Introduction to Problem Solving and Programming

Prof. Deepak Gupta

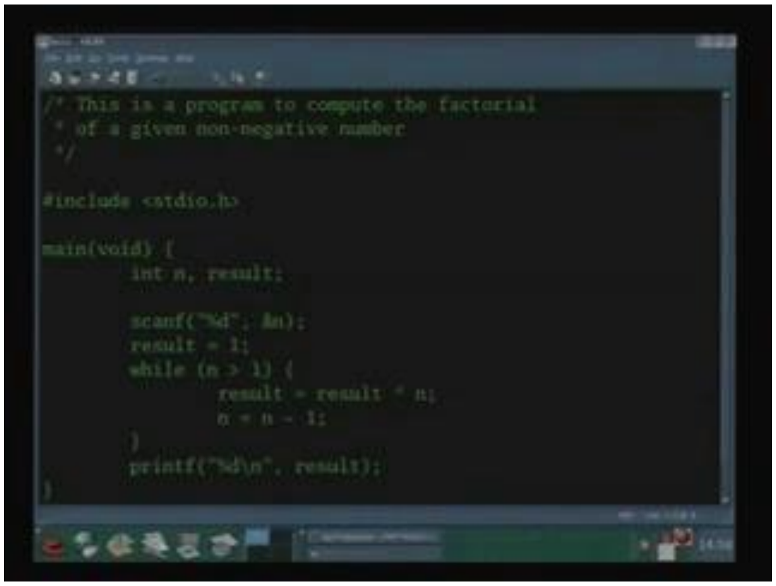
Department of Computer Science Engineering

Indian Institute of Technology, Kanpur

Lecture – 4

Hello we have ended the last lecture by writing a simple C program to compute the factorial of a given non-negative integers. Today what we will do to start again with that program (( )) and try to understand how it works. The idea is to understand basic concept of C language before discussing the more formal settings.

(Refer Slide Time: 00:46)



```
/* This is a program to compute the factorial
 * of a given non-negative number
 */

#include <stdio.h>

main(void) {
        int n, result;

        scanf("%d", &n);
        result = 1;
        while (n > 1) {
                result = result * n;
                n = n - 1;
        }
        printf("%d\n", result);
}
```

So let's look at the program again. So the program that we had written last time, where the few modification, so let's start examine line by line and C what is happening in the program. So what we have here is what is known as command, the command is a piece of permission, the machine will completely ignored. And this is intended only the information for people who are going to read the program. All the not very important that particular program, because if this is a particularly simple program but as we write more and more complex program, it will more importantly to write commands so that we read ourselves and other who read the programs can understand what are the (( )). So in C, (( )) slash star and a star plus is recommend. So the entire thing we recommend from here to here, and this commands basically (( )) from simple information both this programmed as well.

Let’s come to the next line, this is hash include stdio dot h will not explain the entire details in this lecture. But let me just explain what is the purpose of such statements are line in this. Actually there are lot of useful library function that are available to a C program to give some example, we have functions to find trigonometric value for example, the (( )), angle and so on and so forth. Apart from that we have number of useful function which we start seeing slowly 1 by 1. There are 2 library particularly of 2 syntax in this course, that is the (( )) and the math library. They belongs to the math library. In this program, we are going to use 2 functions for input output namely scanf, which you can see here, this is for reading some input from the user. And printf which you see here, which is used for writing some output to the terminal.

And these are not define the C language itself, but are functions are available to us (( )). And this slide here, include stdio dot h, basically tells that in our program we are going to use some functions, I/O related functions from the (( )). So we are going to the next slide, it is main void and an opening curly brace. It is not important to understand exactly the significant of main voids and this pointing time. We will just say at this point of time, we just say that (( )) the compiler (( )). And the entire program is between this opening brace or the parenthesis and this closing curly parameters or the closing brace. So this entire thing is body of our program as it is called.

Within the body of the program, this first line declare 2 variable and result. So this is known as a variable declaration. So this line say that I am going to use 2 variables in my program, those names are effectively n and result. And both the variables are going to hold integer values. So integer is the type, which is denoted in C by int, there are a number of other useful tags in C as well, which will see later on. Right now integer type is enough for us. So essentially this (( )) is saying that for an variables in our program, their names are n and result. And both are going to hold integer value.

In C as in most languages, before we use the new variables, you must clear that variable. And the program, most of the time, the variable declaration will be first followed by the actual program to perform.

