

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

Lecture – 19
Implementing Repetitions (Loops)

We were looking at C constructs by which we can achieve repetitions or loops; that means, a set of statements will be executed repeatedly for a fixed number of times ok.

(Refer Slide Time: 00:38)

while Statement

```
while (condition)
    statement_to_repeat;
```

```
while (condition) {
    statement_1;
    ...
    statement_N;
}
```

as long as

```
/* Weight loss program */
while ( weight > 65 ) {
    printf("Go, exercise, ");
    printf("then come back. \n");
    printf("Enter your weight: ");
    scanf("%d", &weight);
}
```

The slide includes a small video inset of the professor in the bottom right corner.

So, one of such constructs that we came across was the while statement as is being shown here while. And you can see the structure of this that while a particular condition is true, we will carry out a set of statements. It can be one statement or it can be a number of statements as is shown here a number of statements will be repeated as long as this condition is true. An example that we had seen in the last lecture was this that is as long as I can read this while as long as all right. As long as the weight is greater than 65 we will have to do exercise all right, and here you again check the weight and if it is greater than 65 you will carried out that is the structure of the while statement.

Now, how can you use it fruitfully? Let us take an example of using the while loop for some meaningful computation. For example, I want to add 10 numbers the example that we are doing earlier we had shown the 0 count for that. So, I have got a value N.

(Refer Slide Time: 02:05)

```
{
    Sum = 0;
    Count = 0;
    while (Count <= N)
    {
        printf("Please enter a number \n");
        scanf("%d", &num);
        Sum = Sum + num;
        Count++;
    }
    printf("The sum is %d \n", Sum);
}
```

Handwritten annotations on the slide include:

- A box containing the numbers 45, 44, 37, and 22.
- Initial values: $N=5$ and $Count=0$.
- Initial values: $Sum=0$ and $Count=0$.
- Loop condition: $while (Count \leq N)$.
- Print statement: `printf("Please enter a number \n");`
- Input statement: `scanf("%d", &num);`
- Sum update: `Sum = Sum + num;`
- Count update: `Count++;`
- Final print statement: `printf("The sum is %d \n", Sum);`

Let us see let us do it for a small n number N equals to 5. So, I will carry out the sum of 5 numbers. So, I will have a count another variable count which may be initialized to 0 ok.

Now, we can write something like. So, initially count is 0, count is an integer, count is 0 while count is less than N and also I do another thing I am going to add 5 numbers. So, I create another variable sum. So, sum is 0 and count is 0. So, while count is less than N, I can I write a complete program; now printf. Please enter a number and then. So, on the screen I will have please enter the number printed here, then I am doing scanf percentage d and num, num is the variable which I am reading, and then I am adding updating sum initially sum was 0. So, sum plus num all right I do this and then I how many numbers I have read I have read one numbers. So, I will do count plus plus; that means, count is now 1.

So, one number has been read. So, I will be completing here, now let us see what will happen. Suppose M was 5. So, sum is 0. So, the first number has been read. So, count was 0 please enter a number and some number has been entered and suppose that number was 5, then sum is sum plus number. So, this one becomes 5 then I increment count. So, count becomes 1. I come back here, I find count is still less than 5, because I am going to n is 5 right.

So, count is still less than 5 I again do that suppose I did the second number 6. So, then this becomes here I add this becomes 11 and count becomes 2. Please note that count in this case is therefore, keeping a count of how many numbers I have already read because I am reading the number here and then incrementing count. So, count now I have read 2 numbers and I have added the sum ok.

So, now after doing this count is becoming 2 average, 2 numbers I again go here now 2 is less than 5 I take another number say 11. So, I add sum. So, it becomes 22 and I increment count 3. I have read 3 numbers I go up again here and read the forth number 15 add that with sum. So, 22 plus 15 will be 37 and then count will be upgraded it will become. So, I have read 4 numbers 5 6 11 15.

Now, I again go up go up here and check count, count is 15 yeah sorry count is 4 which is less than 5. So, I read another number suppose that is 7 I come here add it, it becomes 44 and I increment count I first add that number. So, it becomes 44 and then I increment count. So, count becomes 5 meaning that I have already read 5 numbers.

