



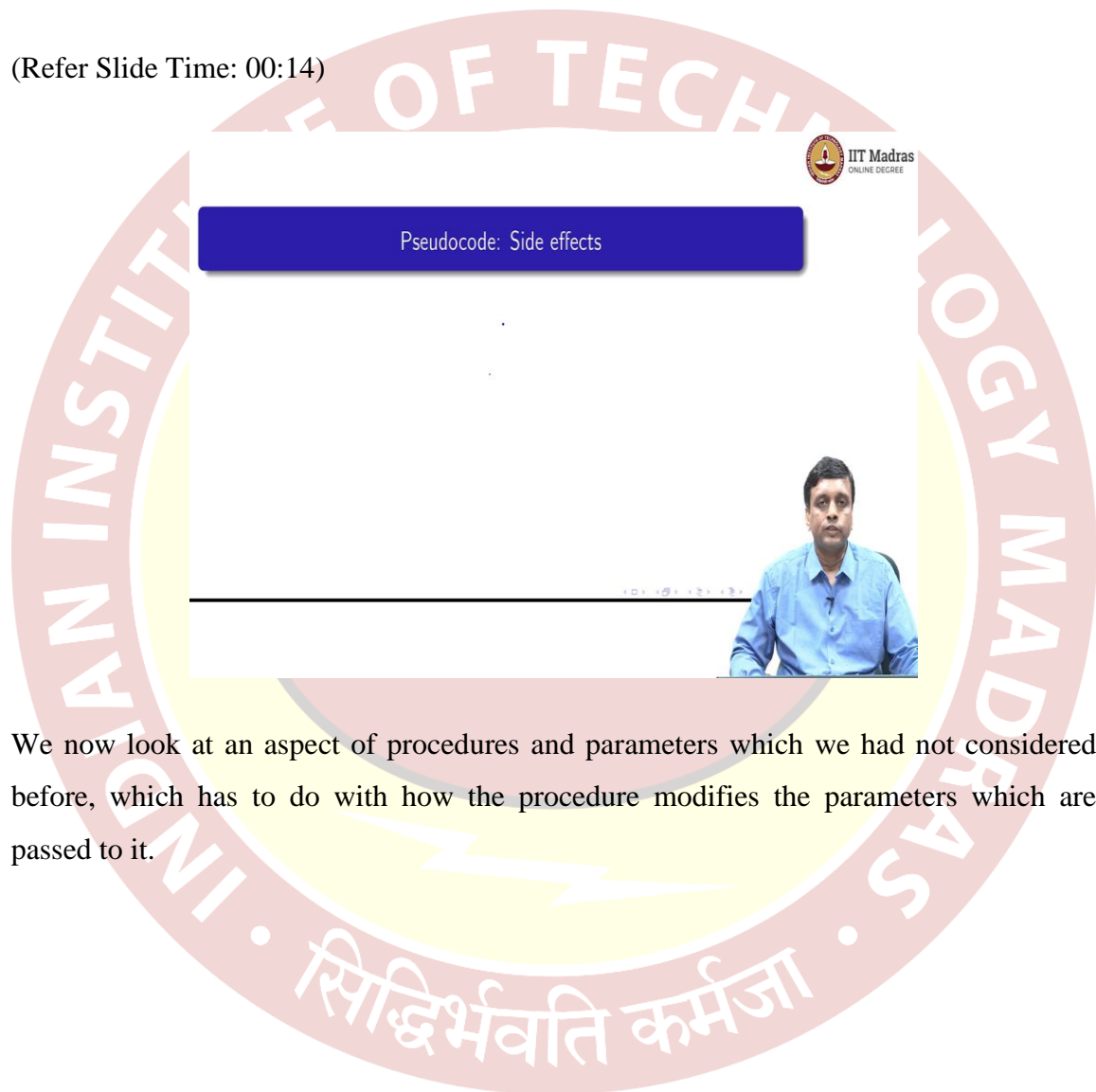
# IIT Madras

ONLINE DEGREE

**Computational Thinking**  
**Prof. Madhavan Mukund**  
**Prof. G. Venkatesh**  
**Department of Computer Science**  
**Chennai Mathematical Institute**  
**Indian Institute of Technology, Madras**

**Lecture – 3.9**  
**Side effects of procedures**

(Refer Slide Time: 00:14)



Pseudocode: Side effects

IIT Madras  
ONLINE DEGREE

INDIAN INSTITUTE OF TECHNOLOGY MADRAS

सिद्धिर्भवति कर्मजा

We now look at an aspect of procedures and parameters which we had not considered before, which has to do with how the procedure modifies the parameters which are passed to it.

