

**Indian Institute of Technology
Kanpur
NP – TEL
National Programme
On
Technology Enhanced Learning
Course Title
Compiler Design
Lecture – 21**

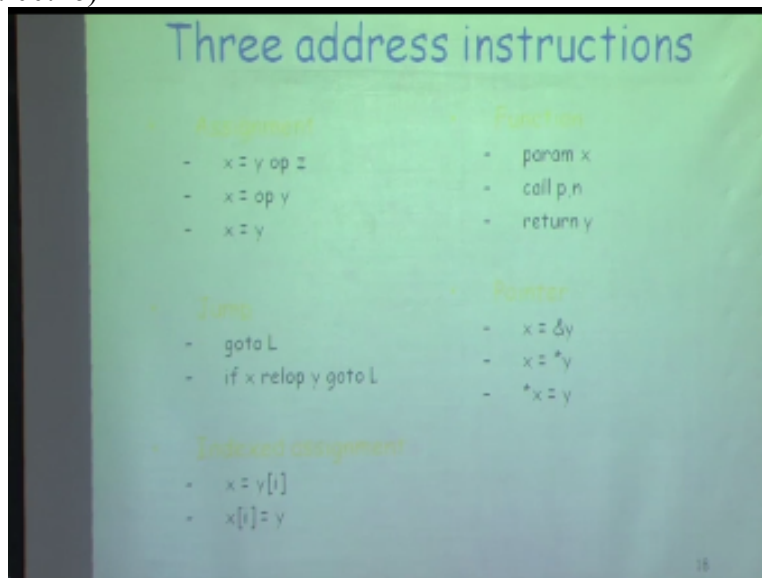
by...

Prof. S. K. Aggarwal.

Dept. of Computer Science and Engineering.

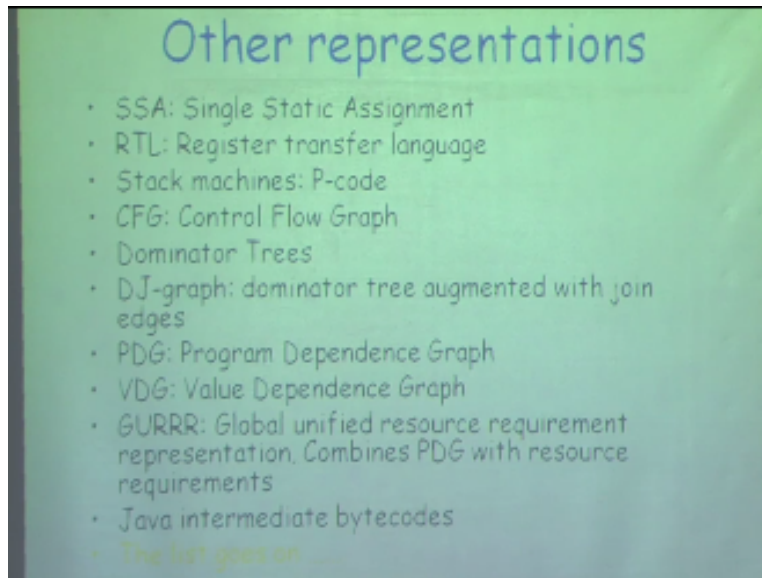
Good morning so let us continue our discussion on towards end of the previous class various ideas what is the kind of property in behind and finally to discussing set of instructions.

(Refer Slide Time: 00:46)



There we had the sign has we had all from there is procedures and defines that more enough set we will captured class of languages and okay and notice as a more we go now we start discussing code generation we will find that more able to handle with this so with this is really you will find that more or less one piece of informations.

(Refer Slide Time: 01:23)

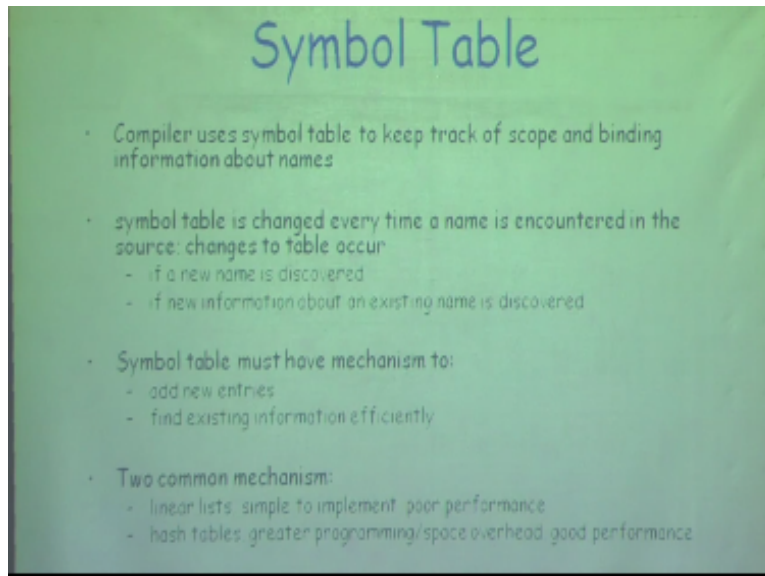


So just to just to conclude this discussion on IR we look at several IRs actually a top of the important information and that console popping that people are not holding context space to that special context it has held of a special control of a language and compilers at which was about various intermediate and context and so this is small less top intermediate specifications and use there is fast compiler and integers such on this presentations.

And we will find the solutions so this is your close our discussion as per this there is a another very important data structures in just to mean to have been define okay apart from the creating to this also and worry about that the code generation what is that stack so stack is the okay various stack is the important structures so because okay.

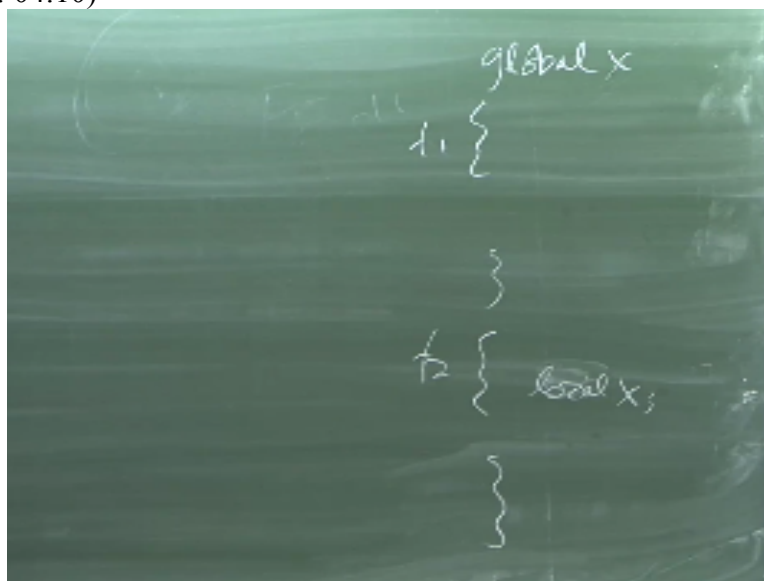
So what are the important of in that part giving stack that is okay but discussing stack that is cross form when the stack point is another point to be this stack simple table so if you re call the initial compiler structure that was a data structure we said we just going to be interface with all of compilation starting complex we can lexical analyzer then co generation and that also we needs to understand how we can organized a single table because this is again as certain properties which will find the interfaces okay.

