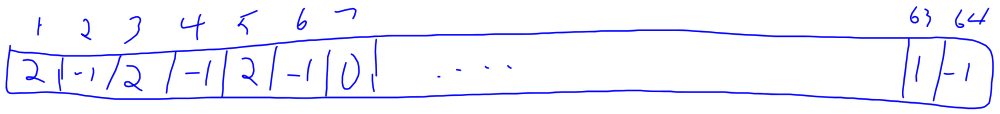
5.3 Multi-Dimensional Arrays

Suppose we wanted to make a 2 player checker game. We could do it with a 1-D array:



To keep track of where the pieces are we could use different int values like this perhaps:

0 - no piece there 3 - black king

1 - black piece 4 - red king

2 - red piece -1 - a square you can't use

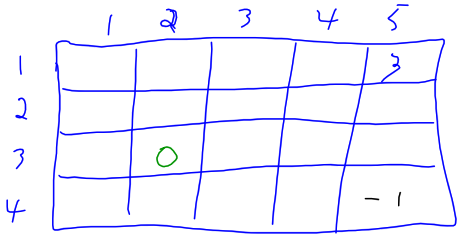
Although we can keep track of the 64 squares of the board, it is hard to visualize the physical board. As we'll see in a moment you can use a 2-D array to more closely reflect the actual board layout.

Let's make a smaller type of checker board, say one with 4 rows and 5 columns.

We could declare it like this: rows columns

```
var board: array 1 .. 4, 1 .. 5 of int
```

We can imagine the array would look like this:




```
board(1,5) := 3
board(3,2) := 0
board(4,5) := -1
board(5,4) := 2
```

this would be illegal since no 5th row

Whenever you want to do something to the whole array you will need a nested for loop (one for the rows, and one for the columns). If you just want to do something to 1 row or 1 column then you just need 1 for loop. Working with just 1 row is like working with a 1-D array like we did in 5.1 and 5.2

Example: To fill the array with zeroes:

```
for row : 1 .. 4
   for col : 1 .. 5
      board(row, col) := 0
   end for
end for
```

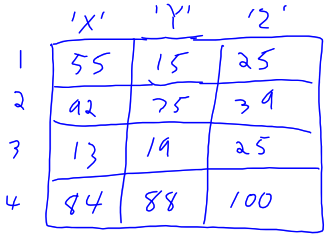
Example: To fill the first row with the number 3:

```
for col: 1 .. 5
   board(1, col) := 3
end for
```

Let's suppose we declared an array like this:

```
var scores : array 1 .. 4, 'X' .. 'Z' of int
```

and filled it with values so it looks like this:



To print all the values in the array showing the actual layout we could do this:

```
for row : 1 .. 4
   for col : 'X' .. 'Z'
      put scores(row, col), " "..
   end for
   put ""
end for
```

Output

```
55 15 25
92 75 39
13 19 25
84 88 100
```

To add up all the numbers in the third row we could do this:

```
var total : int := 0
for col : 'X' .. 'Z'
   total := total + score(3, col)
end for
put "Sum of third row is ", total
```

Sum of third row is 57

To add up all the numbers in column 'Z' we could do this:

```
total := 0
for row : 1 .. 4
   total := total + scores(row, 'Z')
end for
put "Sum of column 'Z' is ", total
```

Sum of column 'Z' is 189

We can have more dimensions than just 2. It is rather rare to have more than 3-D. 1-D and 2-D are most common.

An example of a 3-D array:

```
var names: 1 .. 5, 1 .. 10, 1 .. 3 of string
```

In this case you could visualize it as a 5 x 10 grid (5 rows and 10 columns) duplicated on 3 pieces of paper (or 3 layers). The total number of elements you could store would be:

5 x 10 x 3 = 150

Read section 5.3 in the textbook. Do exercise 5.3 #1-4 (I'll post solutions on Wednesday)

Look for 5.4 on Wednesday.

This section(5.3) is really more enrichment than a core part of the course. The concepts are useful especially if you are going to take ICS 3U. If you're finding this section difficult, don't hesitate to ask for clarification, but in the end don't be too concerned as you will not be tested or have any assignment on it.