Now, when I go back here, I check is count less than N? No therefore, I will come out of that and maybe here I will write something like printf the sum is assuming integers is percentage d back slash n and here I print sum and whatever I do here are the other components of the complete program declarations integers and all those things are there all right.

Now, point to be careful about, I must be very careful about expressing the condition. Look here if instead of this I had if instead of this I had made it less than equal to n then what would have happened? If instead of count less than n, if I had written count less than equal to n what would have happened? After count is five; that means, I have read 5 numbers have added 5 numbers and 44 is my result I would again go back here, and find that count is less than equal to 5 count is 5 therefore, this condition would be true, I would again come and would have read another number sorry I should cut it out. I would have read another number and then count would be incremented 6, I would have 45 which would be along result because here actually I have read 6 numbers. So, you must be very careful to specify this particular statement.

So, that the number of times you want the loop to work should be accurate should be correct. So, I think you have understand this example, a very simple program, but some theory is needed about this. So, that was an example of while statement.

(Refer Slide Time: 10:15)

```
while (condition)
    statement_to_repeat;

while (condition) {
    statement_1;
    ...
    statement_N;
}
```

```
/* Weight loss program */
while ( weight > 65 ) {
    printf("Go, exercise, ");
    printf("then come back. \n");
    printf("Enter your weight: ");
    scanf("%d", &weight);
}
```

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

digit = 0, 1, 2, ..., 8

0 1 2 ... 8 9

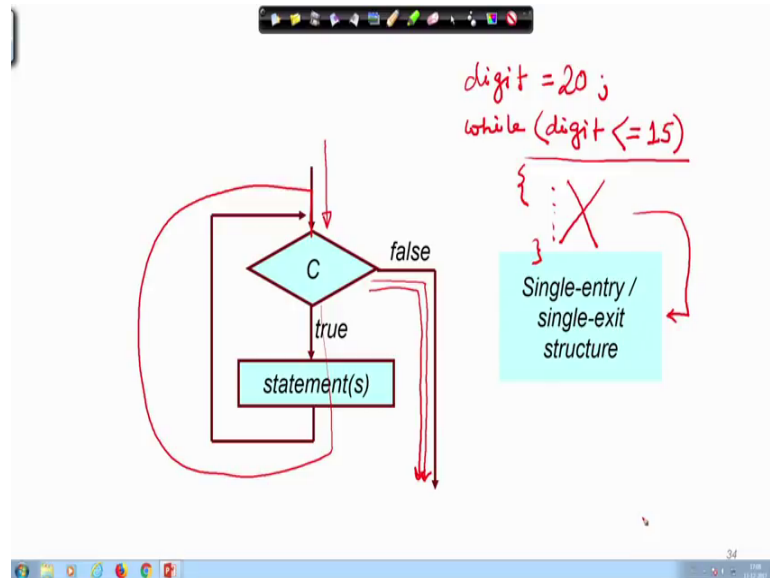
So, here is another example suppose the digit is 0 what would this one do can you find out what this will do? This is a while statement. All of you try to look at it and find out what this loop does this while loop does. If you look at it carefully you will see that I am starting the digit I have initialized digit to be 0.

Now, while digit is less than equal to 9, first time I come here it is less than equal to 9. So, what will it do careful it will come to this printf statement and we will print digit now it is a post increment or pre increment it is a post increment; that means, first digit will be printed. So, 0 will be printed then digit will be incremented. So, this digit will now become 1 I again go back here and find that digit is less than equal to 9 right. So, I will come here, print the digit 1 and then again increment digit, digit will become 2 I go back here check whether it is less than equal to 9, it is still less than equal to 9, I will come inside the loop will print the digit 2 in that way it will go on ultimately 8 will be printed and 8 has been printed, and I go back after printing 8; this has been incremented to 9, I go back here.