Ok the next line is called the library function called scanf array. In earlier the scanf is a standard library function which is used to read the some data from the user. It has 2 what are known as arguments, this percent d encapsulated present double quote character is the first argument, and the ampersand n is the second argument. And this percent d with the double quote character what is known as a string, we are talk about string in more details, later on in this course. But essentially this particular piece of a information which we are applying to the scanf is saying that in the input we are expecting a user to give an integer value.

And the second argument ampersand n says where we want this value given by the user to be stored in. So essentially this this entire line is saying that we are expecting an integer to be input by the user and whatever integer he gives as a input should be stored in the variable called n. This line we have already seen earlier, this is just an assignment statement if we recall, the equal to sign in C is an assignment statement recall that this line is not stating that the result is equal to 1. On the other hand it is an action saying make the value of result to be equal to 1. So it is an initialization, we are setting the result to one.

If you remember the algorithm for the factorial, computing factorial from the previous lecture, and we also need to decrement the value of n by 1, so this line a while loop. And this while loop says that some piece of code has to be repeatedly executed. And along with the while, we have to say 2 things, how long the computation has to be repeated, so this is the condition n greater than 1, which says how long the computation has to be repeated. It has to be repeated as long as the value of n greater than 1. As soon as the value of n becomes less than 1 that is less than or equal to 0 then the computation has to be stopped. And what has to be done in each step or each iteration of the value, that is again enclosed within this curly brace here, and then corresponding closing curly brace here.

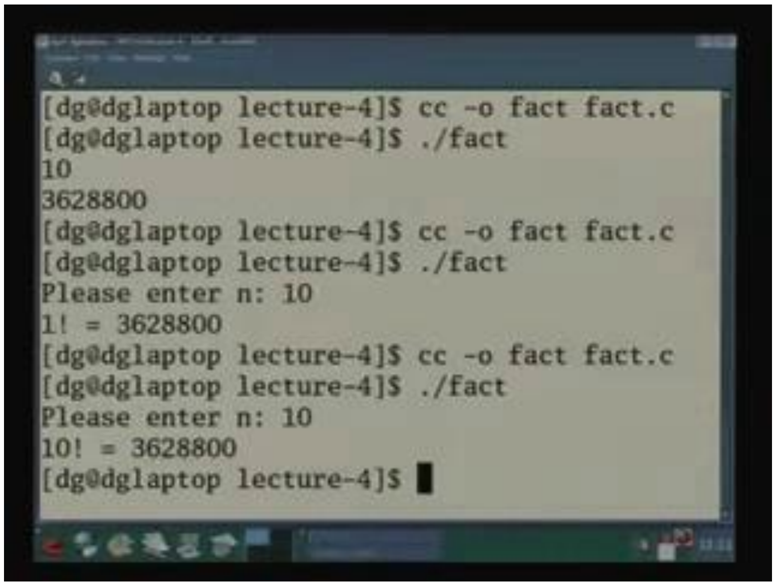
So along with while, we have to give a condition like this and we have to give a single C statement. An assignment is a single C statement as we have already seen, but here as you know we want to 2 things repeatedly in the every iteration of the loop namely assigning result to result into n, and decrementing the value of n, so these are 2 statements that putting them within a curly brace, we make these 2 simple statements is what is known as compound statement. So the result is that entire string is now a single C statement, because it has been now calculated in this opening brace and this closing brace.

And the compound statement itself as you can see the number of simple statements enclosed in the brace here. And finally when the loop is n, as we know the value of result contains the factorial of n, so we need to print that, so we are printing the value of result here in this statement using the C library function again called printf. And again in printf, we have to specify simply what we want to print. You want to print an integer, so that is what this percent d denotes, and this backslash n denotes that after that integer has been printed, the cursor should move on to the next slide. The backslash n stands for the new line character in C. And of course, what integers should be printed is this value, that is the value of the variable called result.

Note that in scanf, when we specify here the value that has read by the user (( )), so the variable has to be specified by the ampersand, whereas when printing, we don’t have to give this ampersand. Right now, we just remember this as rule, we explain later, why this is so. So hopefully this program is correct and we have understood this. Let’s now go ahead and make some minor modification to program to enhance it.

The first simple enhancement that we make to this program, is that when we run this program, we want the user to be stored that the programming expecting an integer as an input.

