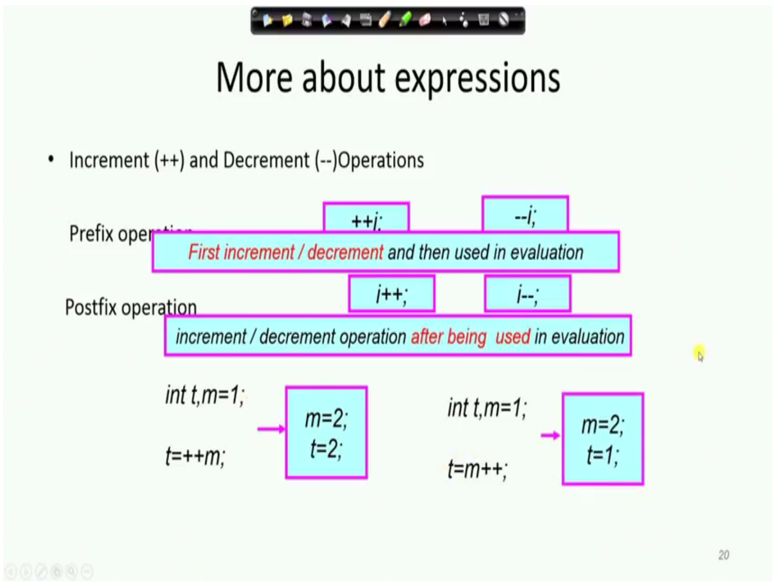
Problem Solving through Programming In C
Prof. Anupam Basu
Department of Computer Science & Engineering
Indian Institute of Technology, Kharagpur

Lecture – 17
Switch Statement

(Refer Slide Time: 00:23)



More about expressions

- Increment (++) and Decrement (--)Operations

Prefix oper

++i; --i;

First increment / decrement and then used in evaluation

Postfix operation

i++; i--;

increment / decrement operation after being used in evaluation

```
int t,m=1;        m=2;
t=++m;       →    t=2;

int t,m=1;        m=2;
t=m++;       →    t=1;
```

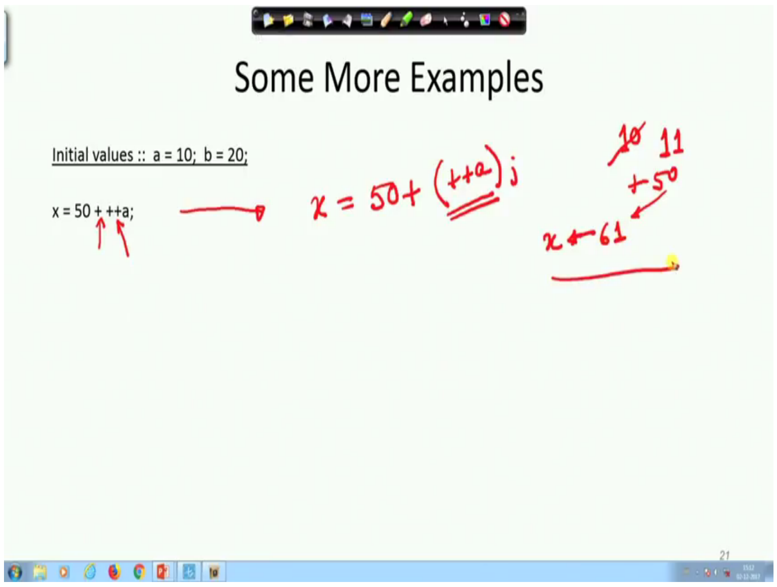
20

In the last lecture we had discussed about 2 new operations, that is post increment and pre increment. Here you can see that we are calling them as prefix operations because the operator is before the operand here you can see also the operator is before operand and what is the operator? Operator is incrementation, but that is being denoted as plus plus twice the addition operation or twice the subtraction operation. And in this here we are talking off we are calling it to be postfix operation, here you can see that the operand is coming first and the operators are following right. So, the operators are following them. So, that is why this is known as postfix here also you can see the operated operand to be I followed by the operator minus minus right. So, that is why these are called postfix or prefix. The significance of these as we discussed in the last class is that a particular variable.

Will be first incremented here in this case it will be first incremented and then used in operation, here it will be first decremented then will be used in the operation. While here it will be first the operation will be done and then the decrementation will be done here first the operation will be done then the incrementation will be done. So, we can, that is what we have written here first increment or decrement and then use it in the evaluation and here increment or decrement is done after being used in evaluation. So, it will be clearer when we come with some examples, see about the example that we have shown in the last class t and m are 2 variables, m is initialized to 1 and when I am saying plus plus m; that means, first I increment m.

So, m will be here you can see as soon as I do that m will become 2 and then that is being assigned to t. So, t becomes 2 whereas, here you can see that m is 1 and we have done post increment. So, first the assignment operation will be done; that means, m was 1. So, sorry m was to here after this I am doing this. So, m will be. So, t will be one. So, what will happen is m is 1. So, that m will be assigned to t. So, t becomes 1and after that it will be incremented. So, ultimately m will be to, but t will be 1. So, if you look at these 2 you will find the difference between these.

