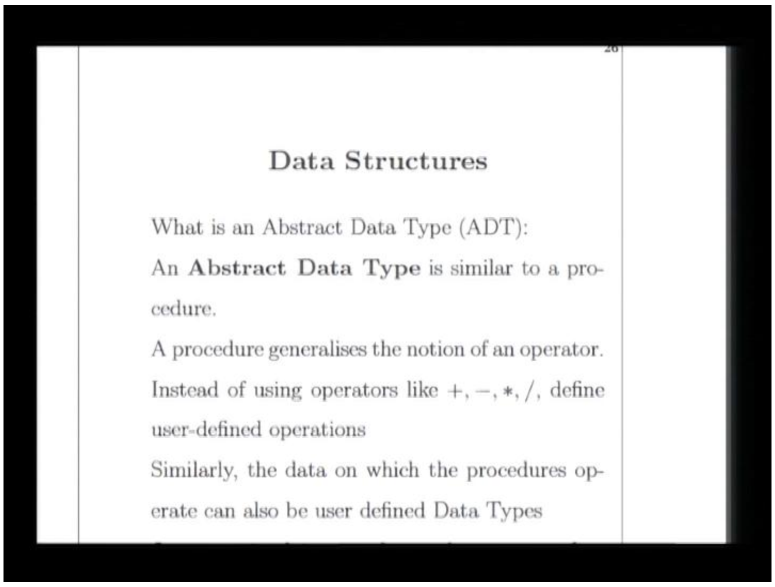
Programming, Data Structures and Algorithms

Prof. Hema Murthy
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Module – 04
Lecture – 42
Data Structures Abstract Data Type (ADT)
Basic ADTs
Advantages of Using ADTs

(Refer Slide Time: 00:08)



Data Structures

What is an Abstract Data Type (ADT):

An **Abstract Data Type** is similar to a procedure.

A procedure generalises the notion of an operator.

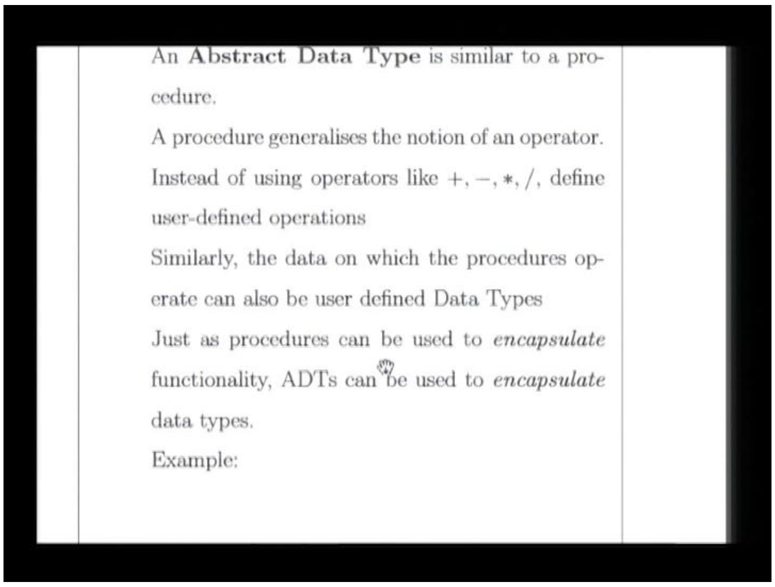
Instead of using operators like $+, -, *, /$, define user-defined operations

Similarly, the data on which the procedures operate can also be user defined Data Types

So, the next lecture is on data structures, but data structures the reason (Refer Time: 00:17) some time on giving a algorithm analysis and things like that. Data structure is encapsulates the what is called a data type? And given a data type it also defines a set of operations. So, basically what we will do is when we talk about data structures here, we going to talk about abstract data types. Let me explain to what is an abstract data type, and very similar to a procedure. What you do the procedure? Now basically if you what we you already seen we saw some functions and procedures, you already seen in your programming module, and we also seen just in the last class how to analysis we found the sum of set of element, we found the search for element in an array, we shorted given in set of elements so and so forth. So, what it is that, that we are saying? We will say short thing and array. What you mean by there? Just I what is short thing? Short thing is an operation, that is being performed on the array.

