

Programming, Data Structures and Algorithms
Prof. Hema Murthy
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Module – 05
Lecture - 43
Lists
Operation on lists: Create, Insert, Delete
Implementation of lists using linked list

(Refer Slide Time: 00:09)

Lists

The ADT List:
A sequence of zero or more elements of a given
ElementType:

$a_1, a_2, a_3, \dots, a_n$

$n \geq 0$ and a_i is of *ElementType*

n – is the length of the List

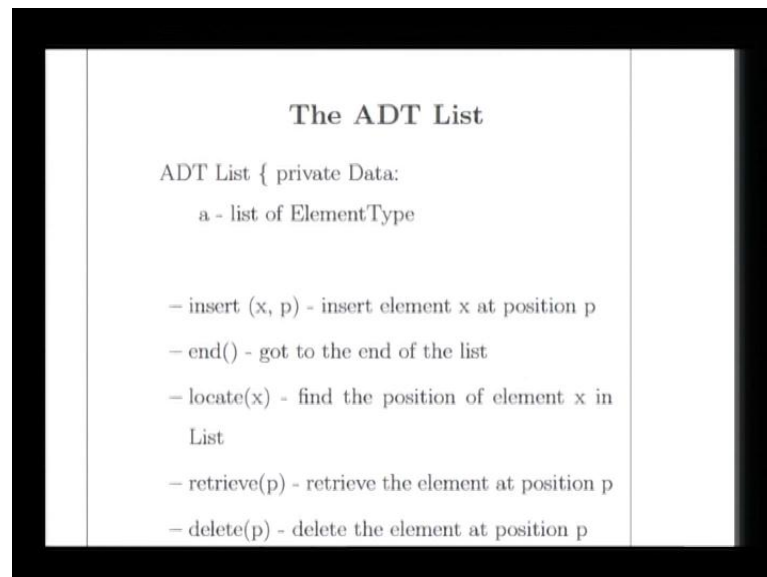
$n \geq 1$, first element is a_1 and last is a_n

Now, what we will do is, we will grocery items that I have. So, this is the simplest thing that we can think of. So, what you do when you are on the list of grocery items, you know name, all of them and put them together and say these are grocery items and send it to your workshop keeper to or call your shop keeper and say please deliver these items and so on. So, when you look ((Refer Time: 00:31)), this is an unordered sequence of 0 or more elements, I have a particular given element type. What is my element type here, in this particular example of grocery items it is element type has grocery items.

So, what do I have? I have elements a_1, a_2, a_3, a_n . The n term is generally greater than or equal to 0 and a_i is some particular type, n is a length of my list. In this particular example, those will illustrate let us say n greater than or equal to 1, first element is a_1

and last element is an.

(Refer Slide Time: 01:01)



The slide is titled "The ADT List" and is enclosed in a thick black border. It contains the following text:

The ADT List

ADT List { private Data:
 a - list of ElementType

– insert (x, p) - insert element x at position p
– end() - got to the end of the list
– locate(x) - find the position of element x in List
– retrieve(p) - retrieve the element at position p
– delete(p) - delete the element at position p

Now, what are the things that you would like to do in an ADT list. So, when you want to define an ADT list, what we will list, for example I can have a list of students who are interested in music, list of students who are interested in sports, so on and so far. It can be any kind of list, for that matter. Why where things it, we would like to do? I would like to perhaps that I have this particular list, I would like to insert elements into the list, I would like to find the end of the list.

Because I would like to know how many people are there, I would have to find suddenly you know, I signed up for let us say know, game and I have registered for playing Tennis or something like that. Let us humbled, suddenly I do not want to play Tennis, I want to cancel my registration. Then I want to locate a particular element in the list and all I find out, who is there at position x. Because I would like to let us say you know, this is an unordered list and I think saying that all these people would have registered for the hostel, you are going to put two pupils together. Then maybe I would like to find out, at my position who is going to be my neighbor and I like to find out, whom neighbor is.

