




# IIT Madras

ONLINE DEGREE

**Computational Thinking**  
**Professor Madhavan Mukund**  
**Department of Computer Science**  
**Chennai Mathematical Institute**  
**Professor G. Venkatesh**  
**Indian Institute of Madras**  
**Tutorial**

(Refer Side Time: 00:13)



```
count = 0
while (Pile 1 has more cards) {
    Read the top card X in Pile 1
    if (X.PartOfSpeech == "Noun" OR X.PartOfSpeech == "Verb") {
        count = count + 1
    }
    Move X to Pile 2
}

count = 0
while (Pile 1 has more cards) {
    Read the top card X in Pile 1
    if (X.PartOfSpeech == "Noun") {
        count = count + 1
    }
    if (X.PartOfSpeech == "Verb") {
        count = count + 1
    }
    Move X to Pile 2
}
```

Hello CT students. In this tutorial we will look a little bit into the OR statement. How to use the OR statement and what is it equivalent to and what is it not equivalent to. So, look at this particular piece of pseudocode. Here we are using the OR statement in here. So let us see what this code does we are initializing a variable called count to 0. And while pile 1 has more cards so as long as the more cards in that pile, what pile is this what data set, so here we are looking at the part of speech so this is the words dataset we are using.

And as long as the more cards left in that pile, we read the top card X in pile 1. And if that top cards part of speech is noun or that part of speech is verb, then we go into this particular if block and we increment the count. So we are counting if our card is a noun or a verb. And then we move this card to pile 2 and we go onto pile 1, checking if pile 1 has more cards.

So what is happening here if your card is either noun or a verb your count will increment that means at the end of the code you will get the total number of nouns and verbs put together as your count variable, which is identical to this code. Here also we are initializing count to 0 and we are going through the cards in pile 1 and we are reading the top card.

And what are we doing we are writing one if statement where we are checking if the part of speech is noun and we are incrementing count. And then we are checking if the part of speech is verb, and then we are incrementing count. So if your card is either a noun or a verb your count will be incremented which is exactly what is happening in this code. So these two are equivalent pieces of code. However, this does not always play out like this.

(Refer Side Time: 02:42)

```
count = 0
while (Pile 1 has more cards) {
  Read the top card X in Pile 1
  if (X.Mathematics > 90 OR X.Physics > 90) {
    count = count + 1
  }
  Move X to Pile 2
}
```

OR is true if even one condition is true.

```
count = 0
while (Pile 1 has more cards) {
  Read the top card X in Pile 1
  if (X.Mathematics > 90) {
    count = count + 1
  }
  if (X.Physics > 90) {
    count = count + 1
  }
  Move X to Pile 2
}
```

```
count = 0
while (Pile 1 has more cards) {
  Read the top card X in Pile 1
  if (X.Mathematics > 90 OR X.Physics > 90) {
    count = count + 1
  }
  Move X to Pile 2
}
```

AND is true only if all conditions are true.

```
count = 0
while (Pile 1 has more cards) {
  Read the top card X in Pile 1
  if (X.Mathematics > 90) {
    count = count + 1
  }
  if (X.Physics > 90) {
    count = count + 1
  }
  Move X to Pile 2
}
```

So consider a situation like this, here let us look at this piece of code now, while pile 1 has more cards we are looking at the X.Mathematics and X.Physics so this is the score dataset and while pile 1 has more cards read the top card which is what we are doing, we are picking a card from a

pile and we are now checking if the mathematics score in that card is greater than 90 or the physics score in that card is greater than 90.

So if at least one of them is greater than 90, so this is what an OR statement does. If you give it two condition A or B, it will check for at least one condition being true. If A is true and B is not, then the OR statement is still true, similarly if A is not true and B is true, then the OR statement is still true. This is contrasted with the AND statement where its check for both conditions to be true.

(Refer Slide Time: 03:41)

```
count = 0
while (Pile 1 has more cards) {
  Read the top card X in Pile 1
  if (X.Mathematics > 90 OR X.Physics > 90) {
    count = count + 1
  }
  Move X to Pile 2
}
```

If Mathematics is 93  
and Physics is 96, count  
is incremented once.

```
count = 0
while (Pile 1 has more cards) {
  Read the top card X in Pile 1
  if (X.Mathematics > 90) {
    count = count + 1
  }
  if (X.Physics > 90) {
    count = count + 1
  }
  Move X to Pile 2
}
```

If Mathematics is 93  
and Physics is 96, count  
is incremented twice.

