

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

Lecture – 14
Use of Logical Operators in Branching

In the last lecture, we had an idea of what control structures are. Control structures are some language structures which by which we can change the flow of control which normally is sequential that means, one executed after another.

Now, in the language we just as we have statements arithmetic expressions logical expressions all those are statements, in our language c language we also have some control statements for achieving the control structures.

(Refer Slide Time: 01:02)

The slide contains the following content:

- Allow different sets of instructions to be executed depending on the outcome of a logical test.

The flowchart diagram shows a diamond-shaped decision node containing the logical test $x > 0$. An arrow enters the top of the diamond. Two arrows exit the diamond: one pointing downwards labeled 'Y' (Yes) and one pointing to the right labeled 'N' (No). From the 'N' path, an arrow points upwards and then another points downwards to the right.

A small video inset in the bottom right corner shows a man speaking.

What they do? As I have said allow different sets of instructions to be executed depending on the outcome of a logical test. So, if we go back to the flowchart, we are coming in a sequential flow, at this point we encounter decision point and at this decision point we carry out some logical tests for example, is x greater than 0 yes or no. If it be yes there will be some flow and if it be no there will be some other flow all right some of not necessarily backward it can go somewhere else etcetera all right. So, based on the logical test; that means this test which will result in a logical value and what are the logical values? Logical values are either 0 and 1 either true or false.

(Refer Slide Time: 02:09)

What do they do?

- Allow different sets of instructions to be executed depending on the outcome of a logical test.
 - Whether TRUE or FALSE.
 - This is called branching.

The diagram shows a red diamond shape with an arrow pointing down into the top vertex, an arrow pointing right from the right vertex, and an arrow pointing down from the bottom vertex. This represents a branching point in a control flow graph.

So, whether true or false that is a; this is called branching. So, again whenever I am coming to a decision box and from there I am branching out, either in this direction or in some other direction all right. So, this is a branching branch point. So, that is change of the control flow.

(Refer Slide Time: 02:37)

What do they do?

- Allow different sets of instructions to be executed depending on the outcome of a logical test.
 - Whether TRUE or FALSE.
 - This is called branching.
- Some applications may also require that a set of instructions be executed repeatedly, possibly again based on some condition.
 - This is called looping.

Some applications they also require that a set of instructions be executed repeatedly, they will be going they will be executing as in the case reading the numbers c numbers. I will be repeating reading a number adding that with some I mean decrementing count

checking whether count is equal to 0, and then again go back and read the number and this slope will continue based on some condition. How long in the earlier example how long I will where we doing that based on what condition? The condition was whether count is 0 or not. As long as the count is not zero we are going on doing this. So, this is also known as looping doing the same repeatedly, this is also known as looping. So, branching or if we go back to the earlier one then it is called a loop.

(Refer Slide Time: 03:48)

How do we specify the conditions?

- Using relational operators.
 - Four relation operators: $<$, $<=$, $>$, $>=$ (L.E)
 - Two equality operations: $==$, $!=$
- Using logical operators / connectives. (L.E) || (L.E) && (L.E)

Now, do we specify the conditions? By now I am sure you have get how we do that. We specify the conditions using relational operators or there are four relational operators you have seen; less than, less than equal to, greater than, greater than equal to. I have said not equal to also, but not equal to can also be seen as the negation of the equality operator. The equality operator is this double equal to sign and not equal to is exclamation mark for the equal to now. So, using this relational operators, we will get using one single any of these one single relational operator, we will get one logical expression and using logical operators and connectives, we will be able to connect a number of logical operations logical expressions and logical another logical expression. In that way we can carry out we can generate the condition using this.

(Refer Slide Time: 05:32)

How do we specify the conditions?