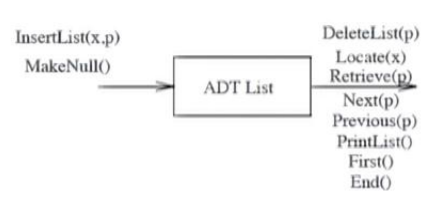
(Refer Slide Time: 02:14)

- insert (x, p) - insert element x at position p
- end() - got to the end of the list
- locate(x) - find the position of element x in List
- retrieve(p) - retrieve the element at position p
- delete(p) - delete the element at position p
- next(p) - get the position of element at the position next to p
- prev(p) - get the position of element at the position before p

Hema A Murthy IIT, Madras

And I already said I deregistered let us say, from game or whatever it may be. Now, I delete the element from the list, then I find out who is next, who is previous and so on, so forth. So, all these operations if I want to perform on the list, we will not yet talked about, how the list is going to be implemented, we will do that a little later.

(Refer Slide Time: 02:33)



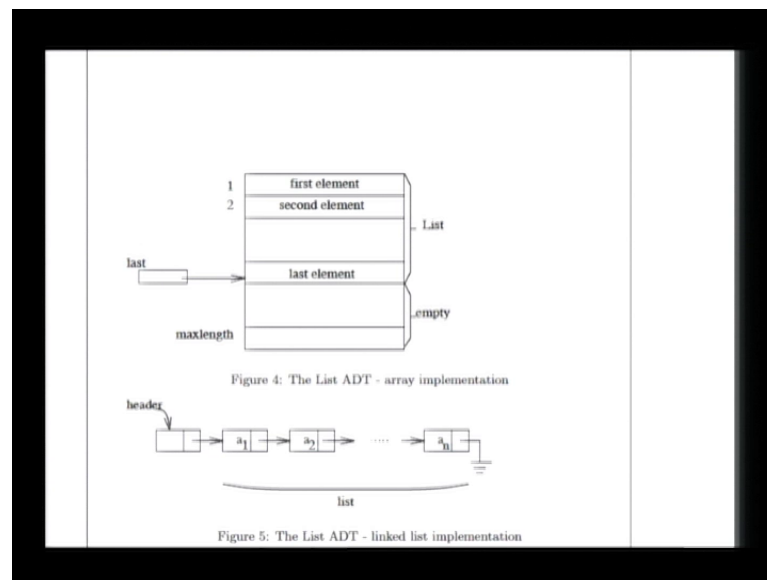
The diagram shows a central box labeled "ADT List". To its left, two operations "InsertList(x,p)" and "MakeNull()" have arrows pointing towards the box. To its right, a list of operations "DeleteList(p)", "Locate(x)", "Retrieve(p)", "Next(p)", "Previous(p)", "PrintList()", "First()", and "End()" has an arrow pointing away from the box.

Figure 3: The List ADT

- makeNull() - create an empty list
- first() - get the position of the first element
- print() - print all the elements in the list
- print(p) - print the element at position p

So, what we are first defining is we define an ADT list and this tells me, what operations going to the ADT list. We first creating an empty list, then insert into the list at a particular position. Then you delete elements in the particular position, you can locate whether particular element is in the list, you retrieve the element it the position p, find the next element of position p, previous element of position p, print the entire list. Find out what is there at the first of the list, what is there, get the position of the first element in the list, get the position of the last element in the list and so on.

(Refer Slide Time: 03:09)



So, now how do I implement the list? Now, what we do is it could be implemented using either an array or a linked list. These are the basic two data structures that are available, we already talked about it in the last class. An array is basically enables random access, whereas we already said a linked list on the other hand is a recursive data structure and you have to start from the head of the list. So, you can do the implementation of the list using either the array or the linked list. But the operations on the list are the same. And as far as the users concern, the user is quit oblivious to how the implementation of the list is being done? So, this is essentially the idea and as finally the user concern, he or she is only going to do insertions, deletions, so on and so forth. Let us see, how this can be done order.

(Refer Slide Time: 04:12)

```
Implementation of Lists

/*****
 * Program      : LinkList.cxx
 * Function     : Defines a class List
 *****/

#include "iostream.h"
/* Definition of element of List */

typedef struct CellType* Position;
typedef int ElemType;
struct CellType {
    ElemType value;
    Position next;
};
```

So, what we will do is this is implementation of the list which I have showing over here. So, what do I do, I have a program called LinkList dot cxx, I have implemented this in C plus plus and noticed that I have a recursive implementation here. You can also use an array, array is easier. Notice that I have basically, what is that I am making available to the user. I am giving doing this typedef position, which is the p in the implementation that we have over here. P corresponds to the position in the list over here, p is typedef to something called it is position. And essentially, the user also get me an element at position p, that is the way you will look at it.