Similarly when I am looking at searching, searching is an operation that is being performed on an array. So, the way we would like to look at is an abstract data type is kind of similar to procedure. So, what is a procedure do, a procedure kind of generalizes the notation of an operator. What you mean by that? Your familiar with a standard arithmetic operators plus minus multiplication and division, addition, subtraction multiplication, division. What we can do in a procedure is that, we generalizing is particular notation and we are able to define our one operation. Shorting is an operation that is defined on the array. Similarly searching is an operation that we define on the array.

(Refer Slide Time: 02:08)



An **Abstract Data Type** is similar to a procedure.

A procedure generalises the notion of an operator. Instead of using operators like $+, -, *, /$, define user-defined operations

Similarly, the data on which the procedures operate can also be user defined Data Types

Just as procedures can be used to *encapsulate* functionality, ADTs can be used to *encapsulate* data types.

Example:

So, what we do in and in a data structure is similarly what we do is, we also generalize the idea of a idea of user define data types, just I like we have operations which are generalizable in procedures. In when you talking about abstract data types, we also have a facility by which begin generalize data types. We can define user define data types. Just I queue have what are data types now integer, float, let us say pointers, so and so forth. Now you can do generalization of the whole things and say I am going to give string for example, is another is a data type, character is a data type, float is a data type, double is a data type, int is a data type. In see what we ((Refer Time: 02:51)) generalize is the whole notation and say I am only give new data type. I call this list, I call this stack, I call this a queue. Then what happens? Just is you had addition, subtraction, multiplication, and division defined on let us say the integer data type, when I define a new data type, I have to define operation on them. So now, that is what is abstract data type? And abstract data type is one just like your procedures wish encamps encompass encapsulate functionality, the abstract data types can be use to encapsulate data types and we also define operations on this. So, let me go back this.

(Refer Slide Time: 03:35)

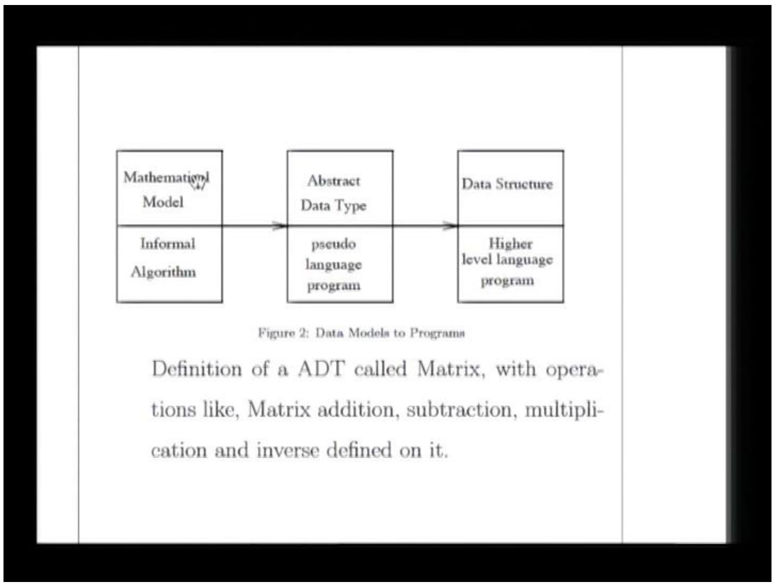
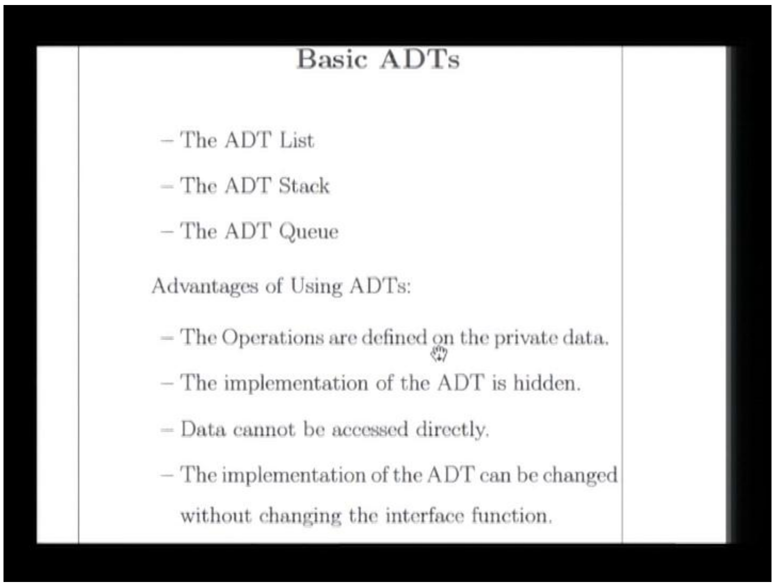