(Refer Slide Time: 10:02)



```
[dg@dglaptop lecture-4]$ cc -o fact fact.c
[dg@dglaptop lecture-4]$ ./fact
10
3628800
[dg@dglaptop lecture-4]$ cc -o fact fact.c
[dg@dglaptop lecture-4]$ ./fact
Please enter n: 10
1! = 3628800
[dg@dglaptop lecture-4]$ cc -o fact fact.c
[dg@dglaptop lecture-4]$ ./fact
Please enter n: 10
10! = 3628800
[dg@dglaptop lecture-4]$ █
```

As of now the user doesn't get information. Let's compile the current program and see how it is run. You see the program that appears to saw and the user is supposed to try some number here, so let us say we take ten, and the answer is printed, but the user doesn't really know what is going on here, when the integer is expected. So we could give some message saying that we want to input a number, then that will be useful. So how do we do that. Well we already know how to print some message using the printf function, so all we need to do is to add printf statement, before this scanf statement. So that before the program stop, wait for the input, print say message on the screen saying that some integer input is expected.

So let's add a printf statement, note that we have not added the (( )), at the end of this string, because after this message is printed, we don't want to cursor to go next line, we will see in a minute, what is happen when we run this program. Let's make another simple change, when we show the output, it also say this number that we are printing is actually the factorial of number that was input. So we want something what we want to appear is something like 5 factorial is equal to 1 twenty, so to do that we need to print another integer, and therefore another percent d, and after this integer, we want the factorial sign to come and then equal and then this. The result of the the factorial of the given number. And how this printf string contains 2 percent d.

And therefore, as the following argument to the printf function called which would supply to integers. The first integer is n, and second integer is result. So what is going to happen is that when this line is executed, this string will be printed on screen as it is except that the first percent d will be replaced by the value of n, and the second percent d will be replaced by the value of result. So let's save this program, compile it again and see how it is (( )). We see that it is (( )) to enter an integer n, and because we didn't n the string with backslash n, the cursor is on the same line, and it doesn't go to the next line. So let's enter again ten, oops there is an error. So let's go back and correct this error. Why has this error occurred, this error has actually occurred, because) the value of n has decremented in this loop ten times, and nine times, and has finally become 1. Remember that in every iteration of the loop here, decrementing n by 1, so at the end of the loop, the value of n has actually become just 1. We called what is the original value is. We want to print the initial value of n here, so we have to save the old value of n, so we can save that into another variable.

Let's called that initial n. And the initial value of this will be this n. Note that you are using a new variable now. And so therefore, we need to declare this variable as well. The type is again integer, so here is the declaration of initial n, and this n must be replaced by initial n. So all has been done with this change the program, that is write it out again, compile it again, and run it again, ten and you see the ten factorial is equal to the value of ten factorial.

Let's now make some more changes, what happened to this side, when running this program is given negative value of (( )). Let's try that n sorry, let's give minus 2, the result comes out to be 1, which obviously is wrong. In fact the factorial of negative number is not defined at all. So how did we get this result 1, while if you examine this program, suppose the value of n is minus 1, what is going to happen. The result is initialized to 1, initialized n is initialized to minus 2, and then we execute the loop. While n is greater than 1, what is the value of n, the value of n is minus 2, which is not greater than 1, so the loop terminate right away, and these 2 statements do not get executed even 1. And as the result, the value of variable result doesn't change at all, it remains whatever it was before the execution of the loop, which is 1. And therefore, we are getting (( )) result, minus 2 factorial equal to one.

What we should have the program do insert is that the negative value for n is given; it should print the message, saying that the factorial function is only defined for non-negative integers. Well how do we do that (( )), after writing the value of n, we have to check whether or not it is more than 0. If it is more than 0 or greater than or equal to 0, we do all these computation; otherwise, we simply print a message saying that the factorial is defined only for non-negative integers. And do that, we will need to use another statement of C, which is if else context which essentially does something if the condition is true and something else possibly if the condition is false.

So what should be the condition here, the condition should be if and if greater than (refer time: 16:00) equal to 0. So if n is greater than equal to 0, then we want to perform all the computation from here to here. So therefore, this will be the n part of the if statement that is the part that will execute, if this condition is indeed true that if n is greater than equal to 0. Note that n greater than equal to condition written in C, greater than sign followed by equal to sign and just like the while loop, here along with the condition we are supposed to give a single C statement, which says what should be done. If this single C statement, which would be a executed if the condition actually evaluated is true. And since we have number of C statement here, we need to include these statements.