(Refer Slide Time: 03:32)



Some More Examples

Initial values :: a = 10; b = 20;

```
x = 50 + ++a;
```

$x = 50 + \underline{(++a)};$

$\cancel{10}\ 11$

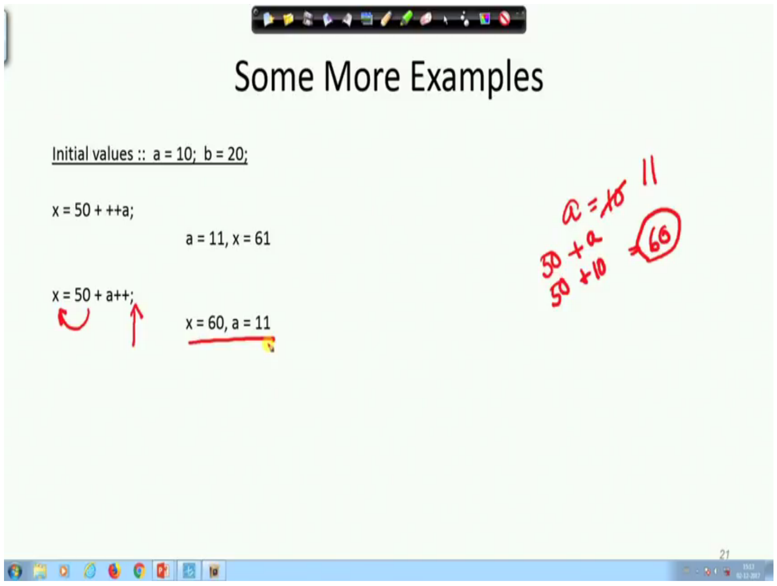
$+50$

$x \leftarrow 61$

Let us have a few more examples, suppose we have got 2 variables a and b whose initial values are 10 and 20. They are initialized to 10 and 20 next we carry out look at this operation very carefully, it looks a little clumsy here there are 2 operators one is this one another one is this one. So, this is pre increment.

So, this is equivalent to x is assigned 50 plus as if I have put a parenthesis, but that is not required that first a will be incremented and then that will be added to 50. So, what result do we expect a was 10. So, first that will be incremented. So, that will be 11 and that will be added to 50 and. So, x will be x we get the value 61 all right that is what will happen in this case. So, a is 11 and x is 61 as we have shown.

(Refer Slide Time: 04:54)



Some More Examples

Initial values :: a = 10; b = 20;

x = 50 + ++a;

a = 11, x = 61

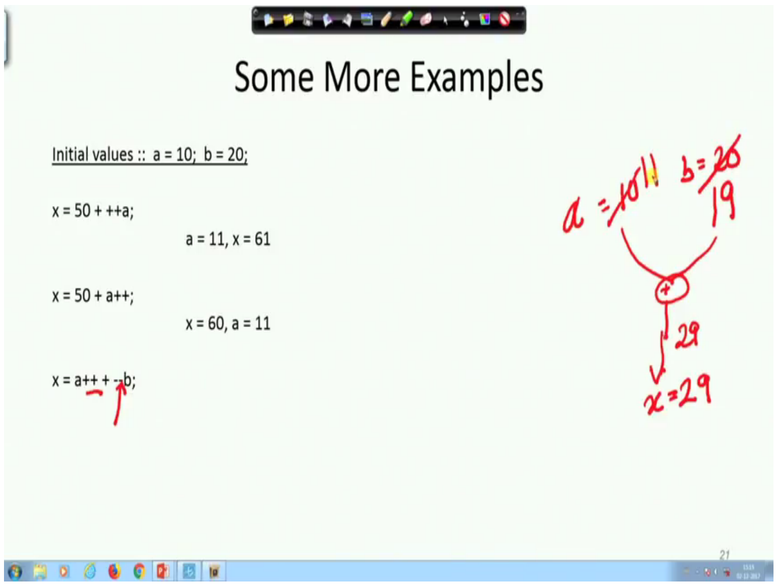
x = 50 + a++;

x = 60, a = 11

Now, let us take this again this is the; this is post increment. So, what do we expect here let us see what will happen in this case, first here after the operation is done then a will be incremented. So, what will happen? X was 50 sorry sorry I am sorry a was 10 and I am adding 50 and 10 and a; that means, 50 plus 10. So, that becomes 60 and that 60 is being assigned to x.

Because, the assignment operation, is taking precedence over this post increment. So, we get x to be 60, but I have my job is not it clear because after is participation in this operation, it will be incremented by 1. So, a was 10 then we will be made incremented by 1. So, it will become 11. So, consequently this is the result what we will get.