Figure 2: Data Models to Programs

Definition of a ADT called Matrix, with operations like, Matrix addition, subtraction, multiplication and inverse defined on it.

So, what is it that we have two kind of formalize this, you have a mathematical model, and in informal algorithm, let me give you an example. Suppose I have sets then I have a mathematical model has to how you represent the sets and then you perform a certain set of operation on sets; your union, your intersection, your complement, and so on so forth or set of operations that operator on set so begin. Now from the mathematical model we goes to the abstract data type, and we define a pseudo language program, and this becomes and data structure and higher level language program which operates on this. So, this is how the data models, this is how you go from data models to programs, and this is what is x this figure is illustrating. Let me give you one in other realization of this, what is this mathematical model. Just I like this saw the ((Refer Time: 04:29)) something that is closer for example, I can define an abstract data type called matrix with operation like matrix addition, subtraction, multiplication and inverse defined on it. So, the operation is not performing on the matrix or let us say on this, but I am need some kind of representation in the matrix, and that kind your representation is hidden or encapsulated in the abstract data type.

(Refer Slide Time: 04:58)



Basic ADTs

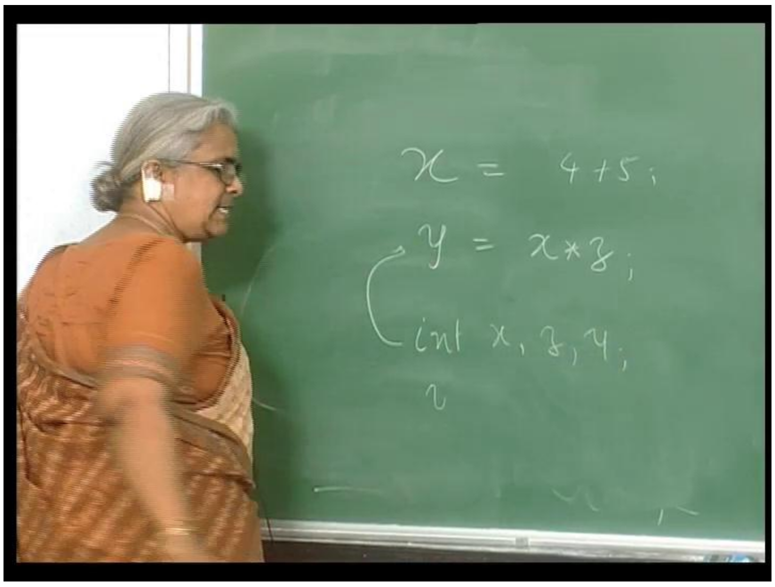
- The ADT List
- The ADT Stack
- The ADT Queue

Advantages of Using ADTs:

- The Operations are defined on the private data.
- The implementation of the ADT is hidden.
- Data cannot be accessed directly.
- The implementation of the ADT can be changed without changing the interface function.