(Refer Slide Time: 00:27)

### A procedure to sum up any type of marks



- Two parameters, gender (**gen**) and field (**fld**)
- What about the set of cards?
- This procedure works for a fixed set of cards
- Pass the deck of cards through a third parameter!

```
Procedure SumMarks(gen,fld)
  Sum = 0
  while (Pile 1 has more cards) {
    Pick a card X from Pile 1
    Move X to Pile 2
    if (X.Gender == gen) {
      Sum = Sum + X.fld
    }
  }
  return(Sum)
end SumMarks
```



So, let us recall our procedure which would add up the marks for either gender in any subject. So, we passed two parameters gender and field and we asked it to calculate the total marks in the particular field, physics, maths or chemistry, or total, for that gender male or female. But notice that this procedure assumes that there is a fixed set of cards which it is operating on.

So, there is no mention in this about what pile 1 is. So, pile 1 is assumed to be already existing, so this procedure exist in a context where pile 1 is fixed. So, if we want to make this work across different sets of cards different grade cards for different classes for example, then we need to also pass the deck of cards as a parameter.

(Refer Slide Time: 01:16)

Sum up any type of marks for any deck of cards



- Third parameter **Deck**
- New variable **SeenDeck** for second pile
- **Deck** is modified as the procedure executes

Procedure **SumMarks**(gen,fld,Deck)

```
Sum = 0
while (Deck has more cards) {
  Pick a card X from Deck
  Move X to SeenDeck
  if (X.Gender == gen) {
    Sum = Sum + X.fld
  }
}
return(Sum)
end SumMarks
```




So, this becomes now a third parameter, right. And now instead of pile 1 and pile 2 we have to have another value which accounts for what we call pile 2. So, we have a deck of cards which we are processing the deck in which we have starting to look for values, and the deck of cards which we have finished processing pile 2 which we will call seen deck.

So, what happens in this procedure is that cards are taken, so we have this original deck, right, and then one by one we look at them and then we put them into the seen deck. So, gradually as we go along this number of cards keeps dwindling, right. So, eventually we come to a situation where the deck has no more cards.

(Refer Slide Time: 02:07)


Sum up any type of marks for any deck of cards


 IIT Madras  
ONLINE DEGREE

- Third parameter **Deck**
- New variable **SeenDeck** for second pile
- **Deck** is modified as the procedure executes
  - Cards move from **Deck** to **SeenDeck**
- At the end of the procedure, **Deck** is empty!
- Procedure should also restore **Deck**
- Is this sufficient?

```
Procedure SumMarks(gen,fld,Deck)
    Sum = 0
    while (Deck has more cards) {
        Pick a card X from Deck
        Move X to SeenDeck
        if (X.Gender == gen) {
            Sum = Sum + X.fld
        }
    }
    Restore Deck from SeenDeck
    return(Sum)
end SumMarks
```

Pseudocode: Side effects





So, this is something that we passed to the procedure. So, we passed a deck of cards in some format to the procedure and as the procedure went through the cards the deck disappeared.

At the end of the procedure the deck that we passed to the procedure is empty. Now, this is not acceptable because we do not want the cards to be lost in some sense in data. I mean they are not lost in a physical sense, but as data because we cannot proceed because the value deck that we were dealing with is no longer meaningful.

So, in this procedure there has to be some way to undo what is been done here, right. So, what has been done here is to take everything from one deck and move it to another deck and we want to restore it. So, the question that we are asking in this lecture is what does it mean to restore, and is it just sufficient to say the procedure should restore the deck.

