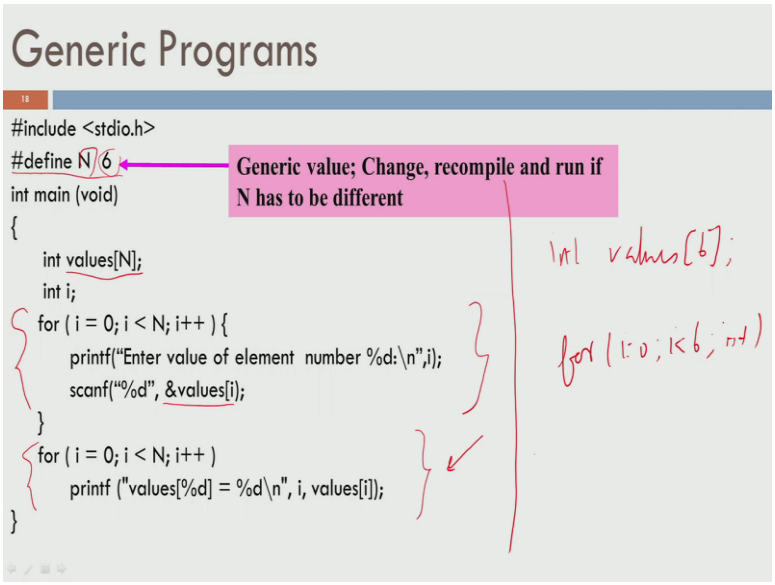
Programming, Data Structures and Algorithms
Prof. Shankar Balachandran
Department of Computer Science and Engineering
Indian Institute Technology, Madras

Module - 8B
Lecture – 10
Contents
Rcode segments for reading into and printing out of arrays
Example: Fibonacci unmbers
Example: Using arrays as counters

In this module, we look at how to work with one dimensional array. And I am going to show more detailed examples and also write small programs for some sample problems.

(Refer Slide Time: 00:23)



So, let us look at this piece of code. What this piece of code does is actually read values. So, it says enter value of element number; percentage d. You are reading some values into values of i and you are printing them. So, it is a fairly simple program. So, I am reading values and I am printing the values.

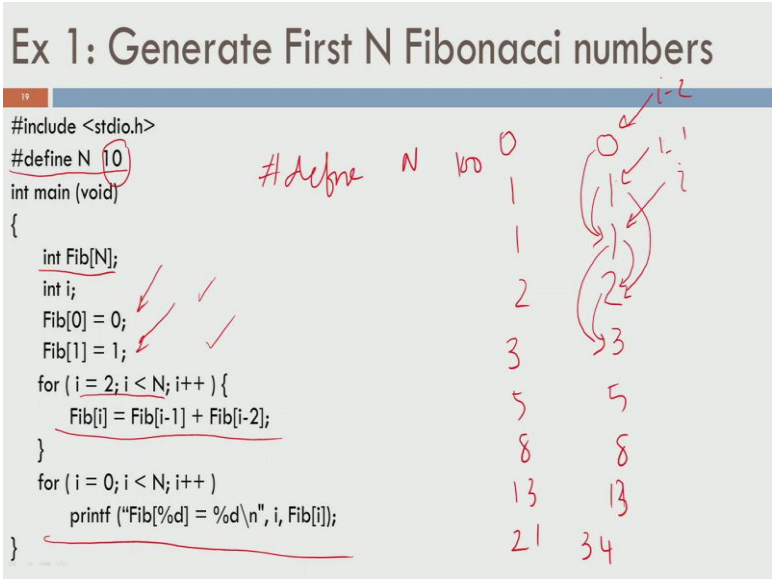
So, one thing that happens in real world is that, many times you do not know what is the number of things that you want ahead of time. So, we want to be able to write generic programs; which means, I do not want a program to be of fixed size. So, let us say in the

previous example, temperature, I declared it to be 365. What if I had a leap year? In which case, I need 366 days and not 365 days. So, I need a mechanism by which I want this in a generic fashion.