(Refer Slide Time: 03:26)



So what compiler this going to do use symbol table to keep stack of information and how all the variables and node the reason additional issue that will come to that some point of time we have to worry about it so what happens in have positions and functions I have no local variables and there is a source we need to a symbol table these somehow able to capture when I have look at the real when I want to look at the example what will happen I have a same user name offering to different source and I refer to a variable I need to they should be get the right variable access with the right variable now where that we need so that something from languages which are like language as C.

(Refer Slide Time: 04:10)



So I can have some global X here and then I may have let's say function at one which may have which local X but then I have other functions which may have let me type to this let me just

copying locals okay it and now question that we come is that when I have person X in this scope and refer from 2x is this scope okay when I refer to X in this scope and I referring to variable and I refer to this scope and initial to this state and then.

If you have nested procedures and functions okay then issue is going to be lot more complex in this case so need to worry about how the scope information is will be kept because that ultimately will tell me how the finding will going to became a variables when I say a variable use here so if I say u defines here this case attributes to handle that to the kind of symbol tables I had okay so also we need to understand.

Certain property that symbol table is need to check and all the kinds of compiler to give the potentially interchange for changing is that keep on adding new on informations so if you use lexical analyzer is receive information and co generation relates are set of that particular variable okay so in this also whenever a new name is discovered a new information about this thing is found out and I am doing a modify the information.

In the symbol table that means we all s aid worry about the how fast I can access my symbol table now simple tables therefore must have mechanism to take care of these new functionalities one and also I should be able to add new informations and the new entries and find the variable in the single table therefore I should add that and also I can able to find all the information find existing information.

Because if variable is already there and some new information has to be added about it there it was first locate by the variable is so I should be able to do that again efficiently.

Because that is important issues and what is the kind of common data structures for as we already goes with the data structures which can be a data structures and of course so linear list this is one way to implement this so how would I implement linear list we have this informations and that I can have structures a list of structures and every structure we have an information about on the symbols and that.

We also know that this is very inefficient because every time you will have to do a full traversal of this a kind of grammars okay more efficient way of course is that I use hash tables and hashing obviously but then you have to design good hash functions and here became very bad and because we may have one buffer then we have to design good hash functions and her the programming and you may have slightly more space overhead.

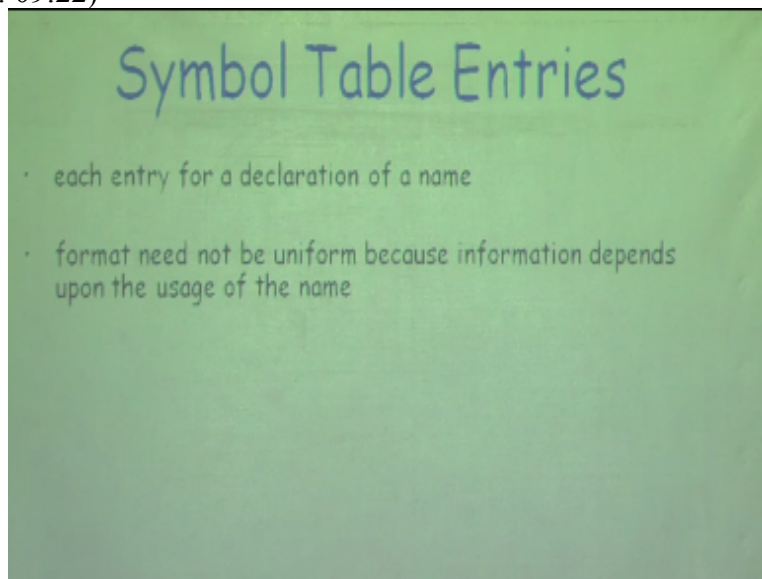
And programming node okay this definitely is going to give you much better performance to compare to leaner variables so you have to now find out and hash tables you will again find actually the combination of fashion in linear linked list because you may not be able to map on to

unique and trees and then whenever we map then on the same thing what are going to do you actually have the liner list of on all the information which is mapping onto that but also keep that as known as possible so if it is string of other kind and the small enough and form particular point of you do not worry about the information so if you find the regions that only now how much time for take us okay.

Also compiler should able to to go simple table dynamical what is that mean why I point have a symbol table of more stack to begin this you can I start reading my program for compilation so one way I can say is that I can have a static structure like a simple table I say area fill box and contact is very inefficient space bugs so what we are likely to do is be a small simple table and as it can require possible also what will happen is that.

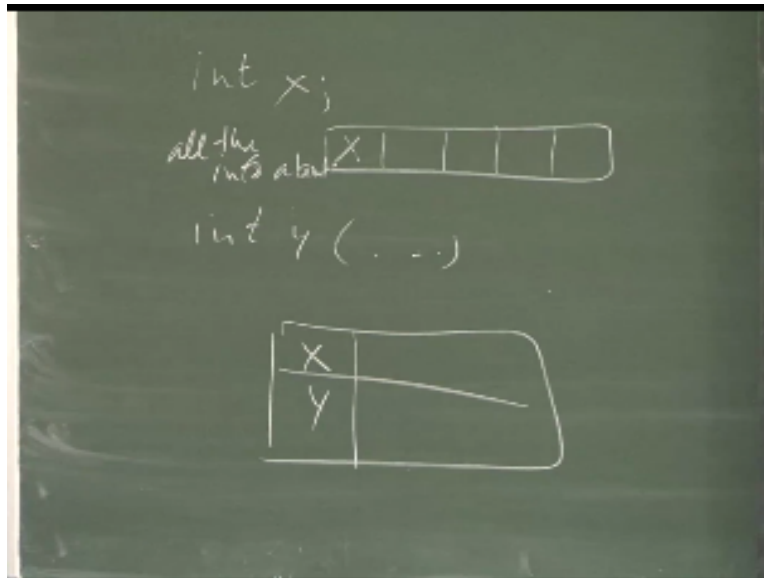
We take care of the scope information I may start creating symbols to for in this scope and once I find in those scope and I start creating a new symbol table for that and this is some information so I would like to make sure that I start with reasonable size symbol table and information for more and more symbols and more and more scopes I just know this information and if size is fixed then it should be large enough for the largest program which is inefficient for the capacity to basis so you should not even that.