(Refer Slide Time: 06:21)



Some More Examples

Initial values :: a = 10; b = 20;

x = 50 + ++a;

a = 11, x = 61

x = 50 + a++;

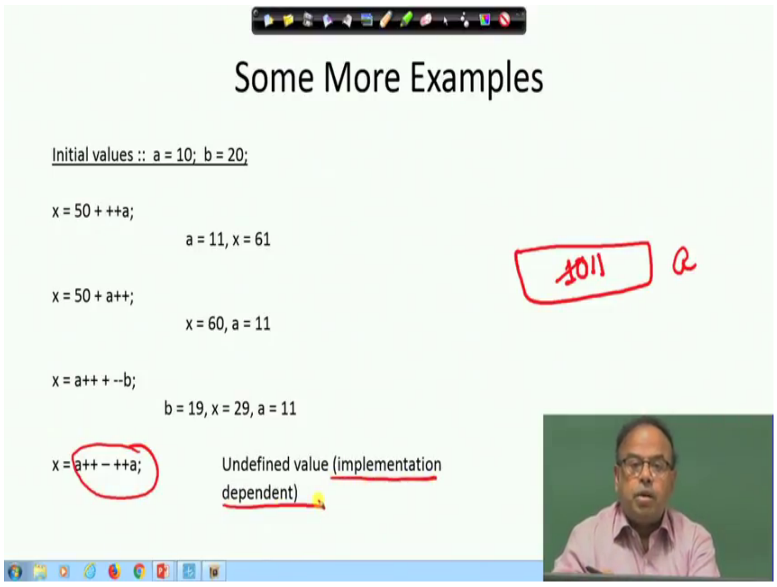
x = 60, a = 11

x = a++ + --b;

Now, let us see again here; what do you expect to happen. A is 10 as usual, a is 10 and b is 20 what is being said here? A is post increment, but b is pre increment pre decrement. So, first b will be decremented. So, b will become 19 decrement means by 1. So, it will be decremented it will be nineteen it will be added to a; what was a? 10.

So, these 2 will be added and what is that 29 and that 29 will go to x. So, x will be 29 and then my job is not it clear not it over. So, this will be incremented and a will be 11 all right. So, this is how the thing will happen.

(Refer Slide Time: 07:36)



Some More Examples

Initial values :: a = 10; b = 20;

x = 50 + ++a;

a = 11, x = 61

x = 50 + a++;

x = 60, a = 11

x = a++ + --b;

b = 19, x = 29, a = 11

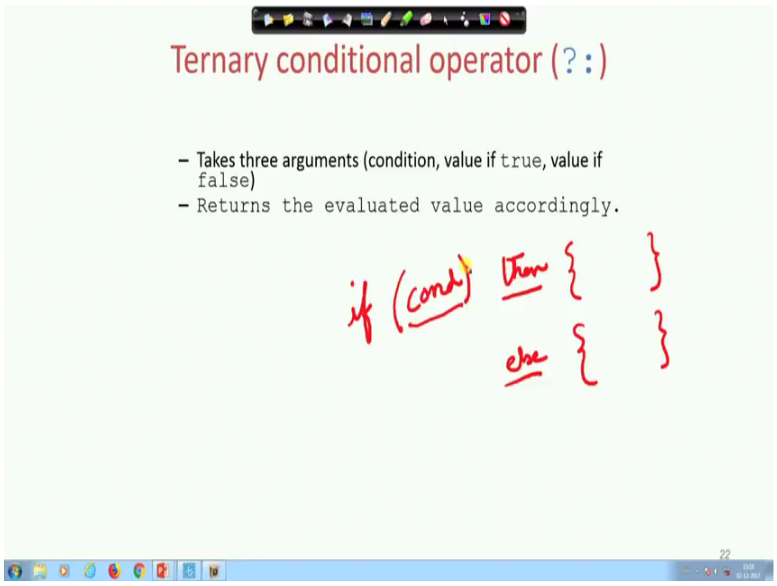
x = a++ – ++a;

Undefined value (implementation dependent)

So, the results that we get in this case will be b will be 19, because b has been decremented x will be 29 and a will be 11. I hope you have understood the flow. So, now this is another one, here we really do not know what to do because here I am incrementing pre incrementing a post incrementing a here now I will take this value of a now what will happen. Suppose the problem here is suppose a is let us take the case here a to be 10 now first.

I will decrementing I will incrementing. So, a will be 11, but a was this a is not incremented. So, a minus 11. So, 11 minus 11, will it be 0 and then this will be a will be 12 or which one will be done first this is a particular scenario, for the same variable in the post increment the post decrement is given often that is implementation dependent and the compiler does not take it well. So, it often results in undefined value. So, you should try to avoid this as much as possible actually you should not use this you should use you can use the pre increment post increment, but as I said in the last class I had strongly suggest that at the initial phase of programming you avoid using this pre increment post increment operations, instead although it will be a little laborious you write x assigned x plus 1 or x assigned x minus 1 for pre increment and pre decrement, but it is good to know that c allows us with this facilities.

