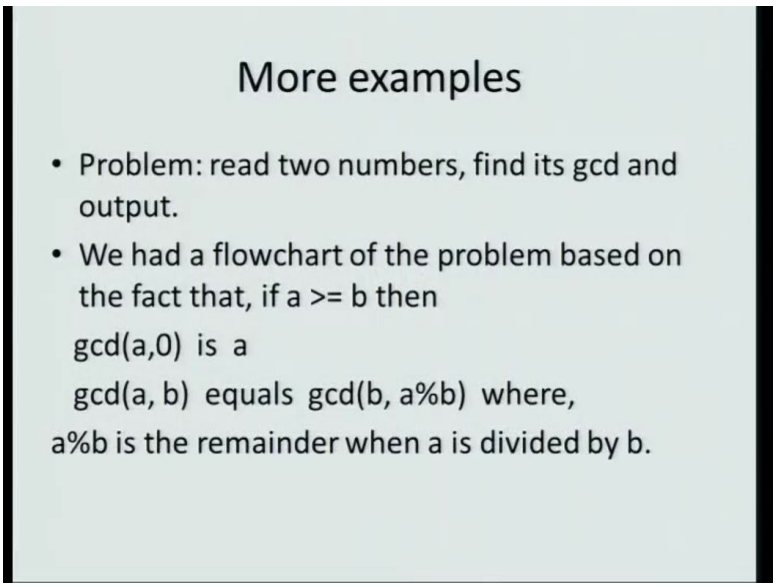
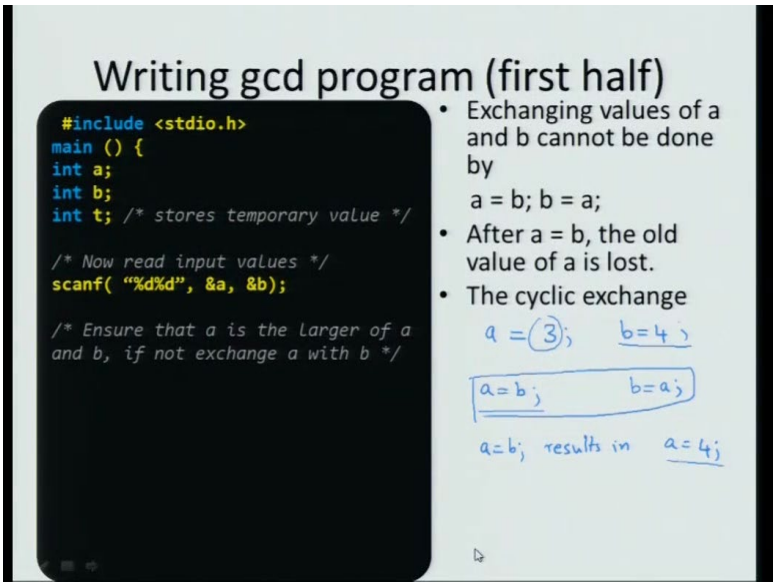
We will see a few more examples, because loops are really important. Let us go back to the first problem that we discussed, which was the problem of computing the greatest common divisor of two positive numbers. So, the problem is to read the two numbers, find their GCD and compute the output.

(Refer Slide Time: 00:37)




Now, we had a flowchart of the problem based on the fact that, if a >= b, then GCD(a,0) if b is 0, then GCD(a,0) is a. Otherwise, GCD(a,b) is the same as GCD(b, a%b), with a % b is a % b is the remainder, when a / b. So, let us now try to write the program in C using a while loop. So, we have to do a few preliminary things.

(Refer Slide Time: 01:12)




So, let us call up the first half of the program. In the first half I declare three variables, a, b and there is another variable t, whose need we will see right now. But, let us say that I need an extra variable for now, let us just take it on faith. So, what I will do is scan two variables a and b. Now, recall in the GCD equation that we saw right now, we assume that a >=b. Now, what if the user is unaware of his condition and enter the lesser number first.

So, he just entered the numbers in such a way that, a < b. So, we need to correct that, we need to make sure that a is the greater number. So, we need to exchange the values of a and b, if it is true that a is less then b. Now, how do we do this? So, the first thing to note to try will be to say that for example, if I say that let us say a was 3 and b was 4. And suppose, I just said a = b I want to exchange the values of a and b. I just said a = b and b = a.

