

Problem Solving through Programming In C
Prof. Anupam Basu
Department of Computer Science & Engineering
Indian Institute of Technology, Kharagpur

Lecture - 20
Implementation of Loops with for Statement (Contd.)

We were looking at new construct for building up loops in C language and that construct is for construct.

(Refer Slide Time: 00:26)

for Statement

```
for (initial; condition; iteration)
    statement_to_repeat;
```

```
for (initial; condition; iteration) {
    statement_1;
    ...
    statement_N;
}
```

All are expressions.
initial → expr1
condition → expr2
iteration → expr3

```
fact = 1;
for (i = 1; i <= 10; i++)
    fact = fact * i;
```

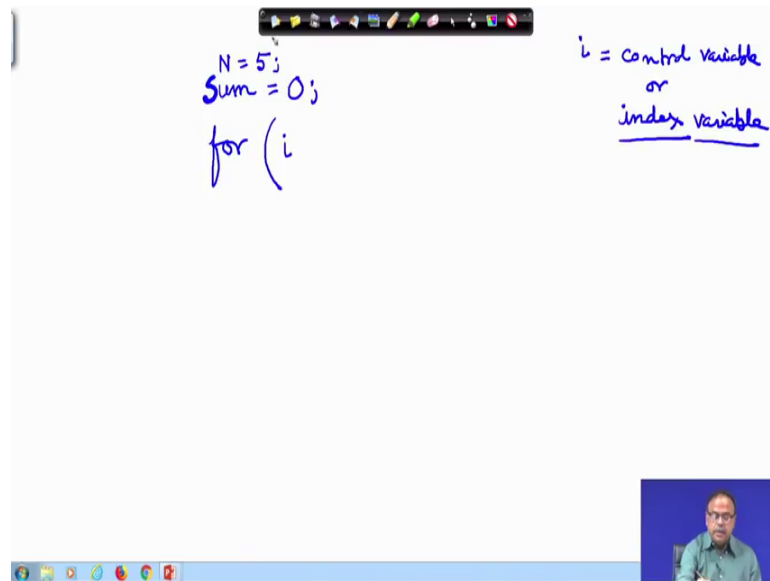
No semicolon after last expression

$(i+p) * 2 < 25$
 $i = i + 5$
 $i = i * 5$

So, let me first write a simple program again the same program that we are writing for reading for finding the sum of 10 numbers. So, just I am writing and you try to follow what the meaning of this program is, then we will go and further explain it. My intention is to read n numbers and find their sum the simple thing, how can I write that.

So, I put a variable again sum is 0 and I have read some value n whatever that value is.

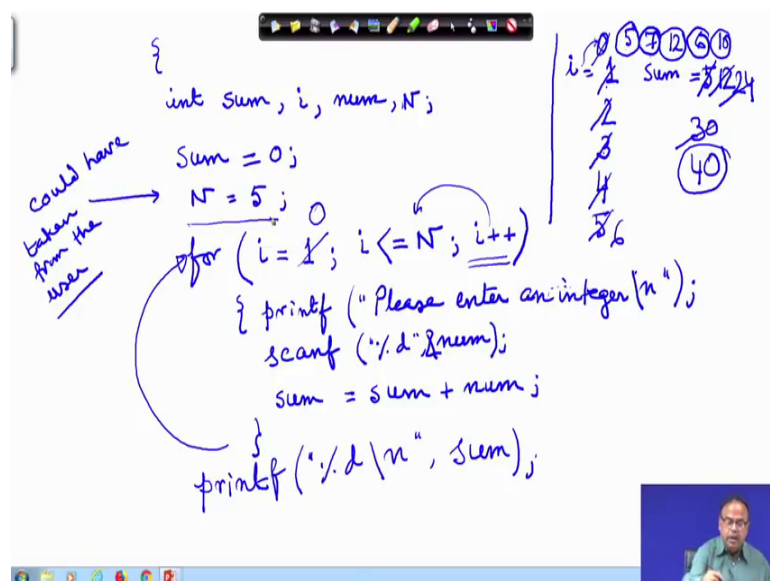
(Refer Slide Time: 01:04)



Suppose say n equals 5 user has provided me or I can initialize it to 5. So, 5 numbers I will read and add. Now, I write for i. What is this i? i is an control variable or we also call it as very common name of this is an index variable, index variable. It need not be i it can be anything, but an integer variable. So, that has to be declared at the beginning.