Again within braces, to make a compound statement out of these. And if the value of n is indeed less than 0, but it is not greater than equal to 0 then we want to execute something else, which is just printing a message saying that factorial is defined only for non-negative integers. Note that in the else part, there is only a single C statement that we want to execute, namely this printf function. And therefore, we don’t need to put this within braces, even though, if it is put in within the braces, it could be fine. So this program will work, but I am not quite satisfied with this, because even though this tactically correct and it will actually work, but this is hard to read ah.

Note that so far we have been following a style of the C program, which we have not (( )) discussed, but let me now illustrate, what I am trying to say. These 2 statement constitute the body of the while loop. And because these 2 statements are (( )), to the left, to the right I am sorry by the certain amount that is being start here; whereas the while started at this location, so it was quite clear that these statements are within the while loop, this is called indentation and this is very necessary to make our program readable. Similarly note that here, since this printf is within the else, we have started it at a horizontal location; after the place where the else. So what is the wrong with this program, well if you see, this entire sequence of statement, is within the if statement, which is the kind of proper statement for all these statement.

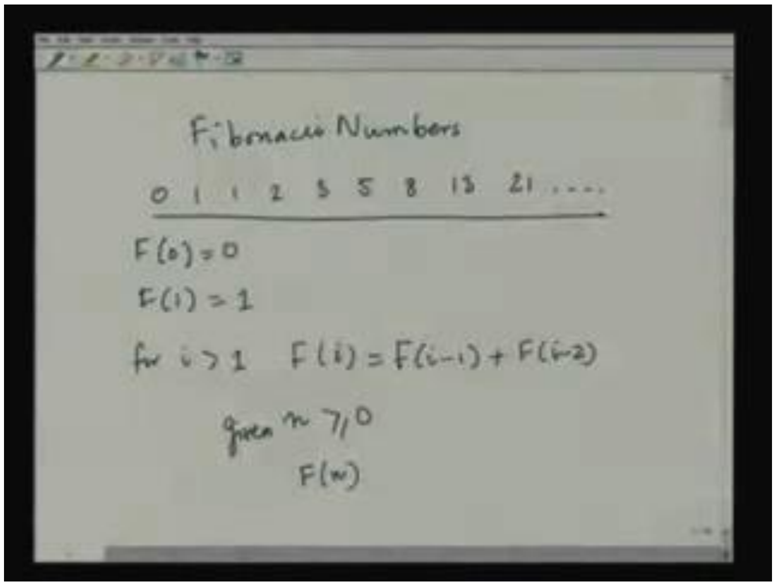
And therefore, these should start further towards to be right, then the program will be much more readable. So need to insert some blank spaces here, so that this not moves to the right. In C actually liberally you know in the (( )), statement with number of spaces and blank lines as we wish, but a certain programming style as I said is (( )) that the program is readable. Instead of inserting number of spaces, it is usually convenient to just insert a tab character, this is left top area of the keyboard, so let’s just press the tab character here. And as you see a tab character is roughly equivalent to number of spaces.

So this statement is now gone to the left, to the same thing is with this statement. With this statement, what about these 2 these are actually within void, they should be further intended to the right, so 1 more tab here, 1 more tab here. This break is the handling of the void, so should be at the same level (( )). And this printf again is at the same level at the while. It is directly if n, so it should come here. Now you can see the program is much more readable. It is clear that these 2 statements, for example of are printf. The while is (( )) f, but these 2 statements are within the while (( )). So let's now save this program again, and try to compile it, and hope that there are no error.

Let's try with a normal number non-negative integers first, 120 was prime. Then try with this negative integers, and we get what is the expected namely a message saying that the factorial function is defined only for non-negative integers. So there are 2 things that we have learnt by during this simple enhancement program, the first which is very important is that the program must check that all the inputs given to the programs by the user are within the acceptable range. Because if the inputs are not within the acceptable range, the program is (( )) behave unpredictable fashion. And the output is slightly to confuse to the user.

So therefore, we have to check that the value of n, indeed greater than equal to 0. The second thing we have learnt is importance of (( )) fashion. And we have seen how this makes the program much more readable and in all programs that we write from now on therefore, we will consistently use the indentation types that we have introduced in this lecture. Now we will take another example problem, and try to first develop an algorithm for it. And then writing a C program for solving that problem. So this problem is very famous 1. And it has do with what are known as Fibonacci number.