(Refer Slide Time: 09:22)



So looking at the information which is going to go in the simple table it has declaration of the name that will happen there and this is something we see when we start doing code generation for all the declarations for the symbol table okay so importantly not uniform because information depends upon use of the denied so for example when I say that so let us keep this okay.

(Refer Slide Time: 09:52)



But let us look at something like this suppose I have an intense and I also have let us say in Y but it is actually the function so this is just the variable in this declaration of a function.

Now if I say that in my symbol table I am keeping structure may I have information about X and information about Y need the same pair of structure clear that in this is the information which is going to come in is very different than this kind of information so for it need not be uniform because information is how this so in some cases when I say that I have a different kind of utilization of the name my form it will be different in each end piece.

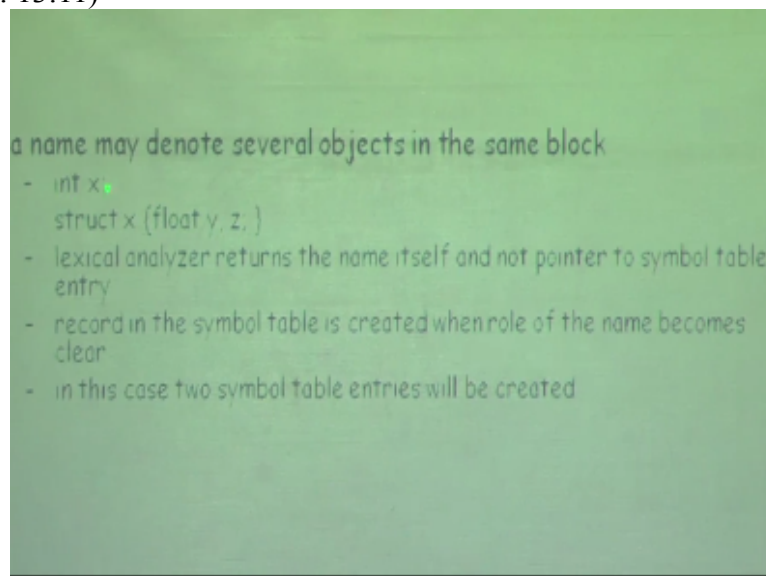
A define which consists of so normally what I will assume from this point onwards is that at least information about each variables continuative so I will if I take this variable I will say that this report now it doesn't matter how I access this recording whether this is part of hashing whether it is part of an area of structures whether this is part of a linked list that is immaterial but as far as this record is concerned.

This will be in continuous locations in memory and this will have all the information about this so this is making sure that I have some kind of format and some kind of location because I do not want to keep for example our token ID at one place I do not want to give this type information at some other place and I do not want to keep its offset for that reasons for so at least for as far as memory location and this allocation.

We want to have if I have entries which are uniform then I may want to keep them outside is not it ok so this example and we were keeping some information because the outside symbol table because I have got to keep my the court uniform the lexical analysis we were saying that we want remember the lexical analysis we were saying that complex is and the let's say new information lexis but there are the four bytes.

And actual lexis information was that so what that did what that may to the inform and some of the informations which is actually part of the record and the part of the symbol you can outside the particular side and information is going to be entered in simple table at various times so keywords are going to be entered initially and ID fires are going to be entered by lexical analyzer sometimes even information about type of this various times I am going to add this information and simple table entry may be set up when role becomes clear so sometimes it is possible analyzer may not know the role of can we shot the stack for example of that in that time I will say let us delay this position so let us look at this example of particular declaration for this now like this.

(Refer Slide Time: 13:11)



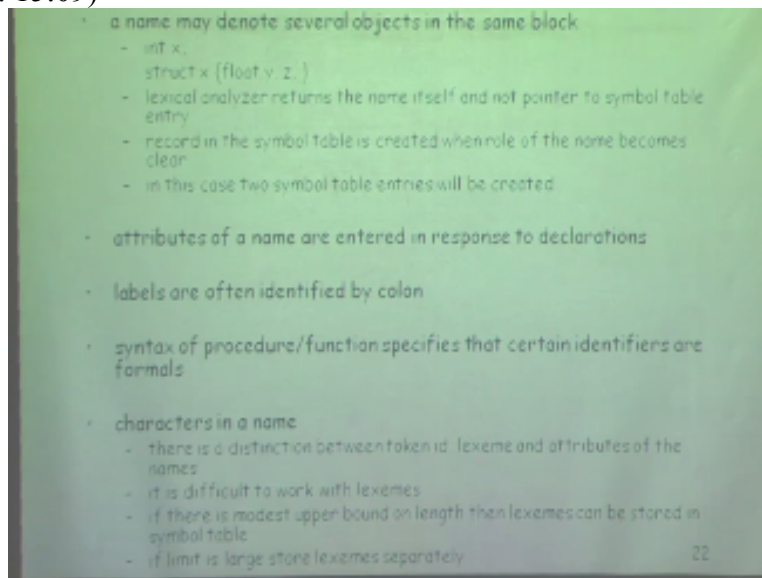
Now this is a possible declaration without an index in our struct x and this structure is than having peace now when I encounter X lexical analyzer remember that lexical analyzer was going to give me a token identifier but at the same time it is it was going to give me information like saying that pointer to symbol table okay.

Now when I say that this is my let us see when this let us see what it does point where I will not know at this point of time when I am just doing lexical analysis that is the point for that maybe know that at what point of time you say we node compiler what is the role of X so I need not move the time but at least I know that this is a variable of credible and this is actually structure so basically I mean again not important.

When we you know it but important things remember is that sometimes we may have to delay the decision and we may have to say that I do not know what is the pointer to the symbol table at corresponding to this heat the range and only when I am able to resolve it that point of time I

place that once okay so in case two symbols table entries are created and only this is created when the rule of the variable becomes clear that then I have keep the names and note the point the important point is this is the take away point that regards with the simple table is created but role is not more whenever I know the rule and that point of time otherwise I will keep handling to the case point clear with everyone okay.

(Refer Slide Time: 15:09)



So attributes are attribute of the name our impending responsibility provision so what are the attributes for kind information is for example an attributes okay and also levels are names that you will have labels and labels so this is again going to some form grammar find out grammar find that we will give the labels that labels that can identify so when I say that how have a token identify sometimes by looking at the syntax.