Si, I can declare let me start with normal declarations int, some program has start at main int sum i and the number that I will read, I will see if I need any other variables later and also n sorry and also n.

(Refer Slide Time: 02:32)



And here I do `sum = 0` and `n` equals to 5. Now, this thing I could have read from the user, could have taken from the user the value. How could I take it from the user? By doing a `scanf`, `scanf` `printf` please enter the number of numbers you want to add then `scanf` `percentage d` `&n` in that way I could read this. But here for the sake of saving time I am just initializing it `N` is 5.

Now, what I do for `i`, this `i` equals 1 `semicolon`. So, this one complete statement `i` less than equal to `N` `semicolon` `i` plus plus. What does it mean? We will see what it means then what would I do I will `printf`. Please enter number let us say integer I call it please enter an integer `scanf` `percentage d` `num` `&num`. Then `sum` assigned `sum` plus `num` that is all.

Here I do not have to increment this control variable why, after I add the number that has been read here after I add that with `sum` I am going back to this for loop and where am I going to, I am now doing this operation that is incrementing `i`. So, `i` will become it was `i` was 1, so `i` will become 2 and then again I add the number. After I make `i` equal `i` increment `i`, initially `i` was 1 here, so what is happening. Let us look at what is happening to the control variable of the index `i` was 1, I do it once. So, the `sum` becomes number suppose the number was 5 suppose the number was 5 that was entered. So, the `sum` becomes 5, I increment `i` so `i` becomes 2, I check it is less than 5 less than this `n`. I again read another number suppose I read 7 as the second number.

So, `sum` becomes 5 plus 7 12 and then I go back here increment it, it becomes 3 all right. I check again after doing this I come here check it is still less than 5, so less than `n` I again do this I need another number suppose it is 12. So, 12 plus 12 it will be 24. Here increment this, it becomes 4, `i` still go over there still it is less than 5. I read the next number which was 6 say, so `sum` becomes 30 here. Then `i` is incremented to be 5, `i` again come here and check that is less than equal to `n` true. So, I come here I read another number say 10 `sum` becomes 40, come back here `i` become 6, `i` come at this point this condition is not true therefore, `i` will come to this point where maybe I will be writing something like `printf` `percentage d` `backslash n` `sum`. So, the `sum` will be printed at this point 40.

Now, you see I have read how many numbers 5 7 12 6 10, 5 numbers and that is what I wanted to do. Now, I can look at this. Now, I could have done several this thing in a

different way also. For example, if I had just instead of this i equal to 1, I make it I initialize this to be 0 all right, I initialize i to 0 and keep everything the same. What will be changing? N is 5, so here i will start not from 1, but from 0. So, first will be 0 then it will be 1, then it will be 2, then it will be 3, then it will be 4, then it will be 5. So, ultimately how many numbers would I read? 6 numbers. But I was actually trying to read 5 numbers. I am sure you are confused. So, let me show it through another example.

(Refer Slide Time: 10:30)

Read 3 numbers

```
for (myindex = 0; myindex < 3; myindex++)
    scanf("%d", &num);
```

	myindex	num	
0	0	15	✓
1	1	5	✓
2	2	14	✓
3	3	X	X

Suppose I read, I want to read 3 numbers. If I write it in this way for i, now by the way it need not be i, I could have written declared it properly and I could have taken any variable to be my index, I could have saved that it is my index But the only constraint is that this must be an integer variable. My index is 0, note the semicolon here because this is one statement then I write my index less than equal to 5 say less than equal to I am trying to read 3. So, my index is less than equal to 3 semicolon again and then my index plus plus and I just read the numbers. So, I am dropping off the printf just writing scanf percentage d and num.

Now, I since its only one statement I can simply remove this parenthesis. Now, let us see I wanted to read 3 numbers. Now, what will happen? Here I write my value the value of my index and the value of num. My index has been initialized to 0 and I check with my index my I check with this statement, it is less than equal to 3. So, I read the number

suppose the number is 15, I increment my index. So, my index now becomes 1, my index becomes one still less than equal to 3, I read another number 5.