(Refer Slide Time: 09:47)



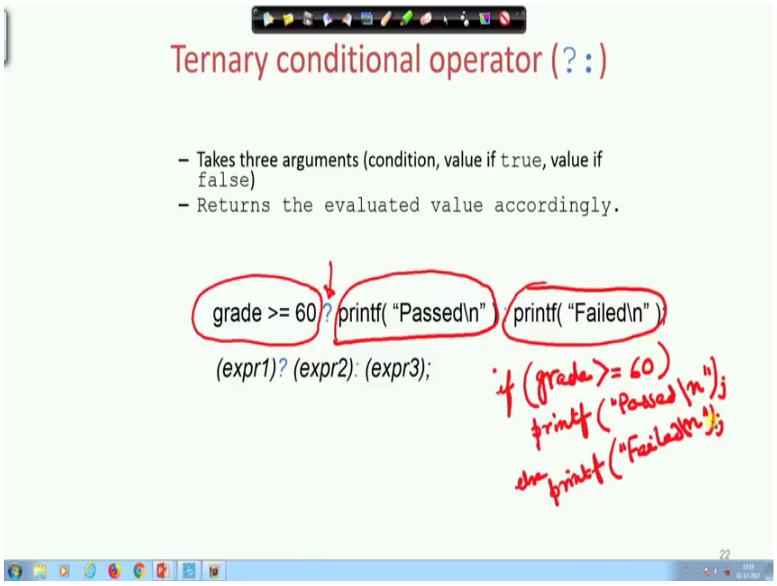
Ternary conditional operator (? :)

- Takes three arguments (condition, value if `true`, value if `false`)
- Returns the evaluated value accordingly.

if (cond) then { }
else { }

Now, here is another one which at the initial phase you should not try to use unless you are very confident. We know if conditions then we do something else something. So, there are three things, if some condition then although we do not write them, then we do some operations else we do some other operation. Then the same thing can be written, it call it a ternary conditional operation here we write it in a different way.

(Refer Slide Time: 10:45)



Ternary conditional operator (? :)

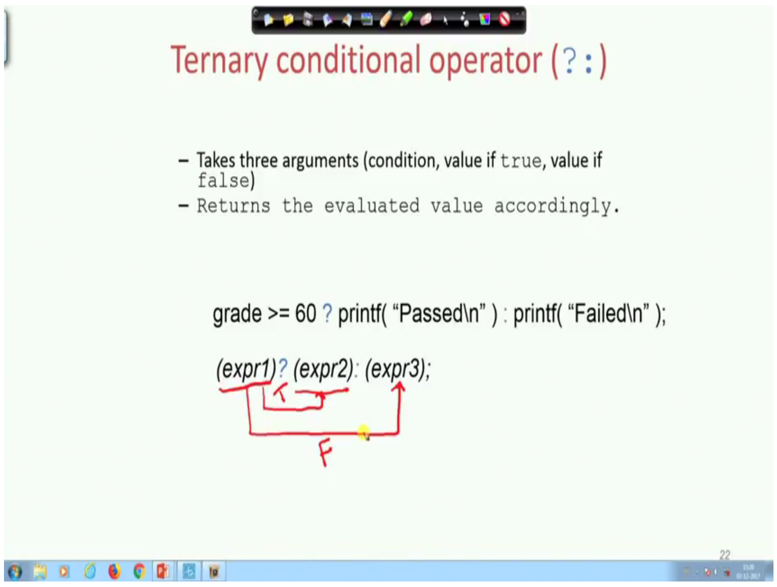
- Takes three arguments (condition, value if `true`, value if `false`)
- `Returns the evaluated value accordingly.`

```
grade >= 60 ? printf( "Passed\n" ) : printf( "Failed\n" );
(expr1)? (expr2): (expr3);
```

```
if (grade >= 60)
  printf ("Passed\n");
else printf ("Failed\n");
```

Just to save space on I mean the size of the program what we do here say grade greater than equal to 60 printf passed otherwise printf failed. So, this means here this is the this part is the condition part and this part is the part that will be computed if the condition is true, and this part will be computed if the condition is false all right. So, this is equivalent to writing if grade greater than equal to 60 printf, past else printf failed all right this is equivalent to that. So, what is happening here in this example? So, the general syntax is if there is an expression there is a condition expression then we carry out if that is true.