So, what will be do is you will start with some very basic abstract data type, which are commonly use in computer science. These data types are the ADT list, the ADT stack, the ADT queue very very ((Refer Time: 05:10)) data types, and what is nice why do we want data types. Let me give you an example again, when you look at when you perform the operations on integer, when I am saying that I want find the sum of 4 plus 6, and you write on the left hand side, I write a variable like this.

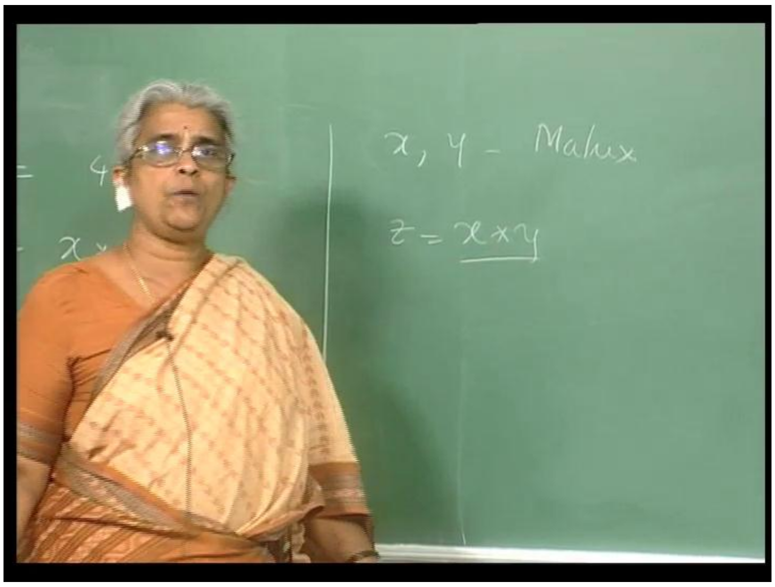
(Refer Slide Time: 05:30)



I say x is equal to let us say 4 plus 5 was something like that. Or y is equal to x star z when you write statement like this in the higher level language program, clearly you are not aware of how x how the operation of multiplication is implemented nor are your aware of how ((Refer Time: 05:53)) store. What we do in programming language, I simply say int x, you know that is in integer data type, and x comma z comma y, and say y is equal to ((Refer Time: 06:06)). Then after your second right something like this.

So, what is happening is even a programming language today actually heights the way in which the integers are represented, and how the operation of addition, subtraction, multiplication, and division heart performer. So, now what is what are the abstract data type are, they give you tool by which you can define your own data types, and define operation on them.

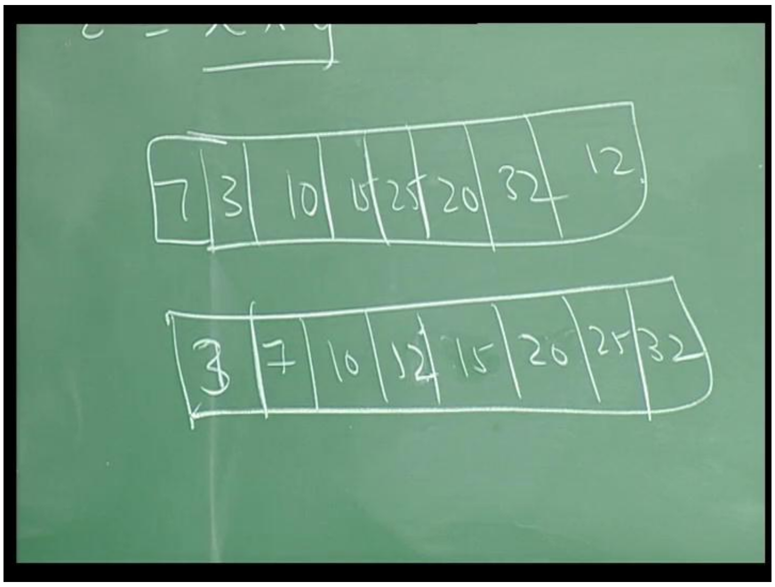
(Refer Slide Time: 06:46)



