Multi-Dimensional Arrays In Pascal

In this section of notes you will learn about how and when to use multi-dimensional arrays.

When To Use Arrays Of Different Dimensions

• Determined by the data – the number of categories of information determines the number of dimensions to use.

Examples:

• (1D array)
  • Tracking grades for a class
  • Each cell contains the grade for a student i.e., grades[i]
  • There is one dimension that specifies the student

One dimension (which student)

• (2D array)
  • Personal finances program
  • One dimension of information specifies the financial category (cash in or cash out).
  • The other dimension is used to specify the time
When To Use Arrays Of Different Dimensions (2)

- (2D array continued)

<table>
<thead>
<tr>
<th>Financial category</th>
<th>January</th>
<th>February</th>
<th>March</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td>Income</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-Rent</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-Food</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-Fun</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-Car</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-Misc</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Net income</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Time

When To Use Arrays Of Different Dimensions (3)

- (2D array continued)
- Notice that each row is merely a 1D array
- (A 2D array is an array containing rows of 1D arrays)

<table>
<thead>
<tr>
<th>[1]</th>
<th>[2]</th>
<th>[3]</th>
<th>[4]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Income</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-Rent</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-Food</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-Fun</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-Car</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-Misc</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Net income</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
When To Use Arrays Of Different Dimensions (4)

- (3D array – take the 2D array but allow for multiple people)
- The third dimension specifies whose finances are being tracked.

When To Use Arrays Of Different Dimensions (5)

<table>
<thead>
<tr>
<th>Bob’s finances</th>
<th>Mary’s finances</th>
<th>John’s finances</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>January</td>
<td>February</td>
<td>March</td>
</tr>
<tr>
<td>Income</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-Rent</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-Food</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-Fun</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-Car</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-Misc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Net income</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Declaring Multi-Dimensional Arrays

Format:

(Two dimensional arrays)

\[
\text{Name} : \text{array } [\text{min..max}, \text{min..max}] \text{ of type;}
\]

(Three dimensional arrays)

\[
\text{Name} : \text{array } [\text{min..max}, \text{min..max}, \text{min..max}] \text{ of type;}
\]

Example:

\[
\begin{align*}
\text{var johnFinances} & : \text{array } [1..7, 1..7] \text{ of real;} \\
\text{var cube} & : \text{array}[1..3, 1..4, 1..6] \text{ of char;}
\end{align*}
\]

Declaring Multi-Dimensional Arrays As A Type

Format:

Type declaration

\[
\text{Type name} = \text{array } [\text{min..max}, \text{min..max}] \text{ of element type;}
\]

\[
\text{Type name} = \text{array } [\text{min..max}, \text{min..max}, \text{min..max}] \text{ of element type;}
\]

Variable declaration

\[
\text{Array name} : \text{Type name;}
\]
Declaring Multi-Dimensional Arrays As A Type (2)

Example
Type declaration
Finances = array [1..7, 1..7] of real;
Cube = array[1..3, 1..4, 1..6] of char;

Variable declaration
var johnFinances : Finances;
var aCube : Cube;

Accessing / Assigning Values To Elements

Format:
name [row][column] := name [row][column];

Example:
finances [1][1] := 4500;
writeln (finances[1][1]);
Example Program: Map Generator And Editor

You can find the full program in Unix under:
/home/231/examples/arrays/map.p

Example Program: Map Generator And Editor:
Breaking The Problem Down

map.p

makeBorder  populate  displayWorld  editWorld
Example Program: Map Generator And Editor

program map (input, output);
const
    MAX_ROWS    = 10;
    MAX_COLUMNS = 10;

    type
        Level = array[1..MAX_ROWS, 1..MAX_COLUMNS] of char;

procedure makeBorder (var world: Level);
var
    r : integer;
    c : integer;
begin
    for c := 1 to MAX_COLUMNS do
        world[1][c] := '-';
    for c := 1 to MAX_COLUMNS do
        world[MAX_ROWS][c] := '-';
    for r := 1 to MAX_ROWS do
        world[r][1] := '|';
    for r := 1 to MAX_ROWS do
        world[r][MAX_COLUMNS] := '|';
end;
Example Program: Map Generator And Editor (3)

