SYNTAX DIRECTED TRANSLATION – NESTED ROCEDURE

- In translation scheme of fig.1 nonterminal P generates a sequence of declarations of the form id:T
- Syntax directed definition (SDD) is used in putting information to the symbol table.
- Two stacks will be used to hold the pointers of the symbol tables and the current offset
- Tblptr – stack to hold the addresses of the symbol tables of nested procedures
- Offset – stack to hold the current offset

Terms Used in SDD

- **offset** – it is initially set to 0 before any declaration.
- **type and width** are the synthesized attributes of the non terminal
- **mktable (previous)** – creates a new table and returns a pointer to the new table. Argument previous points to the previously created symbol table.
- **enter (table, name, type, offset)** - creates an entry for the name in the symbol table pointed to by the table.
- **addwidth (table, width)** - records the cumulative width of all the entries in the table in the header associated with the symbol table
- **enterproc (table, name, newtable)** - creates a new entry for procedure name in the symbol table pointed to by the table

Productions	Semantic rules
P ->MD	{addwidth(top(tblptr),top(offset)}
M->ε	{t=mktable(nil) push(t, tblptr) push(0,offset)}
D -> D ; D	
D->procid; ND; S	{t=top(tblptr) addwidth(t,tblptr(offset)) pop(tblptr) pop(offset) enterproc(top(tblptr),id.name,t)
D -> id : T	{enter(top(tblptr),id.name, T.type, offset) top(offset) = top(offset) + T.width}
N-> ε	{t=mktable(top(tblptr)) push(t, tblptr) push(0,offset)}
T->integer	T.type = integer ; T.width = 4}
T->real	{T.type = real; T.width = 8}

Fig. 1: SDD for processing declarations in Nested Procedures

Keeping track of Scope information

- In a language with nested procedures, names local to each procedure can be assigned relative addresses
- Similar to the approach of SDD for Decalaration statement, this also can be discussed
- Adding the following rules to the SDD of decalaration statement
  P -> D
  D -> D ; D | id : T | proc id; D; S

Example – Pascal Program with Nested Procedure

```
program sort (in,out)
        var a: integer
        var b: real
        proc readarray
                var i: integer
                var j: integer
                begin ….a....end (readarray)
        begin …..b ….end (sort)
```

Symbol tables for nested procedures for the Example Pascal program

Starting with the start symbol P, when M is derived to ε, a new symbol table is created and a pointer to it is returned to t. for example the address for symbol table sort is 100. t=100. t will be pushed to tblptr stack. Two stacks will be maintained. Stack tblptr will hold the addresses of symbol tables for the various procedures involved in the program. Stack offset will hold the current offset with respect to the symbol table