(Refer Slide Time: 12:21)



Ternary conditional operator (? :)

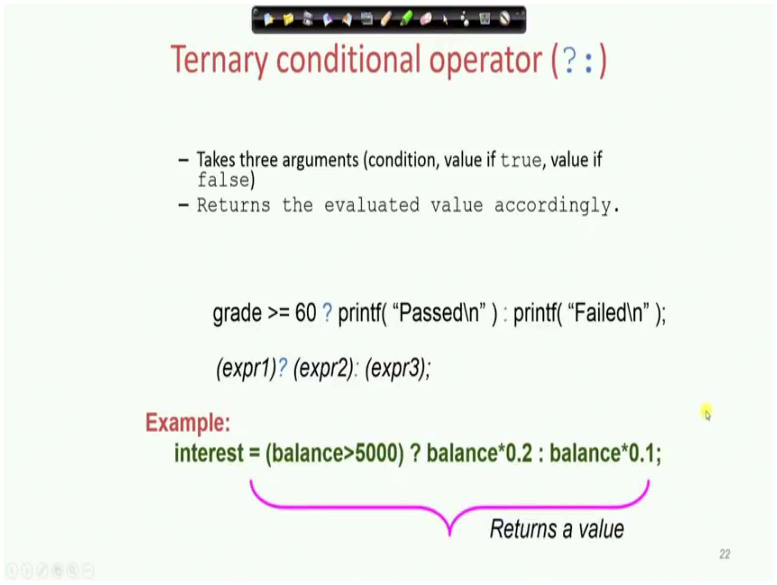
- Takes three arguments (condition, value if `true`, value if `false`)
- `Returns the evaluated value accordingly.`

grade >= 60 ? printf(“Passed\n”) : printf(“Failed\n”);

(expr1)? (expr2): (expr3);

Then we carry out expression true otherwise if it is false, then we carry out if it is true then we carry out this part this part, that is fall following interrogation mark, and we carry out the expression three that is the expression that is following the colon mark for the else case or if the condition is false. So, this is again writing that if then else in a cryptic shorter way this is known as ternary conditional operation, but again I would say that this is a specialty of c it is good to know that, but initial phase at the initial phase of programming I discourage you to use this .

(Refer Slide Time: 13:11)



Ternary conditional operator (? :)

- Takes three arguments (condition, value if `true`, value if `false`)
- `Returns the evaluated value accordingly.`

grade >= 60 ? printf(“Passed\n”) : printf(“Failed\n”);

(expr1)? (expr2): (expr3);

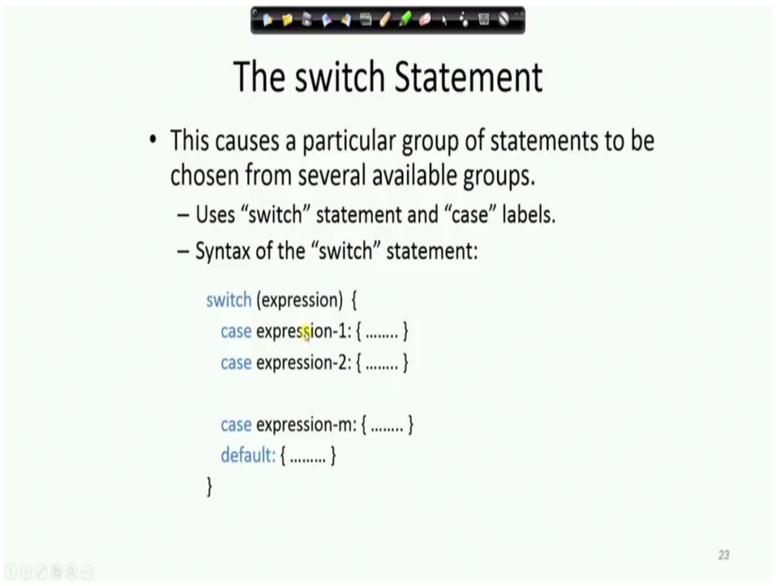
Example:

interest = (balance>5000) ? balance*0.2 : balance*0.1;

Returns a value

So, here is an example what will this v can anyone tell me? We are computing the interest on the amount of money you have got in bank, this means here let us look at this part this is a ternary operator. If the balance is greater than 5000 then you compute balance times 0.2 that means, 20 percent of the balance or compute 10 percent of the balance. If somebody has got a balance 6000 then have 20 percent of that computed and assign it to the interest part. So, you see a bigger statement if balance is greater than 500 then interested equals balance multiplied by 0.02 else interest equal to balance multiplied by 0.01 all these things can be done in one single sentence or one single statement using c all right. So, that is why some people prefer to use such ternary operators and you will also certainly use it, but when you are very confident about your programming. So, this.

(Refer Slide Time: 14:48)



The switch Statement

- This causes a particular group of statements to be chosen from several available groups.
  - Uses "switch" statement and "case" labels.
  - Syntax of the "switch" statement:

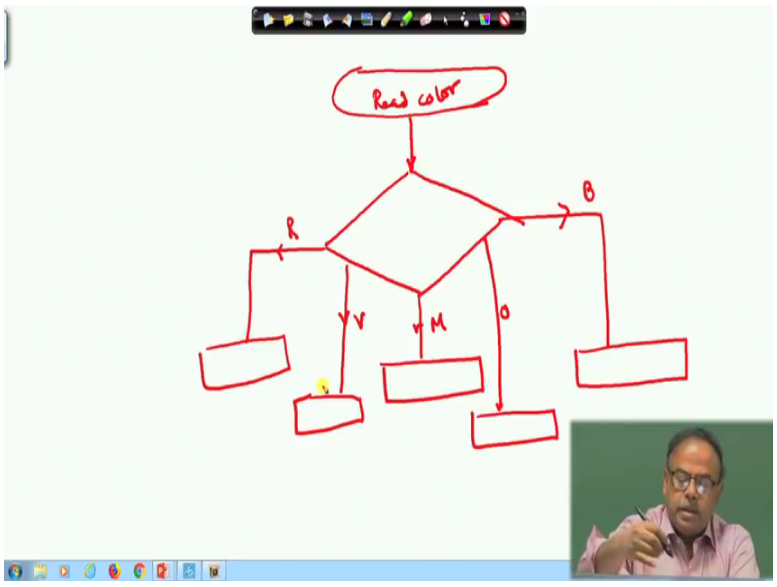
```
switch (expression) {
  case expression-1: { ........ }
  case expression-2: { ........ }

  case expression-m: { ........ }
  default: { ......... }
}
```

23

Now, we come to a new construct you know that is the switch construct. Now in order to understand this. So, let us quickly go to the flow chart for a while see I draw flow chart like this.

(Refer Slide Time: 15:03)

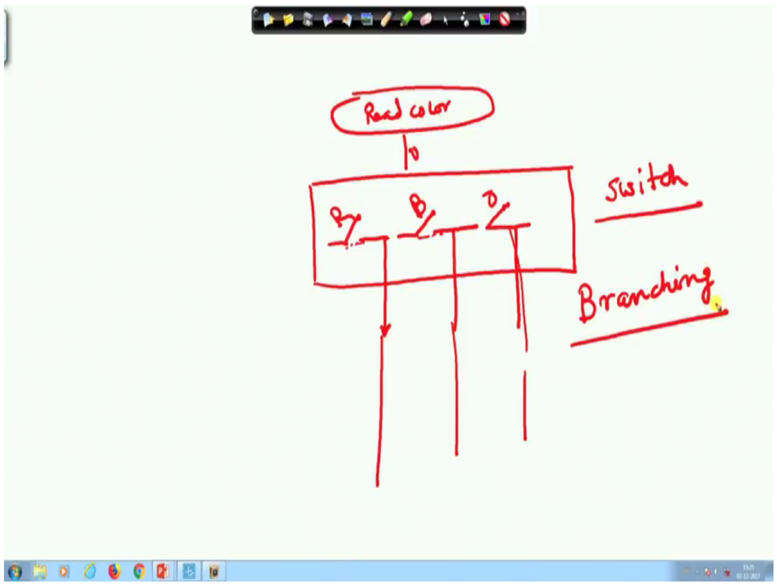


I am evaluating some values here all right or maybe I get some value from the user here, I read some value let me make it clearer.

I read some value from the user all right something read colour now here depending on the colour that the users applies earlier in this decision box whatever you doing? We were either going for true or for false. Now here what we are seeing if the colour is red then we do something, if the colour is blue we do something if the colour is magenta, we do something if the colour is orange then we do something if the colour is violet we do something ok.

So, the same diagram I can draw in a different way that is.

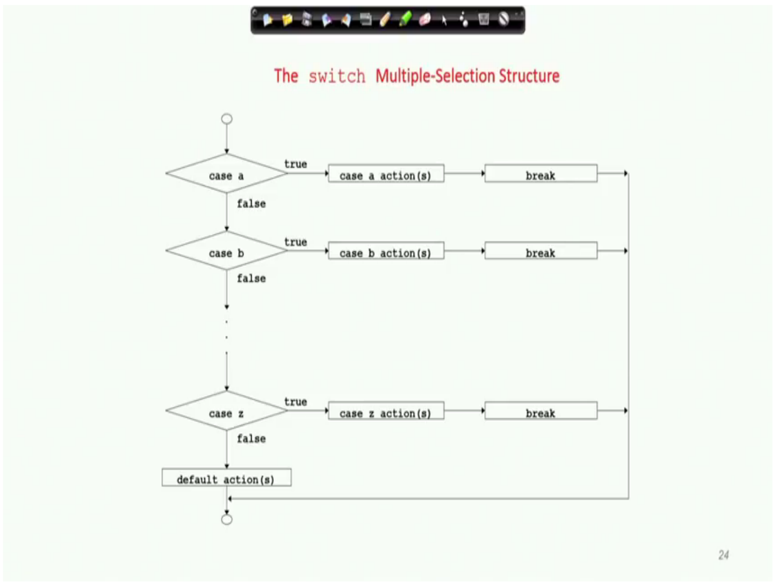
(Refer Slide Time: 16:36)



I am reading colour and then as if I am coming to a switch box, think of this to be a switch box where number of switches are there all right and number of switches are there now depending on what value of the colour is coming. So, this may be the red switch, this will be the blue switch, this one with orange switch, depending on which switch which value is coming. So, let the value be v. If v is r then this switch gets closed and we follow this path. If the v is blue then this switch gets closed and we follow this path. If v is orange then then this switch gets closed and we follow this path all right that is why because of this analogy which switch this statement is also known as switch statement, this is also very much use for branching.