Now, you see here less than equal to 9 therefore, I will still executed; that means, I will print 9 and then increment it, it will be 10 and then when I go up here this condition is no

longer true therefore, I will come out of this loop. So, what will be printed 0 to 9 the 10 numbers will be printed. So, that is how the while loop works.

(Refer Slide Time: 13:05)

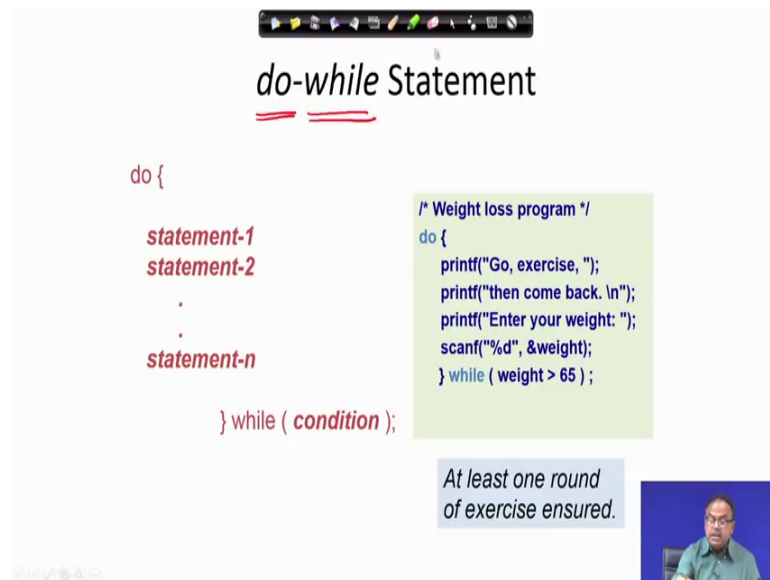


So, this flowchart is very important to remember what while does you have actually understood it by now, but the key point to note the most important point to note is this, that when I am executing the program in this direction, I first compute the condition. First I compute the condition and if the condition is true I execute the body of the loop then I again go up check the condition and this will be going on.

So, first the condition is checked if it is false I will go out of the loop. So, if initially say for example, I write something like this say digit, is equal to 20, and I start something like while digit is less than equal to 15, some things I will be doing. Now in this case when it comes to this condition at the very beginning in will fail, because this condition is not true. So, it will come out to this false path and this loop will not be executed, will be just coming to the next statement after the loop.

So, in while the condition is first evaluated and depending on the success of the test of the condition, will either enter the loop or will bypass the loop. So, this flowchart is very important.

(Refer Slide Time: 15:04)



do-while Statement

```
do {  
    statement-1  
    statement-2  
    .  
    .  
    statement-n  
} while ( condition );
```

```
/* Weight loss program */  
do {  
    printf("Go, exercise, ");  
    printf("then come back. \n");  
    printf("Enter your weight: ");  
    scanf("%d", &weight);  
} while ( weight > 65 );
```

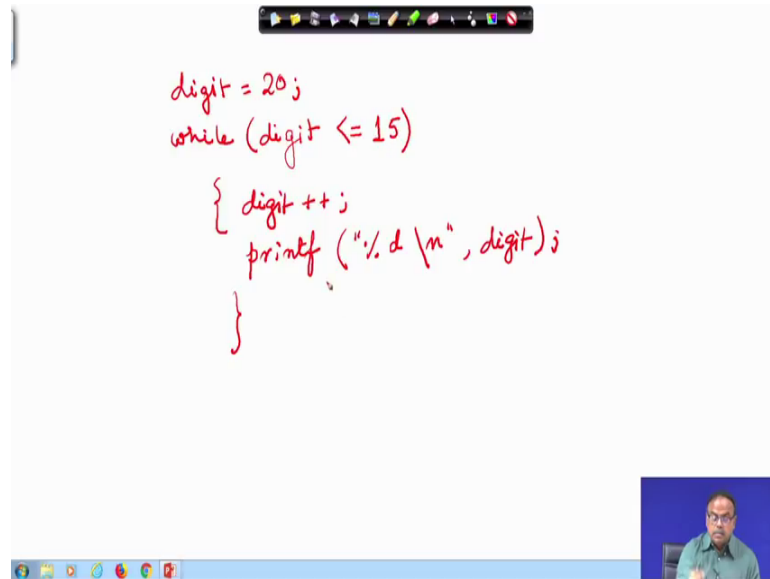
At least one round of exercise ensured.