I need to able to say I know this is an identified so this is are the type labels without having declaration for the syntax itself okay and syntax of procedure function again going to say that what are the formal variables so if I have for example this kind of declaration that syntax will be able to find out the list of all the formal variables and this we have already discussed in case of the lexical analysis that accept point of fine.

I know that now you have some lexemes but I want to eat this information so these are some of the points we have already discussed in lexical analysis in some point of time I know that I have lexis that I want to give some informations separately so that I do not list of the symbol table I do not have to actually have plat form very large bytes okay I can just planned sit just four bytes okay so these are some of the points we have already discussed okay.

(Refer Slide Time: 16:28)

Storage Allocation Information

information about storage locations is kept in the symbol table

Now let us look at how to do storage allocation what kind of information will be allocated okay shortly but some of the reason I am waiting of his background is so that we start writing for symbol table we do not get into the kind of issues business so first I want to discuss all the issues which are required to design of an IR and design a simple table so that when I start writing code which I say that now.

Get information at the symbol table although we see structure of the code as well as the data structures okay so information about storage allocation is also going to be kept in the symbol table so what are the storage I will do we will get it actually code of re locatable now we leave for any so normally for any reasonable size know what is the actual address of variable at time of code generation.

Yes no we do not know it what re locatable code means is that there will be some state occurs and I say that with respect to this base address this variable is at a certain offset so I will only report the offset information and not absolute information we know about the page stack I any place right when the programmer is running so actual address for leading for it which means fill stack it does not mean the actual address okay.

That we will know that means there should be found way I should be able to initialize by same pointer and I code should be able to take care of all the informations where I create the code see this function I should be able to say that the certain information which will be loaded from the either programmers okay.

At the compiler this list phases if you going to get it machine for so compilation did data informations if you not to know time is compilation okay so now it is assembly code using compilation also have to do allocations so names whose storage that look at the run time no

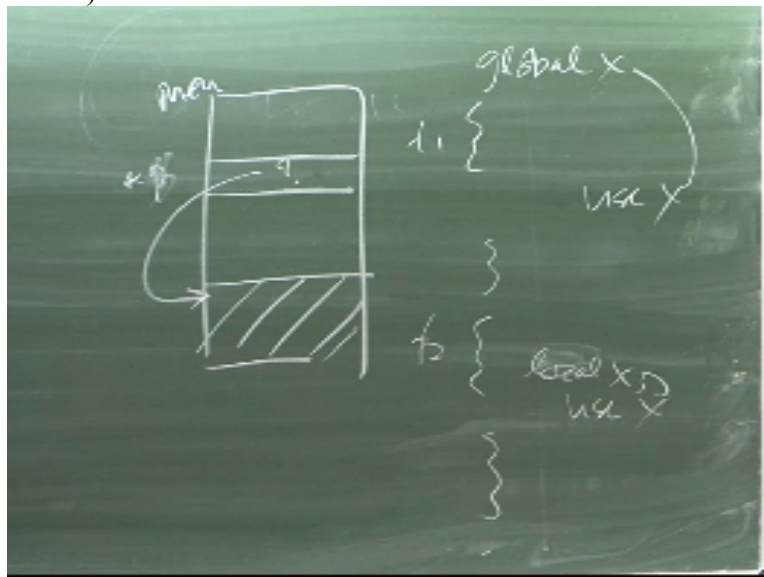
storage allocation is there and what are the variables in which slowly allocating the content what are the kind of variables of the allocation and content and not with the compiler.

Keeps all the object so pointer is I do not get that actually space a type of file pointer when I say something is a file pointer I am located four bytes to this but for the five pointers I allocate four bytes and executing the that object that point of time I am going to allocate such as space on the then we will say that this particular point are did not updated and stop point to this particular allocation right.

So that I will have to be so now when I say has to be run time t o allocate the location is done okay who is going to this information I have none of the therefore this pointer will have this rule like this everyone to get that informations

And suppose I say there is some point there some variable is of type pointer so instead of INT X I suppose I say intense X is the pointer to type A in and how many bytes will I look it for that four bytes and then somewhere I say may not and I get some space of storing that particular INT right okay who will modify the value of x that all and I get some space who will modify the value of x the second question is fear or not okay so suppose this is my memory map.

(Refer Slide Time: 21:06)



And in compiler type and have a compile time I had this is I have said that this is pointer to an object shape so this is start p we will given four bytes now what are the values which is go here a compiler where I knows this value no okay so supported so when my program starts executing at that point of time I will say that this is star P is equal to new some object okay and point of time I will say that here is space.

Which is allocated for this object and this particular point of the star point to resolve it okay so who will modify this value we are still you but who is modified obviously so this has to be done

by the compiler not a manner has been there itself it should pour that I am going to modify all this information.

So basically what happens here is that I have to generate sufficient force in my compiler so that all these modifications one-time modification can take this that immediately tells you something that I must have some control over runtime okay so finally speed index so we should go to manage the runtime system including the activations that are imported and my quote should be such that whenever actually something happens with online.

Then I am going to modify all these locations and so we will able to be store us okay so why there is something is whenever something is happening at runtime still compiling is responsible for doing all those modification and so the variables are changing but you have to modify make sure that the code is such that it will take care of this okay so this we will see when we actually start then the managing the run time stack okay.

So if location is point have been seen and may who storage allocated in run time no storage allocation is going to be allocation for example when I say that something is going to go into me I will know how much space where okay there and therefore no storage allocation is required it comes with some compilations and compiler is only thing it is going to do is it is going to plan out or box or less activation reports.

Anyone know idea of what activation of recode is we take about this later but your assembly programmer language is because we talk about some experience in program you know what activation record is so basically what you want to do is that from each of the slopes I want to score certain information what will be the relevant information relevant information maybe that one of the arguments that are impossible it may require certain space or local variables.

It may require certain space save the status of the machine it may require some space for returning the return value is and so on okay so corresponding to each of this procedures you are going to plan activation record and compiler have to say while spawning to each of the procedures an activation record and compiler we have to say that all this activation record looks how much space and then my or tell you later phase has to fill in information in the activation so we will see this when we do code generation case has to phase information and activation from so we see this can we move code generation for the procedures and functions.

