

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

Lecture – 18
Switch Statement (Contd.) And Introduction to Loops

So we are looking at a new construct that is switch and case.

(Refer Slide Time: 00:23)

```
Example

switch (choice = toupper(getchar())) {

    case 'R':    printf("RED \n");
                 break;

    case 'G':    printf("GREEN \n");
                 break;

    case 'B':    printf("BLUE \n");
                 break;

    default:     printf("Invalid choice \n");

}

Handwritten notes:
- 'getchar()' is circled in red.
- An arrow points from 'operation' (written below) to 'getchar()'.
- An arrow points from 'get the character from the user' (written below) to 'getchar()'.
```

So, here is another example, but this example has got many more things embedded in it. For example, in order to understand this you have to understand two things that are being introduced here, one is getchar. This is a construct we have seen scanf, we have seen printf as input output statements right, but getchar is also another statement another function actually, this is given by you can since we have not considered function in detail there is a built in library function or an operation you can consider that to be an operation also that getchar and then they there is a parenthesis as if we want to have some value here.

It means get the character from the user. So, where from does the user provide you the character whatever the user types, that will be captured by this function and this will get that particular character that is why it is called getchar, all right.

(Refer Slide Time: 01:57)

Example

```
switch (choice = toupper(getchar())) {  
    case 'R':    printf("RED \n");  
                break;  
    case 'G':    printf("GREEN \n");  
                break;  
    case 'B':    printf("BLUE \n");  
                break;  
    default:    printf("Invalid choice \n");  
}
```

toupper ()
char mychar;
mychar = 'a';
toupper (mychar);

That other thing that we have here is another function to upper to upper and then you can see that there is another parenthesis to upper what it does is it converts any character variable to its upper for example, if I type a as a character say I have got a variable mychar. Now suppose I have declared mychar to be a character type of variable and somewhere here I assigned mychar to be a.

Then what does mychar get? Mychar gets a ascii value of small a, now if I say toupper mychar then what it will do? It will take the ascii value of small a and will return me the ascii value of capital A all right to the upper a. So, this one will be returned.

(Refer Slide Time: 03:23)

Example

```
switch (choice = toupper(getchar())) {  
    case 'R':    printf("RED \n");  
                break;  
    case 'G':    printf("GREEN \n");  
                break;  
    case 'B':    printf("BLUE \n");  
                break;  
    default:    printf("Invalid choice \n");  
}
```

So, here you see to upper what not mychar, but what? We have written to upper then getchar character. So, what it means is that, you will get the character from the keyboard suppose it is small a, and then that will be converted to capital A. So, it becomes capital A and then that is being fed assigned to choice ok.

Let us look at it because it gives us the opportunity to look at nesting of functions also here two built in functions which are being used here to get the character and assign it to choice all right. So, you can also have another example similarly we have got to lower getchar something, where if I had given capital A it will be converted to small a ok.

Now, let us look at this. So, we take a choice. So, what is the choice? We are taking a choice and what is the choice; choice is a character that has been given if the user gives a lowercase character then suppose I want a choice between a multiple choice answer scenario, type in the answer a b c. So, somebody can type caps cap a shift a or just a all right whatever you type internally I will convert it to capital.

So, through my; to upper functions. So, now, I take your choice now based on the choice I go on checking is it r if so, print red and then break its not R then let us go and check whether its green G. If it is G then print green and then print green and for break you will come and meet here it is not green also not R not green, I will come to this point is it blue yes then I will print blue, but suppose somebody has printed as typed in give the choice as y all right Y then neither it is R nor it is G nor it is B then we will come to default and we will print invalid choice.

Here since that is a last statement I am closing the bracket immediately after that I may not give the break statement here, I am free not to give the break statement here. Otherwise now suppose let us look at this break statement suppose I do not give the break statement suppose this is not there and my choice is r my choice is small r.

