



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 – 3.6
Pseudocode for procedures and parameters

(Refer Slide Time: 00:15)



So, when we manipulated the cards, we saw the need for procedures to capture repeated computations which we do with minor variations.

(Refer Slide Time: 00:25)

Sum of Boys' Maths marks

Sum = 0
while (Pile 1 has more cards) {
 Pick a card **X** from Pile 1
 Move **X** to Pile 2
 if (**X**.Gender == M) {
 Sum = Sum + **X**.Maths
 }
}

■ What if we want to sum Maths marks of girls?

```
graph TD
    Start([Start]) --> Sum0[Sum = 0]
    Sum0 --> MoreCards{More cards in Pile 1?}
    MoreCards --> End([End])
    MoreCards -- Yes (True) --> PickCard[Pick a card X from Pile 1]
    PickCard --> MoveCard[Move X to Pile 2]
    MoveCard --> GenderCheck{X.Gender == M?}
    GenderCheck -- No --> MoreCards
    GenderCheck -- Yes (True) --> SumAdd[Sum = Sum + X.Maths]
    SumAdd --> MoreCards
```

IIT Madras
ONLINE DEGREE

Pseudocode: Procedures

So, let us see how to describe this in pseudo code. So, here is a procedure to compute the sum of the boys' maths marks in our grade cards. So, what we do is, we loop through all the cards. And if the gender on the card is M, then we accumulate the value of the maths marks for this card in a variable called sum.

So, initialize sum to 0, and we update sum provided the gender is M. So, let us consider a variation on this problem. The obvious variation is that instead of looking for the boys' maths marks, we want to add the girls' maths marks. So, what would change?

(Refer Slide Time: 00:59)

Sum of Girls' Maths marks

Sum = 0
while (Pile 1 has more cards) {
 Pick a card **X** from Pile 1
 Move **X** to Pile 2
 if (**X**.Gender == **F**) {
 Sum = Sum + **X**.Maths
 }
}

■ Only change is the value we check for **X**.Gender
■ Remaining pseudocode is identical

```
graph TD
    Start([Start]) --> Sum0[Sum = 0]
    Sum0 --> MoreCards{More cards in Pile 1?}
    MoreCards --> End([End])
    MoreCards -- Yes (True) --> PickCard[Pick a card X from Pile 1]
    PickCard --> MoveCard[Move X to Pile 2]
    MoveCard --> GenderCheck{X.Gender == F?}
    GenderCheck -- No --> MoreCards
    GenderCheck -- Yes (True) --> SumAdd[Sum = Sum + X.Maths]
    SumAdd --> MoreCards
```

IIT Madras
ONLINE DEGREE

Pseudocode: Procedures

So, quite obviously, the only thing that would actually change is what you check for the gender on the card. Instead of checking that the gender is F, M, you check that it is F. So, this is the only change the rest of the code is identical. We have a value sum a variable which is set to 0, and we increment or we update sum to have the maths marks of the current card provided the gender is now F instead of M.

The remaining code does not change at all. So, this is a good candidate to convert into something where we can parameterize this code that is we can write the same code and use it again and again just changing the value of the gender from outside.

(Refer Slide Time: 01:37)

A procedure to sum up Maths marks

- Procedure name: **SumMaths**
- Argument receives value: **gen**
- Call procedure with a parameter **SumMaths(F)**
- Argument variable is assigned parameter value
- Procedure call **SumMaths(F)**, implicitly starts with **gen = F**
- Procedure returns the value stored in **Sum**

```
Procedure SumMaths(gen)
    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.Maths
        }
    }
    return(Sum)
end SumMaths
```

Pseudocode: Procedures

So, here will be a notation in pseudo code for these procedures. So, this is the procedure which generalizes the sum of the boys' maths marks and the sum of the girls' maths marks into a single piece of code which allows you to choose the gender from outside.

So, this procedure has a name because we have to call it we cannot just have unnamed procedures because we have to know what procedure we are going to execute. And of course, we need to invoke it with the right value to tell it what to do, so it has an argument.

In this case, we call the argument something, so we call it gen inside the procedure. And this value that we pass is supposed to be the gender, and the way it is going to be used in the procedure is going to substitute for the specific check for M or F in the earlier two

pieces of code instead it is going to be checked against the value of this variable `gen` which will be set from outside when the procedure is called.

So, how do we call the procedure well we invoke the name of the procedure with the concrete parameter for this argument. So, if we want the girls' maths marks, then we will pass the parameter `F`; if we want the boys' maths marks, we would pass the parameter `M`.