- Using relational operators.
 - Four relation operators: $<$, $<=$, $>$, $>=$
 - Two equality operations: $==$, $!=$
- Using logical operators / connectives.
 - Two logical connectives: $\&\&$, $||$
 - Unary negation operator: $!$ \rightarrow NOT

Handwritten notes:

- $x = 5;$
- $(x > 5) \rightarrow 0$
- $!(x > 5) \rightarrow 1$
- Diagram: $!(LE) \rightarrow 0$ (with a crossed-out 0)

Now, two logical operators we have seen and or. Another unary negation operator we did not discuss earlier, but we are doing it now that is this means not all right. For example, suppose x is 5 and at some point I compute x greater than 5, the result of this logical expression will be false right this is false, but if I take the negative of this then this false will be negated and the negation of that will be taken. So, that will be one two; that means, what is the meaning of this, is it the case I can read it in this way, is it the case that x is greater than 5 is false is not true yes that statement is true. I can state it in another way is it the case that x is not greater than 5? Yes that is the case again I tell the first one that is more complicated is it the case that the statement is greater than 5 is not true that statement is true? That it is not true is true. So, that is not means whatever logical expression I have after this that will be negated. So, it is equivalent to not of the logical expression.

So, if this logical expression if this logical expression leads to this logical expression is only 1, then not of that will invert it and make it 0 or if this logical expression was 0 then this not of this will make it 1. So, this is the unary negation operation all right.

(Refer Slide Time: 08:08)

Examples

```
count <= 100
(math+phys+chem)/3 >= 60
(sex=='M') && (age>=21)
(marks>=80) && (marks<90)
```

Let us see some examples count less than equal to 0 this is the logical expression it will be either true or false maths plus physics; that means, here maths is the marks obtained by a student in maths plus physics, plus chemistry divided by 3 is greater than or equal to 60 that means, the average of these 3 marks is greater than or equal to 60.

Either the person the sex of that person is male sorry and let us look at this if I am using a logical connective the person this will be true if the person is a male and he age is greater than or equal to 21. So, for a female of age 22 this will be false, for a male age 20 it will be false. A male age 25 it will true. Here again another one with this connective marks is greater than 80 and marks is less than 90 sorry marks is greater than equal to 80 and marks is less than 90; that means, what anything starting from 80 to 89, if the marks is between this then this will be true .

(Refer Slide Time: 09:39)

Examples

```
count <= 100
(math+phys+chem)/3 >= 60
(sex=='M') && (age>=21)
(marks>=80) && (marks<90)
(balance>5000) || (no_of_trans>25)
!(grade=='A')
```

Another one with an or operator balance is greater than 5000 or the number of transaction is greater than 25 say in a bank see situation for example, you are making a lot of transactions what is the transaction when you deposit a money that is a transaction, when you withdraw the money that is also a transaction. So, there are different transactions. So, what we are saying here either I have got 5000 rupees greater than sorry greater than 5000 rupees in my balance or I have done more than 25 transactions. If any one of them true suppose you have got 6000 rupees in your balance and number of transactions is 26, then also this is true. If the balance is 6000 and number of transactions you have done is only 20 then also it is true, we have already explained the role of or operator earlier ok.

Now, this if the grade of the student is a if a student has got grade a what will be the value of this logical expression true or false if the student has actually got the a grade all right. So, this one this part will be true, but my actual logical expression is this one which is a negation of that. So, it will be false because the person has grade a what does it means? It means that is it the case that the student has not got grade a.

(Refer Slide Time: 11:35)

Examples

```
count <= 100
(math+phys+chem)/3 >= 60
(sex=='M') && (age>=21)
(marks>=80) && (marks<90)
(balance>5000) || (no_of_trans>25)
!(grade=='A')
!((x>20) && (y<16))
```


Here is another little more complicated x greater than 20 and y less than 16, but the way we should evaluate this is, I will evaluate each of them separately suppose x is greater than 20 suppose 21. So, this is true and y is 15. So, this is true. So, both these together is true then I take this one with this and so, the result will become false let me clear it again suppose x is 21. So, this part is true then I compute this y is 17, this part is false therefore, these two are connected by this ampersand; that means, logics are not I a m sorry very sorry it should it is not ampersand double and; that means, and operator by logical and. So, these two will result in though the logical and it will result in 0, but my actual expression is the negation of that. So, the result will be 1 clear.