Next I come here. So, I do this and come here and my index becomes 2, still less than equal to 3, I read the number say 14. I again come here my index becomes 3 less than equal to 3, I read another number 5. I come here my index becomes 4 I compare this condition is not satisfied. So, I come out of this loop.

But in the process how many numbers have I read? I have read 4 numbers 1 2 3 4. But what was my intention? My intention was to read 3 numbers. So, where did I go wrong where did my logic go wrong. This is what as a programmer you must be very careful and cautious about. Where did I go wrong? You can say that I have gone wrong in either of the 2 places, one is I could have simply initialized my index not with 0, but I could have initialize it to 1. In that case what would have happened, first this part we forget we will start with my index one, read one number less than 3. So, then become 2 less than 3 I read the other number increment it, it becomes 3 still less than equal to 3 I read this number, whenever I come to 4 then my index becomes 4 this condition is violated. So, I will not be read in this number it will be all right.

Otherwise, another thing I could have done what could I have done? You must have discovered it by. Now, that suppose I had kept my index to be 0, I decide no I like this circular figure 0 very much, so I keep it like that. Then what should I do in order that I can still be logically correct? I would have changed this condition from less than equal to to less than, then let us see what would have happened. Then also my index starts with 0 I read one number incremented. So, it becomes 1, I check for the condition my index is still less than 3. So, I read the other number 5 my index is incremented 2 still less than 3. I read the other number 14 fine as soon as I after I read I make it 3 incremented and I check the condition. Now, it is no longer true it is not less than, but equal to, but less than equal to is not my condition my condition is less than therefore, I will not read the forth number.

So, this is a point where often people make mistakes while writing for loops. So, I encourage all of you to very carefully study this we look into the for loop a little bit more. So, we have seen the initial. So, I have given you some example one point that is very important you must have noticed while I was writing this that there is no semicolon

at the end of this statement why, because the entire for statement has not ended here it is going on for this period to this, that is the end of the statement. So, no semicolon is given for this for conditional part all right. And here all these are expressions you can see that this I am sorry what happened. You can see that this is an expression sorry this is an expression, this is another expression, this is another expression.

Now, since these are expressions it can be very general for example, I could have written initialization is fine, but here I could have done something i plus p times 2 is less than 25 that is very much valid and here also I could have done i assigned I could have written i assign i plus 5 or I could have written i assigned i times 5 anything. This is a modified modulator I am changing the condition and changing the index variable and then testing that index variable with respect to a condition. Only point that you should remember is that that this must be integer variable.

(Refer Slide Time: 19:44)

• How it works?

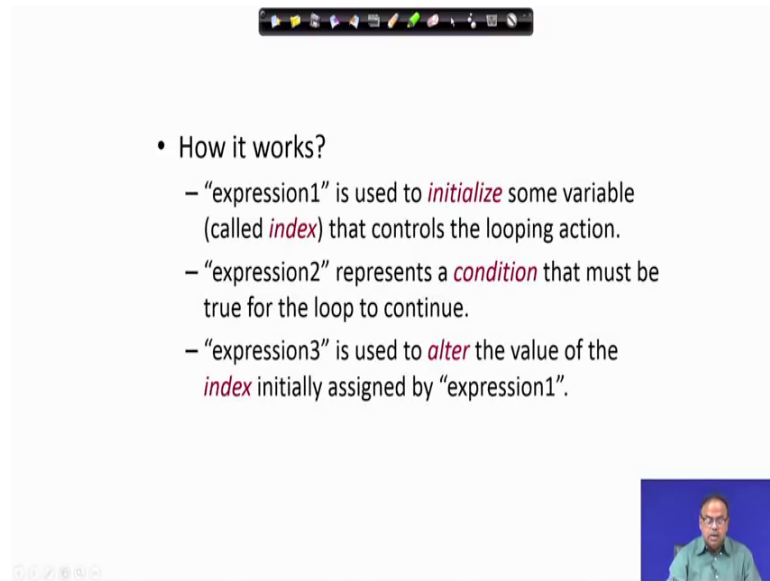
- “expression1” is used to *initialize* some variable (called *index*) that controls the looping action.
- “expression2” represents a *condition* that must be true for the loop to continue.
- “expression3” is used to *alter* the value of the *index* initially assigned by “expression1”.

```
int digit;
for (digit=0; digit<=9; digit++)
    printf ("%d \n", digit);
```

$(i=1; i \leq 5; i=i+2)$

So, how it works? The expression one that is say typically i assigned 1, the expression 1 is used to initialize some variable call index that controls the looping action. Expression 2 i less than equal to 5 represents a condition that must be true so that the loop continues. An expression 3 say i assigned i plus 2 all right that is the bracket, i assigned i plus 2 or i plus plus as you are seeing till now, they are used to alter the value of the index initially assigned by expression 1. We have seen this, so not much to worry about it.