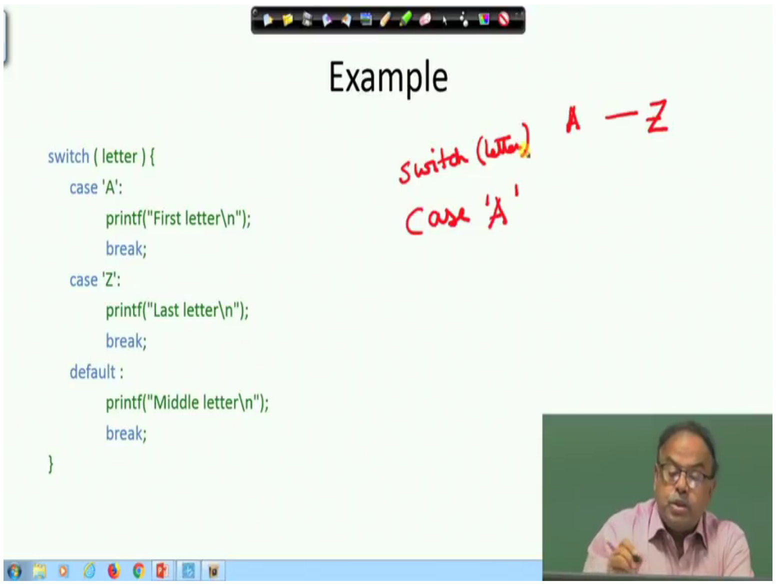
Branching we have seen with this else type of thing and here, also we will see another variety of a c construct called switch using which we will do that. So, let us see here now let us look at the construct of this. The syntax of the switch statement is this switch expression then case expression 1 expression 2 let me give an example first and it will be better I will come back to this.

(Refer Slide Time: 18:36)


The switch Multiple-Selection Structure

So, here is the total switch structure, it is a multiple selection structure. So, I come to this point switch if the case is a, if it is the case that the colour is red like that if the case is a then I take the case a actions and then come out M otherwise I take the case reactions in the case be b then only I will take the case b action and then come out.

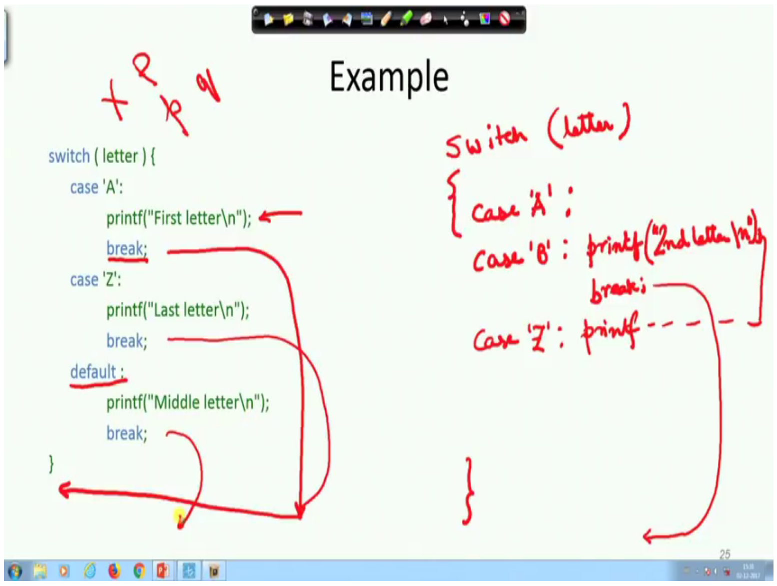
(Refer Slide Time: 19:18)


Example

```
switch ( letter ) {
    case 'A':
        printf("First letter\n");
        break;
    case 'Z':
        printf("Last letter\n");
        break;
    default :
        printf("Middle letter\n");
        break;
}
```

Similarly if there may be different cases. So, what we are getting here is an advant is an example, that we are showing here what we are getting here is using this case construct multiple branching we are accommodating in one shot. L et us look at this I was giving an example with colour here it is an example with a letter. So, I have read some letter from the user some letter has been now that letter can be anything A to Z say A to Z. So, if it be if the case is that the letter is A. Look at how it has been written case A before that switch what is my variable on which way I am switching? Switch on variable letter.