(Refer Slide Time: 04:56)

```
public:
    void makeNull();           // create a new List
    void insertList(ElemType x, Position p); // insert x at Position p
    void deleteList(Position p); // delete element at Position p
    Position first();         // get Position of First
    Position end();          // get Position of end
    Position next(Position p); // get Position of Next
    void printList();        // print List
}; /* end (Definition of class List) */

/* begin (Implementation of class List) */

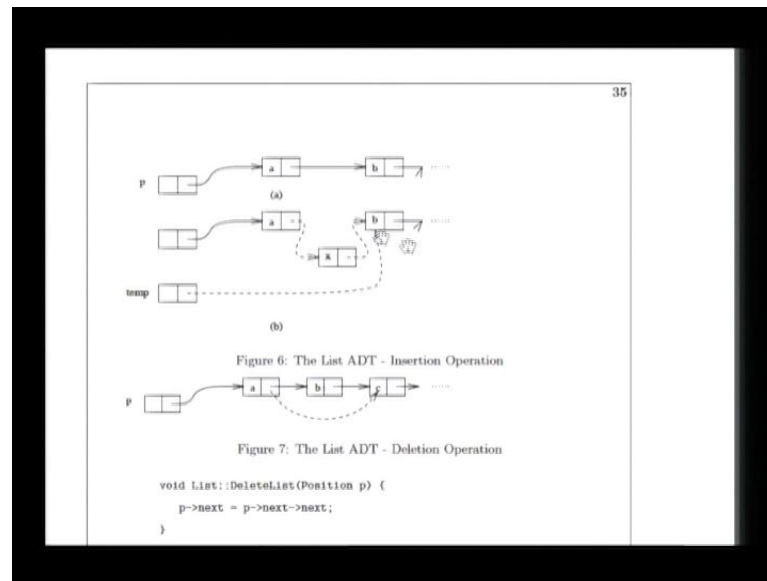
void List::makeNull() {
    listHead = new CellType;
    listHead->next = NULL;
}

void List::insertList(ElemType x, Position p) {
    Position temp;
    temp = p->next;
    p->next = new CellType;
    p->next->next = temp;
}
```

So, what is he or she do? So, we have here, private notice that I have done an implementation using link list and this is hidden from the user and because, it is defined as a private data type. Then I have a facility to create a new list, then I have a facility to insert element x at position p, so element types or something that is specified by the user. Now, I am writing a generic list data structure, which is internally represented as a link list.

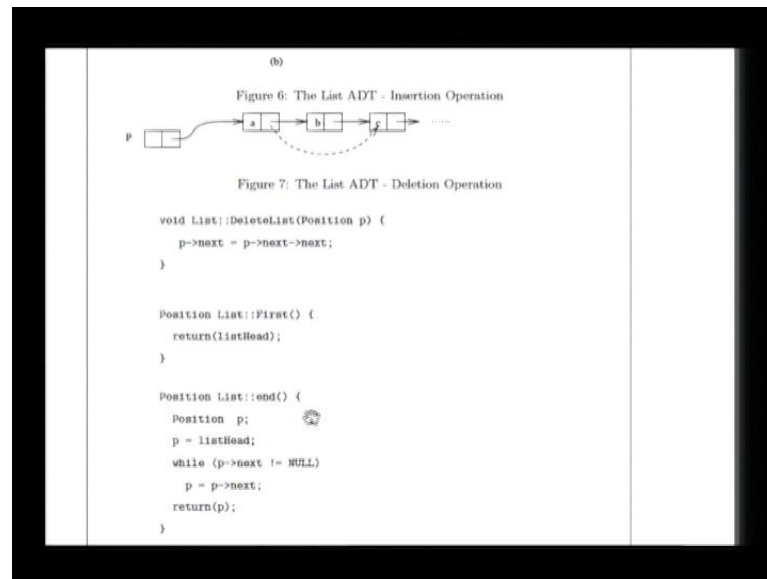
Element type is provided by the user, because the user knows what element he wants to, what set of element he wants to put in the list. For example, is it a list of box, is it a list of plates, is it a list of grocery items, is it a list of students, it may be anything for that matter. That information comes from this element type defined by the user. Then what he or she will do is essentially we have an implementation here. I have created for my convenience, I create one empty node in the list.

