

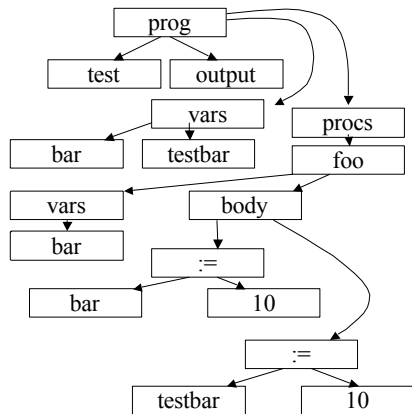
Identifiers

- We can now implement type information, but we need to determine which identifiers are valid (and their meaning) in our program.

— Again, we use the AST:

```

Program test(output);
var bar, testbar : integer;
procedure foo;
var bar : integer;
begin (* foo *)
  bar := 10;
  testbar := 10;
end; (* foo *)
begin (* test *)
  foo;
end.
    
```



The Identifier Table

- To access the information needed, we need an auxiliary data structure
 - We need to know what things are in scope
- Aside: Scope rules
- The scope rules of the language define the *referencing environment* the module sees.
- Pascal has a scope environment that chains back through the nested modules
 - In the example, the local identifier `bar` and the main program's `testbar` identifiers are in scope inside procedure `foo`.
 - The identifier table also needs to implement the scope rules of the language.
- Observation: number the scope regions, the scopes *nest*.
 - We have only *one* thing contributing to the referencing environment at any given level
 - Our scope regions are therefore a simple linear list of blocks in scope.

```

Program Scope(output);
var bar, testbar : integer;
procedure one;
var bar : integer;
procedure two;
begin (* two *)
  bar := 1;
end; (* two *)
begin (* one *)
  bar := 10;
  two;
  testbar := 10;
end; (* one *)
procedure three;
begin (* three *)
  one;
end; (* three *)
begin (* scope *)
  three;
end.
    
```

Scope

```

Program Scope(output);
var bar, testbar : integer;
procedure one;
var bar : integer;
procedure two;
begin (* two *)
  bar := 1;
end; (* two *)
begin (* one *)
  bar := 10;
  two;
  testbar := 10;
end; (* one *)
procedure three;
begin (* three *)
  one;
end; (* three *)
begin (* scope *)
  three;
end.
    
```

Scope

