

Matrices

Matrices are two-dimensional arrays. Since we are not using the C-style unsafe arrays we are going to use a class called *apmatrix* to implement are two-dimensional arrays.

In a matrix there are rows and columns. The first index indicates the number of rows, the second column indicates the number of columns. For example:

```
#include "apmatrix.h"

apmatrix<int> m(2,5);
```

To fill the matrix with values:

```
for (int r = 0; r < m.numrows(); r++)
    for (int c = 0; c < m.numcols(); c++)
        m[r][c] = c;
```

This creates a matrix with two rows and five columns. Just like in vectors, the first row is called row 0, the second row is row 1, etc., and the same goes for the column indices. Notice the *apmatrix* member functions that return the number of rows (*numrows*) and columns (*numcols*) in the matrix.

To display the matrix so that it appears on the screen as

```
1 2 3 4 5
1 2 3 4 5

for (int r = 0; r < m.numrows(); r++)
{
    for (int c = 0; c < m.numcols(); c++)
        cout << setw(4) << setiosflags(ios::right) << m[r][c];
    cout << endl;
}
```

The first subscript always indicates the row location and the second subscript indicates the column location.

You can also resize the matrix

```
m.resize(5, 7);
```

The matrix now has 5 rows and 7 columns. If you resize your matrix smaller, values be will lost.

We can also initialize the matrix when we declare it. The initialized value will be placed in every matrix location.

```
apmatrix<int> m(2, 3, 8);
```

This matrix with two rows and three columns will have the value 8 in all 6 matrix locations.