Now, we will be contrasting while statement with another statement which is the do while statement what is the difference looks like very similar. So, there is a scope of confusion while and do while. So, as the name implies here do comes first do something and then check the condition. What it means is say do some statements here while condition will do all these statements and then check the condition.

So, let us see what will happen here, the weight loss program again really with while is do go exercise. So, we start with a do you do, go exercise printf comeback whatever whatever, then read the weight and while weight is 65. So, at least the condition is being checked here the condition is being checked here. So, it will at least carry out the computation once.

(Refer Slide Time: 16:33)

```
digit = 20;
while (digit <= 15)
{
    digit++;
    printf ("%d \n", digit);
}
```

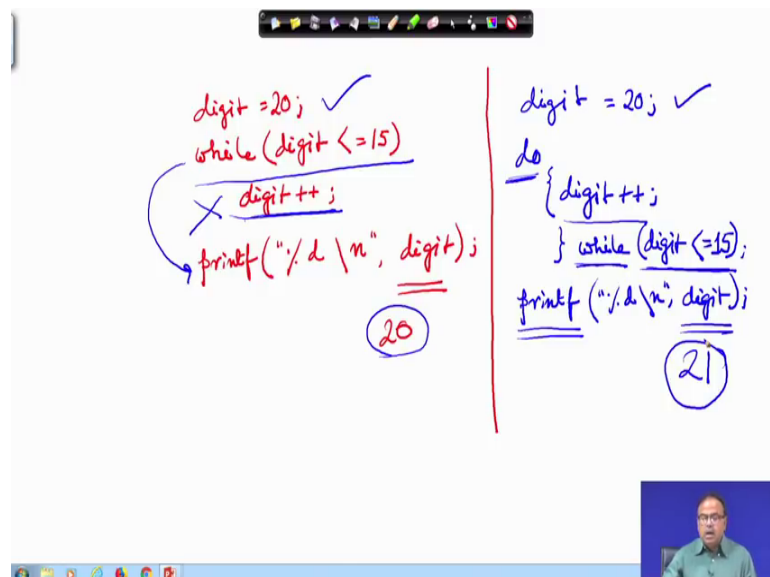


So, the digit thing if we do suppose I write something like this. So, digit is 20, and I write while digit is less than equal to 15 say I do digit plus plus, printf digit. Now in this case since the digit is 20 what will be printed here nothing all right or say let me make it even simpler.

(Refer Slide Time: 17:44)

```
digit = 20; ✓
while (digit <= 15)
digit++;
printf ("%d \n", digit);
20
```

```
digit = 20; ✓
do
{
    digit++;
} while (digit <= 15);
printf ("%d \n", digit);
21
```



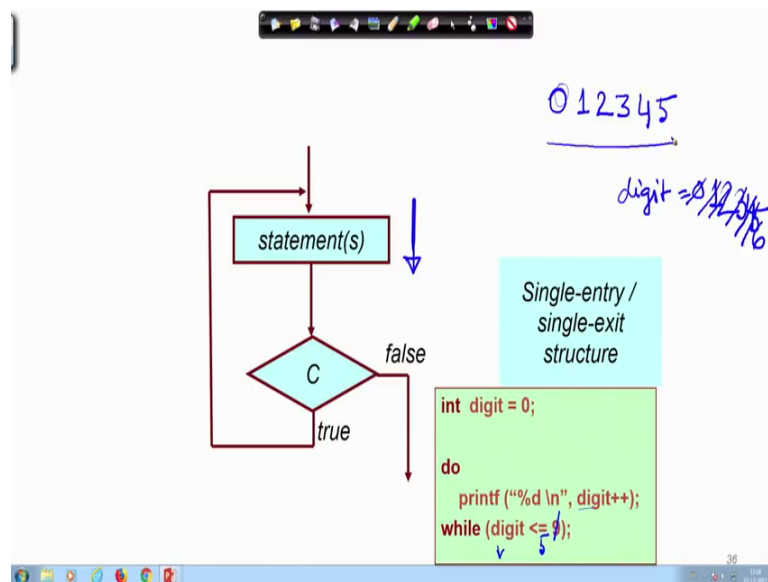
Let me make it a little different digit is 20, while 15 digit plus plus printf percentage d back slash n digit. If I do this since digit is 20 what will be printed here 20 will be printed because this plus plus will not be executed right because here it will feel.