(Refer Slide Time: 07:54)

```
Example  
switch (choice = toupper(getchar())) {  
    case 'R':    printf("RED \n");  
                break;  
    case 'G':    printf("GREEN \n");  
                break;  
    case 'B':    printf("BLUE \n");  
                break;  
    default:    printf("Invalid choice \n");  
}
```

Handwritten annotations: A red arrow points from the 'R' case to a handwritten 'r', which then points to a handwritten 'R'. Below this, 'printf("RED")' is written in red.

So, at this point the small r will be converted to capital R by this statement right by this. So, my choice will be capital R now I go here and I will my system will find that it is r. So, it will printf red I am sorry why should I the printf red will be worked on let me

do it again, my choice was r that was converted to capital R and by this and then I come to this point and I find that it is r.

(Refer Slide Time: 08:28)

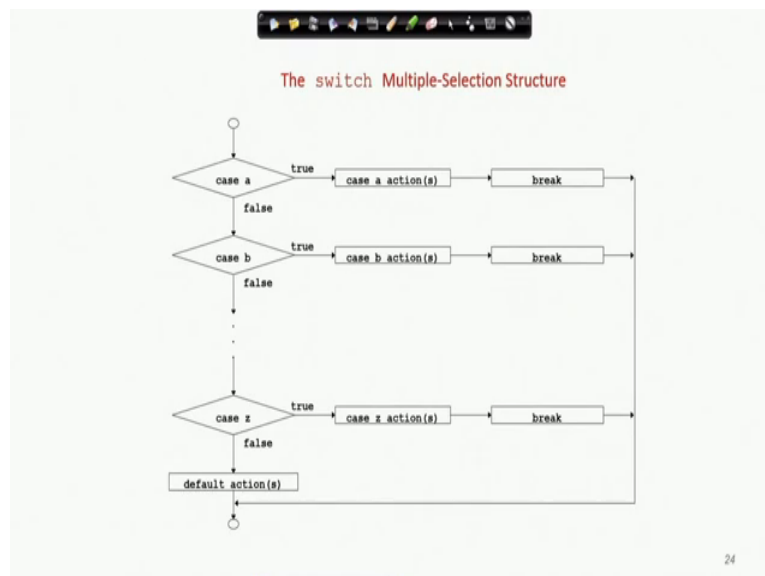
Example

```
switch (choice = toupper(getchar())) {  
  case 'R':  printf ("RED \n"); ✓  
             break;  
  case 'G':  printf ("GREEN \n");  
             break;  
  case 'B':  printf ("BLUE \n");  
             break;  
  default:   printf ("Invalid choice \n");  
}
```

r → R
RED
GREEN

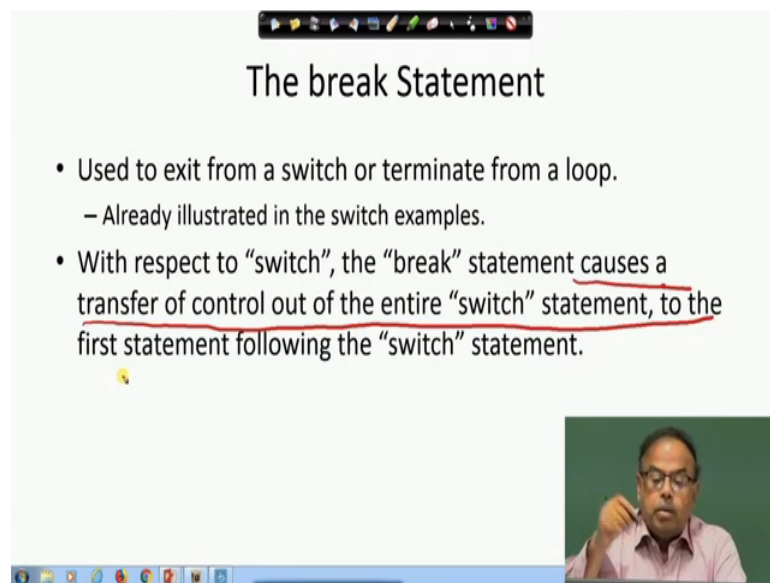
So, my system prints red, red has been painted now this break statement is not there. If the big break statement is not there, then it will not go to check this condition it will simply print all these things. Since it has follow this path unless I force it to break it will go on following this path. So, I will have green printed again which is not what I desired right. So, let us see. So, now, let us see here let us look at this flowchart.