(Refer Slide Time: 24:51)

- List data structure

- simplest to implement
- use a single array to store names and information
- search for a name is linear
- entry and lookup are independent operations
- cost of entry and search operations are very high and lot of time goes into book keeping

So let us look at data structures once again we touch this point so this data structures very easy to implement but and I can use a single area to store names and information and such of or a range will be linear so if you recall we have two functions lookup and update right so when I say look up In that case that itself which is linear and implies which are formal then look up our independent operations and cost of entry is operations are in this case and you have lists there for structures are very high so we do not want to do that hash table obviously I will know the data structures I want to be obvious but one has to very careful of the hash table function design okay. (Refer Slide Time: 25:31)

- entries are declarations of names
- when a lookup is done, entry for appropriate declaration must be returned
- scope rules determine which entry is appropriate
- maintain separate table for each scope
- symbol table for a procedure or scope is compile time equivalent of an activation record
- information about non local is found by scanning symbol table for the enclosing procedures
- symbol table can be attached to abstract syntax of the procedure (integrated into intermediate representation)

So let us take this case okay how do I handle this information so that this variable looks different so in my symbol table so each scope that is having different rules and one way to do scope rules are going to determine these are appropriate at least and we want to for and entries are

declarations of names and when a look up is done entry for appropriate declaration must be returned and the scope rules determine.

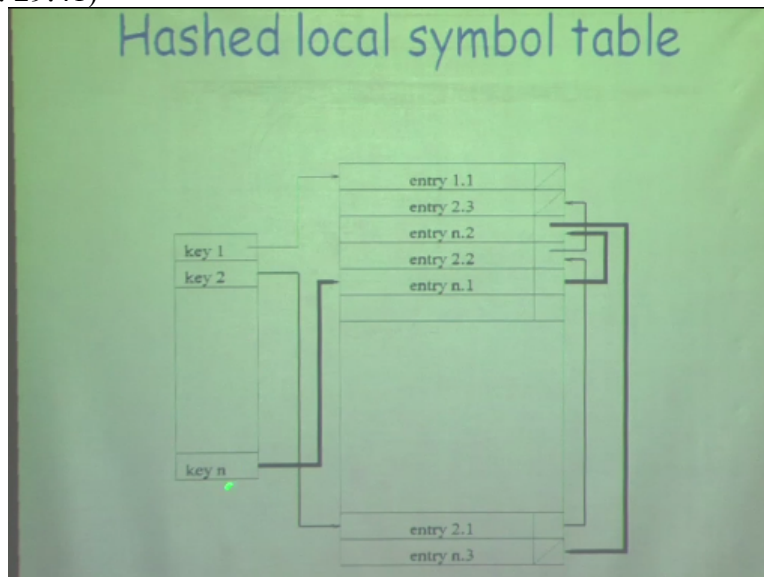
Which entries are appropriate we want to maintain separate table for each scope and that is one way of doing it and symbol table for a procedure or scope is compile time and equivalent of an activation record and information about non-local can be found by scanning symbol table in the enclosing procedures.

What does this mean that if I say that I am looking at use of this particular X and I want to find out do I scan all these symbol tables or do I look at one of the symbol tables where these who so when I go back I say use of this X we like go and look at the symbol table find out use of X no so what I will do is I will say what were the functions and procedures because we are going to use lexical scope use here procedures and functions.

So if I go into deep nesting and I say that I have various levels of nesting if I do not find a variable here then I find that this is being tested by the main function Mexicali and functions or the scopes which are nesting in order to any other that is because of lexical scoping rules and symbol table can be attached to the abstract syntax tree of the procedure itself so what we can do is that I can just take a symbol table and then I can attach this symbol table to the scope information itself which is in the tree.

So let us look at a symbol table structure let me skip this part and come back to this let me show you a symbol table and so that is this okay so here is one way of looking at a hashed local symbol table .

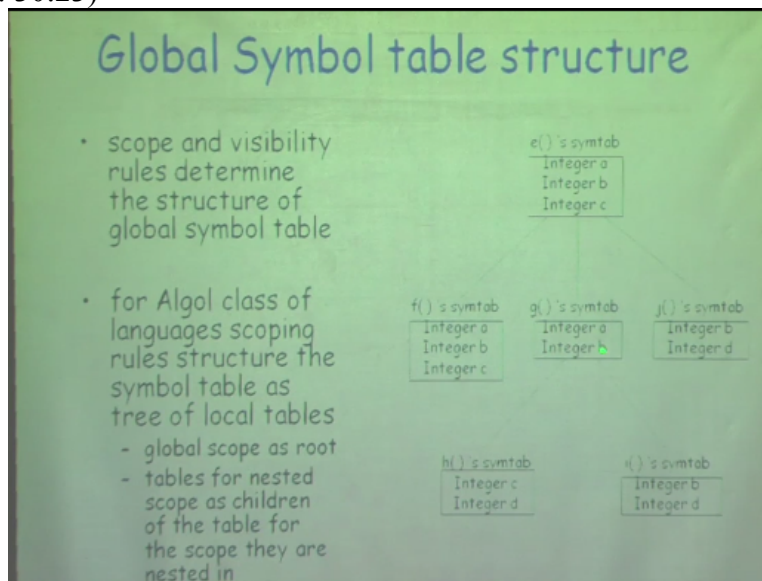
(Refer Slide Time: 29:41)



Now when I say local symbol table what that means is that I am looking single table for only one of the source now these are my entries but you know that if I have entries in the symbol table

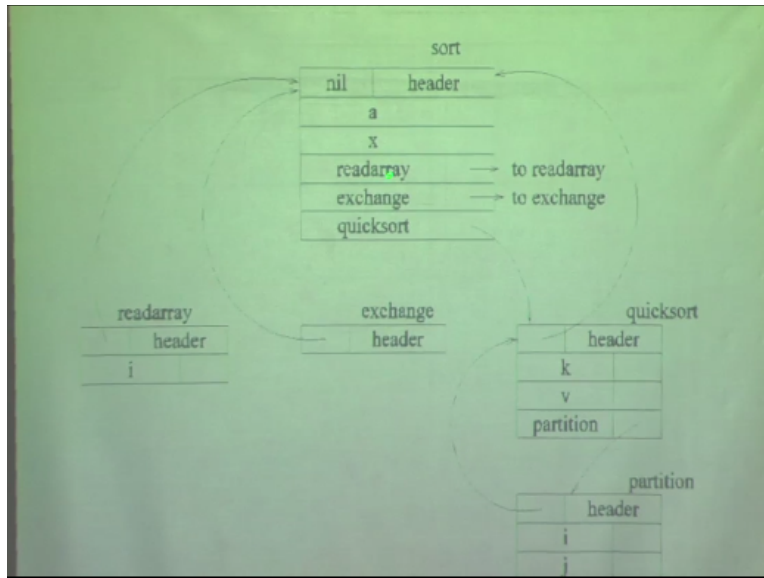
then I am NOT going to have a unique key for it so what may happen is that if I say key n then we take me to entry one but then I have a linked list of all these entries of n so I will have a linked list which say that this is actually now + 1 this is corresponding to n 2 and then I have n3 here so this kind of reduce but if I go for non-local symbol table then I will have symbol table like this .