And that is where C comes in very handy with something called hash define. So as of now, so see this lines, line 2; hash define N 6. So, what this does is where ever you see a symbol N, it is going to replace that with the value 6. So, now if I go and look at this loop; let us see how to make sense of this loop. For i equal to 0, i less than N, i plus plus. Where ever you see N, the C precompiler or pre-processor will replace that with 6. So, this loop will run from 0 to 6. So, what this loop will do is read seven elements; numbered values of 0 to values of 6. And this loop also runs on same N. So, it will print values of 0 to values of 6. So, we have declared the array with values of n; n is not a variable. So, n takes the value 6 here. So where ever you see n, assume that it is number 6 there. So, what you are essentially doing is it is equivalent to say int values of 6 and you have a for loop running from i equal to 0, i less than 6, i plus plus and so on. So, we have indices that go from 0 to 5. I am sorry, 0 to 5 and not 0 to 6. And this loop also runs from 0 to 5. So, we have 6 locations numbered 0 to 5.

So, this generic value is very useful, but one small thing that you have to be careful about is that if I change this n to let us say 10, I expect 10 values; numbered 0 to 9. But if I change, the source code is not enough. Remember, once we have a change in a source code we have to save it, compile it and then run it. So, whenever you have a change in the source program you have to save, compile, and run. If you want to change the value of n, it cannot be a single change in a source program and which will automatically reflect in the binary to be different. So, whenever there is a change in source program, remember you have to save compile and run it.

(Refer Slide Time: 03:36)



So, from now on we will try and use generics as much as possible. So, the first exercise that we are going to look at is the notion of Fibonacci numbers. So, Fibonacci numbers are very interesting they come in various places later. So, Fibonacci number goes like this. The first Fibonacci number is 0, the next number is one. And from there on, the Fibonacci numbers will go in this fashion. You take the previous 2, add them and that gives you the next Fibonacci number. So, 0 plus 1 is 1. Then the next Fibonacci number is take one. And so take the previous 2, add them, then next number is 2. Then again you take the 2 previous ones, add them; it is 3 and so on. So, the Fibonacci numbers go in the sequence 0, 1, 1, 2, 3, 5, 8, 11, sorry, 5, 8, 13 and so on. This is the Fibonacci sequence.

So, let us say I want to write a program to do exactly this. So, till now we were scanning the values from the user. Now, we have a program in which it will actually fill in the values. So, we are going to generate the first n Fibonacci numbers. So as before, we have a hash define N. So, what we want is we want the first 10 Fibonacci numbers starting from 0. So, we declare an array called fib of N. So, this will declare 10 memory locations and you can address them as fib of 0 to fib of n. We already know the value of fib of 0 and fib of one. And what the loop is going to do is it is going to run from 2 to 9. So, you have i less than N; which means i less than 10. So, it runs from 2 to 9. So, i equal to 0 and i equal to one are already taken care of. When you run from 2 to 9, you get another 8

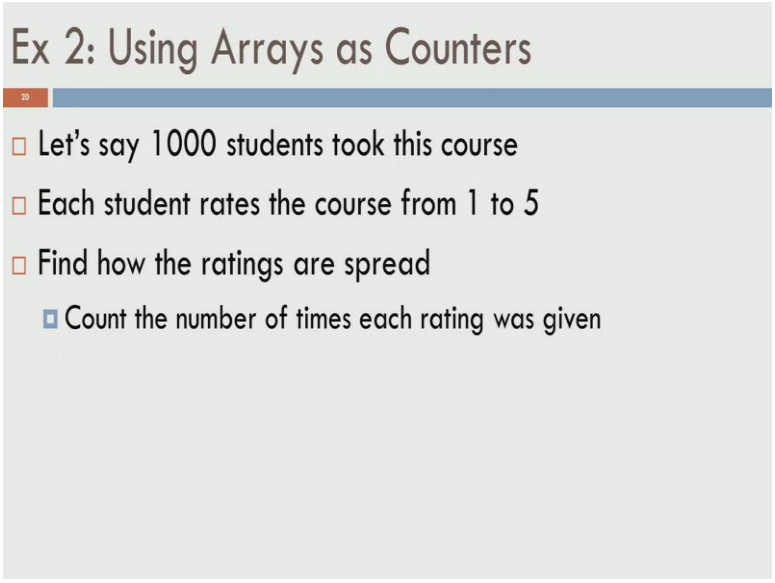
values. So, this is; overall you are filling up 10 values of fib.