(Refer Slide Time: 09:43)



The switch statement if the case is red let us take to the example that we are doing, red then I print red then break. If the sorry if the break was not there then I would be following this path again I mean this path would be followed that is why this break is essential will show that requirement of break in a moment.

(Refer Slide Time: 10:29)



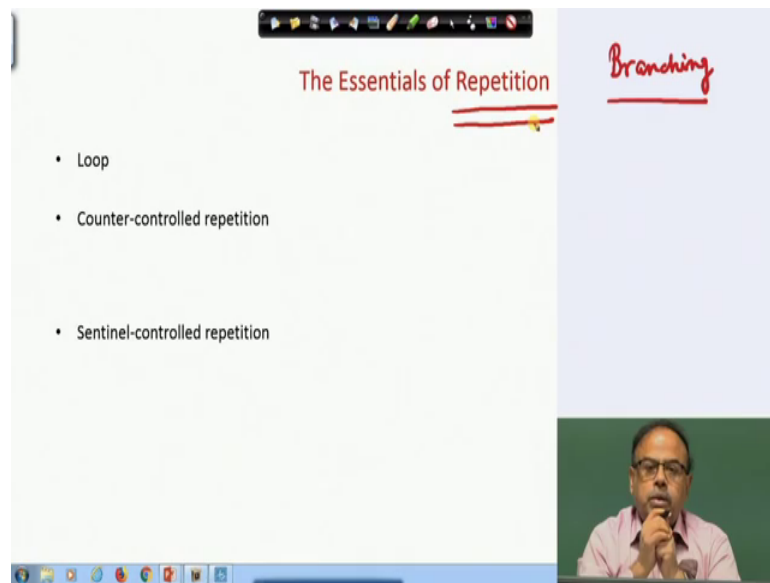
The break Statement

- Used to exit from a switch or terminate from a loop.
 - Already illustrated in the switch examples.
- With respect to “switch”, the “break” statement causes a transfer of control out of the entire “switch” statement, to the first statement following the “switch” statement.

Now, let us look what the break statement is used for. The break statement is used to exit from a switch or terminate from a loop; we have just illustrated it for the switch statement it can be also used for some other purpose.

So, the break statement causes a break statement causes a transfer of control out of the entire switch statement to the first statement following the switch statement.

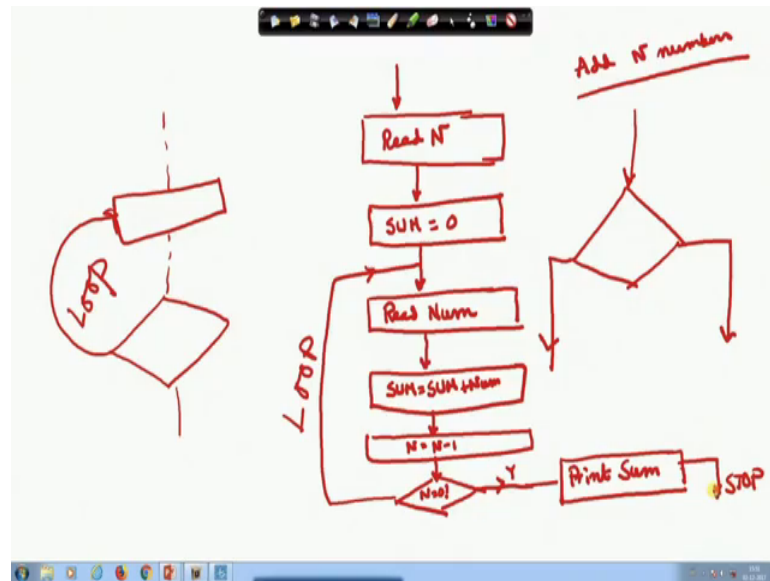
(Refer Slide Time: 11:06)



So, I hope you have understood what is meant by the switch statement how it can be used we will see that, through a number of examples later we will see with the number of examples. So, we will see that the break statement can be used for exiting from loop also we will see that, but what is a loop.

