



**IIT Madras**  
ONLINE DEGREE

**Computational Thinking**  
**Professor G. Venkatesh**  
**Department of Computer Science**  
**Indian Institute of Technology – Madras**  
**Lecture –3.7**

**Pseudocode for Procedures and Parameters (Part 2)**

So, we say how to write pseudocode for procedures. So now let us just go through a small example to see how we actually call a procedure from our main code and use it in our execution.

(Refer Slide Time: 00:26)

The slide is titled "Analysis of top students" and features a list of bullet points. In the bottom right corner, there is a small video inset showing a man speaking. The slide is part of a presentation, as indicated by the "Pseudocode Using Procedures" text at the bottom.

- Is there a single student who is the best performer across subjects?
- Is the highest overall total the same as the sum of the highest marks in each subject?
- Need to compute maximum for different fields in a score card
  - Maths, Physics, Chemistry, Total
- Ideally suited to using procedures
  - Same computation with a parameter to indicate the field of interest

Pseudocode Using Procedures

So, remember this problem that we discussed in the class which is whether or not there is a single student who is the best performer across all subjects. So, we wanted to know if there is a overall top student and for this what we said was that we would check the maximum marks amongst the totals of all the subjects of all the students and compare this to adding up the total marks in each subjects.

So, if some student is a toper in every subject then this student will of course have the overall total which is a top. So, the maximum total would be the sum of the maximum totals in each of the subjects. So, for this we have to do the same thing repeatedly on the scorecard we have to take a look at all the Maths card and find the maximum then we have to take a look at all the physics cards.

All the chemistry cards and finally the total marks and in each case we have to run through the entire stack of students and find the maximum. So, this is ideally suited to using procedures since we are doing the same thing again and again and in this case the parameter

that we will use to differentiate one call of the procedure from the other is to denote which field we are interested on. So which field on the card we are computing the maximum off.

(Refer Slide Time: 01:38)

Finding the maximum in a given field

- As usual, keep track of the maximum using a variable
  - Initialize to 0
  - Update whenever you see a bigger value
- The value to be compared is not fixed
  - Parameter fld determines the field of interest

Procedure Maxmarks(fld)

```
Maxval = 0
while (Pile 1 has more cards) {
  Pick a card X from Pile 1
  Move X to Pile 2
  if (X.fld > Maxval) {
    Maxval = X.fld
  }
}
return(Maxval)
end Maxmarks
```


Pseudocode: Using Procedures

So, here is the procedure that we would use for finding the maximum marks. So, as usual we start off with this variable called Maxval which will be initialized to the smallest value possible so that whenever we see a larger value it will get updated. So, we start off with Maxval equal to 0 and then we do our usual iteration we go through all the cards in the first deck and for each card we compare the current value of Maxval with whatever field has been passed through the parameter.

So, we look at the fld value on card X and if the new card has a value which is bigger than the Maxval then we update the Maxval. So, this is just our usual max computation you iterate through the cards and keep updating max each time you see a larger value, but because we have made it a procedure, we can change what value we are looking for by passing a different field in each iteration.

(Refer Slide Time: 02:31)


Analysis of top students



- Use the procedure Maxmarks to find maximum marks in different categories
  - Four procedure calls, with fld set appropriately
  - Save each return value separately
- Use saved return values to compare the maximum overall total with the sum of the maximum subject totals

```
MaxMaths = Maxmarks(Maths)
MaxPhysics = Maxmarks(Physics)
MaxChemistry = Maxmarks(Chemistry)
MaxTotal = Maxmarks(Total)
SubjTotal = MaxMaths + MaxPhysics
           + MaxChemistry
if (MaxTotal == SubjTotal) {
    SingleTopper = True
}
else {
    SingleTopper = False
}
```

Pseudocode Using Procedures



So, now how do we use this in our actual computation. Well we wanted to compute the total marks maximum and also the maximum for each of the subjects. So, we would call max marks four times once with Math, Physics and Chemistry and once with the total. So, we do that, but now we have to remember the value. So, what we do is we call the max max marks procedure with our parameter and then we store it in a local value to remember it.