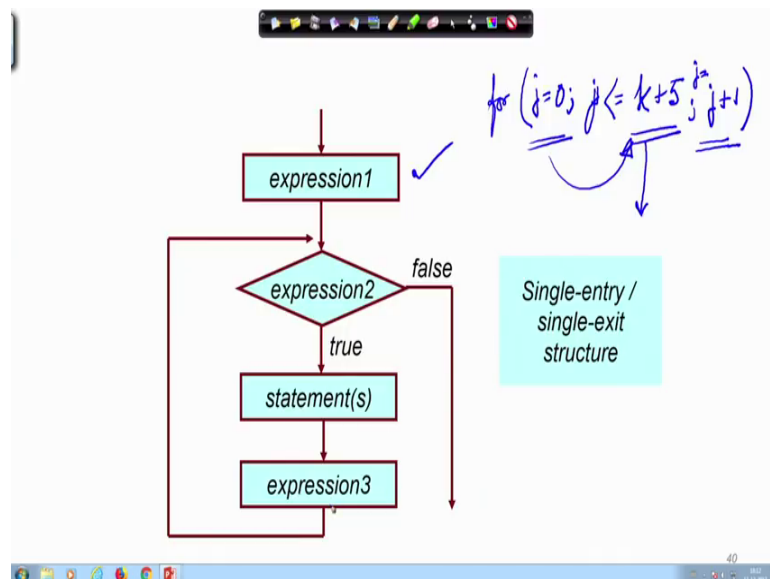
(Refer Slide Time: 20:52)



- How it works?
 - “expression1” is used to *initialize* some variable (called *index*) that controls the looping action.
 - “expression2” represents a *condition* that must be true for the loop to continue.
 - “expression3” is used to *alter* the value of the *index* initially assigned by “expression1”.

So, expression one is used to initialize, so here.

(Refer Slide Time: 20:58)



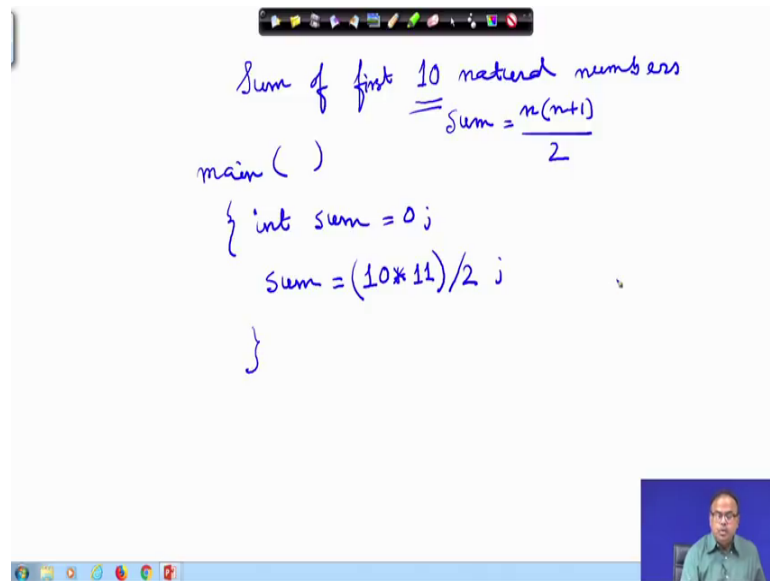
So, what is the way it is done is first expression 1 is executed. So, for j equals 0, j less than equal to k plus 5 can be anything it can be an expression j plus plus. So, first this expression is executed then after we execute this, I check by initialization have I violated the condition in that case of course, there is no point getting into the loop I will go out.

