C –15 Solutions

1. Invert a 3x3 Matrix

Data Structure

A new type my_3x3_matrix as a real array (1..3, 1..3)

Subprograms

1. Function to create the matrix
2. Procedure to display the matrix
3. Function to compute the determinant
4. Function to compute the inverse
5. Main program to invert the matrix

Algorithm

Create_Matrix:

```
For I in 1 .. 3
        For J in 1 .. 3
                Accept Matrix(I,J)
Return Matrix to the user
```

Display_Matrix:

```
For I in 1.. 3
        For J in 1.. 3
                Display Matrix (I,J);
        New_Line
```

Compute_Determinant:

1. Convert the matrix into a real_matrix (as defined in Generic_Real_arrays)
2. Compute the determinant using the det function defined in Generic_Real_Arrays.Operations.
3. Return the computed Value

Compute_Inverse:

1. Convert the matrix into a real_matrix (as defined in Generic_Real_arrays)
2. Compute the inverse using the Inverse function defined in Generic_Real_Arrays.Operations.

3. Convert the real_matrix back into the user defined type
4. Return the inverted matrix

Main Program:

1. Prompt the user to enter a matrix
2. Display accepted matrix to the user
3. Check if matrix is singular by computing the determinant.
4. If matrix is not singular (determinant /= 0)
    a. Compute the inverse
    b. Display inverse to the user
5. Else, Display “Cannot Invert”

2. Code Listing

GNAT 3.13p (20000509) Copyright 1992-2000 Free Software Foundation, Inc.

Compiling: c:/docume~2/joeb/desktop/16070c~1/matrix/pset_4_inversion.adb (source file time stamp: 2003-10-08 14:37:14)

```
 1. ------------------------------------------------------
 2. -- Program to invert a 3x3 matrix
 3. -- Programmer : Joe B
 4. -- Date Last Modified : 10/07/03
 5. ------------------------------------------------------
 6.
 7. with Ada.Text_Io;
 8. with Ada.Float_Text_Io;
 9. with Generic_Real_Arrays;
10. with Generic_Real_Arrays.Operations;
11. with Generic_Real_Arrays.Array_Io;
12.
13. procedure Pset_4_Inversion is
14.    -- create instances of the generic matrix packages.
15.    package My_Real_Array is new Generic_Real_Arrays(Float);
16.    package My_Real_Array_Operations is new My_Real_Array.Operations;
17.
18.    -- create a user defined 3x3 matrix
19.    type My_3x3_Matrix is new My_Real_Array.Real_Matrix
20.         (1 .. 3, 1 .. 3);
21.
22.    -- declare local variables for matrices and determinant
23.    A,
24.    B   : My_3x3_Matrix;
25.    Det : Float;
26.
27.    -- user function to create the matrix
28.    function Create_My_Matrix return My_3x3_Matrix is
29.       Input_Matrix : My_3x3_Matrix;
30.
31.    begin
32.       for I in 1 .. 3 loop
```

```
33.       for J in 1 .. 3 loop
34.          Ada.Text_Io.Put("Please Enter Number in (");
35.          Ada.Text_Io.Put(Integer'Image(I));
36.          Ada.Text_Io.Put(",");
37.          Ada.Text_Io.Put(Integer'Image(J));
38.          Ada.Text_Io.Put(") : ");
39.
40.          Ada.Float_Text_Io.Get(Input_Matrix(I,J));
41.          Ada.Text_Io.Skip_Line;
42.       end loop;
43.    end loop;
44.    return Input_Matrix;
45. end Create_My_Matrix;
46.
47. -- user procedure to display the matrix
48. procedure Display_My_Matrix (
49.       Input_Matrix : in     My_3x3_Matrix ) is
50.
51. begin
52.    for I in 1 .. 3 loop
53.       for J in 1 .. 3 loop
54.          Ada.Float_Text_Io.Put(Input_Matrix(I,J));
55.       end loop;
56.       Ada.Text_Io.New_Line;
57.    end loop;
58. end Display_My_Matrix;
59.
60. -- function to compute the inverse
61. function My_Inverse (
62.       Input_Matrix : My_3x3_Matrix )
63.   return My_3x3_Matrix is
64.    -- local variable to convert the user defined matrix to the package defined
65.    -- matrix
66.    My_Real_Matrix : My_Real_Array.Real_Matrix (1 .. 3, 1 .. 3);
67.    Output_Matrix  : My_3x3_Matrix;
68.
69. begin
70.    -- do type conversion from user defined type to package defined type
71.    for I in 1.. 3 loop
72.       for J in 1 .. 3 loop
73.          My_Real_Matrix(I,J) := Input_Matrix(I,J);
74.       end loop;
75.    end loop;
76.    -- compute inverse using the generic package function
77.    My_Real_Matrix := My_Real_Array_Operations.Inverse(My_Real_Matrix);
78.    -- reconvert back to user defined type
79.    for I in 1.. 3 loop
80.       for J in 1 .. 3 loop
81.          Output_Matrix(I,J) := My_Real_Matrix(I,J);
82.       end loop;
83.    end loop;
84.    --return computed inverse
85.    return Output_Matrix;
86.
87. end My_Inverse;
88.
89.
90. function My_Determinant (
```

```
91.          Input_Matrix : My_3x3_Matrix )
92.       return Float is
93.        My_Real_Matrix : My_Real_Array.Real_Matrix (1 .. 3, 1 .. 3);
94.     begin
95.        -- do type conversion from user defined type to package type
96.        for I in 1.. 3 loop
97.           for J in 1 .. 3 loop
98.              My_Real_Matrix(I,J) := Input_Matrix(I,J);
99.           end loop;
100.       end loop;
101.       -- compute the determinant and return to user
102.       return (My_Real_Array_Operations.Det(My_Real_Matrix));
103.    end My_Determinant;
104.
105. begin
106.
107.    -- create and display the matrix
108.    Ada.Text_Io.Put_Line("Please Enter the Matrix : ");
109.    A := Create_My_Matrix;
110.
111.    Ada.Text_Io.Put_Line("Created Matrix : ");
112.    Display_My_Matrix(A);
113.
114.    Ada.Text_Io.New_Line;
115.
116.    -- compute determinant
117.    Det := My_Determinant(A);
118.
119.    -- check for singularity
120.    if Det = 0.0 then
121.       Ada.Text_Io.Put_Line("Cannot invert the matrix");
122.    else
123.       -- compute inverse and display
124.       B := My_Inverse(A);
125.
126.       Ada.Text_Io.New_Line;
127.       Ada.Text_Io.Put_Line("Inverted Matrix : ");
128.       Display_My_Matrix(B);
129.    end if;
130.
131. end Pset_4_Inversion;
```

131 lines: No errors