(Refer Slide Time: 13:09).

The conditions evaluate to ...

- Zero
- Non-zero

A hand-drawn diagram shows a diamond-shaped decision box labeled "Test". An arrow points from the left side of the diamond to the left, and another arrow points from the bottom of the diamond downwards. There are also red lines underlining the words "Zero" and "Non-zero" in the list.




So, that is how we carry these are the examples of evaluation of logical expressions and logical expressions. So, each of them each of these logical expressions can serve as a condition in our control structure in our this loop in our this diamond box, decision box where we test or let us call it the test box in that any of these sort of logical expressions can reside and the result of that can be either 0 or non-zero false or true ok.

Accordingly we will branch out in either this direction or in some other direction.

(Refer Slide Time: 14:00)

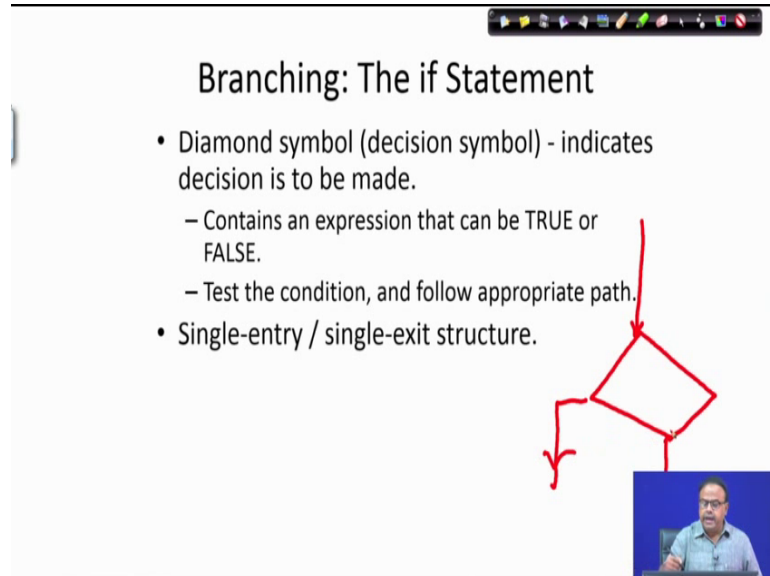
The conditions evaluate to ...

- Zero
 - Indicates FALSE.
- Non-zero
 - Indicates TRUE.
 - Typically the condition TRUE is represented by the value '1'.



So, 0 indicates false and non 0 indicates true typically the true is reprinted by the value one in most of the machines.

(Refer Slide Time: 14:14)



Branching: The if Statement

- Diamond symbol (decision symbol) - indicates decision is to be made.
 - Contains an expression that can be TRUE or FALSE.
 - Test the condition, and follow appropriate path.
- Single-entry / single-exit structure.

The slide includes a red flowchart diagram of a diamond-shaped decision symbol. A red arrow enters the top vertex of the diamond. Two red arrows exit from the left and right vertices of the diamond, representing the true and false paths respectively. A small inset video of a presenter is visible in the bottom right corner of the slide.


This is something that we have already seen, the decision symbol in the flowchart indicates the decision to be made it contains an expression that can be true or false, test the condition and follow the appropriate path that is how we will do is a single entry single exit structure; that means in this diamond box we will enter through any one point and we will exit from only one point either this or this, simultaneously we cannot come out of this because the thing cannot be true or false at the same time therefore, any one exit will take.

(Refer Slide Time: 15:09)

Branching: The if Statement

- Diamond symbol (decision symbol) - indicates decision is to be made.
 - Contains an expression that can be TRUE or FALSE.
 - Test the condition, and follow appropriate path.
- Single-entry / single-exit structure.
- General syntax:
`if (condition) { }`

Handwritten examples in red:
`if (x >= y)
printf ("x is large")`



The general syntax, syntax means what? Syntax means the grammar; the exact grammatical structure that we should write is if condition then I carry out some operations. You have seen in an earlier lecture when we were comparing two numbers x and y and would be printing x is large is x is greater than y then what did we write we wrote something like this, we wrote if x is greater than equal to y, print f what we printed y x is greater than equal to y, x is large or not in this form we wrote down that x is large here all right. So, if this condition is true then I will be doing this print operation, x is large then black slash end I am not being able to write very clearly here.

