Using zero based arrays and lists in Mathematica

Written by Nasser M. Abbasi
Updated October 26,2010

Thanks goes to Sjoerd C. de Vries for letting me know about the Notation package for solving this problem. I did use the notation package before, but only for symbolization. This usage of the Notation is new to me and seems to be very useful to learn about.

This method uses the Notation package to allow one to index an Array, list or a matrix by using a different indexing than the default which starts at 1.

- **The problem**

Sometime there is a need to start the index of an array or a list from zero, as this can be more natural, depending on the problem. To do this in *Mathematica*, all what is needed is to define a notation where the index mapping can occur. An example is shown below to explain how this was done

- **The solution: The Notation package is first loaded**

```
Clear["All`*"]
Needs["Notation`"]
```

- **The array is defined as normal, choosing to use a zero index for the content as this represents better the problem to solve**

```
sizeOfArray = 4;
startingIndex = 0;
u = Array[U[#1] &, sizeOfArray, startingIndex]
```

```
{U[0], U[1], U[2], U[3]}
```

- **A notation is made to access the above array from zero to match the definition of its content. A subscript notation is selected in this example as it was found the easiest and most natural, but other notation could have been used.**

```
Notation[ u_i_ ⇔ u[[i + 1]] ]
```

- **The array u can now be referenced with zero index with the use of u_i instead of the normal u[[i]]**

```
u_0
```

```
U[0]
```

- **The loop now appears more mathematically natural as it starts from zero to length-1**

```
Table[u_i, {i, 0, Length[u] - 1}]
```

```
{U[0], U[1], U[2], U[3]}
```

- **Comparing this to the default case. There is a mismatch in appearance**

```
u[[1]]
```

```
U[0]
```

- **This method can be used with a vector (list) as well**

```
len = 10;
v = Table[i, {i, 0, len - 1}];
Notation[ v_i_ ⟺ v[[i_ + 1]] ]
```

```
v_0
```

```
0
```

```
v_1
```

```
1
```

```
For[i = 0, i < Length[v], i++, Print[v_i]]
```

```
0
1
2
3
4
5
6
7
8
9
```

- **Also for a matrix (table)**

```
(mat = Table[i + j, {i, 0, 2}, {j, 0, 2}]) // MatrixForm
Notation[ mat_i_,j_ ⟺ mat[[i_ + 1, j_ + 1]] ]
```

$$\begin{pmatrix} 0 & 1 & 2 \\ 1 & 2 & 3 \\ 2 & 3 & 4 \end{pmatrix}$$

```
mat_0,0
```

```
0
```

```
{nRows, nCols} = Dimensions[mat];
(Table[mat_{i,j}, {i, 0, nRows - 1}, {j, 0, nCols - 1}]) // MatrixForm
```

$$\begin{pmatrix} 0 & 1 & 2 \\ 1 & 2 & 3 \\ 2 & 3 & 4 \end{pmatrix}$$

- **Even if the list or table is to be accesses using the default indexing (from 1), it might be useful to consider using this notation as it speeds entering the index. Instead of writing v[[i]], one enters v_i which is much faster, and the code appears more clear as well**

```
Notation[ v_i_ ⟺ v[[i_]] ]
v = Table[i, {i, 5}]

(*compare this *)
For[i = 1, i ≤ Length[v], i++, Print[v_i]];

(* to this *)
For[i = 1, i ≤ Length[v], i++, Print[v[[i]]]];
```

- **Less brackets counting and leads to a more clear code.**