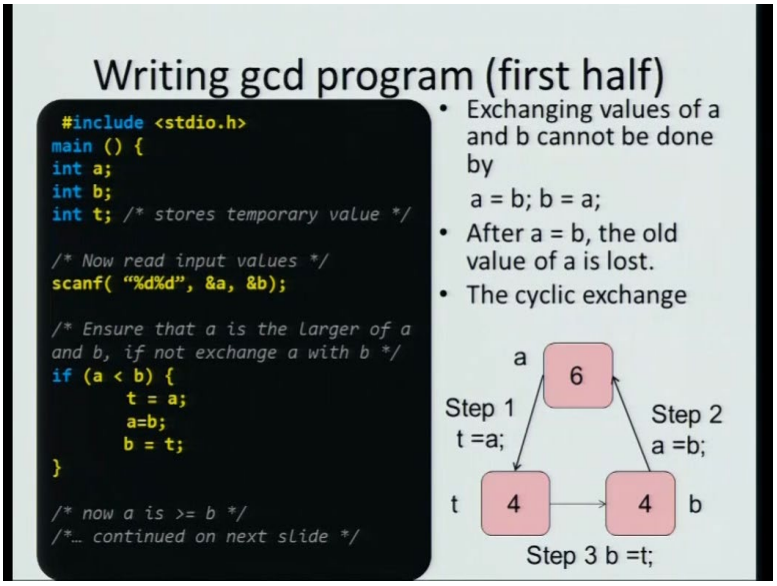
What will be the effect of this? Note that, this is the assignment statement. So, after I execute this line, b is 4 so, a will be 4, a = b results in a equal to 4. After the execution of this line. So, what this situation that we will have is that b equal to 4 and a equal to 4. And we will have no memory of, what was the original value of a? That is lost. So, it is just simply lost.

So, this idea that we can exchange two values by just writing a equal b, b = a does not work. So, what is the correct way to do it? So, we have an idea known as the cyclic exchange and this is a really neat idea. The idea is that… So, how can I motivate it? Let

us say that you have two rooms and these two rooms are full of steps. And I want to change the contents of the first room to the second and the second room to the third.

One way I can do it is that I will move the contents of the first room to a different room. So, have a temporary room and then copy the contents of the second room to the first and copy the contents of the third room to the second. So, this is a very nice intuition and it almost is similar to what we need to do.

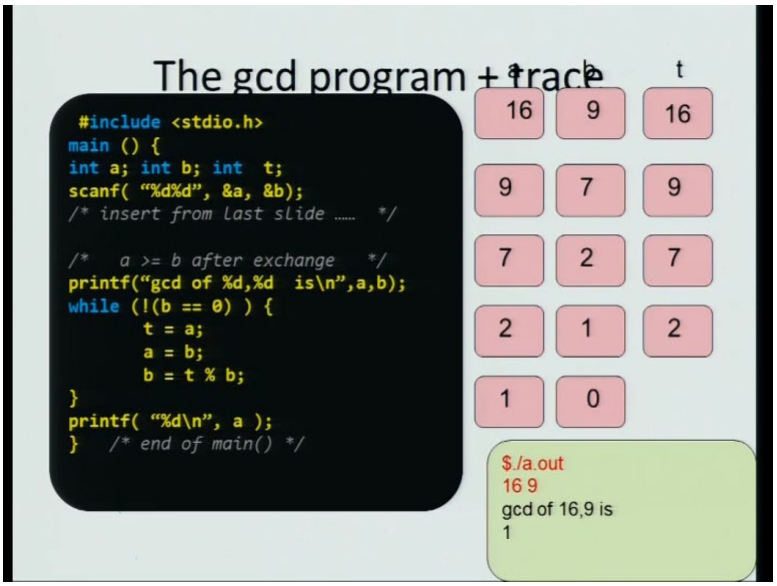
(Refer Slide Time: 04:28)




So, we have two variables a and b that we need to swap. So, one thing we can do is, I will keep a third variable t. First what I will do is, I will copy the value of a to t. So, now I have a backup copy. Now, then I will say a = b,. So, at this point the value of b will be copy to the a, So, now, a is 6, b is 6. But, still we have a memory of what was a before? Because, the old value of a was told in t.

So, in order to complete the routine, all you need to do is to copy the value of t to b and that can be a complete by the equation by the assignment statement b equal to t. So, this idea is known as cyclic exchange. So, recall the physical intuition of swapping the contents of two rooms which are almost full. You have a third room and you move the contents of the first room to the third room, move the contents of the second to the first and move the contents to the third to the second. So, this is similar to what we did. In the case of physical contents, we cannot copy, in the case of variables, we can copy.