And as I showed earlier, we have to take a particular index i and we look at location i minus one and we look at location i minus 2. When we add these 2, we get fib of i; that is exactly what this line is doing. We take fib of i minus one and fib of i minus 2, add that and put it in fib of i. So, the first time when i equals 2, fib of 2 will get updated with 0 plus 1; which is one. Then you go back to the loop, you check whether 2 is less than 10. Yes, 2 is less than 10, so you increment; you get 3. So then you are ready to; you are now ready to do. Fib of 3 is fib of 2 plus fib of one which is one plus 1 giving 2 and so on. So, what you are having is a loop that runs from 2 to 9. And it is taking the previous 2 values and adding them and storing them in a fib of i. So, at the end of the loop you will have numbers of this form; so 0, 1, 1, 2, 3, 5, 8, 13, 21 and 34. So, that gives us the first 10 values of Fibonacci series. So, you will see the array fib filled up from 0 to 34 and you can print them here.

If I want the first hundred numbers, let us say, all I have to do is I do not have to go and touch the program anymore. I go and change the hash define. I, now write hash define N hundred; this changes N to hundred. All I have to do is save it and compile it. I do not have to go and touch these other lines; where is the N. This is very useful. So, usually what we do is we keep the values which are very small, so that we can test the programs. And then maybe I want something some examples where N is very large. I test it with very small values to ensure that it works fine and then I go and change it to some other larger value.

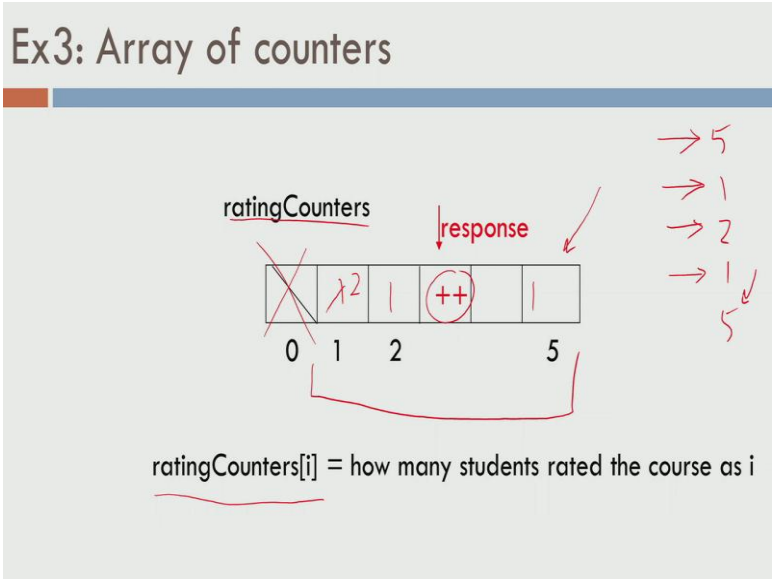
We will see a similar example later. But once we have this, we can now move onto other exercises. In this exercise, the crucial point is we did not have fib of i filled up with values. It starts out with uninitialized values and slowly you fill them up one after the other from fib of 2 to fib of 9.

(Refer Slide Time: 08: 10)



So, now let us move to another example. Let us say you are taking this course. And about, let us say thousand students are taking this course. And I want to find out how you students rate the class. I give a mechanism for you to rate the class. You can rate the class from one to 5. And at the end of the course I may want to see how the rating of this course was spread. So, each student is supposed to give a rating 1, 2, 3 4 or 5; one being bad and 5 being good let us say. And at the end of it I want to see that how many people said good, how many people said ok, how many people said bad and so on. I want that. So, this is something that is very common. And what I want is I want to count the number of times each rating is given. So, maybe I want to see that, "Oh! There are 10 people who gave very good rating; hundred people who gave good and so on". So, I want to see numbers like that. Let us see how to write a program to do that.

(Refer Slide Time: 09:12)