(Refer Slide Time: 22:10)



Fibonacci Numbers

0 1 1 2 3 5 8 13 21 ....

$F(0) = 0$

$F(1) = 1$

for $i > 1$ $F(i) = F(i-1) + F(i-2)$

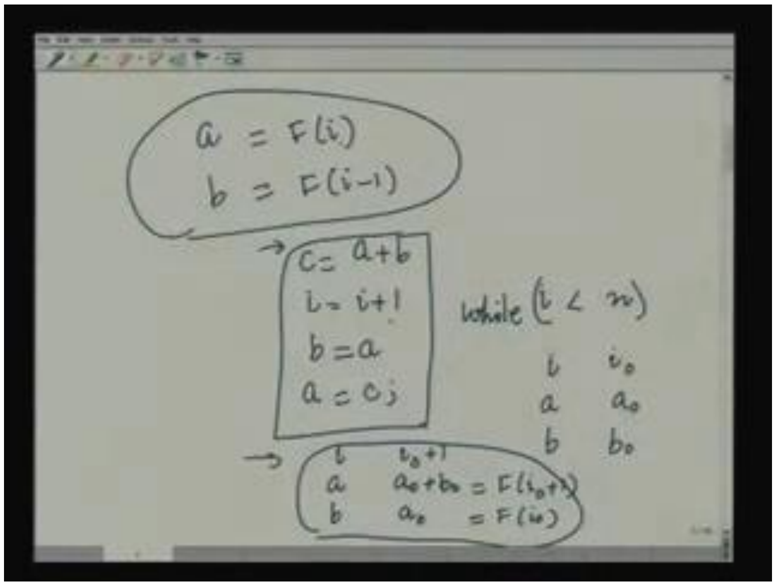
given $n \geq 0$

$F(n)$

[noise] The Fibonacci sequence of numbers are defined as follows. These are the numbers in the sequence. And you can observe in this sequence, any number is the sum of previous 2 numbers in the sequence. So we can define this mathematical terms as follows. So let say F of I denotes the i eth Fibonacci number. So F of 0 is 0, F 1 is 1. And for i greater than 1, F of i is nothing but F of i minus 1 plus F of i minus 2.

So let us see given the expression, we are given an integer n greater than equal to 0. And we want to compute the value of F n that is the n eth number, counting from 0 in the sequence that we just saw. So how do we go about solving this problem. So we can see before solving this problem, what we need to do is to remember the previous 2 numbers in the sequence, and then if we add up these 2 numbers, we get the next number in the sequence.

(Refer Slide Time: 23:44)



$a = F(i)$

$b = F(i-1)$

```
c = a+b
i = i+1
b = a
a = c;
```

while $(i < n)$

i	i_0
a	a_0
b	b_0

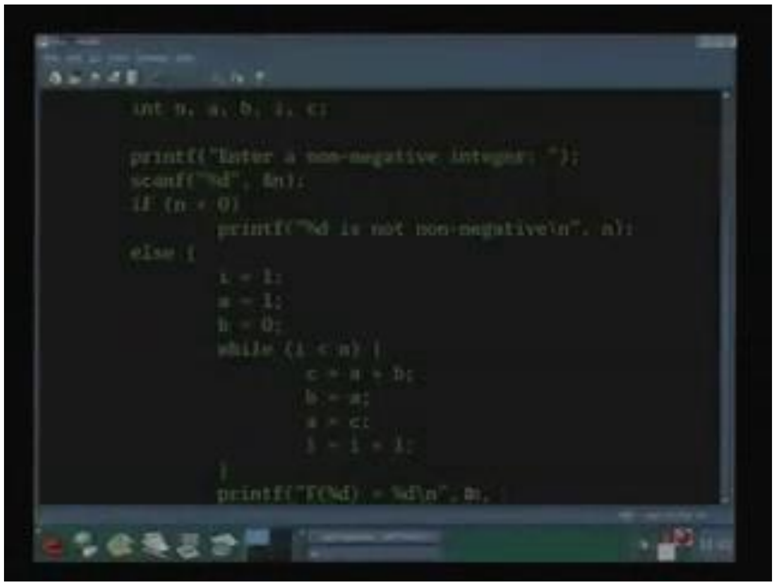
i	i_0+1	
a	a_0+b_0	$= F(i_0+1)$
b	a_0	$= F(i_0)$