Next we come to see repetition, till now what we have seen is branching, but now we are going to look at another important construct which is repetition. Now just for a moment let us go back to our old friend example that is finding the sum of n numbers for example.

(Refer Slide Time: 12:32)



Now, how did you do that what was the flow chart? The flow chart was something like this read N there is a number of numbers all right then we make sum equal 0, I am adding 10 num adding n numbers then read number I am giving a different variable name num, then I am adding that with sum, sum is assigned sum plus num, then I am decrementing N.

Now, I look at is N 0. if yes then I go and print sum all right. So, if there be three then I first read one add it. So, n actually shows how many numbers are yet to be added. So, print sum then stop, but if N is not 0 then what I will do? I will go and continue this path this is another form of structure that we see that we have based on some decision we have got sum statements somewhere, and we are going back to that and repeatedly doing this this is we are repeatedly doing this this is known as loop this is known as repetition or loop all right.

Unlike the earlier case of if then where we look at the condition and then we follow either this path or this path both in the forward direction, here we are also taking up the backward direction. So, this is a very important concept we will need it at every step in our programming exercise. Now there can be different types of loops we will discuss that later, but first let us take.

(Refer Slide Time: 15:43)

The Essentials of Repetition

- Loop
 - Group of instructions computer executes repeatedly while some condition remains true
- Counter-controlled repetition
 - Definite repetition - know how many times loop will execute
 - Control variable used to count repetitions
- Sentinel-controlled repetition
 - Indefinite repetition
 - Used when number of repetitions not known
 - Sentinel value indicates "end of data"

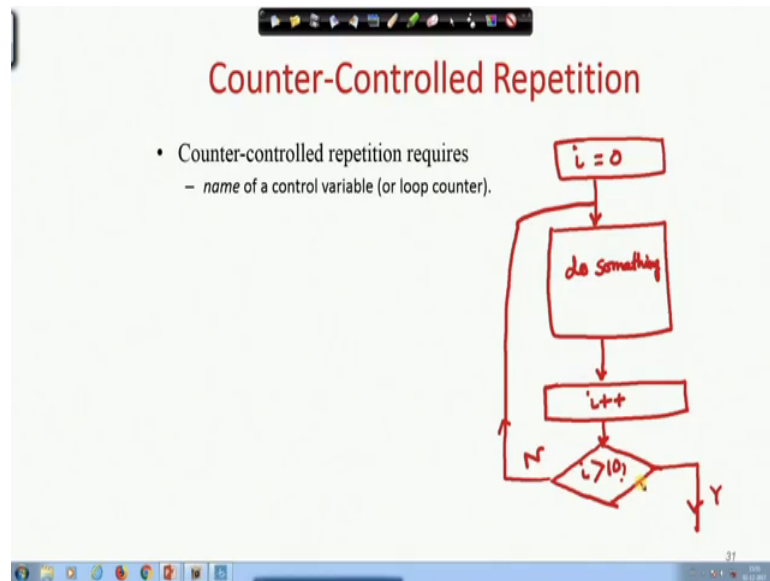
Handwritten notes: "Find the sum of N numbers" with a circular arrow and a large "0".

Let us look at a counter now there are three types of loops briefly let me tell you one is loop is something that is repeatedly executed. There can be counter control repetition where we know that that we know before and how many times I must repeat ok.