(Refer Slide Time: 16:48)

Branching: The if Statement

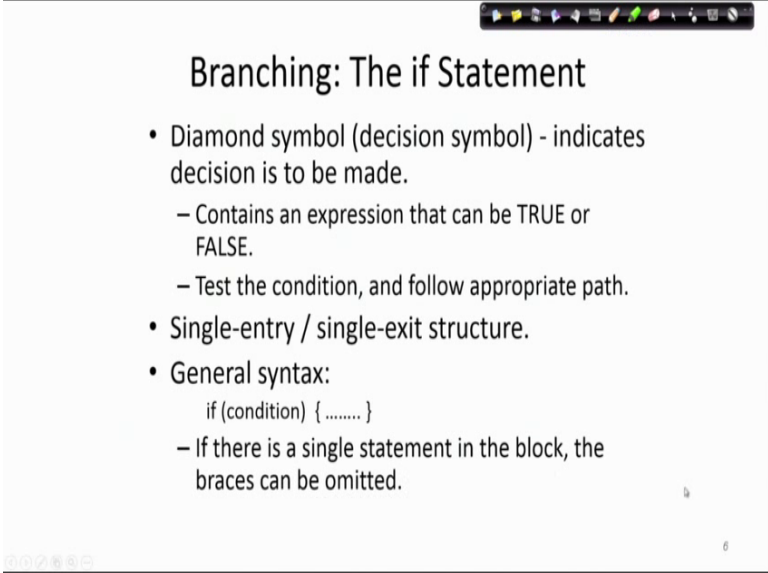
- Diamond symbol (decision symbol) - indicates decision is to be made.
 - Contains an expression that can be TRUE or FALSE.
 - Test the condition, and follow appropriate path.
- Single-entry / single-exit structure.
- General syntax:
`if (condition) { }`

Handwritten examples in red:
`if (x >= y)
{ printf ("x is large"); }`

So, I try again if x is greater than y or greater than equal to y. So, this is the condition you see this is this is the reserved word in c end the condition. If the condition is true then this is followed by a statement. Now if it be a single statement I do not need to give this curly bracket, but just I am giving it just to keep the parity, here I write print f x is large then black slash end right and we end this statement ok.

So, if this condition is true, then this statement will be executed that is my syntax otherwise some other statement will be executed. Now this is something which takes some times to settle down. So, we will explain it repeatedly to some extent.

(Refer Slide Time: 18:15)



The slide is titled "Branching: The if Statement" and contains the following content:

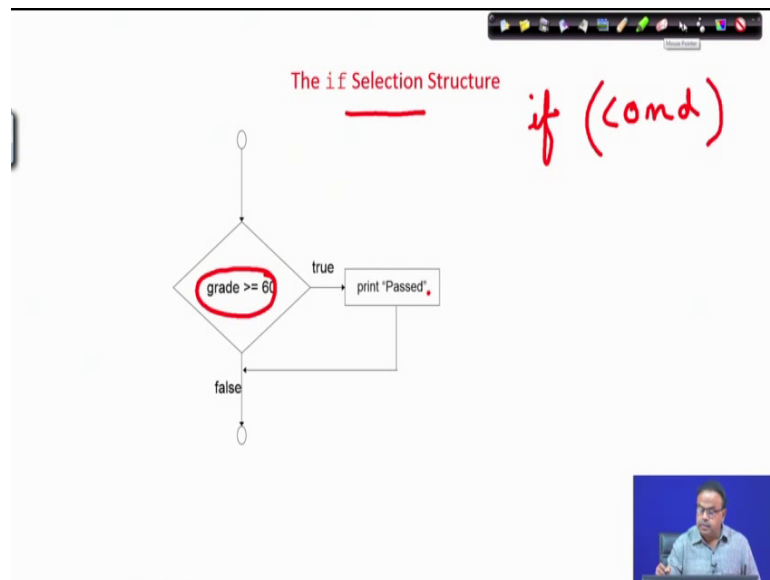
- Diamond symbol (decision symbol) - indicates decision is to be made.
 - Contains an expression that can be TRUE or FALSE.
 - Test the condition, and follow appropriate path.
- Single-entry / single-exit structure.
- General syntax:

```
if (condition) { ..... }
```

 - If there is a single statement in the block, the braces can be omitted.