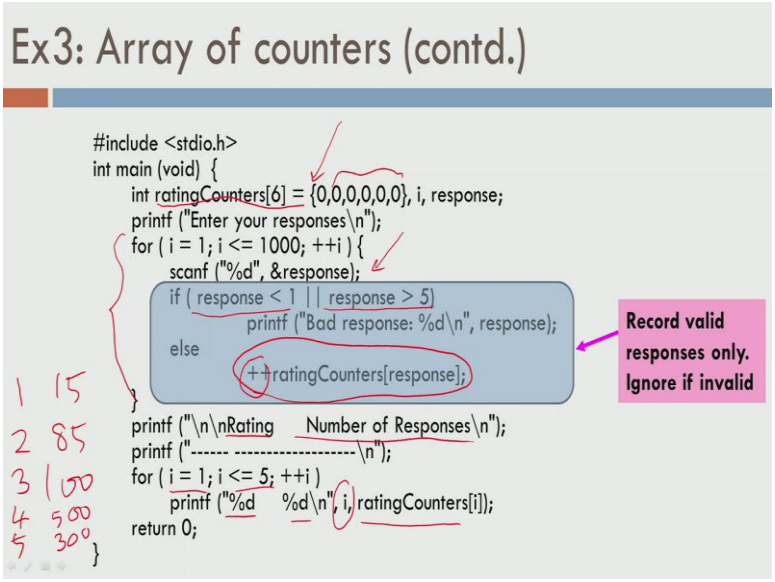


So, I am going to maintain an array called rating counters. That is the array name that I am going to use. And I know that the valid ratings are one to 5. So, I am going to have 1, 2, 3, 4 and 5. I have 5 locations. And rating counters of i is going to maintain the record of how many students rated the course as i. So, 5 students gave rating one. I would expect rating counters of one to be containing the value 5. So one thing is, since the rating are only 1 to 5, however array indices start from 0. What I am going to do is I am going to declare an array of size 6 and I will not use the location. Therefore, even though my arrays actually is 6 elements, I want only 5 of those. I am using part of the array. I am not using the whole array. So, this makes the programming slightly more easier. Let us see how to write this program.

So, let us say there are only 5 students. And the 5 students gave the rating 5; 1, 2, 1 and 5 right. So, let us say these are the rating that were given by the students. And what I would like to do is go one user at a time. So, the first user gave 5; so, I would want this one to be a count of one. Then the second user gave one; I want this to be a count of one. The third user gave 2; I want this to be a count of one. Also, the 4th user gave one; which means, now there are 2 people who gave rating one. I want this to be 2 and so on. In general, what I want is as soon as I get a rating, I want to do a plus plus on the array location, which contains the response. If I get response as let say 5, I want to go to

location 5 and increment the current value. So, however one thing you have to pay attention to is since we are doing increment, you are going to do a plus plus. We should ensure that all of these entries are set up at 0. If we start with the unknown values, you may end up with the counter, which are incorrect. So, we assume that all the counters are initialized to 0. We start from there.

(Refer Slide Time: 11:40)



So, let us see how a program of this kind will work. So, I have in rating counters of 6. As I said earlier, I am going to accommodate 6 locations. But, my actual entries that I am concerned about are rating counters of one to rating counters of 5. So, even though I have initialized this to 0, I am not really going to touch the rating counters of 0 at all. I am going to touch the rating counters of one to rating counters of 5. So, I have this loop which runs from i equal to one up to thousand; including thousand. For i equal to 1; i less than or equal to 1000 plus plus i. So, at this point we are scanning thousand responses. And for each user or each student, we are going to take a response. And we are first of all going to check if the response is less than one or if it is greater than 5, these are invalid responses.

So, I said the ratings have to be from one to 5 as integers. So, if anything less than one or if is greater than 5, it is a bad response. We do not record it. So, we will record only valid

responses. And if it is valid response, you are going to increment the corresponding counter. So, remember there are 5 counters; rating counter of one to the rating counters of 5. And based on the response, you index that particular counter and increment it. So, again if you want to break it down, you can think of it as plus plus on a variable. And which variable are you incrementing? You are incrementing the variable for which you got the response.

So, if the first user gave one as the response, rating counters of one is going to get incremented. If the 500 th user gave rating of 5, i would be 500. And rating counters of 5 will be incremented by one. And once you are done with this loop you can now print the rating. So, we print rating and the number of responses. So, we run from one to 5, both inclusive, and we print the 2 values. What are we printing? We are printing the rating i and print the count.

So, at the end of this if I scan thousand inputs, the output would look something like this. Response of one came from fifteen people, response of 2 came from 85 people, response of 3 came from 100 people, response of 4 came from 500 people and response of 5 may be came from 300 right.

So in this case, all the user let us say gave valid responses; 5 people gave 3 hundred, 4 people gave 5 hundred and so on. The summation of this will be thousand. So, that is the quick way to check whether your program is correct or not. If let us say 5 people gave invalid responses, then the count will not be thousand. The addition of this would give you 995 and not 100. So, I suggest that you take this program and write it in the edited that I showed you in the last lecture and ensure that this actually works.