Somewhere in the early stages of the lifecycle of a 4th Dimension developer, we create procedures that work. Some of those procedures work well. Those that do work well are cobbled together into an application that works. Eventually we reach a stage where we review the work that we have done and we notice that many of the routines that we have written are similar. They are so similar, in fact, that we create some “modular” procedures that allow us to pass a variable number of parameters.

These procedures are coded to be called from different places in our applications. They can handle two parameters from the Company file, four parameters from the Contacts file, three from the Projects file, etc. They are not, of course, truly modular but we are groping our way out of the “hard wired” approach to programming and are attempting to create software that is can be enhanced and maintained.

These “modular” procedures become victims of their own success. They are so useful that they end up being used throughout an application. Since the we can use the Count parameters command in a Case statement, we run around with this “Case of” hammer and beat errant procedures into submission.

There is a better way…

By using a technique called “parameter indirection” we avoid building a Godzilla-line Case statement yet we retain flexibility.

The best way to explain parameter indirection is by using an example. When displaying arrays in a scrollable area, the developer may want to allow a User to delete a row. If the scrollable area displays only a few arrays, it may be sufficient to code a procedure in the following manner:

```
DELETE ELEMENT(alFldFilID;vLine;1)
DELETE ELEMENT(asFldName;vLine;1)
DELETE ELEMENT(alFldID;vLine;1)
```

However, one line of code must be written for each array to be modified. Further, this code must be repeated throughout the database.

As an alternative, we can use a procedure that uses parameter indirection. We accomplish the same result with fewer errors and less code.

```
Case of
:  (◊vonR)
   `Procedure: Delete_Elements
   `© SE Software 1991
  `Douglas von Roeder
   `$1 = the element number where deletion will take place
   `$2 = the number of elements to delete
   `$3 through $n are pointers to the array from which elements will be deleted
   `E.g.  Delete_Elements(vLine;1;»alFldFilID;»asFldName;»alFldID)
End case
C_LONGINT($i;$1;$2)
C_POINTER(${3})
For ($i;3;Count parameters)
 DELETE ELEMENT(${$i}»;$1;$2)
End for
```

This procedure uses only 5 lines of executable code but it can be called from any location in the application. The more it is used, the more valuable it becomes since we are reducing errors and minimizing the size of the structure.

An example that reduces the need for a large Case statement is shown below. In this procedure, the enterability of the fields will be determined by the evaluation of the first parameter.

```
Case of
:  (◊vonR)
   `© 1994 SE Software, Inc.
   `Procedure: Handle_Set_Enter
  `Douglas von Roeder
  `Desc: Sets the data areas enterable or not based on the test passed in.
   `E.g.:
Hndl_Set_Enter([Agents]Type="Rep";»[Agents]Comm_Payable;»[Agents]Comm_Amount)
End case
C_INTEGER($i)
C_BOOLEAN($1;$Enterable)
C_POINTER(${2})
$Enterable:=1
For ($i;2;Count parameters)
 SET ENTERABLE(${$i}»;$Enterable)
End for
```

Parameter indirection can be very useful when you have to export information from different files with varying number of fields. Note that the procedure uses Send packet to and appends a Tab to all packets except for the last packet in the record which received a Char(13).

```
Case of
  : (◊vonR)
     `© SE Software, Inc.
     `Procedure: Expt_Recs
    `Douglas von Roeder
```

```
        ` E.g.  Expt_Recs("Contacts";»[People]First_Name;»[People]Last_Name;»
        `«[People]Company_Name;»[People]Purch_Amt)
      `This proc will export a variable number of fields
End case
C_POINTER($FilePtr)
C_LONGINT($i;$Size;$j;$Parms)
C_TEXT($Text;$FilePath)
$FilePtr:=File(File($2))
$FilePath:=PutFileName ("Save file as...";$1)
If  ($FilePath#"")
  SET CHANNEL(10;$FilePath)
  If (OK=1)
    $Parms:=Count parameters
     $Size:=Records in selection($FilePtr»)
    FIRST RECORD($FilePtr»)
      For ($i;$Size;1;-1)
     $Text:=""
        For ($j;2;$Parms-1)
     Case of
              : (Type(${$j}»)=0) | (Type(${$j}»)=2)
          $Text:=$Text+${$j}»+◊Tab
       Else
            $Text:=$Text+String(${$j}»)+◊Tab
     End case
     End for
         $Text:=$Text+String(${$Parms}»)+◊CR
    SEND PACKET($Text)
     NEXT RECORD($FilePtr»)
   End for
 End if
 SET CHANNEL(11)
  $OK:=SetFileCreator ($FilePath;"XCEL")
End if
```

The final example, shown below, has been published previously in Dimensions. This procedure uses Display List from Foresight Technology to allow the developer to views the contents of arrays. The developer passes in the names of the arrays which then displayed in a Display List's modal window. In the For loop, the procedure builds a text command which is invoked by using the Execute command.

```
  `DoDisplayList
  `© SE Software 1991
   `Example DoDisplayList("aFirst";"aLast")
C_STRING(255;vString255)
C_STRING(15;${1})
C_LONGINT($i)
SetListHeaders (String(Size of array(Get pointer($1)»))+" Elements")
SetListSize  (600;800;1)
SetListBehavior  (1;1;0;0;1;0;0)
vString255:="$Temp:=DisplayList ("   `Initialize the local string variable
For ($i;1;Count parameters)  `Initialize the loop
```

```
 vString255:=vString255+${$i}+";"    `Build the string variable
End for
  `Truncate the last semicolon and add the closing parenthesis
vString255:=Substring(vString255;1;Length(vString255)-1)+")"
EXECUTE(vString255)
```

The code in these sample procedures demonstrates that parameter indirection is a powerful capability. It is not commonly used but when it is used, an astute developer can create powerful, error-free routines that can be moved from one application to the next.