Unified Engineering
Fall 2004

Problem Set # 5
Solutions
PSET 5 Solution

CP11) Define an Ada95 record with four fields: [5 points]

a. Aircraft_ID of type integer

b. Airline of enumeration type with possible values {United, Delta, SouthWest, JetBlue, American}

c. Direction of type character

Aircraft_Type of type integer

type Airline_Type is
(United, Delta_Airlines, Southwest, JetBlue, American);

type Aircraft is record
    Aircraft_Id : Integer;
    Airline     : Airline_Type;
    Direction   : Character;
    Aircraft_Type : Integer;
end record;

CP12) [35 points]

Write an Ada95 program to:

i. Define four arrays with 5 elements each of types Aircraft_ID, Airline, Direction and Aircraft_Type

ii. Accept user input for 5 aircraft as shown in Figure 1 below.

iii. Sort the elements based on Aircraft_ID array as shown in Figure 2.

iv. Display the sorted arrays to the user

<table>
<thead>
<tr>
<th>Aircraft_ID</th>
<th>Airline</th>
<th>Direction</th>
<th>Aircraft_Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Delta</td>
<td>N</td>
<td>777</td>
</tr>
<tr>
<td>4</td>
<td>American</td>
<td>S</td>
<td>737</td>
</tr>
<tr>
<td>6</td>
<td>JetBlue</td>
<td>E</td>
<td>767</td>
</tr>
<tr>
<td>3</td>
<td>SouthWest</td>
<td>W</td>
<td>737</td>
</tr>
<tr>
<td>1</td>
<td>United</td>
<td>N</td>
<td>747</td>
</tr>
</tbody>
</table>

Figure 1: Four User Input Arrays

<table>
<thead>
<tr>
<th>Aircraft_ID</th>
<th>Airline</th>
<th>Direction</th>
<th>Aircraft_Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>United</td>
<td>N</td>
<td>747</td>
</tr>
</tbody>
</table>

Figure 2: Sorted Arrays
Figure 2: Sorted Arrays Displayed to the User

Turn in a **hard copy** of your **code listing** and an **electronic copy** of your **code**.


Compiling:
c:/docume~1/kristina/mydocu~1/underv~1/unifie~2/fall20~1/pset/pset5~1/aircraft_using_arrays.adb
(source file time stamp: 2004-10-14 03:50:10)