But if I do it now if I do it as digit equals 20, do I need to give the bracket, I may or may not, but let me give it it really does not matter digit plus plus; while digit is less than 15 less than equal to 15 and here I write printf percentage d back slash n digit all right what will happen let contrast this here also digit is 20 here also digit was 20.

Now, here it was tested at this point. So, this digit plus plus will not be executed, it will straight way come here and will print digit it will be 20 whereas, in this case I first encountered this do, and as I do I check this digit I make this digit plus plus; that means, it becomes 21, and then I check while digit is less than equal to 15. Now I find digit is not less than equal to 15 of course, I will not do it again, but once I have already done it. So, when I take the printf digit will be printed to be 21.

So, here is the difference between while and do while. So, these are 2 constructs by which just 2 examples constructs by which we can carry out the loop. So, at least one round is carried out. So, if I take if I look at the flowchart, it will be looking like this it will have a set of statements which will be executed at least once whatever the condition b condition s. So, this will be executed first it will be first executed here.

(Refer Slide Time: 21:16)

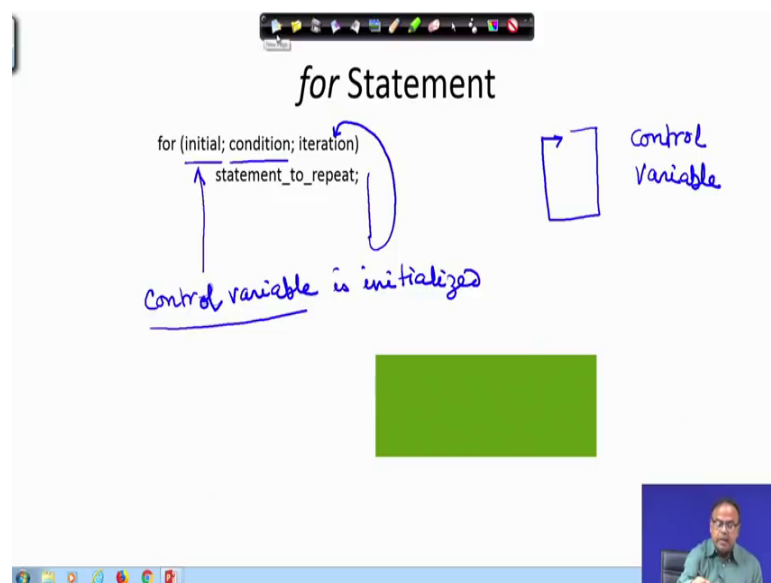


And then we check the condition if it is false I will not do it again, I will come out otherwise I will repeat. So, here again that old example digit was 0, do printf digit plus plus while digit is less than 9. So, you remember in the earlier case we had printed 0 1 2 3 4 5 6 7 8 9; now what will be printed in this case let us see digit is 0 initially.

Now, we come here first we print digits. So, 0 is printed, 0 is printed then digit plus plus is done. So, digit is 1 right. So, digit now becomes 1 digit was 0. So, it becomes 1 and then I check is digit less than 9 yes it is less than 9. So, I again print digit one is printed let us make this 9 let us say that is 5 all right let us say it 5. So, we have done make made it one and after printing one digit has been made 2 plus plus has been done, I check again it is less than 5. So, I will do. Go to the loop again I will do it again. So, I will be printing 2 and then make digit to be 3 it is still less than 5, I will go again I will print 3 and then I make digit to be 4 still less than 5, I print 4 I make it 5 all right less than equal to 5 right. So, I print 5 print 5 I first print 5, then increment it to 6 as I do it I increment to 6 and then I check whether it is a less than equal to 5.