(Refer Slide Time: 30:23)



Where I say I have multiple symbol tables which are nested now but then I can put one more structure which actually gives you not only pre structure because this is a problem of saying that how do I know what my parent node is so this is the outer node and then these are the inner scopes of which I have symbol table and I want to find use of a non-local in F then I must find out what is that and closing procedure which is this one .

(Refer Slide Time: 30:50)



So I should be able to go back actually what I use is something like this I will go back and start discussing properties of this but I first want to show you a symbol table so that at least to have a look at what is going on so now we will have a look at this program which is having a lot of non local.

(Refer Slide Time: 31:03)

Nesting structure of an example Pascal program

```

program e;
var a, b, c: integer;

procedure f;
var a, b, c: integer;
begin
  a := b+c
end;

procedure g;
var a, b: integer;

  procedure h;
  var c, d: integer;
  begin
    c := a+d
  end;

  procedure i;
  var b, d: integer;
  begin
    b := a+c
  end;

  procedure j;
  var b, d: integer;
  begin
    b := a+d
  end;
end;
end;
end;
  
```

So I have a program which has so it is actually two columns because so I have variables A B and C and the reason I am using this Pascal program because so what you have is program so this is a procedure F this is a procedure to G this is a procedure I and this is a procedure to J and within G H and I are nesting so at top level I have three procedures so basically this yellow color are that just nested inside this and this green colors are nested inside G and then it has its own local variables.

So you can see that global variables are a B and C I also have variables ABC in F and then within G I have a and B and during H I have C and d and so on so now if I put this information in the symbol table what I will have is that I have definitions of ABC here which is the global single table with that I have after G and J and it has definitions of ABC this has definition of A B now you can see that if I am in this particular scope and I want to access C which is a non local variable immediately.

What I need to do is I cannot find it in the local and therefore I must go back and look at this if I go and find it here then I must be able to go back further and therefore I need pointers to go back and the way this is implemented is something like this we say that I have these local variables so corresponding to sort I then have local variables here and then I may have calls to other functions which are like read array exchange and so on and then corresponding to each of this I have local information you can see I have nesting .

And this is the pointer which takes me to the parent node and this point that you can see further to see here at the highest level obviously is needed because it has open except more information which is here so at least I mean you should have mental model of possible before we go too deep into the properties of this we need to make sense that what kind of symbol tables are talking any questions on this clear this discussion 3d is not so difficult to so that we can get into our discussion on how do I generate code okay.

So when we have most closely nested scope rule this thing can be implemented in data structures discussed so far and each procedure can be given a unique name and block must also be numbered procedure number so when I say that I do not have tree structure then I have nested structure name of the procedure and I can have a symbol scope that I want to look up in certain and I want to remove certain symbol tables.

(Refer Slide Time: 35:00)

- Symbols have associated attributes
- typical attributes are name, type, scope, size, addressing mode etc.
- a symbol table entry collects together attributes such that they can be easily set and retrieved
- example of typical names in symbol table

| Name | Type |
|-------|------------------|
| name | character string |
| class | enumeration |
| size | integer |
| type | enumeration |

So symbols have their own attributes and typical attributes are going to be the name which is the Lexi the type scope information size addressing mode etc which I want to associate that for scope generation and symbol table and please may connect together attributes which are going to be then easily an example of typical names in this symbol table maybe that name I will have character string with class and I have enumeration size and integer and so on . Okay and type again can be something which is enumeration this mission type checking when you are saying that I know what present I know one of my basic types and therefore I can always fix number of bits and say that what possible.
(Refer Slide Time: 35:58)

- A major consideration in designing a symbol table is that insertion and retrieval should be as fast as possible
- One dimensional table: search is very slow
- Balanced binary tree: quick insertion, searching and retrieval; extra work required to keep the tree balanced
- Hash tables: quick insertion, searching and retrieval; extra work to compute hash keys
- Hashing with a chain of entries is generally a good approach

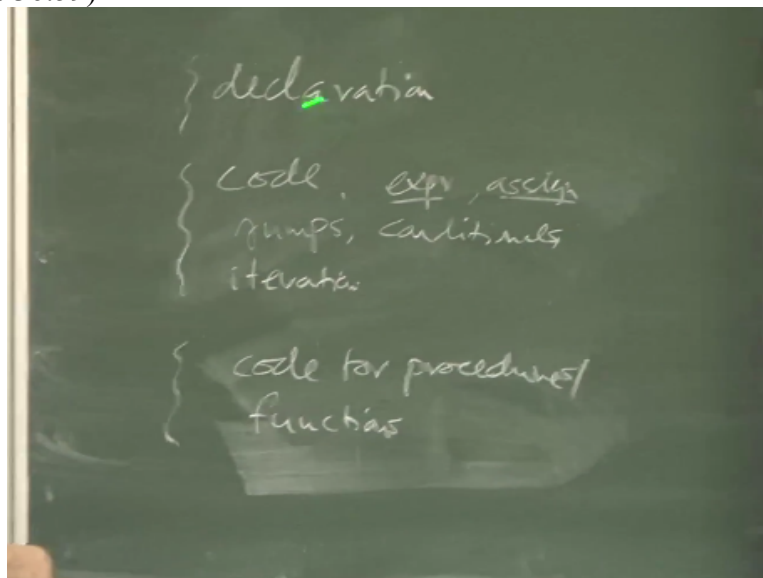
So a major consideration in designing symbol table is that insertion and retrieval should be as fast as possible one dimension symbol table we have seen are slow I hope I am moving forward

and balanced binary trees hash tables and hashing with the chain of entries generally a good approach.

So this is normally what I will assume now so let us take two points from here that I want insertion and retrieval which is fast and I want hashing with which is a chain of entries and there is a Pascal program which I showed you the symbol table which is the in the tree structure and then I wanted to point this back and therefore I now with this background we are ready for code generation.

Now how do I do code generation so first let us look at the kind of things or the parts of the program for which I need to do the code generation and then start looking at each part.