1. -------------------------------------------------------------------
2. -- Program to create four user input arrays of type aircraft_ID,
3. -- Airline, Direction, and Aircraft_type. Sort the arrays based on
4. -- the contents of the aircraft_id array and display the results to
5. -- the user.
6. -- Programmer : Joe B
7. -- Date Created: October 10, 2004
8. ---------------------------------------------------------------------
9. with Ada.Text_Io;
10. with Ada.Integer_Text_Io;
11.
12. procedure Aircraft_Using_Arrays is
13. 
14.   --aircraft ID array contains 5 integers
15.   type Aircraft_Id_Array is array (1 .. 5) of Integer;
16. 
17.   -- airline is an enumeration type with 5 elements as shown below
18.   -- delta is a reserved word in Ada95 and hence we use Delta_Airlines
19.   type Airline is
20.     (United,
21.       Delta_Airlines,
22.       Southwest,
23.       JetBlue,
24.       American);
25. 
26.   --define a new package called airline IO to carry out Airline specific IO
27.   package Airline_Io is new Ada.Text_Io Enumeration_Io(Enum => Airline);
28.   --define an airline array of 5 elements
29.   type Airline_Array is array (1 .. 5) of Airline;
30. 
31.   --define a direction array with 5 elements of type character
32.   type Direction_Array is array (1 .. 5) of Character;
33. 
34.   --define a aircraft array of type integer
35.   type Aircraft_Type_Array is array (1 .. 5) of Integer;
36. 
37.   --define four arrays to store the user input data
38.   My_Aircraft_Id      : Aircraft_Id_Array;
39.   My_Airline_Array   : Airline_Array;
```ada
--define four temporary variables for carrying out the swaps
Temp_Aircraft_Id  : Integer;
Temp_Aircraft_Type : Integer;
Temp_Direction    : Character;
Temp_Airline      : Airline;

begin
--get 5 inputs from the user
for I in 1 .. 5 loop
   Ada.Text_Io.Put("For Aircraft ");
   Ada.Text_Io.Put_Line(Integer'Image(I));
   Ada.Text_Io.Put("Please enter Aircraft ID : ");
   Ada.Integer_Text_Io.Get(My_Aircraft_Id(I));
   Ada.Text_Io.Skip_Line;
   Ada.Text_Io.Put("Please enter Airline: ");
   Airline_Io.Get(My_Airline_Array(I));
   Ada.Text_Io.Skip_Line;
   Ada.Text_Io.Put("Please enter Aircraft Direction: ");
   Ada.Text_Io.Get(My_Direction_Array(I));
   Ada.Text_Io.Skip_Line;
   Ada.Text_Io.Put("Please enter Aircraft Type: ");
   Ada.Integer_Text_Io.Get(My_Aircraft_Type_Array(I));
   Ada.Text_Io.Skip_Line;
end loop;

--display input elements to the user
Ada.Text_Io.Put_Line("Unsorted Array is: ");
for I in 1 .. 5 loop
   Ada.Text_Io.Put_Line("Aircraft ID      Airline      Direction      Type");
   Ada.Text_Io.Put(Integer'Image(My_Aircraft_Id(I)));
   Ada.Text_Io.Put("      ");
   Ada.Text_Io.Put(Airline'Image(My_Airline_Array(I)));
   Ada.Text_Io.Put("      ");
   Ada.Text_Io.Put(My_Direction_Array(I));
   Ada.Text_Io.Put("      ");
   Ada.Text_Io.Put(Integer'Image(My_Aircraft_Type_Array(I)));
   Ada.Text_Io.Put("      ");
end loop;
Ada.Text_Io.New_Line;

--sort the elements based on aircraft_id
for I in 1 .. 4 loop
   for J in I+1 .. 5 loop
      if My_Aircraft_Id(I)> My_Aircraft_Id(J) then
         --store the temporary values
         Temp_Aircraft_Id := My_Aircraft_Id(I);
         Temp_Aircraft_Type := My_Aircraft_Type_Array(I);
         Temp_Airline := My_Airline_Array(I);
         Temp_Direction := My_Direction_Array(I);
      end if;
   end loop;
end loop;
```

-- store the values in jth location into ith location
My_Aircraft_Id(I):= My_Aircraft_Id(J);
My_Aircraft_Type_Array(I):= My_Aircraft_Type_Array(J);
My_Airline_Array(I):= My_Airline_Array(J);
My_Direction_Array(I):= My_Direction_Array(J);

--store the values in temp into the jth location
My_Aircraft_Id(J):=Temp_Aircraft_Id;
My_Aircraft_Type_Array(J):=Temp_Aircraft_Type;
My_Airline_Array(J):= Temp_Airline;
My_Direction_Array(J):= Temp_Direction;
end if;
end loop;
end loop;

-- display the sorted arrays to the user
Ada.Text_Io.Put_Line("Sorted Array is: ");
for I in 1 .. 5 loop
Ada.Text_Io.Put_Line("Aircraft ID  Airline  Direction  Type");
Ada.Text_Io.Put(Integer'Image(My_Aircraft_Id(I)));
Ada.Text_Io.Put(" ");
Ada.Text_Io.Put(Airline'Image(My_Airline_Array(I)));
Ada.Text_Io.Put(" ");
Ada.Text_Io.Put(My_Direction_Array(I));
Ada.Text_Io.Put(" ");
Ada.Text_Io.Put(Integer'Image(My_Aircraft_Type_Array(I)));
Ada.Text_Io.Put(" ");
Ada.Text_Io.New_Line;
ext loop;
end Aircraft_Using_Arrays;

CP13) [60 points]

a. Define your own package, with the type definition of an array with ten aircraft records (as defined in question 1 above), and four functions/procedures to:

i. Sort the array in ascending order based on Aircraft_ID.

ii. Display the contents of records in the array

iii. Read in aircraft information into the array based on user input.

iv. Read in aircraft information into the array from an input file called aircraft_record_input.txt

b. Write a test program that uses your package to:

i. Create an array with 5 records input by the user, and 5 records read in from the file called aircraft_record_input.txt
ii. Sort the records based on the Aircraft_Id

iii. Display each record of the array as shown in Figure 3.

