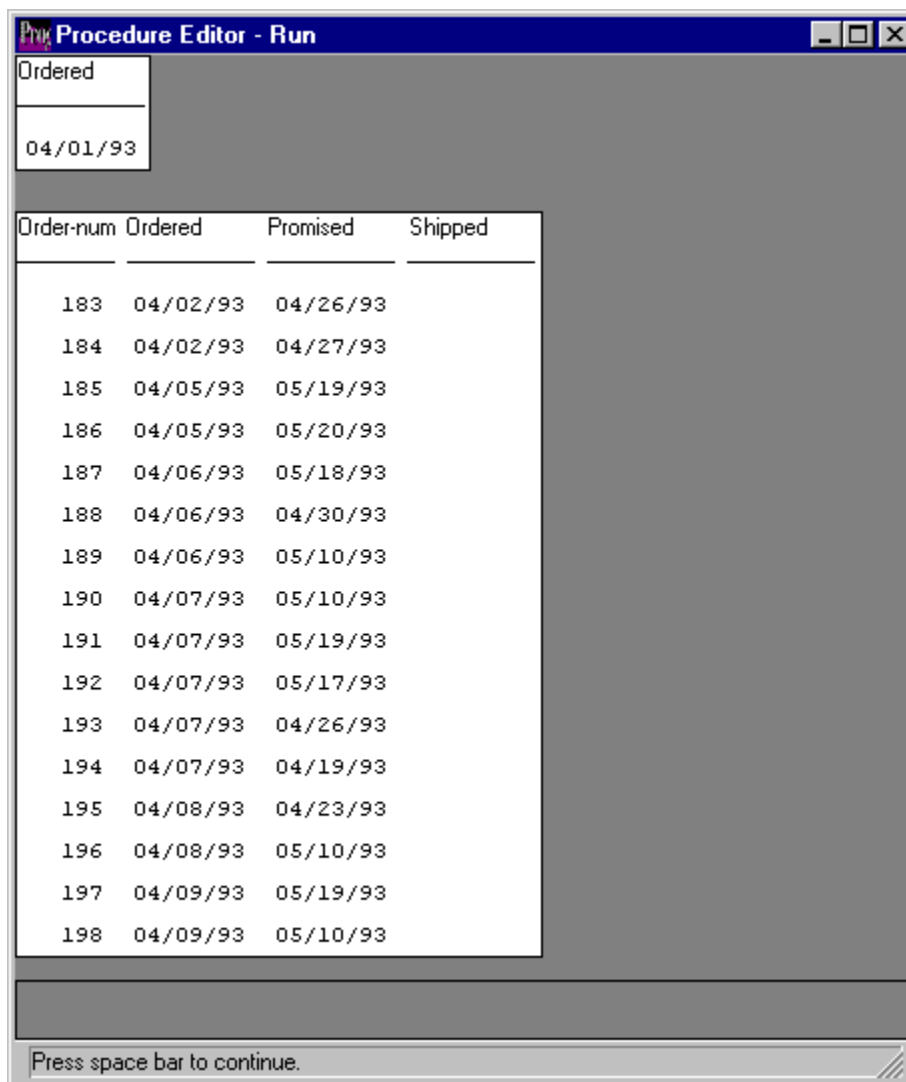


Selecting Records with Variables

slctvar1.p

```
/* slctvar1.p */  
  
def var a like order-date.  
  
repeat:  
  update a.  
  for each order where order-date > a:  
    display order-num order-date promise-date ship-date.  
  end. /* for each */  
end. /* repeat */
```



Procedure Editor - Run

Ordered
04/01/93

Order-num	Ordered	Promised	Shipped
183	04/02/93	04/26/93	
184	04/02/93	04/27/93	
185	04/05/93	05/19/93	
186	04/05/93	05/20/93	
187	04/06/93	05/18/93	
188	04/06/93	04/30/93	
189	04/06/93	05/10/93	
190	04/07/93	05/10/93	
191	04/07/93	05/19/93	
192	04/07/93	05/17/93	
193	04/07/93	04/26/93	
194	04/07/93	04/19/93	
195	04/08/93	04/23/93	
196	04/08/93	05/10/93	
197	04/09/93	05/19/93	
198	04/09/93	05/10/93	

Press space bar to continue.

Selecting Records with Variables

- Use **LIKE** to define a variable that is similar to a field or another variable.
 - ◆ A variable defined with **LIKE** inherits the data type, format and label of the field or variable you specify.
 - ◆ **LIKE** and **AS** are mutually exclusive. You can't use both to define the same variable.
- An **UPDATE** statement does two things:
 - ◆ It displays the value of the variable.
 - ◆ It allows the user to enter data for that variable.
- In this example, we use variable **a** as a starting date for orders by only reading those records in the **FOR EACH** statement whose **order-date** is greater than **a**.
 - ◆ We also use a **REPEAT** block so we can loop through the program and get new starting dates.

PROGRESS Syntax

UPDATE statement

```
UPDATE [UNLESS-HIDDEN]
  [{field [format-phrase] [WHEN expression]} |
   {TEXT(field [format-phrase]...)} |
   {field = expression } | {constant [AT n|TO n]} |
   {^} | {SPACE [(n)]} | {SKIP [(n)]}]...
[GO-ON(key-label...)] [frame-phrase] [editing-phrase]
[NO-ERROR]
```