So, if it is since it is less than not less than equal to 5, I will come out of the loop. So, what I will be printing is 0 1 2 3 4 5. So, this is an example of while do while next.

(Refer Slide Time: 23:55)



So, this is one type of statement that we have encountered here, that is while and do while. Here we are not specifying any number of times that it will be doing; as long as a condition is not met I will be doing it.

Now, how is it that the condition is being affected it is through the computation inside. So, for example, the computation here this is plus plus, that is being done by that this condition variable is changing all right there are cases where I know a priori beforehand

I know beforehand that I will have to carry it out 10 times I will have to carry it out 20 times so and so forth ok.

So, for that another construct is very important and is used in C language that is the; for construct. Welcome to the; for construct this a little complicated, you should be attention there 3 components of this for one is an initial a value, now this initial value can be assigned to a control variable. So, there is a loop program and how many times it will loop. How many times it will loop that is being determined by some control variable. In the case of while what was our control variable? Say for example, in the earlier is a case of earlier example the value of digit was the control variable. Here we put some control variable and initialize the control variable with some value; here some control variable is initialized and then we have a condition. If that condition is true, then I will enter the loop and do that and after doing this loop I have got some iteration parameter by which we update the control variable and check the condition in that way it will go on.

So, let us have a look at the structure in a little more detail.

(Refer Slide Time: 27:13)

The slide is titled "for Statement" and contains the following C code examples:

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

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

Handwritten annotations include:

- A vertical line next to "10" with a bracket underneath, indicating the range of iteration.
- A circled sequence "1.2.3.4.5" with a bracket underneath, representing the first five iterations.
- A sequence "1.2.3.4.5.6.7" above "8.9.10", representing the remaining iterations.
- A green box containing the code:

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

with a blue arrow pointing from the loop body to the "8.9.10" part of the handwritten sequence.

A small video inset of a person is visible in the bottom right corner of the slide.

See here we have got some initial value expression some condition and iteration and we have got some statements which will be looping. Here is an example say I am calculating the factorial of 10. So, I am trying to compute the factorial of 10, some people write it in this way some people write it in this way all right.

Now, here what is being done I will explain it in a little bit more detail, fact is a variable which is standing for factorial it is initialized to 1. Now you know factorial of 5 is what? 1 times 2 times 3 times 4 times 5 right. So, I am started with some variable and multiplying it with its successor, and then the product I take and multiply with the successor of the last integer and in that way it goes on. So, here is a look of how it will look like we have got a for statement here, and I am saying for I assigned one now this is what I was talking about I is a control variable here that is being initialized to 1.

So, that is less than 10 of course, because here I am trying to compute factorial 10 and then what I do fact was 1. So, fact will be fact multiplied by i what was i? I was 1. So, it is one times 1, then after this operation is done I go and do this iteration operation iteration operation what it is doing it is incrementing i and i is becoming 2, and then before I enter I just check whether it is still less than 10 yes it is less than 10, I will again do that. So, fact will be it was 1, 1 times what is i now 2, 1 times 2 after I compute 1 times 2 I will increment i. So, that will become now 3 it is still less than 10 please follow my pen it is still less than 10, I go in compute what was I three. So, I whatever was fact was the product of 1 and 2. So, I multiplied that with 3, and then I increment the iteration variable it becomes 4 again I come and check is it less than 10 yes it is I go on here and I multiply it in that way it goes on and on.

Now, ultimately it will be ultimately it will be like that 5 6 7 8 now suppose i is 9 sorry suppose i is 9. So, I come here I multiplying fact with 9 i plus plus. So, i becomes 10, i come here, i is still less than equal to 10 is it is equal to 10, I come here multiply 10 i plus plus it becomes 11, I come here this condition is not satisfied I come out all right, this is the for statement will look at it in the little more details in the next class.