(Refer Slide Time: 02:58)

Side effects

- What is the status of **Deck** after the procedure?
- Is each card the same as it was before?
  - We certainly expect so
- Is the sequence of cards the same as it was before?
  - Perhaps not
  - Depends what we mean by "restore" **Deck**

Procedure SumMarks(*gen, fld, Deck*)

Sum = 0

while (Deck has more cards) {

Pick a card **X** from Deck

Move **X** to SeenDeck

if (**X**.Gender == *gen*) {

Sum = Sum + **X**.fld

}

}

Restore Deck from SeenDeck

return(Sum)

end SumMarks

Pseudocode: Side effects

So, what is the status of deck if we have restored? It is it exactly the same as we passed it? Well, first what is it mean to be exactly the same, are the cards in the deck the same? Well, we certainly expect that each card is exactly as it was before we passed it. So, the deck consists of exactly the same cards that we gave it. If we gave 30 cards with certain data on it, each card comes back with the same data, none of the cards is updated or changed in any way.

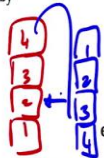
What about the sequence of cards? So, if we gave the deck of cards in a particular sequence can we be sure that the sequence that we get back is exactly the sequence that you gave. Now, this may not be the case. For instance, this is where this term restore deck becomes important. So, supposing as we said, our cards were say numbered 1, 2, 3, 4, 5 and this is how we add them.


(Refer Slide Time: 03:57)


Side effects


- What is the status of **Deck** after the procedure?
- Is each card the same as it was before?
  - We certainly expect so
- Is the sequence of cards the same as it was before?
  - Perhaps not
  - Depends what we mean by "restore" **Deck**

```
Procedure SumMarks(gen,fld,Deck)
Sum = 0
while (Deck has more cards) {
  Pick a card X from Deck
  Move X to SeenDeck
  if (X.Gender == gen) {
    Sum = Sum + X.fld
  }
}
Restore Deck from SeenDeck
return(Sum)
end SumMarks
```









Pseudocode: Side effects

So, we add, so we add card 1 and card 2, and this is our deck from top to bottom. So, we had four cards numbered, 4, 3, 2, 1 from top to bottom. Now, as we transfer them what happens in the second deck is the 4th card goes to the bottom, then the 3rd card comes, then the 2nd card comes, and the 1st card come, right.


So, the second deck is actually inverted with respect to the first deck. Now, if restoring just means copying this back or moving this back to this side then what we have is a reverse deck compared to what we started with, right.


(Refer Slide Time: 04:26)


Side effects

- What is the status of **Deck** after the procedure?
- Is each card the same as it was before?
  - We certainly expect so
- Is the sequence of cards the same as it was before?
  - Perhaps not
  - Depends what we mean by "restore" **Deck**
  - SeenDeck** would normally be in reverse order
- Side effect** Procedure modifies some data during its computation

```
Procedure SumMarks(gen,fld,Deck)
Sum = 0
while (Deck has more cards) {
  Pick a card X from Deck
  Move X to SeenDeck
  if (X.Gender == gen) {
    Sum = Sum + X.fld
  }
}
Restore Deck from SeenDeck
return(Sum)
end SumMarks
```








Pseudocode: Side effects



So, the SeenDeck would normally be in the reverse order. So, this is what is called a side effect. So, a side effect is when we pass some information to a procedure and the procedure actually inverts or reverses this thing or does something to it which was not part of the procedures requirement.

(Refer Slide Time: 04:41)


Side effects ...



- Sequence of cards may be disturbed
- Does it matter?
  - Not in this case — adding marks does not depend on how the cards are arranged
- Sometimes the side effect is the end goal
  - Procedure to arrange cards in decreasing order of Total Marks
- A side effect could be undesirable
  - We pass a deck arranged in decreasing order of Total Marks

```

Procedure SumMarks(gen,fld,Deck)
  Sum = 0
  while (Deck has more cards) {
    Pick a card X from Deck
    Move X to SeenDeck
    if (X.Gender == gen) {
      Sum = Sum + X.fld
    }
  }
  Restore Deck from SeenDeck
  return(Sum)
end SumMarks
          
```



Pseudocode: Side effects

So, in the case of the cards a side effect is that the sequence of cards may be disturbed. So, this was not necessary, I mean, so the fact is that when we asked it to find something the sum of the marks we did not require it to be disturbed. So, it being disturbed to the consequence of how it was computed.

