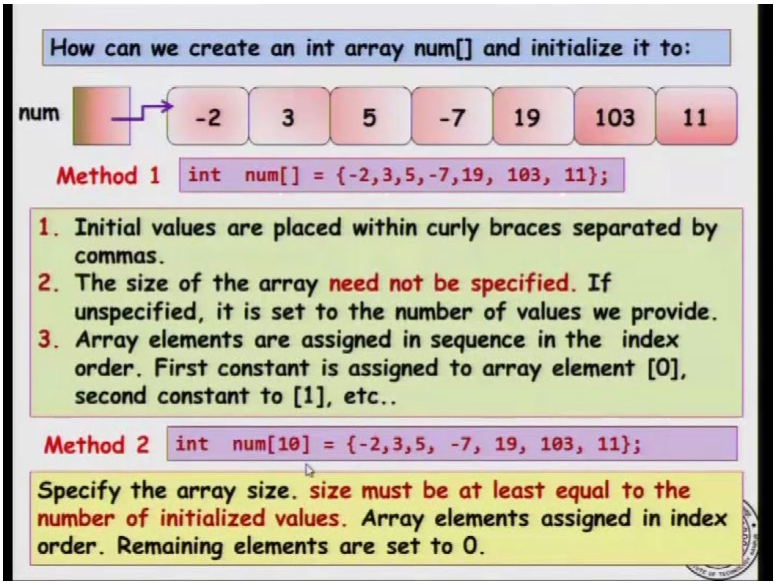
In this lecture will just talk about how to initialize arrays. So, recall that we have defined arrays as follows, if you declare an array float w[100], it will declare an array of floats 100 floats consecutively allocated in memory. And we have also mention the fact that there is a separate box w, which will point to the first location in the array. So, it contains the address of the first location. In num[10] will declare and integer array of 10 integers plus one box which will hold the address of the first location, and so on. So, the arrays names, the cells of the array or the elements of the array are index from w[0] through w[99] the indices start from 0. And we also mentioned that conceptually there is a separate variable called w, the name of the array which stores the address of w[0].

(Refer Slide Time: 01:11)



Now, it is not important that we use numbers as the size of the arrays, we can also use constant expression; for example, we can say float w[10 * 10]. So, instead of saying 100, we can give an arithmetic expression which evaluates 200, and this has the same effect it will evaluate an array of 100 elements staring from w[0] to w[99]. And there is a separate box called w which move to the address of the first location, but what about using variables or variables size arrays, this is feature that we often which we had. So, what do I mean by that? I could declare the following code int size, and then float w size and I could say scanf ("%d". &size);. So, user enters the size of the array, and then I can enter 10 elements in to the array for example, but here the size of the array itself is a variable which depends on the user input. And we often wish that we would we would be able to allocate variable size arrays, but this is not allowed in Ansi C, it is allowed in the latest versions of C 99 C 11 and so on. We will avoid this feature for the purposes of this course, let us assume that array means they are declared to be of constant size. By constant size you can give the size as a particular number or you can give it as a constant expression, that is an arithmetic expression involving constants, but not general expressions.

(Refer Slide Time: 03:16)



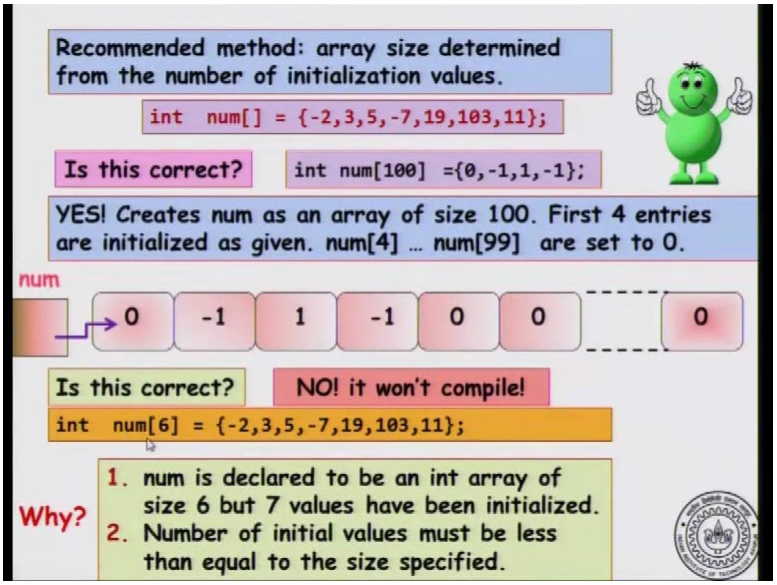
Now, let us just look at how can we create an integer array num and also initialize it to particular values; for example, I want the num array to look like the following, it contains 7 cells having the values - 2, 3, 5, - 7, and so, on. Now I know that if C did not allow me to initialize arrays when I declared it, I could declare the array as int num 7 and then I will just write num[0] = - 2, num[1] = 3, and so on until num[6] = 11. So, here is a way that I can create an array and ensured that this state is reached, but is there more convenient way of doing it. Can I start of the array with these contents. So, C allows you two ways do it. The first is I declare an int num[] and then specify what are the initial values, so - 2 so on up to - 11 within {}.