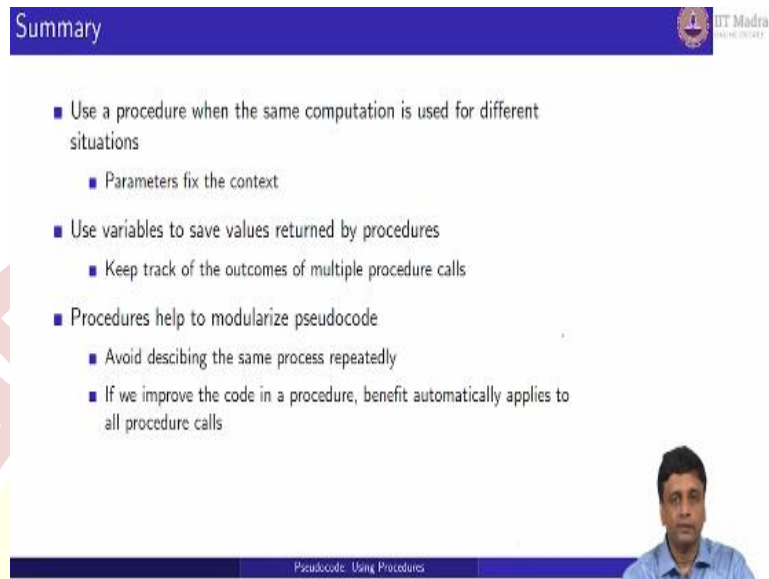
So, we can store the max Math marks in one variable the max Physics marks in other variable and so on. So, at the end of this we have done four calls to this procedure and we have got four maximum marks in four different categories and now we can apply the criterion we want to check. So, we first compute the subject total by adding three of them. We add max marks, max Math, max Physics and max Chemistry to get the subject total.

And now we want to determine whether there is a single topper in the class or not. So, we check whether the max total is equal to the subject total. If the max total is equal to the subject total then there must be some student who has got the maximum marks in every subject and also in the total. If not, then there is some variation in the subject wise marks even though the total marks belongs to one student that person has not done the best in all the students.

So, in this case if max total is equal to subject total we said the single topper variable to be true otherwise we said it to be false. So, this illustrates how we use procedures in our code. We just call it whenever we need it and then we store the value so that we can use it later. So, it is pretty much like calling any other functions or expression so if you are trying to do x

plus x and store it in y and the same thing except that the value is returned not by a simple arithmetic operator, but by a procedure.

(Refer Slide Time: 04:12)



Summary

- Use a procedure when the same computation is used for different situations
  - Parameters fix the context
- Use variables to save values returned by procedures
  - Keep track of the outcomes of multiple procedure calls
- Procedures help to modularize pseudocode
  - Avoid describing the same process repeatedly
  - If we improve the code in a procedure, benefit automatically applies to all procedure calls

Pseudocode: Using Procedures

So, to summarize as we have seen we use the procedure when we are going to use the same computation in many different situations and the way we distinguish these situations is by passing a parameter to tell the procedure what context we are in. So, in this case the context is which field on the card to examine in order to compute the max. So, we pass the field name as a parameter and then these procedures will return values.

But we have to keep track of these values because we will call these procedures multiple times. So, we use variables to keep track of these things across multiple procedure calls. So, as we saw last time the main value of doing this by making our code or breaking it up into procedures is to modularize our code. So, one thing is that if we are doing the same thing again and again, we do not have to duplicate the code.

So, we do not accidentally make mistakes and also our code is easier to read because we do not have to write the same thing again and again, but more importantly because we are describing how to do it only once if we come up with a better way of doing it or a different way of doing it, but we take the same input and return the same output then without affecting the rest of the code we can change our procedure to make it better.