So, does it matter to us? Well, it does not matter as long as we are just going to use these decks for adding cards because adding the card does not matter because 5 plus 7 is 7 plus 5. So, if I can add it in any order I will get the same sum. So, if we pass it again and it is in a different order again the result will be the same.

Sometimes, on the other hand, the side effect is really what we want to achieve. For example, supposing we wanted to arrange the cards in ascending order or descending order, say in decreasing order of total marks we wanted the highest mark and the second highest and so on.

We might have a procedure which takes the cards in some arbitrary order and gives it back to us neatly arranged. So, here the goal is actually to rearrange the cards it does not



have to compute any value for us, but just give back the deck in a prescribed order which is different from the order that we gave.

And sometimes, this order side effect could be undesirable. If we have spent some effort to arrange the cards for example, in decreasing order and then we give it to a deck to a procedure which will shuffle it randomly then the effort that we put in to get it arranged in decreasing order is destroyed.

(Refer Slide Time: 06:04)

**Interface vs implementation**

Each procedure comes with a **contract**

- **Functionality**
  - What parameters will be passed
  - What is expected in return
- **Data integrity**
  - Can the procedure have side effects?
  - Is the nature of the side effect predictable?
    - For instance, deck is reversed

Contract specifies **interface**

- Can change procedure **implementation** (code) provided interface is unaffected

**Procedure SumMarks(gen, fld, Deck)**

```
Sum = 0
while (Deck has more cards) {
  Pick a card X from Deck
  Move X to SeenDeck
  if (X.Gender == gen) {
    Sum = Sum + X.fld
  }
}
Restore Deck from SeenDeck
return(Sum)
end SumMarks
```

Pseudocode: Side effects

So, sometimes we may not want the side effect to happen. So, the way to think about this is that each procedure actually we have to specify what it does. So, think about signing a contract. Remember this delegation idea.

You are delegating this work to someone else. So, when you delegate work to someone else you have to clearly define what the goals of this process are, what is the person supposed to do, and what are the expectations on both sides. So, the first part of this contract is a functionality, what is the procedure supposed to do and how is it going to achieve it.

So, from the person getting the work they have to know what you will give them. So, you have to; you have to specify what parameters you are going to pass and in terms of the work being done you have to know what is going to come back, so what is the property of the values. For example, if I get back this value from this procedure sum of

marks I would like to be sure that the value that comes back is actually the sum of the marks that I requested.

The second part which is more tricky is this business about side effects. So, we have to specify what we expect from the data, can the procedure have side effects, and if side effects are allowed then do we know what the side effect is. So, do we know for instance that it is going to be arranged in decreasing order or do we know that it is going to be a reversed with respect to what we had before or is it random, right. So, this contract specifies what is called an interface, right.

So, it is like I hand over a job to somebody else and I have told this person what I am going to give that person to do and I know what is going to come back, after this I really do not care how that job is done, right. It is not my responsibility anymore to figure out what process the other person takes to achieve this job, so long as they return the information to me that I need and the information that I passed is not disturbed outside the data integrity part of the contract.

So, in computational terms what we are saying is that we have an interface, we have a boundary between the procedure that is doing the calling and the procedure that is being called.

This interface specifies how information flows across this boundary, and as long as this information flow across the boundary is preserved, this interface is preserved what happens on the other side of the boundary which is how the procedure actually works the implementation does not matter. So, tomorrow if you replace that implementation by different implementation, but you preserve the interface everything is fine.

(Refer Slide Time: 08:28)

Side effects ...

Associating personal pronouns with nouns

- Fix a pronoun
- Search text **before** pronoun backwards
- Stop at the nearest **name**

Write a procedure for pronoun matching

- **FindMatch(before, pronoun, after)**
  - Three parameters
- **FindMatch** should not disturb **before** and **after**
  - Sequence of words, position
- No side effects should happen

It was Monday morning. Swaminathan was reluctant to open his eyes. He considered Monday specially unpleasant in the calendar. After the delicious freedom of Saturday and Sunday, it was difficult to get into the Monday mood of work and discipline. He shuddered at the very thought of school: that dismal yellow building; the fire-eyed Vedanayagam, his class-teacher; and the Head-Master with his thin long cane.