So let us say at any point in times, you want to maintain the value of a as the value of F i, and b let us as the value of F i minus 1. So suppose at any point in time our program for some value of i less than or equal to n. The value of a is same as the value of F i, and the value of b is same as value of F i minus 1. And then what do you do, in the next step, what will do is to use these 2 values, to compute the value of F i plus 1, which will simply the a plus b. So let's write c as a plus b; and in the next step, we still want these variants to hold that is a is F of i, and b is F of i minus 1. So what will need to do is increment i is i plus 1, and the value of a will be c, and the value of b will be the whole value of a. But there is a problem, where is the old value of a, we have already overwritten with c, right, so you can do these 2 in the reverse format.

Let's just erase this little bit, and write. So in every step of the program, this is the computation that we are going to perform. As you can imagine, this will be done repeatedly as long as, i is less than n, so this will be repeatedly as long as i is less than n, because as soon as i becomes equal to 1. Then this variant tells us that the value of F n is nothing but a, so we stop doing this computation at that point in time. So note that, if that this point in time, it was the case that a is equal to F i, and b is equal to F i minus 1. Then at this point in time, it could also be the case that a is equal to F i, and b is equal to F i minus 1, why because we have (( )) i by 1, but the number a and b now are the old values of a plus b and the old values of a respectively. That is if the value of i, i 0 to begin with, a was a 0 to begin with, and b was b 0 to begin with, then after the end of this computation, the value of i will be i 0 plus 1, the value of a will be a 0 plus b 0, because a is assign c here, which is a plus c. And the value of b will be b 0, no the value of b will be a 0.

And you can observe that a 0 plus b 0 is simply F of i 0 plus 1. And a 0 is of course F of i 0, why a 0 plus b 0 is F of i 0 plus 1, because a 0 is F of i and b 0 is F of i minus 1 and by the definition of the Fibonacci series, F of i plus F of i minus 1 is F of i plus 1. So at the end of the this particular computation, we will again end of these value will study for the invariance that at any point in time, a is equal to F i and b is equal to F of i minus 1. Based on these idea, we can now easily develop a C program for computing the n eth number in the Fibonacci series using that 2 we have already seen by writing the factorial program. So let’s go and do it now. Ok So let’s now write a C program based on an algorithm that we have just discussed to compute n eth Fibonacci number.

(Refer Slide Time: 27:43)



```
int n, a, b, i, c;

printf("Enter a non-negative integer: ");
scanf("%d", &n);
if (n < 0)
        printf("%d is not non-negative\n", n);
else {
        i = 1;
        a = 1;
        b = 0;
        while (i < n) {
                c = a + b;
                b = a;
                a = c;
                i = i + 1;
        }
        printf("F(%d) = %d\n", n,
```

So, let’s start a editor window. Name of the program is going to fib.c. So let’s start with the command explaining what the program is going to do. So the program to compute the n eth Fibonacci number. And remember that this is the end of the command start and forward slash. And again we are going to use the scanf and printf function from the standard C library. So we include stdio.h as the (( )), here comes our program. We intend all statement remember by 1 dash space, because these are all part of the main program. And first thing if you remember that you have to declare the variable (( )), we going to use variable n, n will whole the number that the user has given to us. A and b will use as in the algorithm, that is a will hold the i eth Fibonacci number F of i, b will hold F of i minus 1, and i itself of course is the current index of the Fibonacci number that we are computed. And c remembered we use for computing the term of a and b. So these are variables that we are going to use.