So, this is one way to that C allows you to do this. The initial values are placed within curly braces and separated by ,, the size of the array need not be specified. So, I need not say that num[] has size 7, it will allocate an array with enough space to hold 7 integers. Array elements are assigned in the order that you specified. So, num[0] will be - 2, num[1] will be 3, and so on. So, it is done in a reasonable manner. This also another way to do it, which is slightly different from way above, I can declare the size of an array. So, I declare an array of size 10, and then give this initial value. What will happen in this case, is that it will make sure that the size of the array is at least equal to the size of list that I have given. So, I have given 7 elements, and I have declared an array of size 10, 7

is less than 10. So, it is fine. So, I can declare an array of size 10, I should give a value, I should give values at most 10 in number. So I can give a 10 or below. In this case, I give 7 numbers. So, what happens is that, array is initialized in the order elements given num[0] will be - 2, num[1] will be three and so on, until num[6] will be 11. 7 elements are filled; the remaining 7 elements are unspecified. So, they will be initialized to 0.

Now let me just remained you that if I had just declared an array int num[10], and then put a semicolon. So, I had just declare an array without saying any initialization at all, then you should assume that the array contains or arbitrary values, you should assume that array contain Junk values, but if you initializes an array of size 10, and give only 7 initialization values, then the C standard gives you the guarantee that the remaining elements are initializes to 0. So, they are not junk.

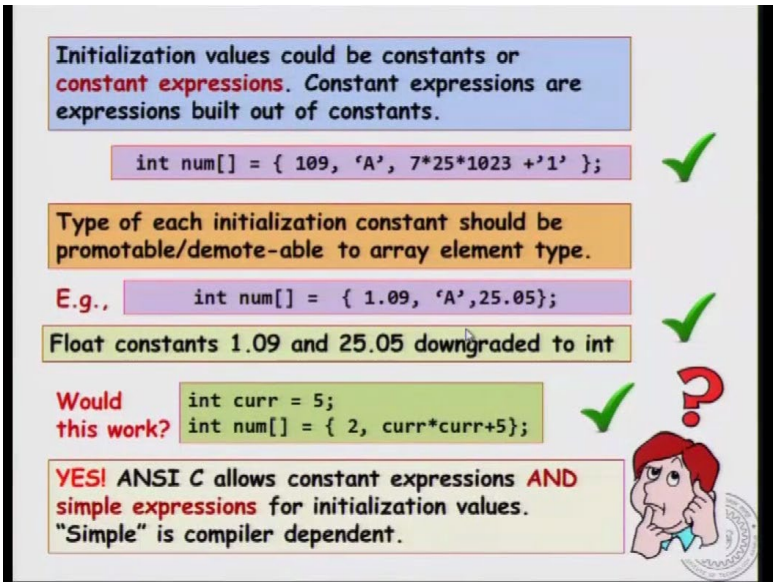
(Refer Slide Time: 06:52)



The recommended method to initialize an array is to give the list of initial values, and let the compiler decide what the size of the array it should be. So, if you give 7 initial values, it will decide that the array is of size 7. Now is the following code correct, if I declare an array of size 100 num and give four initial values. So, this is correct, it creates num as an array of size 100, the first four entries will be initialized as given. So, num[0] will be 0, num[1] will be - 1, num[2] will be 1, num[3] will be - 1, and then num[4] until

num[99], they are all set to 0. So, after the initializations the array will look as follows; the first four values are what we given and the remaining value are 0's. Now is the following code correct, num[6] = and then you give a list of 7 values to initialize, is this correct? The answer is no, it will not compile. So, if you right this code, and compile it using gcc, it you will get a completion error. Why is that? We have declared an array of size 6, but we have given 7 initial values. So, there is no way to do this. So, the rule of thumb is that either give no size for the array, and let the compiler figure out or if you do give a size it has to be at latest 7, which is the number of values that you give, it can be 10, it can be 100, but it cannot be less than 7. Now just like size can be not just numbers it can also be constant expressions, we can also have constant expressions as initialization values ok.