Pseudocode: Side effects

So, as an example where we do not want side effects, let us look at this problem that we had of associating personal pronouns with nouns in a piece of text. So, here is our familiar piece of text about Swaminathan. And what we do is we decide a pronoun. So, for example, we have this he and we want to know which he this refers to.

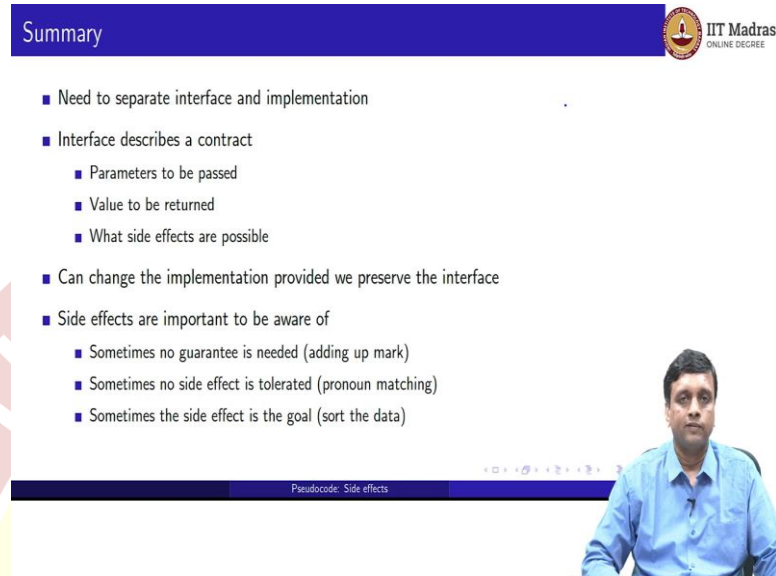
So, the procedure that we described was to take this and search backwards to find a noun which occurs before it. So, we split the text into the text that comes before the pronoun and the text that comes after the pronoun, and we go backwards till we find the nearest name which in this case is Swaminathan. And, we stop there and we declare that Swaminathan is the value that is referred to by this he.

So, if we were to formalize this as a procedure for pronoun matching, then we would have basically the pronoun that we are looking at the text before it and the text after it. So, these are the three parameters. And the pronoun matching function or procedure should start at the current tech pronoun and go backwards through the before text till it finds the matching noun.

In this process, the sequence of before and after should not be disturbed because this is a sequence of word which defines a piece of text and as we have seen the sequence actually determines the answer to the problem. If you shuffle the nouns then the noun that refers to he will change. So, in this case, we definitely do not want this procedure to

have any side effects. So, there are situations where side effects are totally undesirable and unacceptable.

(Refer Slide Time: 10:00)



Summary

- Need to separate interface and implementation
- Interface describes a contract
  - Parameters to be passed
  - Value to be returned
  - What side effects are possible
- Can change the implementation provided we preserve the interface
- Side effects are important to be aware of
  - Sometimes no guarantee is needed (adding up mark)
  - Sometimes no side effect is tolerated (pronoun matching)
  - Sometimes the side effect is the goal (sort the data)

Pseudocode: Side effects

So, to summarize when we write a procedure we need to be clear about separating the interface and the implementation. So, we draw up a contract in some sense with the procedure which describes what parameters we will pass, what they are supposed to mean, and what values are going to be returned.

And in addition, we also specify what side effects are possible, so that we know whether the procedure is allowed to change values that we pass or not allowed to change and perhaps tell us in what way these changes can happen.

So, the advantage of doing this is that we can change the implementation provided we preserve the interface. For instance, we might have a more efficient way of doing something. So, if we replace the current procedure by a more efficient procedure it should not in any way affect the way that it is called and the values returned are used in the main procedure.

So, side effects are definitely something we need to be aware of. Sometimes it does not matter, adding up marks for instance does not matter whether the side effect is there or not, as long as the cards individually are preserved it does not matter if the sequence is disturbed.

Sometimes, we want the order to be exactly as it was for example, when the sequence matters in the pronoun matching problem, and sometimes the side effect is actually the desired goal, for instance a procedure which does sorting. The desired goal is to produce a side effect of a predictable and desired version which is to arrange it in decreasing or in increasing order.