Suppose I know that find the sum of N numbers whatever that be n I know that this will have to be repeated N times all right N times, but there is another case where we actually go on repeating a particular set of sentences we go on repeating that until a particular condition occurs; that means, some value has become negative or something that say for example, I am taking the sum I am adding that and the sum goes beyond 9999 and if it goes beyond that I will stop, that sort of situation is known as sentinel control or even controlled repetition this will come later.

But most importantly we will look at this counter controlled repetition where we know how many times we should look and for that we need a control variable we will see what that is.

(Refer Slide Time: 17:38)



So, counter control repetition requires name of a control variable or a loop counter. If you recall in when we added n numbers what was my counter variable? What was my control variable or what was the loop counter, how many times will it do n the number of numbers right.

So, similarly I can have another some other variables say I say I can just draw arbitrary flowchart for you say is and integer initialized to 0, then I come here do something and then i increment i let me use what we learnt I increment i plus plus; then I check, i greater than 10 if it is yes I will again, if it is no sorry if it is no I will again do that if it is yes i will come out all right. So, I am doing it as long as i is not greater than 10.

So, this here what is my control variable? My control variable is i or this is a loop counter based on which how many times I will be carrying it out is determined. Now we also had an initial value of the control variable what was the initial value that I did in this case.

(Refer Slide Time: 19:32)

Counter-Controlled Repetition

- Counter-controlled repetition requires
 - *name* of a control variable (or loop counter).
 - *initial value* of the control variable.
 - condition that tests for the *final value* of the control variable (i.e., whether looping should continue).
 - *increment* (or *decrement*) by which the control variable is modified each time through the loop.

```
int counter = 1; //initialization
while (counter <= 10) { //repetition condition
    printf( "%d\n", counter );
    ++counter; //increment
}
```

Handwritten annotations: *while it is raining*, *stay home*, *while*, *counter++*

The initial value of *i* was 0 it was initialized to 0 if you have seen the condition that tests for the final value of the control variable whether the loop should continue or not, what was my condition test in the earlier case? It was *i* greater than 10 if *i* is greater than 10 then I will come out otherwise I will go on continuing with the looping and another increment or decrement operation here *i* was 0. So, I implemented it until it comes to 10 or crosses 10.

If you recall in the earlier example there was *n*. So, I have to read *n* numbers after reading one number I decremented *n*. So, depending on what I want to do I will have to increment or decrement the control variable and based on this condition I will come out. So, here is an example an initialization is done, you can see here a counter has been set to one it is an integer counter that has been set to one while counter is less than equal to 10, while it is a statement that you are getting. As long as counter is less than equal to 10. So, this is one new term you are learning.

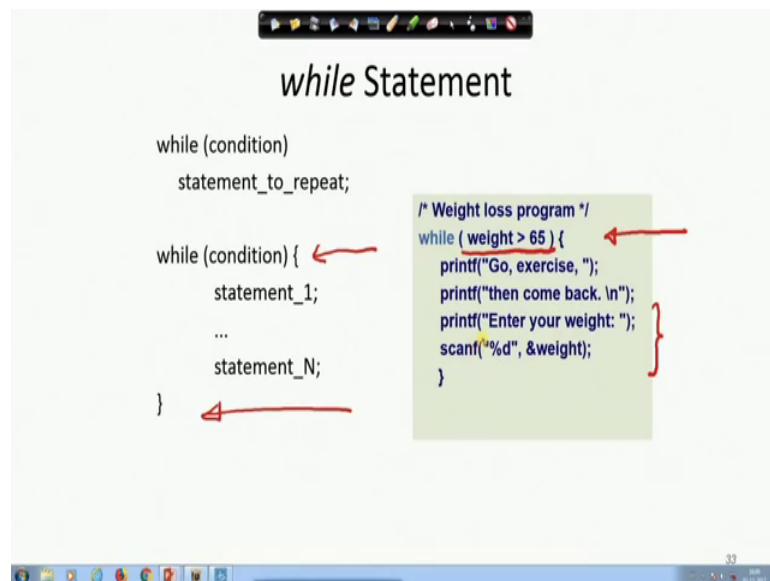
While counter is less than equal to 10 do this printf the counter value and then increment the counter and then go on doing it; now this is this really does not make any difference if I had written counter plus plus that we equivalent because these are singleton here there is no assignment or nothing addition with this value. So, its standing alone it really does not mean me and does not matter whether it make it counter plus plus or counter

minus minus. So, this one is. So, here is the condition check counter is my control variable and increment and decrement operation.