(Refer Slide Time: 06:02)



So, this is what I do, my empty list make talent for example, creates a list with one single element. Only one element is there and it points to the next, I make a dummy header for that factor. Then when I am inserting I always insert at the end of the list. Whatever I am doing here, notice that I want to insert it at position p, I am inserting at p pointer next, this is again not known to the user. So, when the user asked me for an element at p, I will return the element at p point to next, that is the point of all of this.

(Refer Slide Time: 06:44)



Similarly, when I am doing deletion. The reason why I have done this dummy node is primarily because, deletion becomes very easy, p pointer next becomes p pointer next next and whatever it may be and return will give you the... Notice that please observe this, obey here, we make position list, position list end. So, as far as the users concern, he or she does not know, whether the position is pointer or to a recursive list or it is a index into an array, because we have done the typedef in the definition itself.

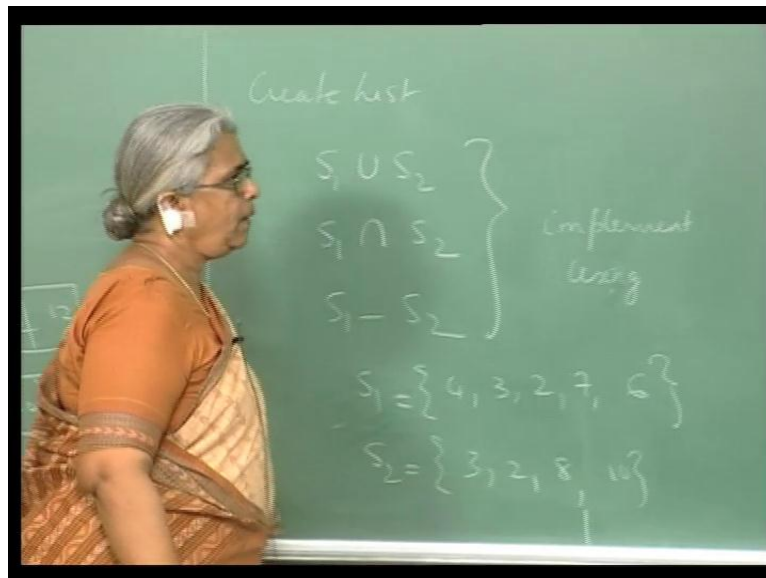
Going back to this part, how I have done over here. We done a typedef here and all that user uses a user position, in all hers programs. So, it does not matter if as long as I keep, I can change the internal representation of position, here it is pointing to the pointer. It could be 0. For example, corresponding to the index of the 0th element in first element in the array. It could be done that way too, but I have just tells the user, link list operation could be used.

So, this source the deletion operation and this source the insertion operation. So, basically what is that we are doing here, we are essentially creating a new element over here. Inserting it and then this store this pointer in a temporary variable and then, put it back. So, if you notice how to find and what we do over here, if you notice here again, let us look at the end over here, it is an interesting function. And for example, returns the

element noticed that we are continuously while p point to next, not equal to NULL.