If there is a single statement in the block this braces can be omitted I am sorry this braces can be omitted.

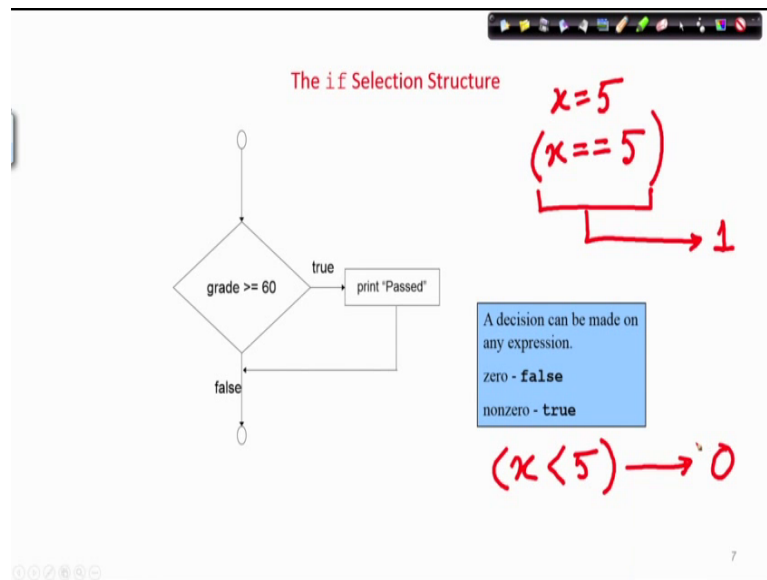
(Refer Slide Time: 18:19)



Now, let us come to this structure. If statement first of all we are looking at the first and very important control statement in c that is the if statement, if some condition. So, we come here the structure is we carry out some tests on some condition grade is greater than equal to 60 if it is true, then we do something I print past otherwise I would write something else. So, this is a selection structure what is it selecting? It is selecting which path should I go all right.

Now, you see here we have come we are coming to this path, if this here I carried out a test if this test is true I do this do this execute this statement, and come back and continue not come back and then continue to this part. If this statement is not true; that means, if this test fails then I will not take this I will not select this path, I will continue in the path which in which I was going right. So, that is why this is often known as a selection structure.

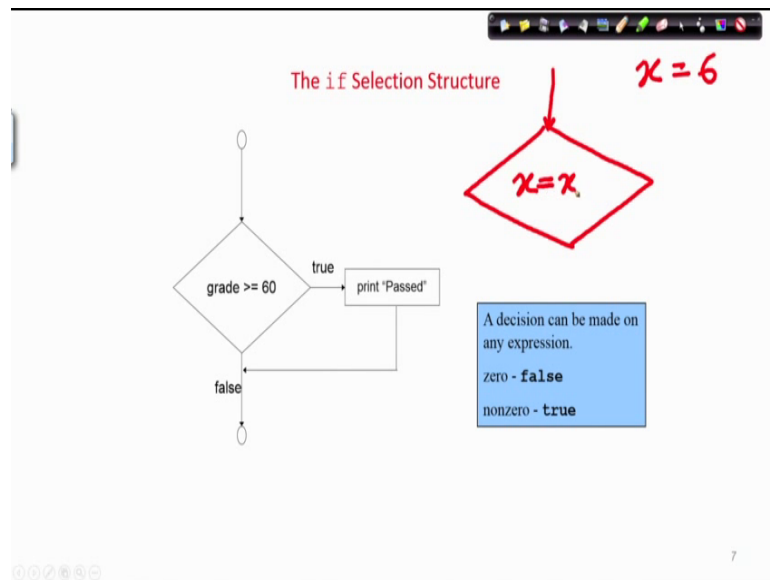
(Refer Slide Time: 20:06)



So, a decision can be made on any expression. Here you see only thing that we are trying to do is test whether some condition is true or false. Now we if the result of an expression is 0 then we take it to be false and if it be non 0 then we will take it to be true. If I evaluate something to be true like say for example, if x was 5 and I carry out x is x equal to 5, this logical expression we will evaluate to true; that means 1 that means, it is non zero.