So, what happens inside the procedure is that this particular assignment happens, so we have it is as though we insert a statement there. So, when we call it with `F`, it is as though this variable `gen` is assigned the value `F` inside the procedure that is how you should think about it. So, from outside the variable gets the value `F`.

So that when it comes here the value that is being checked is the value that has been passed from outside that is `F`.

The other difference between doing this as a part of the main code and doing it as a procedure is, how to extract the value that we computed. If we do this as part of the main code, we are keeping track of the variable `sum` in our main code. So, after we execute these lines of code, the variable `sum` has the value that we want, the sum of either the boys' maths marks or the girls' maths marks.

But remember in a procedure we are delegating this work to somebody else. So, we are asking somebody else to do the work, and then report back to us. So, this reporting process has to be formalized in the pseudo code as to what is the value that the procedure gives back to us, what do we get back.

So, in this case, what we get back is the value `sum`, and this is indicated in the procedure by having the statement called `return` which returns the variable whose value the procedure is supposed to compute and give back right. So, this is how we write our pseudo code. So, we have the procedure name with a parameter with an argument which is instantiated by passing a parameter. And we have a return value which tells us how the computed value in the procedure goes back to the code that called the procedure.

(Refer Slide Time: 04:21)

A procedure to sum up Physics marks



- Only change is the field we examine in the card
- X .Physics, instead of X .Maths
- For Chemistry, add up X .Chemistry
- For Total, add up X .Total
- Pass field name as parameter

```
Procedure SumPhysics(gen)
    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 +  $X$ .Physics
        }
    }
    return(Sum)
end SumPhysics
```



So, let us transform this problem in another dimension. So, we looked at the transformation from boys' maths marks to girls' maths marks where what was changing was the value of the gender that was being checked. Now, supposing we transform it from maths to physics.

So, now we are transforming not what gender, but also what field we look at on the card, what attribute on the card we are adding up. So, if we try to modify the code for the sum of the physics marks like we had for the sum of the maths marks, it takes the gender as before; and the only difference is that instead of taking X dot maths as the value to check on the card, it takes X dot physics and adds that to sum provided the gender matches.


And what would happen if we wanted to do it for chemistry? Well, again this thing would just change from X dot physics to X dot chemistry. And there is no reason to restrict our self only to these three subjects we could also do it for total we might want the total marks of all the boys or the total marks of all the girls.

So, this would again be the same thing we will take X dot total instead of X dot maths or X dot physics. So, this seems to suggest that all these procedure are also minor variants and they could be combined into a single procedure if we could pass this field name as a parameter.

So, this is a slightly different type of parameter. Earlier we were passing a value as a parameter. We are passing M or F and say compare a given field to this value; now we are saying this is the name of an attribute, look up this attribute and add it up. So, here is a generic procedure which will add up any subject or the total marks for either male or female.

(Refer Slide Time: 06:01)

A procedure to sum up any type of marks




IIT Madras
ONLINE DEGREE

- Two parameters, gender (**gen**) and field (**fld**)
- **gen** is assigned a value, M or F, to check against **X.gender**
- **fld** is assigned a field name, to extract appropriate card entry **X.fld**
- Single procedure **SumMarks** to handle different requirements
 - **SumMarks(F,Chemistry)**
Sum of Girls' Chemistry marks
 - **SumMarks(M,Physics)**
Sum of Boys' Physics marks
 - ...

```

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
            
```



Pseudocode: Procedures

So, it takes two different arguments first it takes the gender argument as before. And the second thing is it takes a field argument. And remember the difference between these two is slightly subtle, but it is important. That gender is a value. So, gender is a value that we pass, and inside there is a check which compares the value of something inside the card to this value that we have passed. So, we are passing a value.

Field is an attribute name. So, this is used to extract an appropriate part of the card and this varies from different calls of the procedure. So, now, by instantiating this to different values, we can get different results.


So, if we call sum marks with the argument F and chemistry, then this says add up those for gender F in the field chemistry, so we get the girls' chemistry marks. If we say M and physics, then we get the boys' physics marks. So, by varying the parameters, we can get any combination of gender and subject total.

(Refer Slide Time: 06:57)


Calling a procedure

- Use procedure name like a math function, as part of an expression

```
GirlChemSum = SumMarks(F,Chemistry)
BoyChemSum = SumMarks(M,Chemistry)
if (GirlChemSum > BoyChemSum) {
    "Congratulate the girls"
}
else {
    "Congratulate the boys"
}
```

**IIT Madras**
ONLINE DEGREE

$$y = \text{sqrt}(5)$$



Pseudocode: Procedures

So, how do we call a procedure in our code? Well, we call it by just using it like a function in mathematics. So, if we write something like if we had a in a programming notation, we could have a function say which takes a number, and takes its square root which we call back say a square root of 5, and then we could assign it to 5 to a variable y. So, in pretty much the same way, we use the name of the procedure with its arguments like a function in an expression.