procedure populate (var world: Level);
var
  r                     : integer;
  c                     : integer;
  randomValue : real;
begin
  for r := 2 to (MAX_ROWS-1) do
  begin
    for c:= 2 to (MAX_COLUMNS-1) do
    begin
      randomValue := random;
      if (randomValue <= 0.05) then
        world [r][c] := '~'
      else if (randomValue <= 0.25) then
        world [r][c] := '^'
      else if (randomValue <= 0.30) then
        world [r][c] := 'C'
      else if (randomValue <= 0.40) then
        world [r][c] := 'T'
      else
        world [r][c] := ' ';
    end; (* inner for *)
  end; (* outer for *)
end; (* procedure populate *)

Example Program: Map Generator And Editor (4)

if (randomValue <= 0.05) then
  world [r][c] := '~'
else if (randomValue <= 0.25) then
  world [r][c] := '^'
else if (randomValue <= 0.30) then
  world [r][c] := 'C'
else if (randomValue <= 0.40) then
  world [r][c] := 'T'
else
  world [r][c] := ' ';
procedure displayWorld (world: Level);
var
  r : integer;
  c: integer;
begin
  (* Display world *)
  for r := 1 to MAX_ROWS do
    begin
      for c := 1 to MAX_COLUMNS do
        begin
          write(world[r][c]);
          end;
        writeln;
    end; (* for loop - displays world *)
end; (* Procedure displayWorld *)

procedure editWorld (var world : Level);
var
  editChoice        : char;
  charToChange : char;
  rowToEdit       : integer;
  columnToEdit  : integer;
begin
  writeln;
  writeln('Enter ''Y'' or ''y'' if you wish to edit the world or the return key otherwise: ');
  readln(editChoice);
  writeln;
  write('Enter "Y" or "y" if you wish to edit the world or the return key otherwise: ');
  readln(editChoice);
Example Program: Map Generator And Editor (7)

if (editChoice = 'Y') OR (editChoice = 'y') then
begin
  writeln;
  write('Enter row (2 - 9) to edit: ');
  readln(rowToEdit);
  write('Enter column (2 - 9) to edit: ');
  readln(columnToEdit);
end; (* if edit mode chosen. *)
end; (* Procedure editWorld *)

Example Program: Map Generator And Editor (8)

if (rowToEdit < 2) OR (rowToEdit > (MAX_ROWS-1)) OR
  (columnToEdit < 2) OR
  (columnToEdit > (MAX_COLUMNS-1)) then
  writeln('Value for row and column must be in the range of 2 - 9')
else
begin
  writeln('What do wish to change this square to? Choices include:');
  writeln('"~" for water');
  writeln('"^" for trees');
  writeln('"C" for a city');
  writeln('"T" for a town');
  writeln('" " (A space) for an open field');
  write('Enter choice and hit return: ');
  readln(charToChange);
  world[rowToEdit][columnToEdit] := charToChange;
end; (* else *)
end; (* Procedure editWorld *)
Example Program: Map Generator And Editor (9)

begin (* Start of main program *)
    var world : Level;
    var quitChoice : char;

    makeBorder(world);
    populate(world);

    (* A loop that displays the world and allows the user to edit it. *)
    repeat
        displayWorld(world);
        editWorld(world);
        write('Type ''Q'' or ''q'' to quit, or return to continue: ');
        readln(quitChoice);
    end; (* repeat loop *)
    until (quitChoice = 'Q') OR (quitChoice = 'q');

end. (* End of main program *)

You Should Now Know

• The number of dimensions that should be set for an array
• How to declare arrays of multiple dimensions
• How to access and assign values to different parts (elements, rows etc.) of multi-dimensional arrays
• How to scan selected parts of the array using loops