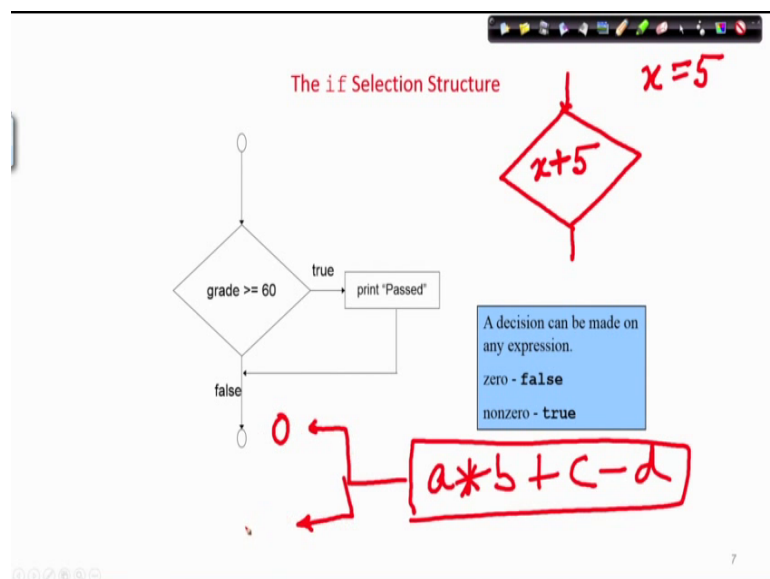
If I had written x less than 5 then it would have resulted in false it would be 0. Now suppose I simply do something different I write in this decision box I am coming like this.

(Refer Slide Time: 21:40)



And inside the decision box I write and x was 6 say I do not write a logical expression here I write an arithmetic expression x or let me x was 5, I carry out some expression x plus 5.

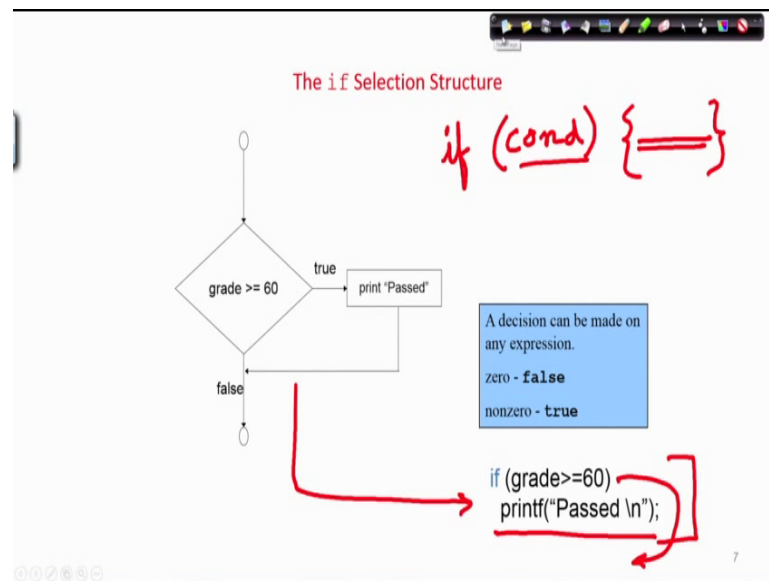
(Refer Slide Time: 22:13)



What will be the result of this expression be? It will be ten that is it is non 0 then also it will be taken as true although we will we do not need it all the time you need not bother about it, but in general a decision can be made on any expression. Typically and most conveniently we will be using logical expressions, but if I carry out an arithmetic

expression something a times b plus c minus d, then also this will give me a value and this value gives 0 means it is false and not 0 then it will be taken as true however, this is just a matter of detail you need not get too much bothered about it right now, we will try to keep as much as possible relational expressions or logical expressions inside this text blocks.