Aircraft_ID: 1
Airline: United
Direction: N
Aircraft_Type: 747
...
...
Aircraft_ID: 10
Airline: Delta
Direction: N
Aircraft_Type: 777

Figure 3. Sorted Records Displayed to User

iv. Store the contents into a sorted array called sorted_aircraft_records.txt

Turn in a hard copy of your code listing and an electronic copy of your code.
package My_Aircraft_Package is

type Airline_Type is
  (United,
   Delta_Airlines,
   Southwest,
   Jetblue,
   American);

type Aircraft is
  record
    Aircraft_Id    : Integer;
    Airline       : Airline_Type;
    Direction     : Character;
    Aircraft_Type : Integer;
  end record;

type Aircraft_Array is array (1 .. 10) of Aircraft;

procedure Get_Aircraft_From_Array (My_Aircraft_Array : in out Aircraft_Array);

procedure Get_Aircraft_From_File (My_Aircraft_Array : in out Aircraft_Array);

procedure Display_Array (My_Aircraft_Array : in Aircraft_Array);

procedure Sort_Array (My_Aircraft_Array : in out Aircraft_Array);

procedure Store_Aircraft_In_File (My_Aircraft_Array : in Aircraft_Array);

end My_Aircraft_Package;

40 lines: No errors
package body My_Aircraft_Package is

  -- define a new package called airline IO to carry out Airline specific IO
  package Airline_Io is new Ada.Text_Io.Enumeration_Io(Enum => Airline_Type);

  -- procedure to get user input into array
  procedure Get_Aircraft_From_Array (
    My_Aircraft_Array : in out Aircraft_Array ) is
  begin
    -- get 5 inputs from the user
    for I in 1 .. 5 loop
      Ada.Text_Io.Put("For Aircraft ");
      Ada.Text_Io.Put_Line(Integer'Image(I));
      Ada.Text_Io.Put("Please enter Aircraft ID : ");
      Ada.Text_Io.Get(My_Aircraft_Array(I).Aircraft_Id);
      Ada.Text_Io.Skip_Line;
      Ada.Text_Io.Put("Please enter Airline: ");
      Airline_Io.Get(My_Aircraft_Array(I).Airline);
      Ada.Text_Io.Skip_Line;
      Ada.Text_Io.Put("Please enter Aircraft Direction: ");
      Ada.Text_Io.Get(My_Aircraft_Array(I).Direction);
      Ada.Text_Io.Skip_Line;
      Ada.Text_Io.Put("Please enter Aircraft Type: ");
      Ada.Integer_Text_Io.Get(My_Aircraft_Array(I).Aircraft_Type);
      Ada.Text_Io.Skip_Line;
    end loop;
  end Get_Aircraft_From_Array;