Then give this abstract data types to user hooking use it without knowing how the implementation of the data types as being done, already do is I will say for example, let me give your example, I would say x and y are of type matrix, this is what I have define. Add I can write let say z is equal to x star y. The place it will for perform matrix multiplication, the star is the operation of multiplication time into perform. So, it automatically does in. Clearly these are not available in current day programming languages. So, what we do is the whole idea of the using ADT is that the operations are define on the private data, the representations completely hidden, and the data cannot be access directly, there are operation that can be perform on the ADT to access the data.

And what is very nice is, suppose today you know I representing for shorting I represented the date, I store the data in an array; maybe there will be a more efficient algorithm which were I can change the internal representation from array to some list or link list are something like that, and it my become more efficient. Or even in 5, I am representing in an array, first let us say I did simply linear search. You ask for an element, I return the element whether present in the array or not. Then after words I am become a little cleverer and I said let me do, let me short the elements and perform operation and so on. What is the operation and performing a performing search, then when what happens, if I am let us go backs to example of an array.

(Refer Slide Time: 08:16)



So, initially I had in unsorted list. On the unsorted list I was looking for the element 12 which was found at the end of the array if remember it, we had some 7 3 10 15 25 20 32 and something different. So, what we it was we set search throw the array find the element. So, clearly is very inefficient, because 12 is at the end of the array. Then next ((Refer Time: 08:42)) what is you know let us short this remember the binary search algorithm that we did, we set 3 7 10 15 20 25, sorry we should be a 12 15 20 25 and 32, you what we did? Then what we set was now clearly, I can do little bit better. What can I do? When I looking at the middle, I can I can just search until I find the final elements which is larger than the key. So, if I am looking a key equal to let us say 14 then I compare with this, compare with this, compare with this, compare with this, compare with this. Now what happens? Key is become smaller than 14, I can say elements is not found in and come out, the one way of doing it efficiently.

Now next what it I do? As a know I can even make it even better, I can divide the array into to once is short cut and do binary shorts. This algorithm even a 5 do linear search key equal to 14, if I will ask the key equal to 35, I would had to search the entire array. Then what we did was, we said now we do not, we are not expecting the property that is actually shorted and I can do certain things, then we define the binary search algorithm, and we said… Now I can divide the array into two and search only in that part where the ((Refer Time: 10:03)). So, one we have looking ADTs, if I could provide an ADT which operates an arrays, can do searching or shorting for that matter and are different. And then what ((Refer Time: 10:15)) every time you have been first time I gave, because I initially wrote in inefficient algorithm for searching, which will you give me the location of ((Refer Time: 10:22)).

And then next I given other little more in efficient algorithm, I can using linear search, and finally I may do binary search, but as far as the user is concern your giving and algorithm called search. Is like the addition being implemented maybe you know first when you use this operation of here, x equal to 4 plus 5 may plus was inefficiently is implemented. Later on it became more efficient, but that is kind of hidden from the user, that is the big advantage of ADTs. And what is nieces the implementation of the ADT can be changed continuously without changing the interface function. I want you give you… I want you leave you with slightly kind of abstract notation of this.

If you look at human beings, how do we you know when you when you see somebody after 10 years, suddenly you are recollecting something about the person and your able to identify, he was ((Refer Time: 11:19)) 600 classmate. How you doing this, clearly the way we have represented the data are information the brain is not note anybody, and what is a operation that we are performing, you see this person and immediately you want to recollect, you perform in operation on the data, the result come soon. So, basically the representation of the data is completely hidden and so on abstract data type is kind of nationalizes this particular property.