(Refer Slide Time: 05:52)




So, now let us complete the GCD program. We have just done the first part of the program which is to ensure that a is actually the greater number. If it was not the greater number, you swap or exchange. Now, So, after exchange we have ensure that a >=b. Now, we have to write the main loop for the GCD function. So, just by translating the flowchart, what we will do is while b is not 0. What you do is, you say that store the value of a and t. Assign a to b, a = b and b becomes t % b.

Recall the equation was written as follows that. So, recall that the equation was GCD(a, b) is the same as GCD(b, a % b). So, when you assign a to b, the old value of a is lost and we can no longer do a % b. So, the way to do that is, you use the idea of temporary variable, store the old value of a and t, before you do a = b. So, that finally, a % b can be done by t % b. I do not want the new value of a, I want the old value of a.

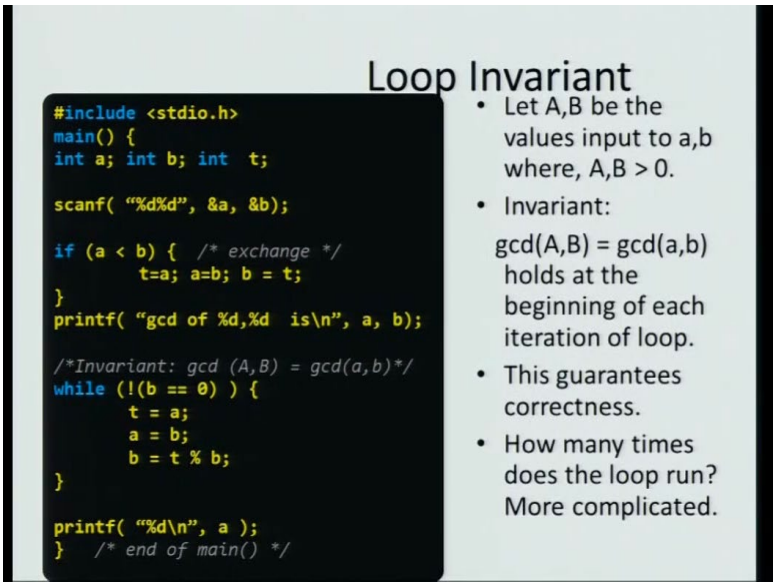
So, let us just trace the execution of this program. Let us say that I scan two numbers a and b and the user was correct in entering it. So, he actually enter the greater number first. So, we have 16 and 9, a equal to 16 and b equal to 9 and t is undefined. So, after you read the numbers, you just say GCD of… After these you enter a message which is printed message which is the GCD of a and b is,. So, GCD of 16 and 9 is and then you enter the loop.

So, in the initial execution of the loop you have t equal to a, which will store t equal to 16, a = b, which is a will become 9. And then, but you want to compute the modulo 16 % 9, but 16 was lost in a, because a is not 9. So, you have do t % b,. So, 16 % 9 which is 7.

So, you go back to the while loop and then you see that b is not 0, b is 7. So, you enter the loop again. T is equal to a,. So, t is 9, a = b,. So, a become 7, b becomes 2.

Again b is not 0, So, you enter the loop again. So, t is 7, a equal to 2 and b equal to 1. Again you enter the loop, b is not 0, t is 2, a becomes 1 and b becomes 0 at this point, you exit the loop and at this point a that we ended with is the GCD of these numbers. So, 16 and 9 are relatively prime. Therefore, their GCD is 1. Now, let us think a minute about what is the loop invariant in this program?

(Refer Slide Time: 09:42)




What is it that? We have a central while loop which computes the GCD. What was the invariant in that loop? So, for this I will just introduce a slight notation which makes it easier to discuss this invariant. So, let capital A and capital B be the original numbers that I input. And little a and little b represent the numbers which are involved in the loop. So, capital A and capital B are the original input and the invariant that I have is that at every stage, the GCD of the original inputs are the same as the GCD of little a and b.

We call that little a and b are the loop variables involved in the loop. So, little a and b keep changing through the loop, whereas, capital A and B are fixed, they are the input. So, the invariant that I have is that every time you enter the loop, the GCD of the input where the same as the GCD of the variable. Now, this guarantees the correctness. Because, when you exit out of the loop, you will correctly compute. You exit out of the loop, because b equal to 0, and by the original equation, you know that when b is equal to 0, a is the GCD of a and b. So, this guarantee is correctness.

Now, you could also ask other questions like, how many times has the loop run? And this question is big complicated, because you have to compute it based on the input numbers. So, such questions are of interest to computer science. But, we will not going to computing the efficiency or the performance of this code. But, it is also a very crucial question.