  -- procedure to get 5 records from a file.
  -- notes:
  -- 1. you will have to have an extra line after the inputs if you
  -- 2. use the skip_line. An alternative is to check for end_of_File
  -- 3. before the last skip_line in the loop.
  -- 4. The input file has to be in the same folder as the code
  -- 5. The output file is created in the same folder as the code
  procedure Get_Aircraft_From_File ( 
    My_Aircraft_Array : in out Aircraft_Array ) is
  begin
    Ada.Text_Io.Open(My_File, Ada.Text_Io.In_File, "aircraft_record_input.txt");
    -- get 5 inputs from the file
    for I in 6 .. 10 loop
      Ada.Integer_Text_Io.Get(My_File, My_Aircraft_Array(I).Aircraft_Id);
      Ada.Text_Io.Skip_Line(My_File);
    end loop;
  end Get_Aircraft_From_File;
52.          Airline_Io.Get(My_File, My_Aircraft_Array(I).Airline);
53.          Ada.Text_Io.Skip_Line(My_File);
54.          Ada.Text_Io.Get(My_File, My_Aircraft_Array(I).Direction);
55.          Ada.Text_Io.Skip_Line(My_File);
56.          Ada.Integer_Text_Io.Get(My_File, My_Aircraft_Array(I).Aircraft_Type);
57.          Ada.Text_Io.Skip_Line(My_File);
58.       end loop;
59.       Ada.Text_Io.Close(My_File);
60.       end Get_Aircraft_From_File;
61.       end loop;
62.       -- procedure to display contents of the array to the user
63.       procedure Display_Array ( 
64.             My_Aircraft_Array : in     Aircraft_Array ) is 
65.       begin 
66.             -- display the array to the user 
67.             for I in 1 .. 10 loop 
68.                 Ada.Text_Io.Put_Line("ID      Airline      Direction      Type");
69.                 Ada.Text_Io.Put(Integer'Image(My_Aircraft_Array(I).Aircraft_Id));
70.                 Ada.Text_Io.Put("      ");
71.                 Ada.Text_Io.Put(Airline_Type'Image(My_Aircraft_Array(I).Airline));
72.                 Ada.Text_Io.Put("      ");
73.                 Ada.Text_Io.Put(My_Aircraft_Array(I).Direction);
74.                 Ada.Text_Io.Put("      ");
75.                 Ada.Text_Io.Put(Integer'Image(My_Aircraft_Array(I).Aircraft_Type));
76.                 Ada.Text_Io.Put("      ");
77.                 Ada.Text_Io.New_Line;
78.             end loop; 
79.       end Display_Array; 
80.       --procedure to sort the array 
81.       procedure Sort_Array ( 
82.             My_Aircraft_Array : in out Aircraft_Array ) is 
83.             Temp : Aircraft; 
84.       begin 
85.             --sort the elements based on aircraft_id 
86.             for I in 1 .. 9 loop 
87.                 for J in I+1 .. 10 loop 
88.                     if My_Aircraft_Array(I).Aircraft_Id > My_Aircraft_Array(J).Aircraft_Id then 
89.                         Temp:=My_Aircraft_Array(I);
90.                         My_Aircraft_Array(I) := My_Aircraft_Array(J);
91.                         My_Aircraft_Array(J) := Temp;
92.                     end if;
93.                 end loop; 
94.             end loop; 
95.       end Sort_Array; 
96.       --procedure to store the sorted array into the file 
97.       procedure Store_Aircraft_In_File ( 
98.             My_Aircraft_Array : in     Aircraft_Array ) is 
100.       begin 
103.             --get 5 inputs from the file 
104.             for I in 1 .. 10 loop 
105.                 Ada.Integer_Text_Io.Put(My_File, My_Aircraft_Array(I).Aircraft_Id);
110.  Ada.Text_Io.New_Line(My_File);
111.  Airline_Io.Put(My_File, My_Aircraft_Array(I).Airline);
112.  Ada.Text_Io.New_Line(My_File);
113.  Ada.Text_Io.Put(My_File, My_Aircraft_Array(I).Direction);
114.  Ada.Text_Io.New_Line(My_File);
115.  Ada.Integer_Text_Io.Put(My_File, My_Aircraft_Array(I).Aircraft_Type);
116.  Ada.Text_Io.New_Line(My_File);
117.  end loop;
118.  Ada.Text_Io.Close(My_File);
119.
120.  end Store_Aircraft_In_File;
121.  end My_Aircraft_Package;
122.