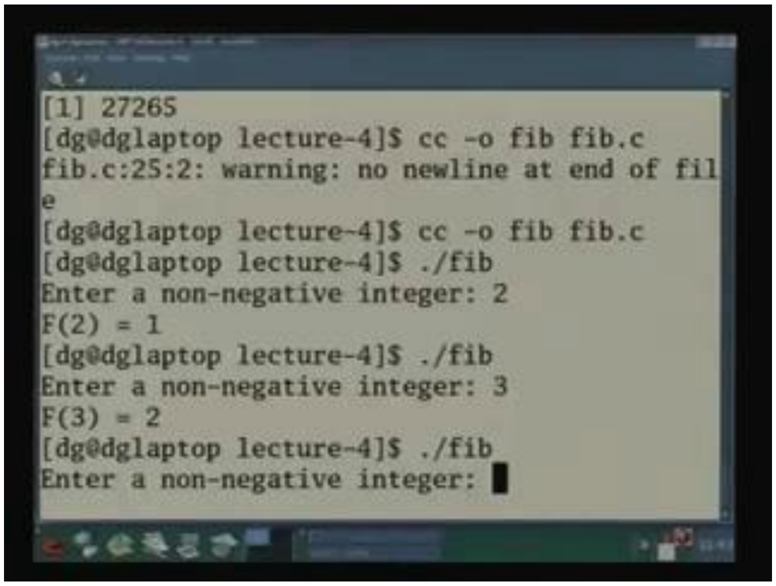
So what do we next, let's again from the user that a no negative integer is expected using printf statement. And then we read the value of n as before right. And again we need to check whether n is less than 0 or greater than equal to 0, because if n is less than 0, that is not a valid input; and you must (( )) the program telling the user that you must enter a non-negative integer. So let's check whether n is less than 0, if n is less than 0, then all we want to do is to print a message saying that that the number you have entered that is n is non –negative integer. So let's this write not non-negative. And since we are using percent d here, you must specify what actual value from the d replace should replace the percent d and of course z.

So far example, if the user enter minus 2, this printf statement will result in output saying minus 2 is not non-negative ok. And since this is the single statement that we want to execute, if the value of n is greater than 0, we don't need braces here. Otherwise, that is if n is greater than equal to 0, you want to do a computation of F n, so that will require multiple statement, and so therefore we need to enclose them in braces and makes in a compound statement for reference let's write the definition of the function F here, as the (( )) numbers. So we are ready to start. So we need to initialize the integer a, b and i, the value n has already given; the c, we don't need to initialize, the variable C, we don't need to initialize, because that will replaced space of the value of a and b in every iteration of the loop. But a, b and i have initialized and remember, the constant we have that we must ensure that at any point in time, before starting any iteration of the while loop, the value of a must be equal to F of i, and b must be F of i minus one.

So let's initialize i to 1, a to 1, which is F of one; and b to 0, because that is F of i minus 1, there is 1, i minus 1 is 0 and F of 0 is 0. So therefore b has to be displayed as 0. Note that all these statements are within the else. And therefore they are intended to the right by some place after the else. And next comes our loop while i is less than n, we need to do certain computation. Why we have to do them only as long as i is less than n, because remember that if the i is becomes equal to n, then we have already found the answer, the answer is a. So by (( )), remember what we need to do, compute the c a plus b. Then assign b to b to a, and a to c; and of course increment i, i plus 1. And that's the end of the body of the while loop.

And now what we do, this is again same within the else part, this is the computation we are doing if n is less than, if n is greater than equal to 0. Now we have found F n, so we want to print the value of F n, so just print again using the print statement F of n is equal to that F n we have computed, and what should go here, the first percent d should be replaced by the value of n. Which in this program, we have not modified anywhere else as you can see after reading this from the user. So we don't need to save it's (( )) variable, we just use n. And this should be replaced by the answer which is F of n, and F of n remember now is equal to a, because i has become equal to n, so that's all we need to do. And this is the end of the else block, and this is the end of the main program, so let's save this and compile this.

(Refer Slide Time: 34:24)



```
[1] 27265
[dg@dglaptop lecture-4]$ cc -o fib fib.c
fib.c:25:2: warning: no newline at end of fil
e
[dg@dglaptop lecture-4]$ cc -o fib fib.c
[dg@dglaptop lecture-4]$ ./fib
Enter a non-negative integer: 2
F(2) = 1
[dg@dglaptop lecture-4]$ ./fib
Enter a non-negative integer: 3
F(3) = 2
[dg@dglaptop lecture-4]$ ./fib
Enter a non-negative integer: 
```