(Refer Slide Time: 36:59)

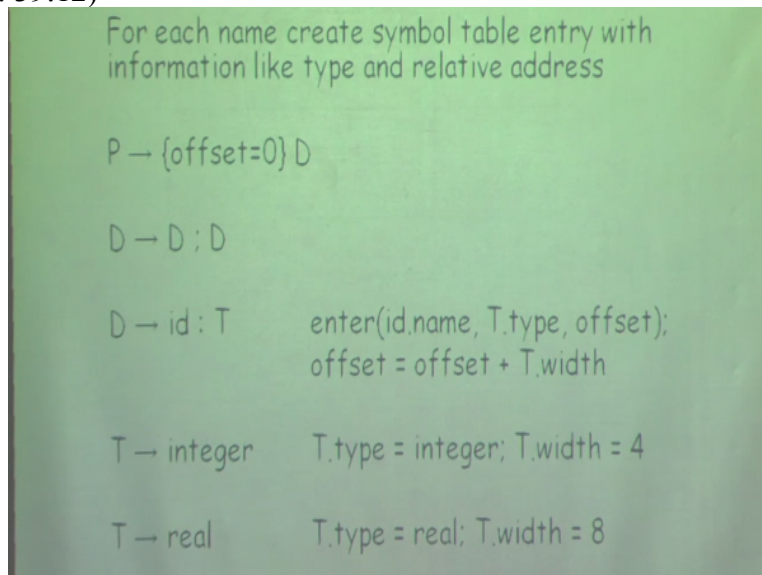


Now my programs are going to have declarations so begin with let us assume that I have only one scope okay so if I have only one scope then what are the kind of code I can write what is the type of code I can write in a single scope I can have silence I can have conditionals so I am going to complete syntax so basically in code part you will see that I will have expression assignment kind of code then I have jumps and which jumps I will have conditionals I will have iteration . And of course when I talk about expressions here all kinds of data structures may come in so I may have symbol variables I have arrays here I may have structures and so on okay but this covers more or less what we can have in one scope and then you may go into code for procedures or functions now when it comes to code for procedures and function how it is different from this because body of each procedure and function is going to be similar . So code for body of the function and procedure is going to be something similar to this but then there are linkages between the functions and procedures so when it comes to specific interfaces

of procedure then we look at all the code generation procedures okay so let us go in this order and if I know that how to handle declarations that means how do I put information in a symbol table how do I actually generate code and how do I make linkages between procedures and function .

Then I know that for the whole program are you there and we use the same method which was syntax directed translation and in syntax directed translation will start now actually writing grammar and seeing that how do I keep on pushing information as I am going through the process of okay .

(Refer Slide Time: 39:12)



```
For each name create symbol table entry with
information like type and relative address

P → {offset=0} D

D → D ; D

D → id : T      enter(id.name, T.type, offset);
                  offset = offset + T.width

T → integer     T.type = integer; T.width = 4

T → real        T.type = real; T.width = 8
```

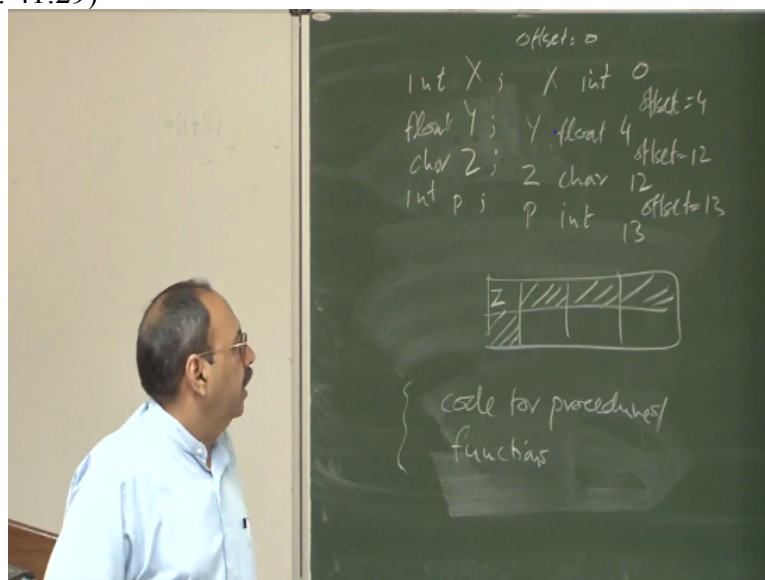
So declaration is the first thing are you that again for each name we want to create symbol table entry next ability which has information like type and its relative address so let us write a grammar rule or a set of grammars so what we are doing here is we are saying that so at this point of time are I am not even putting code information you can see that that we can easily insert it later what we are saying is a program is nothing but the set of declarations and each declaration can make a concatenation of two declarations.

So you can see this is the recursive rules and each declaration is of the form here I have actually a variable and a type integer now you can also see that this declaration could have been of the type of list off identifier followed by a type but that will not bother anymore because we know that even if I have a list here how to handle that we have already seen so what you want to do here is as we are generating relative code .

We are saying that the first variable will be at an offset of you if so we say there is some base address we took an initialized data but I say that the first variable I encountered will be at an

offset of zero and then if I know the size of the variable that means if I know how many bytes this is going to take that is going to determine the offset of the next so at this point of time I will not worry about whether I want to do and find out word boundaries I just both in the size. So what we say is that offset is zero and when I have a declaration like this which says identifier is of certain type then what we want to do is we want to interrupt this information in the symbol table which says that for this particular name type information is given here and offset is what is my global variable globally but as soon as I have to modify my offset I change my offset to whatever was my old offset so then you will start seeing immediately that when I have multiple variables first.

(Refer Slide Time: 41:29)



So if I have variables X Y in Z and let us say these are integer ,float and character okay what will I say that to begin with offsets will be zero okay and then I will make this entry in the symbol tables and I just say that in my symbol table x is of type x at an offset zero but as soon as that happens I will say that in type x this is where I will see that if I have interior type I will say what is the width if I have real type what is the width and who determines what these numbers are already.

And so on we specify these numbers you see a language for bytes always this is partly implementation if you have to be clear on that because depending on the machine for implementing it your file size 15 so if compiler on a 16-8 machine then you are saying it was two bytes and short equal one byte and longing was four bytes you go for a 128-bit machine on that then you say that it integer is 8 bytes and floating-point is 16 bytes .

So this is issue of implementation or now you have to be aware of what your machine is and these numbers 4 and 8 are coming not from the language specification but depends on which language and you have to be now aware of what is being provided on the machine which is no longer you can see I am getting into code generation I now cannot live with just the machine but the source languages.

So these numbers are going to be determined by the implementation platform and therefore I will say that here if I say it takes 4 bytes then immediately I change my offsets add 4 to that and therefore offset will become 4 and when I have float y the information will go the symbol table is something like this and then I will say that float takes 8 bytes and therefore my offset becomes now 12 right and entry corresponding to this is going to be now Z character at an offset of 12 and now suppose character takes one bytes.