(Refer Slide Time: 20:19)



Example

```
switch ( letter ) {
    case 'A':
        printf("First letter\n");
        break;
    case 'Z':
        printf("Last letter\n");
        break;
    default :
        printf("Middle letter\n");
        break;
}
```

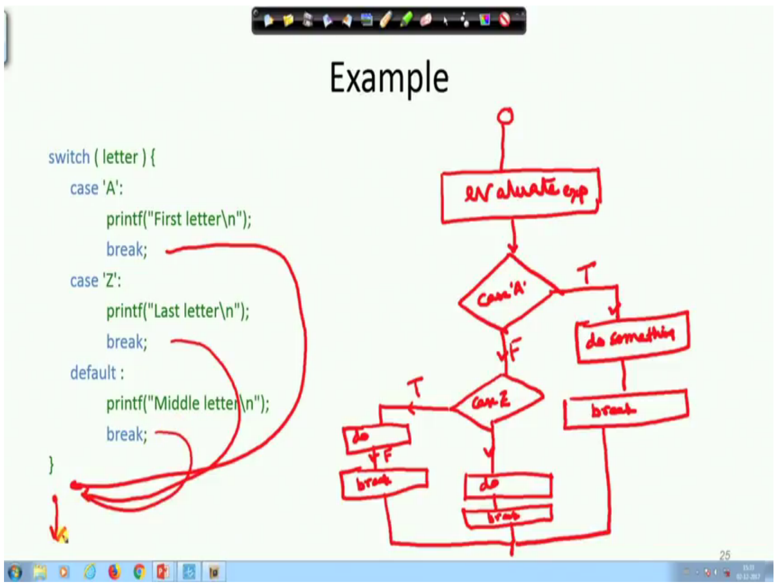
In the earlier example it was switch on the variable colour. So, here we start with switch on the variable letter.

And what are the possible switches I encapsulate that in 2 braces just as we do. Now within this I am taking multiple decisions multiple possibilities are there for example, if the letter is a; that means, if it is the case that the letter is A then we put a semicolon then I write down print the first later. So, what will be printed first later and then there is a break statement what this break statement does? That you execute this and then you come out of this entire switch box come out here; that means join the next statement. If the letter was B then here it is not written I am writing case B printf second letter backslash n and then break; that means, if the case is B then.

I will print this and after that with the break I will come out of this now after that I have not written anything else, but I have written case z only this is case z all right printf last letter as is shown here and then break. Now if somebody this letter was p if this letter was q or some other alphabet sorry I mean anything small alphabet or capital alphabet x whatever in that case what do I do? This one only tells me what I can do if the letter is A or if the letter is Z or in my case if the letter is B, but if it B something else otherwise then we use this statement called the default statement if anything is other than what has been specified here then we will say that it is some other letter and.

Then break it you see here if the letter is a then only I entered here and then went out if the letter is b then I only entered here if the letter is c then I will come to default if the letter is c it is not specified here I will come here and I will escape I will go right it is an intermediate letter and then come out. So, with this understanding if we can go back to the earlier slide, the flowchart then we can see that the flow chart will be something like this.

(Refer Slide Time: 24:00)



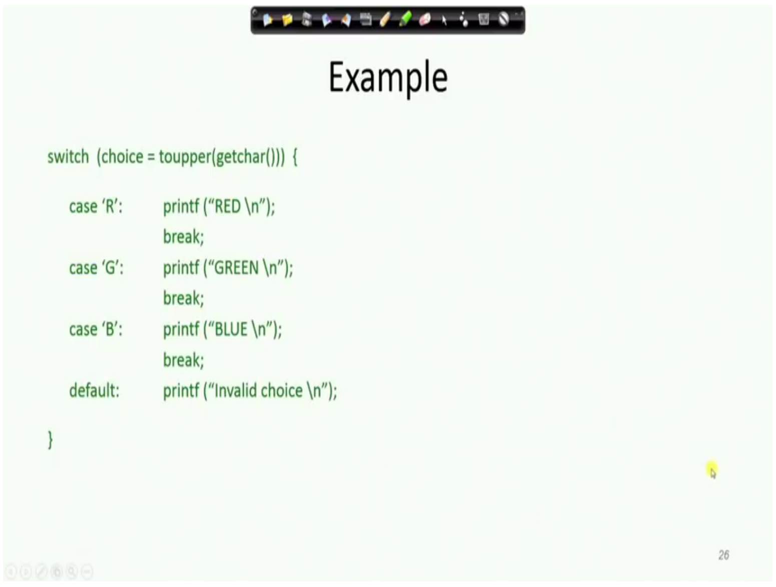
Example

```
switch ( letter ) {
    case 'A':
        printf("First letter\n");
        break;
    case 'Z':
        printf("Last letter\n");
        break;
    default :
        printf("Middle letter\n");
        break;
}
```

We start then we evaluate expression, we evaluate expression come here its equivalent to case a yes do something do something and then break otherwise it will come here is looking at I am just doing this case z, if true then do something oh and then break right otherwise.

So, this is true this is false, this is true this is false otherwise default I will come here do something and then break now all these breaks all these breaks brings me to the next statement after the case statement. So, anything that is here whenever I encounter a break this comes brings me here this also brings me here, this also brings me here and I continue following the sequential nature of program from here all right.

(Refer Slide Time: 25:43)



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");

}
```

26

So, we have a seen. So, here is another nice example we can look at it say here switch, I am switching on an expression what is that? Choice to upper now if this has got some multiple parameters into this, let me take it up in the next lecture so that I can devote some time on this.

Thank you.