122 lines: No errors
-- Program to create an array of 10 records of type aircraft, half
-- of which are entered by the user and the remaining are read in
-- from a file called aircraft_input_records.txt
-- Programmer : Joe B
-- Date Created: October 10, 2004

with Ada.Text_Io;
with My_Aircraft_Package;
use My_Aircraft_Package;

procedure Aircraft_Using_Records is
begin
    Get_Aircraft_From_Array(My_Aircraft_Array);
    Get_Aircraft_From_File(My_Aircraft_Array);
    Ada.Text_Io.Put_Line("Unsorted Array");
    Ada.Text_Io.New_Line;
    Display_Array(My_Aircraft_Array);
    Sort_Array(My_Aircraft_Array);
    Ada.Text_Io.New_Line;
    Ada.Text_Io.Put_Line("Sorted Array");
    Ada.Text_Io.New_Line;
    Display_Array(My_Aircraft_Array);
    Store_Aircraft_In_File(My_Aircraft_Array);
end Aircraft_Using_Records;
aircraft_record_input.txt
120
SouthWest
N
737
110
JetBlue
S
737
130
American
W
777
121
United
E
767
140
JetBlue
S
767

aircraft_record_output.txt
1
UNITED
N
747
3
SOUTHWEST
W
737
4
AMERICAN
S
737
6
JETBLUE
E
767
10
DELTA_AIRLINES
N
777
110
JETBLUE
S
737
120
SOUTHWEST
N
737
121
UNITED
E
767
130
AMERICAN
W
777
140
JETBLUE
S
767
setting up loop eqns.

for \( i_a \):

\[-V_7 + i_a R_1 + (i_a - I_e)R_2 + (i_a - I_e)R_3 = 0\]

for \( i_b \):

\[(i_b - i_e)R_3 + (i_b - I_e)R_4 + i_b R_5 = 0\]

gathering terms

\[(R_1 + R_2 + R_3)i_a - R_3i_b = I_e R_2 - V_7\]

\[-R_3 i_a + (R_3 + R_4 + R_5)i_b = -I_e R_4\]

plugging in given terms

\[7 i_a - 4 i_b = 1\]  \(\text{(1)}\)

\[-4i_a + 11i_b = 18\]  \(\text{(2)}\)

Solving these eqns.

multiply \(\text{(1)}\) by 4, \(\text{(2)}\) by 4

\[28i_a - 16i_b = 4\]

\[-28i_a + 77i_b = 126\]

\[61i_b = 130\]

\[i_b = \frac{130}{61}\]

so \(i_e = \frac{83}{61}\)}
so the voltages for all the components are:

\[ V_1 = -i_2 R_1 = -\frac{23}{61} V = -1.36 \]

\[ V_2 = -(i_2 - i_4) R_2 = \frac{200}{61} V = 3.28 \]

\[ V_3 = -(i_2 - i_4) R_3 = \frac{153}{61} V = 2.50 \]

\[ V_4 = -(i_3 - i_4) R_4 = \frac{318}{61} V = 5.22 \]

\[ V_5 = -i_5 R_5 = -\frac{130}{61} V = -2.13 \]

\[ V_6 = e_1 - e_5 \text{ but we don't know } e_1 \text{ or } e_5 \text{ so we find } \text{ using known } v_{17} \]

\[ v_{17} = \frac{5}{61} V \]

The currents are:

\[ i_1 = -i_2 = -\frac{83}{61} A = -1.36 \]

\[ i_2 = -(i_2 - i_4) = \frac{130}{61} A = 1.60 \]

\[ i_3 = -(i_2 - i_4) = \frac{47}{61} A = 0.77 \]

\[ i_4 = -(i_3 - i_5) = \frac{53}{61} A = 0.87 \]

\[ i_5 = -i_2 = -\frac{130}{61} A = -2.13 \]

\[ i_6 = 3 A \]

\[ i_7 = i_6 = \frac{83}{101} A = 1.36 \]
Use node method:

\[ e_1 \text{ supernode:} \]

\[ \frac{e_1 + v_2 - e_2}{R_s} + \frac{e_1 - e_2}{R_b} + \frac{e_1 - v_1}{R_3} + \frac{e_1}{R_4} = 0 \]