So here if the student has greater than 90 in maths but not necessarily greater than 90 in physics this OR statement is still true and we increment the count. So what is happening in this piece of Sudo code is we are counting the numbers of cards where at least one of maths or physics is greater than 90. If both are greater than 90 you still just count that one time.

Now look at this piece of code which is exactly written like the previous example, where we check for each condition independently and then we increment count. The problem here is a student who has got greater than 90 in both mathematics and in physics is going to lead to count being incremented twice in that if statement and this if statement, both these statements will come out true. This student is got greater than 90 in both of these.

So count is incremented twice here, whereas in this piece of code count is incremented only once. Even though the student has greater than 90 scores in both subjects.



(Refer Slide Time: 04:58)



```
count = 0
while (Pile 1 has more cards) {
    Read the top card X in Pile 1
    if (X.PartOfSpeech == "Noun" OR X.PartOfSpeech == "Verb") {
        count = count + 1
    }
    Move X to Pile 2
}
```

The conditions cannot both be true simultaneously. They are mutually exclusive.

```
count = 0
while (Pile 1 has more cards) {
    Read the top card X in Pile 1
    if (X.PartOfSpeech == "Noun") {
        count = count + 1
    }
    if (X.PartOfSpeech == "Verb") {
        count = count + 1
    }
    Move X to Pile 2
}
```



```
count = 0
while (Pile 1 has more cards) {
    Read the top card X in Pile 1
    if (X.Mathematics > 90 OR X.Physics > 90) {
        count = count + 1
    }
    Move X to Pile 2
}
```

The conditions can both be true simultaneously. They are not mutually exclusive.

```
count = 0
while (Pile 1 has more cards) {
    Read the top card X in Pile 1
    if (X.Mathematics > 90) {
        count = count + 1
    }
    if (X.Physics > 90) {
        count = count + 1
    }
    Move X to Pile 2
}
```

So what is the key difference here is, in this particular piece of code, one card can only be a noun or a verb it cannot simultaneously be both. In which case you can write the equivalent code like this where you check for each condition, if this condition is not satisfied this might be satisfied. However, if this condition is satisfied this cannot be satisfied so they are exclusive to each other.

Whereas in this code the condition is not exclusive they can both simultaneously be true in which case splitting up the if statements like this is a bad idea because you are going to count twice instead of once.

(Refer Side Time: 05:43)



```
Ncount = 0
Vcount = 0
while (Pile 1 has more cards) {
    Read the top card X in Pile 1
    if (X.PartOfSpeech == "Noun") {
        Ncount = Ncount + 1
    }
    if (X.PartOfSpeech == "Verb") {
        Vcount = Vcount + 1
    }
    Move X to Pile 2
}

Ncount = 0
Vcount = 0
while (Pile 1 has more cards) {
    Read the top card X in Pile 1
    if (X.PartOfSpeech == "Noun" OR X.PartOfSpeech == "Verb") {
        Ncount = Ncount + 1
        Vcount = Vcount + 1
    }
    Move X to Pile 2
}
```

So here is another example where an OR statement is not equivalent to two separate if statements. So here what is going on we are initializing two variables Ncount and Vcount to 0. What might these be we are looking at noun and verb here. So this must be the noun count and verb count in the word dataset.

So while pile 1 has more cards, we check if pile 1 has more cards we read the top card so we basically looking at the top card of pile 1 and if that part of speech is noun, we increment Ncount. So we are counting the noun here and if that part of speech is a verb we increment Vcount.

And then we move that card X to pile 2 so what will this piece of code do it will basically give you a count of nouns and counts of verbs separately because Ncount is incremented only if the part of speech is Noun. And Vcount is incremented only if the part of speech is verb.

Now if we tried clubbing this two if statements with an OR, suppose X dot part of speech is noun OR X dot part of speech is verb and then we do these increment operations here, the problem here is if the part of speech is noun, if a word is a noun, you will get into this if block and you will also increment V count likewise if a word is a verb then we are incrementing the N count along with the V count.

Which means what you will get at the end of it is not two separate numbers, now N count and V count will both come out to be equal because they are both being simultaneously incremented

each single time. So you will just get the sum of the numbers of nouns and verbs for both of these variables, they both will give you the sum. But that is not what is happening in this piece of code because here noun count is incremented only if the word is the noun and verb count Vcount is incremented only if the word is the is a Verb.

So clearly these two are not identical pieces of code. So in this way you should be careful about writing equivalent statements and also you should remember that an OR statement will be true if even one condition is true unlike an AND statement which is true only if all conditions are true.

Thank you.