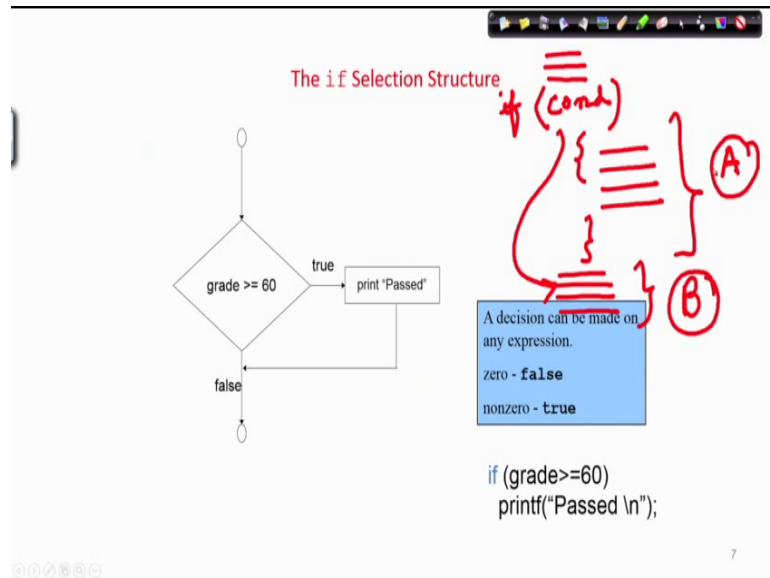
(Refer Slide Time: 23:45)



So, here if grade is greater than or equal to 60 being passed 10; that means, this is the equivalent of this structure that we have shown here. This flowchart can be simply written as this statement I think probably you will be able to read it now, if has been given in a different colour because this is what we have introduced here this if is a use of if the selection structure followed by a condition if the condition is true then this statement will be executed.

If the condition is false, then this statement will not be executed we will bypass this statement. Now please remember this, when I write if some conditions some statements this statements will be executed only if this condition is true, I write it in a different way now.

(Refer Slide Time: 24:49)

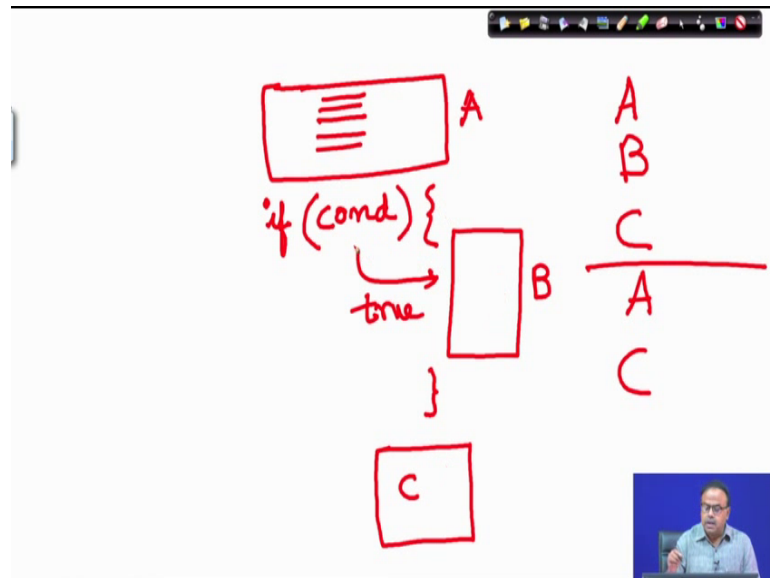


Say for example, I have a got a number of statements, some statements are there and then here there is an if statement. If condition then some statements here may be more than one statements are in a brace. Now if the condition and after that there are other statements.