Node e_2:

\[ e_2 - \frac{(R_4 + v_2)}{R_b} + \frac{e_2 - e_1}{R_b} + \frac{e_2}{R_7} = 0 \]

Gathering terms:

\[ (G_5 + G_4 + G_5 + G_6)e_1 - G_6e_2 = -G_5v_2 + G_5v_1 \]

\[ -(G_5 + G_6)e_1 + (G_5 + G_6 + G_7)e_2 = G_6v_2 \]

Plug in numbers:

\[ \frac{5}{4} e_1 - \frac{1}{8} e_2 = \frac{35}{8} \quad (1) \]

\[ -\frac{1}{4} e_1 + \frac{1}{2} e_2 = \frac{5}{8} \quad (2) \]

Solve for e_2:

Multiply (2) by 5

\[ \frac{5}{4} e_1 - \frac{1}{8} e_2 = \frac{35}{8} \]

\[ -\frac{5}{4} e_1 + \frac{35}{8} e_2 = \frac{25}{4} \]

\[ 6/24 e_2 = 15/2 \]

\[ e_2 = 150/64 = V_0 \]
Because this circuit is a voltage divider, we can calculate the open-circuit voltage directly. We have that:

\[ V_T = \frac{R_1}{R_1 + R_2 + R_3} \cdot V_4 \]

\[ \Rightarrow V_T = \frac{2 \Omega}{2 \Omega + 4 \Omega + 3 \Omega} \cdot 12V = \frac{2 \cdot 12V}{9 \Omega} = 2.6667V \]

We can find \( R_T \) by setting all voltage sources to zero:
\[ R_T = R_1 \parallel (R_2 + R_3) = 2 \Omega \parallel (4 \Omega + 3 \Omega) = 2 \Omega \parallel 7 \Omega \]

\[ = \frac{2 \Omega \times 7 \Omega}{2 \Omega + 7 \Omega} = \frac{14}{9} \Omega = 1.556 \Omega \]

So we arrive at our final solution:

| \( V_T \) | 2.667 V |
| \( I_N \) | \( \frac{V_T}{R_T} \) |
| \( R_T \) | 1.556 \Omega |
| \( R_N \) | \( R_T \) |

Thevenin Equivalent Circuit

Norton Equivalent Circuit

2. The most convenient method of solving this circuit is to add a test current source.

We can then solve directly using the loop method.
Applying Kirchhoff's Voltage Law:

\[ i_A : R_1(i_A - I) + R_2i_A - V_5 = 0 \]
\[ i_A - I_S : R_2(i_A - I_S - I) + V_5 + R_4(i_A - I_S) = 0 \]

Eliminating \( V_5 \),

\[ (R_1 + R_2 + R_3 + R_4)i_A = (R_1 + R_2)I + (R_2 + R_4)I_S \]

Now we can plug in our numerical values:

\[ (1.2 + 3.2 + 3.2 + 1.2)i_A = (1.2 + 3.2)I + (3.2 + 1.2)(8A) \]

\[ \Rightarrow 8i_A = 4I + 32A \]

\[ \Rightarrow i_A = \frac{1}{2}I + 4A \]

We can now solve for the voltage across the terminals:

\[ V = V_1 + V_2 = R_1(I - i_A) + R_2(I - (i_A - I_S)) \]
\[(R_1 + R_2)I - (R_1 + R_2)i_A + R_2I_S\]

\[= (4\Omega)I - (4\Omega)i_A + (3\Omega)(8\text{A})\]

Plugging in our expression for \(i_A\),

\[V = (4\Omega)I - (4\Omega)(\frac{1}{2}I + 4\text{A}) + 24\text{V}\]

\[= \frac{(2\Omega)I + 8\text{V}}{R_T} + V_T\]

So we arrive at our final solution:

\[
\begin{array}{c}
R_T = R_N = 2\Omega \\
V_T = 8\text{V} \\
I_N = \frac{V_T}{R_T} = 4\text{A}
\end{array}
\]

**Thevenin Equivalent Circuit**  **Norton Equivalent Circuit**