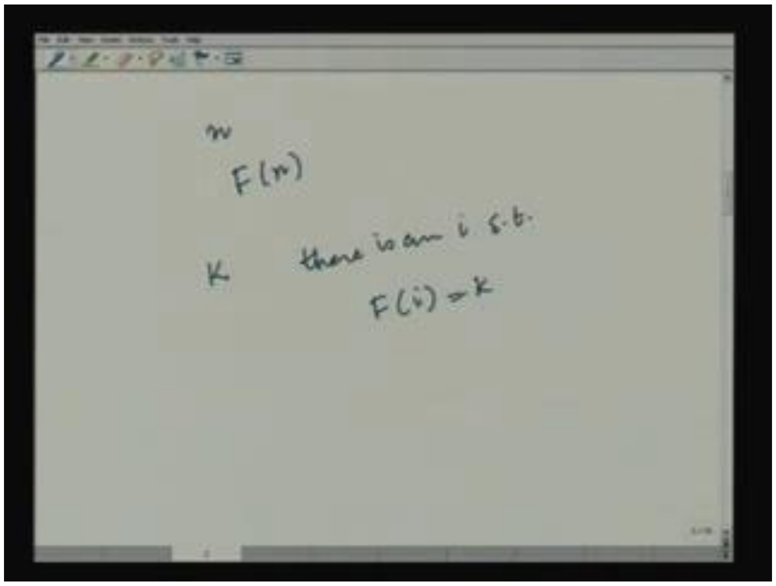
So this warning based on because adding new line at the end of the program, not that if (( )) too much, so let's compile it again. So let's enter the 2, F of 2 is 1, is that correct, yes it is, because F of 2 is F of 1 plus F of 0, which means 1 plus 0, which is 1. So let's try for some more values, 3 should be 2. 4 should be the value of sum of last 2 values, this 2 plus 1, that's correct. 5 should give us 3 plus 2 is 5, it's working. So let's also try with border cases, border cases are ones which loop will run at most 1 for not at all and so on and so forth. So let's try with minus 1 – a negative integer. We get the correct method, minus 1 is not non-negative. Let's now try with 1, yes this is right answer. Let's try with 0, that's wrong, because F of 0 by the definition around is 0, not 1. So what went wrong in the program.

So suppose, n was equal to 0 then what happen. It came here, initialize i to 1, a to 1, b to 0; while i less than n, i is less than 1, and the value of n is 0, so i not less than n. And in fact it is not even equal to n. So here in the print statement, we assume that at the end of the loop, I will be equal to n, and therefore F of i will be equal to F of n, which will intend the equal to a. But here is the special case, in which the value of i has not become equal to n, at the end of the loop. In fact, I here is 1 while n is 0. So in this case, we should print not the value of a as the answer, but the value of b is equal to 0. So therefore, this print statement has to be checked. So we want to the result is a, only if n is equal to i; otherwise the answer is b. So we need another if else, so let's add that if n is equal to i, we have it's the first time we using the equality of greater in C. Note that this is different from the assignment, which is the single equal to and this is equality or comparison between n and i, whether n is actually equal to i or not.

Say n is equal to I, then the answer that you are printing is indeed correct that is f of n is equal to f of y is equal to a. But if it is not the case, and this number will happen only when n is 0. Say if n is 0, the answer of course, is 0. So you can just print this, or of course, we could have taken them longer form of printf, F of percent d is equal to percent d backslash n, n, b not a, or comma n, 0. But because we know that in this case, n is going to be 0, and (( )) is also going to be 0. We can just simplified like this. And that should follow our problem. So let's compile this program again.

And let's write this time with 0, it was fib 1, 2, 3, 4, 5, see it was for all possible integers value. So the program is finally correct. So in this lecture, we have add a general introduction to the basic element of the C programming language. In the next lecture onwards, we will start looking at the elements of C in a little bit more details and somewhat more formally. But before we end today's lecture, I would like to leave with an assignment that you should try out in your own, very similar to the problem that we have already solved in this lecture.

(Refer Slide Time: 38:49)



Based (( )) Fibonacci number, so the problem that we solved was given n, we try to compute the value of F of n, but suppose we are given a K, and we want to write a program, which says whether or not K is a Fibonacci number, that is whether there exist an I, such that F of i is equal to k. And if such an i does exist, then the value of i should be printed, otherwise the program should say that K is not a Fibonacci number. And to solve this problem, there is a little hint that you might want to use, and the hint is that the Fibonacci number sequence is (( )) increasing, that is every number except in the beginning, is more than in the previous number in the sequence. So the essential idea to solving this problem is to generate

Fibonacci number 1 by 1. Still we get numbers which will either equal to k or more than K. If we get the number is equal to k, then we know that the Fibonacci number and by keeping track of i, we should be able to say, which Fibonacci number it is that is corresponding i, that F of i is equal to K. On the other hand, if we likely go from a number less than k to a number greater than K, then clearly says K is not a Fibonacci number, and then the program should says that the given value is not a Fibonacci number.