Now, you are getting this while statement and now let us try to find the simple English meaning of while. We often write something like this while it is raining stay home; that means, as long as it is raining stay home. So, this is the conditions condition as long as the condition is true do this, here you see while the counter is less than 10 as long as this condition is true do this. If this condition is false then come here do not do this that is the meaning of the semantics of while.

So, while is used for one of the methods by which we can achieve counter control repetition now counter control this we have seen.

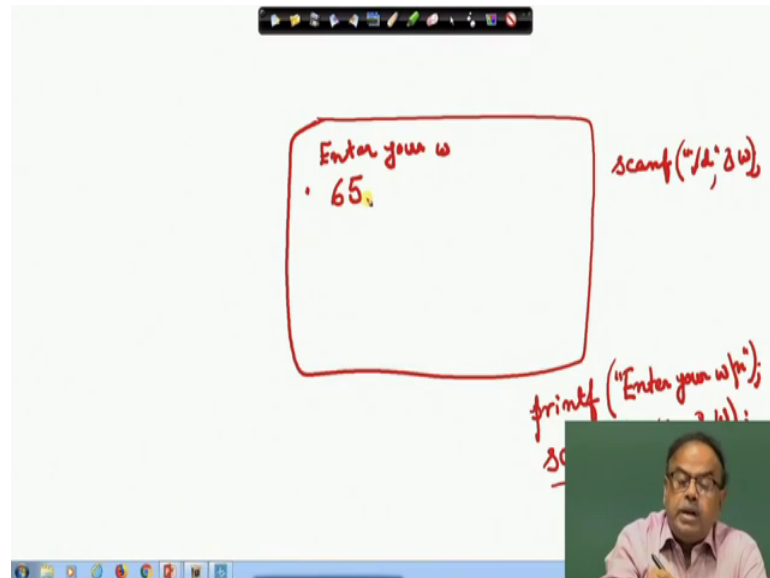
(Refer Slide Time: 23:11)



So, the while statement actually looks like this, while some condition the statements to repeat it can be one statement or it can be multiple statements as is shown here within this block all right, this entire statement should be this entire thing should be repeated; example here suppose here is a weight loss program I mean as if you have being told to lose weight by following this while weight is greater than equal to 65. While weight is greater than equal to 65; that means, as long as your weight is greater than 65, then printf go exercise ok.

Then come back again enter your weight and then read the weight, now here there is a nice thing often we had encountered the scanf and printf independently. Now look at this statement printf enter your weight scanf weight what does this two together mean.

(Refer Slide Time: 24:31)



I have got my screen here and a program is running and it just says scanf percentage d and w say weight, you do not see anything on the screen, but suppose somebody who is a little more helpful to the user writes printf enter your weight all right. Then here on the screen it is printed enter your weight and then since backslash n is there you are here.

Now, scanf percentage d and w. Now here then when you type your weight say 65 then that is shown here you know what data you are supposed to give. So, this sort of thing makes it more interactive all right. So, here you see these two I have made the program more interactive, but what is the program do? It checks at every point first it checks weight. If the weight is not greater than 65, it will simply come out here because this condition is false otherwise it will carry out all these statements and then it will read the weight again you see again, the weight has been read and after the weight has been read it is coming back here and checking it again whether the weight is greater than 65. If by one days exercise you have reduced your weight then; obviously, you need not exercise more you can come out otherwise you will have to again go and do this. So, while is a very important statement you should understand the meaning of this, and we will try to do a couple of more exercise on examples on this.

Thank you.