(Refer Slide Time: 08:54)



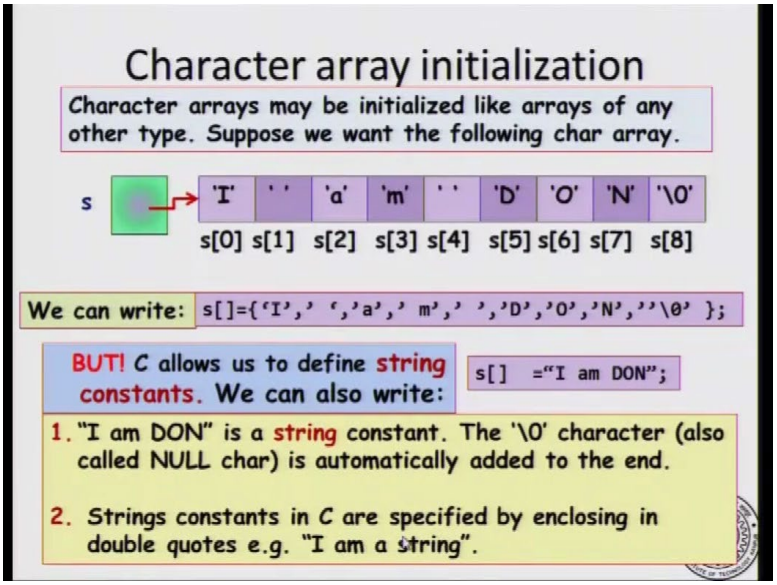
So, for example, I can give num[] = 109, then the character value A, character value A means it will take the ASCII value of A, 65 or whatever it is. So, the first number will be 109, the second number will be 65, let us say if the ASCII value A is 65, and the third value will be 7 * 25 * 1023 + '1'. So, whatever the ASCII value of the character one is let say 90 or something. So, it will be added two this, constant expression, and it will be initialize to that value; num[] two will be the result of evaluating this expression. So, the type of each initialization constant should be promotable or demotable to the array

element type. So, the each value in the initialization list should be compatible with let us say integer, because we have declared the array of size of type integer. So, what do I mean by that, for example I can initialize an array num[] with initialization list 1.09, then , A , 25.25. So, this is ok, because the floating point values can be downgraded to integers. So, may be this will be initialize to one then whatever the ASCII value, A is let us say 65, and then 25.

Now, these are about constant expression. What about expression involving variables when we initialize an array. So, can we do something like this. int curr = 5, and then the num[] array is initialize with {2 , curr*curr+5}, will this work. The answer surprisingly is yes that it will work on most compilers. So, the ANSI C allows constants expressions, and simple expressions for initialization values. Now simple is of course dependent on which compiler we are using. So, if you write a code, and compile using gcc with such an initialization may be or code will compile, and the movement you compile your code with a different compiler it may not compile.

So, earlier I had said that the size of the array cannot be initialized using variable expressions. In ANSI C that is forbidden, but the initialization value, so the value that goes in to the array can involve variable expressions, this may or may not be supported. So, it is safe to assume that both the size of the array, and the initialization value can be done only using constant expressions, even though some compilers allow simple initialization values using variable expressions.

(Refer Slide Time: 12:08)



Now how do we initialize character arrays? Character arrays can be initialize like arrays of any other type, suppose we want the following array. s[] = 'I' , ' ','a', 'm',' ', etcetera. So, I can initialize it just like a initialize the other array, I will not specify their size of s and then give this characters, I am DON. So, this is another way to specified and the last character is a null character, but C 'also allows you to define what are known as string constants. So, we can also write s[] = "I am DON", but now with in double quotes. So, this is known as a string constant, the null character is an implicit ending character inside a string constant. So, it is automatically added to the int. Now the string constants in C are specified by enclosing it in double quotes.