(Refer Slide Time: 07:29)

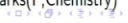
Calling a procedure

- Use procedure name like a math function, as part of an expression
- Assign the return value to a variable
- A procedure may not return a value
- Correct marks for one subject on a card
 - Procedure
UpdateMarks(CardId,
Subject, Marks)
- Procedure call is a separate statement

```
GirlChemSum = SumMarks(F,Chemistry)
BoyChemSum = SumMarks(M,Chemistry)
if (GirlChemSum > BoyChemSum) {
    "Congratulate the girls"
}
else {
    "Congratulate the boys"
}

Sum = 0
...
UpdateMarks(17,Physics,88)
...
GirlChemSum = SumMarks(F,Chemistry)
```

**IIT Madras**
ONLINE DEGREE



Pseudocode: Procedures

And then we take this expression and then we assign it to a value, so that we can keep it for later. So, here for instance, we are computing the sum of the girls' chemistry marks and the sum of the boys' chemistry marks by calling this procedure twice with different parameters F dot chemistry and M dot chemistry. And now that we have computed it, we have stored these values in these two variables GirlChemSum and BoyChemSum. Then for instance we could do something based on the values we have got back

So, for instance, if the sum of the girls' chemistry marks is bigger, we might have some I mean this is not of course real code this is just that indicate that you could do two different thing. So, you could congratulate the girls if the girls sum is bigger; otherwise you congratulate the boys.


Now, it is not required that a procedure return a value. Sometimes the procedure may be doing some update to a value or it might be organizing the information in some way, and not really computing something and returning a value. So, a typical example would be supposing you wanted to write a procedure which corrects a card right. So, it takes a card and then it says update the marks for a given subject.

So, it takes three arguments – the card id, the subject to be updated, and the new marks to be assigned. So, the effect of this procedure is to actually change the card value. So, this we would just write as a statement in our function in our code without assigning the return value to anything because the return value is not useful to us right.

So, in such a case the procedure call would just appear like an assignment statement as a separate statement in our sequence of statements to be executed. So, these are two different ways in which we would call a procedure. If the value that we want is useful to us we call it as part of an expression and we assign it to a variable; if it is doing something whose return value is not important to us, we just call it as a statement with the appropriate arguments.

(Refer Slide Time: 09:19)

Summary

 IIT Madras
ONLINE DEGREE

- Procedures are pseudocode **templates** that work in different situations
- Delegate work by calling a procedure with appropriate parameters
 - Parameter can be a value, or a field name
 - SumMarks(M, Total)
- Calling a procedure
 - Procedure call is an expression, assign return value to a variable
 - GirlsChemMarks = SumMarks(F, Chemistry)
 - No useful return value, procedure call is a separate statement
 - UpdateMarks(17, Physics, 88)
- 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: Procedures



So, to summarize a procedure is a template of pseudo code which works in different situations. So, this is a way of delegating work outside the main procedure by calling this sub procedure with appropriate parameters.

So we saw that this parameter could be of two different types; it could be a concrete value of a data item or it could be an attribute name or a field name saying which item to pick up from a card. So, as examples, we had this sum of marks procedure which would take a gender and say the total, and this would give back the total of the boys; marks across all the cards.

And to call a procedure we can either call it like an expression and assign the return value to a variable like we did when we got the girls' chemistry marks by adding up the marks of the girls in chemistry; or if there is no useful return value, we could make it a separate statement like the UpdateMarks situation.

So, why do we use these procedures? Well, an important part of writing these kind of systematic procedures is to build them modularly. If we just write a long sequence of instructions, is very hard to recognize what are the patterns that are repeated and which can be updated.

So, we want to avoid describing the same process again and again and write it only once, and use it in many different ways as far as possible. So, this is one of the important reasons why we want to organize our code into procedures.

Another reason to do this is that suppose we find a better way to do something. So, we have a way of computing a value and somebody shows us a more clever way something that works faster or which takes less space. Now, if we have moved this code out into a procedure, then wherever this procedure was used it would implicitly call the new code and it will run in this more efficient way.

Whereas, if we had written this code explicitly hundred different places, then we have to go back and update each of these hundred different places to incorporate the new idea. So, by modularizing, we avoid repeating, and we also make it easy to substitute new code for old code provided it computes the same value that we wanted to compute.