Okay and I have another information which says int P then what is the information I will enter it in the symbol table I will say p is at an offset of but with offset of P now as far as this type of writing code is concerned this may be at an offset 13 so as I told you that at this point of time I am not too much worried about that line all my variable said work boundaries or addresses and so on.

But we will see at later point of time that for efficiency reasons accessing because if it takes 4 bytes and imagine you have a machine with 4 bytes then what is happening to your integer is actually it is going into 2d works so it will start storing from here and then it stays this pipe this is highly inefficient so what we may have to do therefore is we may have to you know expand we may have to say that leave these three bytes empty and start allocating from this point onwards me so when pattern happens I have to modify my code .

Now there are various methods of and in this we may say that you take all the variables and variables of different types so this is kind of saying that as I encounter a variable I will enter it in the symbol or I am say sort these variables on their sizes and take the larger size first and then take the smaller size at the end and then you will save some space on padding and so on .

There are various allocation strategies which are used but as far as this code is concerned it does not take care of any issue and is very vanilla kind of code saying that whenever you encountered misinformation for this information symbol and that information is going down so you can see that initially I was able to put only the symbol information and I am able to put type information through type checking and time of code generation I am also able to put now of offset input .

(Refer Slide Time: 46:21)

```

T → array [ num ] of T1
    T.type = array(num.val, T1.type)
    T.width = num.val × T1.width

T → ↑T1
    T.type = pointer(T1.type)

```

So handling declarations suppose now I have a declaration like this say array num of T₁ so what information will go in the symbol table type of t this is only a declaration and so go back and see what I did here I said declarations is an identifier of certain type and then I had the t type with an offset and these are two variables I need before I can make this entry and process further that means in this subsequent part I only need to compute two things.

What is T type and what is the T width right so what is T type here array one dot one of t₁ type and what is t bit here so I will just make width of t₁ and multiply that by number of T type in an array of number of even type and T with this whatever is the number of entities so that can I actually have a declaration I can do this similarly I say that T is a type of pointer t₁ type it will be a pointer to be one type and what will be t width .

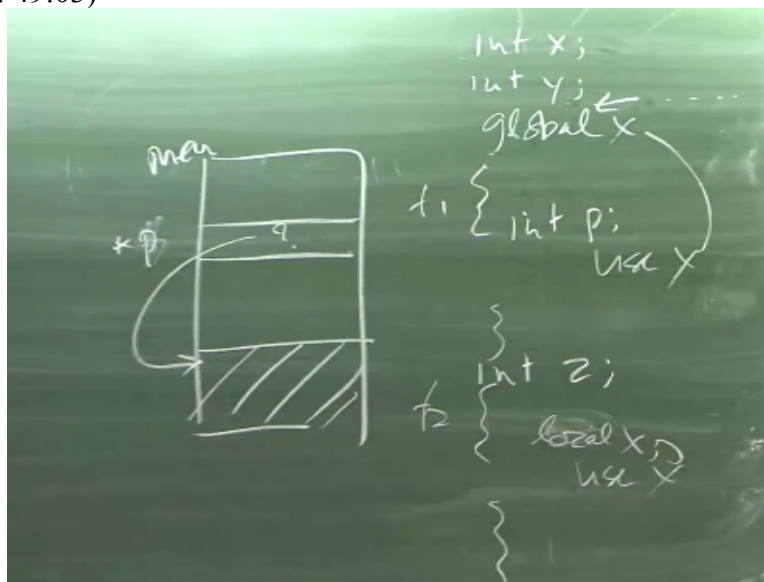
So number which on that particular machine whether my pointer is four bytes or eight bytes t will be a fixed number so this is how I process all my information and say that first I am going to determine so this part you have already seen in type checking that is how I was finding out type of each of the variable so I just need to compute this information which is t-type and t width and when I actually process this particular declaration then I make this entry in the single that is it but this is still saying that all my variables are in the same information I need to do something more so what is that something more I need to do.

(Refer Slide Time: 48:51)

Keeping track of local information

- when a nested procedure is seen, processing of declaration in enclosing procedure is temporarily suspended

So now I want to keep track of all the local information assuming that doing only in the single scope so when nested procedure is processing then temporarily suspended and the deeper message so for example if I am processing for this particular scope.
(Refer Slide Time: 49:03)



And now I say that I start computing certain offsets for this and now suppose I have let us say integer p .So when I was processing variables here so let us put some names here I say integer x ,integer y and I started computing from here offset was zero this took 4 bytes this took 4 bytes and then I come to p what will be offset of p this you should know from the video program for to be offset of P from the code I have shown will it be zero or it will be 8.
It is a different scope right if it is a different scope then how can offset be the same offset is going to go into the same symbol table so when I start looking at a new scope first thing I have to

do is I have to suspect this operation and I have to say whatever was the offset here let us keep track of that now I can have two type of syntax in the language one syntax could be that I can have this declaration then I can have procedures function then I can have all the languages say no

And the local variable that means if I say integer z here what should be offset of Z should be 8 but then I must remember that in between I had a function for which I had to change my offset to zero so I must remember this information it is quite clear so what we therefore do is that whenever a nested procedure is seen processing of declaration of the including procedure is really suspect when I see this scope.

I suspend this and start a new process start the new offset and when I suspect I like to divide waste so what is the good data structure for doing this stack so what I can do is I can put this information on the stack can start processing this and once I finish this I will pop and so what we will do is continue this discussion in the next class .

Acknowledgment

Ministry of Human Resources & Development

Prof. Phalguni Gupta

Co-ordinator, NPTEL IIT Kanpur

Satyaki Roy

Co Co-ordinator, NPTEL IIT Kanpur

Camera

Ram Chandra

Dilip Tripathi

Padam Shukla

Manoj Shrivastava

Sanjay Mishtra

Editing

Ashish Singh

Badal Pradhan

Tapobrata Das

Shubham Rawat

Shikha Gupta

Pradeep Kumar

K.K Mishra

Jai Singh

Sweety Kanaujia

Aradhana Singh

Sweta

Preeti Sachan

Ashutosh Gairola

Dilip Katiyar

Ashutosh Kumar

Light& Sound

Sharwan

Hari Ram

Production Crew

Bhadra Rao

Puneet Kumar Bajpai

Priyanka Singh

Office

Lalty Dutta

Ajay Kanaujia

Shivendra Kumar Tiwari

Saurabh Shukla

Direction

Sanjay Pal

Production Manager

Bharat Lals

an IIT Kanpur Production

@Copyright reserved