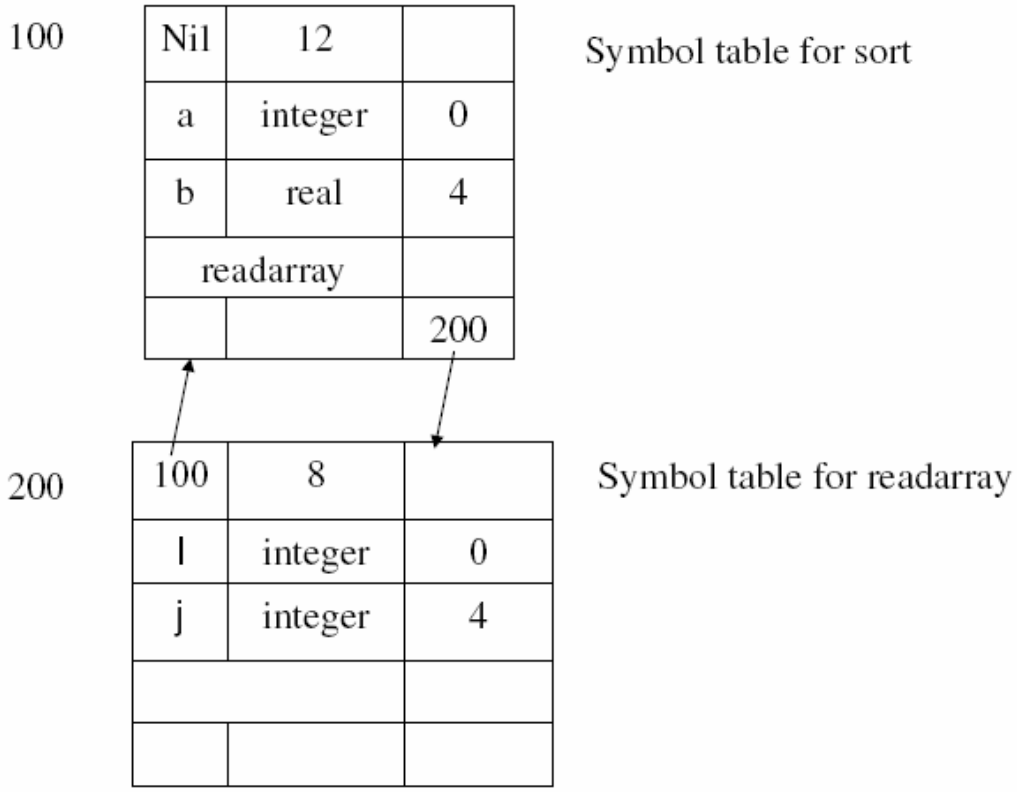
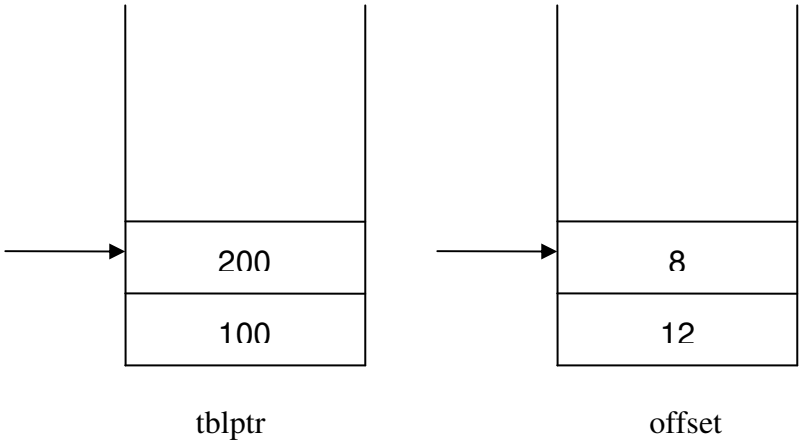
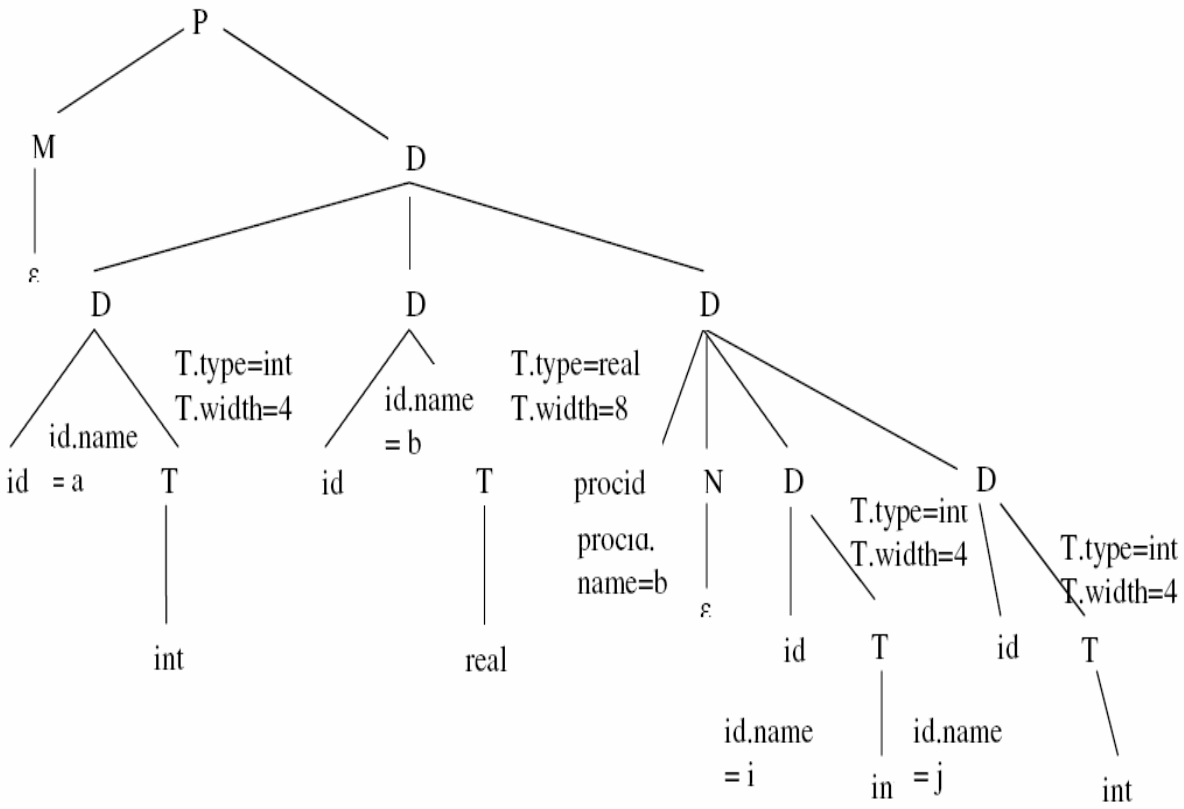


Fig. 2: Symbol table for nested procedures





When a declaration statement is encountered, an entry is made into the symbol table. For example, when var a: integer is encountered var name, its type and the current offset is entered into the table.

When a new procedure is encountered, a new table is created and pointer to it is pushed into the tblptr stack and also the current offset is pushed into the offset stack.

Each when all the entries for declaration statements are over, entry for nested procedure call made in the symbol table and also the cumulative width is added in the header part of the corresponding symbol table