Now, if the condition is true, then only these statements will be executed lets name this statements to be a then this block a will be executed this block will be executed. If this condition is false then this block A will be bypassed and on false I will come here and come to so, this block b. So, if the condition is false then the block B will be executed, if the condition is true then this block A will be executed all right.

Now, suppose let me explain it in a little different way I am drawing another diagram.

(Refer Slide Time: 25:59)



Now, here is some block of statements, let us call the block of statements to be a and then I have one statement, if condition n whatever the condition is, then within parentheses there is a block of statements B. And then there is a block of statements C. So, the flow of I mean the flow will be that first A will be executed then if the condition is true, then if the condition is true then B will be executed after that what will happen? After that again we will come here and C will be executed.

Now, if the condition were false condition were false, I just computed a after that what would have computed condition will be checked B will not be evaluated not be executed because condition is false after that C will be executed. So, please note that in this structure I am computing I am executing C either way, now the choice of whether b is being done or not is dependent on the condition. So, this gives you an idea of procedure if the condition is false then also this part will be executed. So, here we will conclude today with this example, small example program lets try to understand each line of them.

(Refer Slide Time: 28:07)

Example

```
#include <stdio.h>
main()
{
    int a,b,c;
    scanf ("%d %d %d", &a, &b, &c);
    if ((a>=b) && (a>=c))
        printf ("\n The largest number is: %d", a);
    if ((b>=a) && (b>=c)) X
        printf ("\n The largest number is: %d", b); X
    if ((c>=a) && (c>=b)) X
        printf ("\n The largest number is: %d", c); X
}
```

a - 20
b - 10
c - 5

8

We start with include s t d i o dot h I hope you have not forgotten that any c program must have a main program main function and here is the main function. Inside the body of the main function what is being done I have declared what is being done here, I have declared 3 variables a b and c. Then what am I doing here observe this line scan f percentage d percentage d percentage d and a and b and c. So, I am needing 3 variables a b c ok.

Now, what am I doing here, I am evaluating an if condition if a is greater than equal to b and a is greater than c. So, you understand that if a is a number b is a number c is a number a is greater than b and a is also greater than c; that means, a is the largest number. So, then I change this, I change this only if this condition is true otherwise I will do something else. Now suppose a is greater than b suppose a is 20, b is 10 and c is 5. So, this one will be true then the largest number is a that we printed, next statement will come to is another if statement.

What is being done here? If b is greater than equal to a and b is greater than or equal to c b is not greater than equal or to a therefore, this does not evaluate it to be true therefore, this statement it does not execute. Next I come to this statement I am showing this I am following this flow, but only branch that I took is if this condition was true, then I printed this in this particular case of data in this case of data for some other case should be very

different now if c is greater than a not true. So, this is false this will not be executed. So, the only thing that we printed is a largest number is a.

(Refer Slide Time: 30:43)

Example

```
#include <stdio.h>
main()
{
    int a,b,c;
    scanf ("%d %d %d", &a, &b, &c);
    if ((a>=b) && (a>=c))
        printf ("\n The largest number is: %d", a);
    if ((b>=a) && (b>=c))
        printf ("\n The largest number is: %d", b);
    if ((c>=a) && (c>=b))
        printf ("\n The largest number is: %d", c);
}
```

$a = 10$
 $b = 20$
 $c = 5$

Now, if for example, a was 10, b was 20 and c was 5 then what would have happened let us see here a is greater than or equal to b I try to test this fail. So, this will not be executed. So, I come here right and then I test this b is greater than a right and b is greater than c that is also right. So, this condition passes.

So, then I execute this I print larger number is b. I print that and then I come to the next statement what happens in this statement? C is greater than or equal to a false. So, the only thing that is printed is the largest number is b all right. So, we stop here we will continue our explanations in the next lecture further, we will have we will need to discuss it and also try it a little bit more, because from this point onwards our logic and logical constructs are becoming started becoming just a little more complicated, but is very interesting and it is very easy to understand, if you just try to think about it logically.

Thank you very much.