CS100J 27 October 2005

Rectangular arrays. Secs. 9.1 and 9.2

Gries is in town. But he is talking about undergraduate education with the Engineering College Council, a group of people who advise Dean Kent Fuchs.

Do as many of the exercises on pp. 311-312 as you can to get familiar with concepts and develop a skill. Practice in DrJava! Test your methods, both by hand and on computer!

Quotes that relate to specifying a method before writing it.

A verbal contract isn't worth the paper it's written on.

What is not on paper has not been said.

If you don't know where you are going, any road will take you there.

If you fail to plan you are planning to fail.

	0	1	2	3	b.length
b	5	4	7	3	

one-dimensional array

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

rectangular array: 5 rows and 4 columns

Type of d is **int**[][] ("**int** array array", "an array of **int** arrays")

To declare variable d:

int d[][].

To create a new array and assign it to d:

d= **new int**[5][4];

To reference element at row r column c:

d[r][c]

number of rows

number of cols

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

Type of d is **int**[][] ("**int** array array", "an array of **int** arrays")

To declare variable d:

int d[][].

To create a new array and assign it to d:

d= **new int**[5][4];

To reference element at row r column c:

d[r][c]

number of rows

number of cols

Number of rows: d.length

Number of columns in row r: d[r].length

"Length of one array in array of arrays"

Using an array initializer:

int[][] d= **new int**[][]{ {5,4,7,3}, {4,8,9,7}, {5,1,2,3}, {4,1,2,9}, {6,7,8,0} };


```
/** = sum of first elements of rows of d. e.g. for array to
      right, it's 5 + 4 + 5 + 4 + 6. */
public static int sum0(int[][] d) {
    int x= 0;
    // inv: x = sum of first element of rows d[0..r–1]
    for (int r= 0;  r != d.length;  r= r+1) {
        x= x + d[r][0];
    }
    // x = sum of first element of rows d[0..d.length–1]
    return x;

}
```

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

Pattern for processing all the elements of an array

Row-major order (first row 1, then row 2, etc.)

```
// Process elements of b[][] in row-major order
// inv: rows 0..r-1 have been processed.
//      In row r, b[r, 0..c-1] have been processed
  for (int r= 0; r != b.length; r= r + 1)
     for (int c= 0; c != b[r].length; c= c+1) }
         Process b[r][c]
}
```

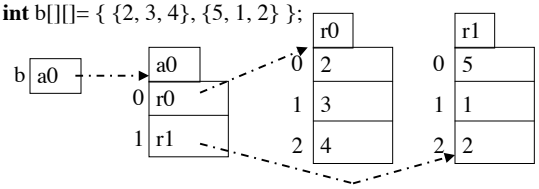


```
/** = a String rep of b[][] (as in an array initializer) */
public static String toString(int b[][]) {
   int s= "{"
   // inv: Rows 0..r–1 have been added to s */
   for (int r= 0; r != b.length; r= r + 1) {
      // Add row r to s
          s= s + "{";
          // inv: the partial row b[r][0..c–1] has been added to s
          for (int c= 0; c != b[r].length; c= c + 1} {
              if (c != 0)  s= s + ", ";
              s= s + b[r][c];

          }
          s= s + "}";
   }
   return s + "}";
}
```

How multi-dimensional arrays are stored: ragged arrays

int b[][]= { {2, 3, 4}, {5, 1, 2} };



b is a one-dimensional array of b.length elements

Its elements are one-dimensional arrays.

b[0] is a one-dimensional array of **int**s of length b[0].length.
Must all these arrays have the same length? No!

How multi-dimensional arrays are stored: ragged arrays

int[][] b; Declare variable b of type **int** [][]

b= **new int**[2][] Create a one-dim. Array of length 2 and store its name in b. Its elements are **null**, have type **int**[]

b[0]= **new int**[] {2, 3, 4}; Create **int** array, store its name in b[0].

b[1]= **new int**[] {5, 6}; Create **int** array, store its name in b[1].