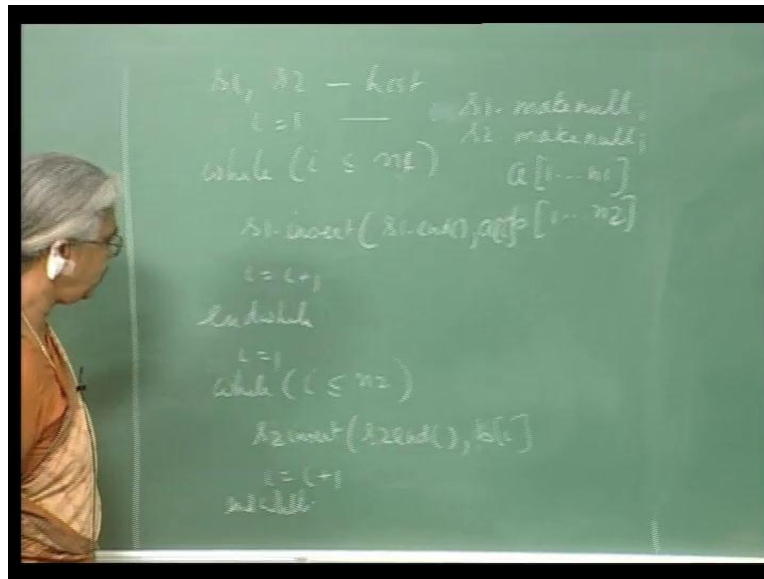
So, it returns the actually the pointing to null at the end of the list. So, the primary reason for this is that when I want to insert, I can very easily insert a key at the end of the list. P pointer next is NULL, in that particular list. So, this is the list and I want to leave you with a set of operations, so let me get back to a certain last the picture over here. Now, one of the applications of list that you can do, let me I give you an example. I put it have it here.

(Refer Slide Time: 09:06)



We can do things like, you can create a list with as half, let us say I have sets for which I want to perform, s_1 union s_2 , s_1 intersection s_2 , s_1 minus s_2 all these can be implemented using s. Let me give you one example that I would, so what would I do? I would create a list of elements, let us say, s_1 is made up of the following elements, let us say 4, 3, 2, 7 and 6 is the something I have that and s_2 is made up of 3, 2, 8 and 10. So, what could I do? I would create, I would say that if you remember here, I would say that going back to this.

(Refer Slide Time: 10:33)

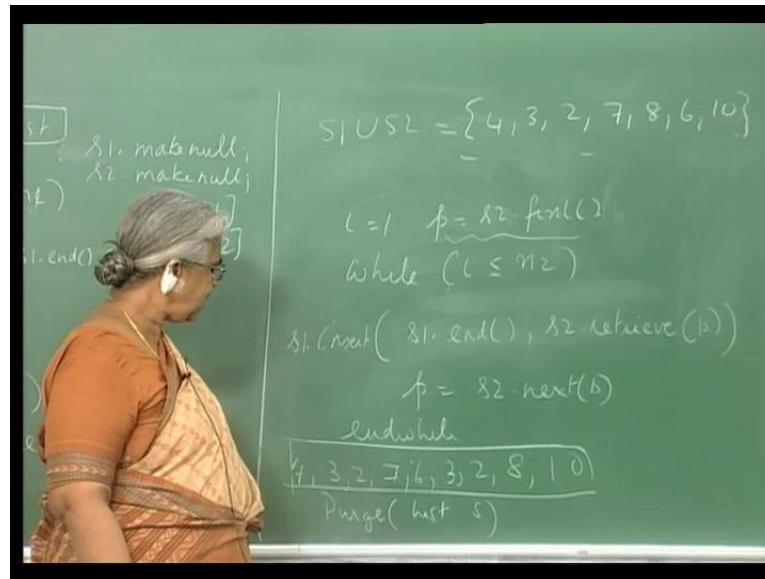


I would say that my $s1$ comma $s2$ are of type list first. Then what I would do is I am reading, let us say this input from somewhere and so I would have, while i less than or equal to $n1$, i equal to 1, let us say. I would say first all, I will make a empty list. I will do $s1$ dot make null and I will do $s2$ dot make null, which will create two empty list. Then what will I do? And let us say $n1$ is the number of elements and I got all these elements let us say in some array.

Then let us say that I have an array a of elements 1 to $n1$ and I have an array b of elements 1 to $n2$ which need to be inserted into the list. So, what I would do is I would say now $s1$ dot insert, I would say here, $s1$ dot end, because end of the list comma a of i , i equal to i plus 1, end while. So, it is much as the same, I would do for $s2$. So, while again I will say i equal 1, while i less than or equal to $n2$, $s1$ dot insert $s1$ dot end comma. I am inserting at a position b of i , i equal to i plus 1.