Otherwise if it is not done then I go inside the loop carry out the loop and then instead of going into the loop straight back I will first come to this alteration or modulator

statement. So, it is I am sorry what did I write here. I wanted to write j assign j plus 1 here or j plus plus whatever. I do that and after I do that what do I do, I immediately come back to this expression and test it again is it true, if so I will get into otherwise I will exit. So, this is the structure of the for statement.

Now, there are a couple of critical issues, but before that let us write a simple program with for. Suppose I want to find the sum of first 10 natural numbers.

(Refer Slide Time: 23:01)



Sum of first 10 natural numbers
$$= \text{sum} = \frac{n(n+1)}{2}$$

```
main ( )  
{ int sum = 0;  
  sum = (10*11)/2;  
}
```

What are the first 10 natural numbers? 1 2 3 4 5 up to n. So, in school mathematics you know that the sum for 10 numbers it is the formula is n times n plus 1 by 2 right. So, one simple program is if I want to find the sum of n numbers you can simply do in your main int sum assign 0 and then in one short you can write sum assign 10 because I want to write 10 natural numbers, 10 times 11 divided by 2 and then printf sum, that could be one way. But that is not what I am going to do. I want to illustrate the for loop.

(Refer Slide Time: 24:33)

```
main ( )
{
  int sum = 0;
  int myindex;
  for (myindex = 0; myindex < 10; myindex++)
    sum = sum + myindex + 1;
}
```

5
5 * 11 = 55
2
myindex = 2
sum = 1

1	1	1	3
3	2	1	6

So, the way I can do it is main main function and here I put in int sum equals 0. Now, I know that it is not I am not going to find the sum of n natural numbers I am going to find the sum of 10 natural numbers. So, I can write it in this way for. So, in sum equal to 0 another variable I have to initialize that is my index I write it int my index not necessarily i or j all right, but it must be an integer. So, for, but I am not initializing it here, for my index assigned 0 my index less than 10, please verify whether I am right or wrong my index plus plus. Sum assigned sum plus my index plus 1 what will happen? Sum was 0, my index was 0, so my index plus 1, the sum will be 1. What am I expecting? 10 times 11 by 2 right. So, that will be 55 right, sum of first n natural numbers.

Now, my index becomes 1. So, my index is 1 now, less than 10 I again add that. So, sum was what was my sum? My sum was 1. So, here is first iteration sum was 1, then sum was 1, I have written it here. So, sum plus my index my index was 1. So, 1 plus 1 am I right. So, here it was right. Now, sum was 1 and my index is 1 plus, 2, 1 plus 1. So, it will be 1 plus 2, this will be added so the sum will be 3.

Next my index will be implemented to 2. So, now, sum is 3 my index is 2 plus 1, so 3 plus 2 5 and 1 so sum will be 6. In that way it will go on and ultimately I will come to this point where it will exceed 10 and then I can stop.

So, here is now, you can also do that, some of natural numbers you can do that all in a loop using the very the index itself being updated and creating the different natural numbers every time.

(Refer Slide Time: 28:58)

for (i = 1; i <= 5; i++)
sum = sum + i;

$\frac{5 \cdot 6}{2} = 15$

i = 1	Sum = 1
2	3
3	6
4	10
5	15

So, if I had written it in this way for i equals 1, there is the first one less than 10 less than 10 would that be i plus plus sum equals sum plus i. What would have happened? i is 1, sum is 1 right.

Now, next i is incremented i is 2 and then sum was 1 so that will be added sum is 3. Again this will be added 3 sum is 4 sum is 3. So, then be 6 again 4 then be 10 in that way it will go on ultimately. So, suppose I was trying to do it for 5 numbers. So, i just i less than 5 first 5 natural numbers. So, sum is 5. So, now 4, I have already taken. Now, i plus plus it becomes 5, but I will get stuck here.

So, if I start it with 1 what should I do? I just did it in a couple of moments earlier in the earlier lecture. So, what should I do here? I should make it i less than equal to 5, in that case I will take this 10 plus 5 15. So, 5 natural numbers sum of 5 times 6 by 2 is 15 right. So, I can compute that using this loop all right. So, that is a very interesting application. So, for is a very powerful will see, in future application that for is a very powerful construct using which we can do many things.