So, now what have we done now, we have created two lists, let me call this $s2$ here, $s1$ and $s2$ into which we have inserted elements into the list. Now, next what we want to do? We want to find the union, union means what now, I have to look at the union of these two sets are what, a set is always made up of a unique number of elements.

(Refer Slide Time: 13:09)



So, now I look at s_1 union s_2 , clearly what do we required now, s_1 union s_2 should be made up of the elements, what do we have to do now? It should consist of 4, 3, 2, 7, 8, 6, 10, something like this I am keeping it unordered here. So, what is it mean, it only should consist of unique elements, that is the definition of a set. So, what could we do is, we could start off with set s_1 , when we know that the elements in set s_2 are and till the end of the list.

Take all of them, this insert it at the end of s_1 with you do that. So, what would I do now, while i less than or equal to n_2 I have already done this over here. So, I can start with i equal to 1 and while i less than or equal to n_2 . What I am going to do? I am going to say s_2 dot retrieved, what I will do first, p is a position of the first element s_2 dot first, use need the first element. And what to be do, we put the outside over here, p is equal to s_2 dot first.

Then I will retrieve what is there it p and we will insert into s_1 , s_1 dot insert s_1 dot end comma s_2 dot retrieve at position p . What is the meaning of this? I am inserting the element at position p from s_2 to the end of the list s_1 . So, I would do this. So, now what will I do, I will say update p , p is equal to s_2 dot next of p . Again, get the next position and so on and so forth, I would do this.

So, this completes, so what is it do now, basically this will give me a new list, such that I have 4, 3, 2, 7, 6, 3, 2, 8 and 10. So, clearly it does not satisfy the definition of a set, here all the elements have to be unique. Next, what I can do is I can set this new list that I have. I can try to another function which take as a argument a list, I can say list s purge the duplicates innovation. I can write a separate function for that.

So, what is that mean, I take this list here, find out I go through and traverse the entire list, see if the element is found. If that element is found, I delete that element, if there are duplicates in this list, I simply keep on deleting it. Whereas I have stopped with this position, start with the first element, then traverse to the entire list over here, I can go to the end of the list, see that element is do we repeated, if it repeated, repeat the duplicate. Then go to the next element and so on.

So, you can do this, so list is what is that, what is that interesting is I want to see this particular segment of code here which I have written. I have not simply, I do not even know, what the implementation of the list is. It may be an array implementation, it may be a link list implementation, a and b are arrays which are provided, which the users provided with data in them of type element type i and all around we use, insert at the end.

So, this is the most important criteria about abstract data types, where we use only the operations. We know that just like we have integer, what have you done here, s1 s2 corresponds to a abstract data type called list and what is that we are doing now, into the list we have inserting elements. Where are the elements is coming from, from a user defined array, he is doing it, he or she used doing it himself. So, what is it doing insert?

Insert it at the end of the list, we not bother because, initially what is going to happen, either create an empty list, it creates an empty list. There it creates the empty list, where is the end point to. End points to the first dummy element that I have in my implementation and I am say, end of the list just keep on appending these items, that is what it does. Similarly, that is done another list is created and you have done this.

Then to form union whatever we have doing, we first putting all the elements of s2 at the

end s1. So, it forms the union, but a problem is it may not have unique. Then what I do? I write a separate function called purge which simply deletes the duplicate. How do I perform purge? I start with the first element, check what is the end of the first element and see, if you can locate the particular element in the rest of the list.

When I forget to locate the element in the rest of the list, if it is already there, then I delete that element at a particular position. The important point I simply do not need to know, what is the implementation of the abstract data type list is. So, I would like you to go back and try the other operations. In particular, I am not given you this implementation of purge. I want you to think about this, I talked about union, I like you to think about complement and intersection on sets, whereas set is represented using a list.

And also I like you to think of how can you use only the list operations to order the elements in the list. So, when you are ordering the elements in the list, what do I need, I need something to compare. Because an order means putting in an ascending order to putting in a descending order, in which integer or real I can do it. Suppose it is names of people, how do you want to order them. So, the comparison operation is provided outside the list implementation again. And just like here, to locate the element we are doing a comparison, the comparison is provided by the user. So, let us stop with this list ADT, and next we will look at stacks and queues.