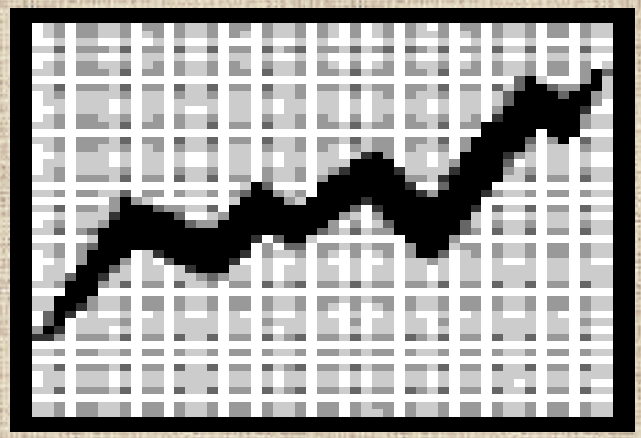
Chapter 8: Matrices
The AP Matrix Class



Mr. Dave Clausen

La Cañada High School

Introduction To Two Dimensional Arrays

- Two Dimensional Arrays work well with data that is represented in tabular form (table).
  - Some examples include:
    - Baseball statistics for an entire team.
    - A teacher's grade book for a class of students.
- A two dimensional array is an array where each data item is accessed by specifying a pair of indices.
  - The first index refers to the Row, while the second index refers to the Column. (You can remember this by thinking RC Cola: Row before Column)

Matrices

- A vector is a one-dimensional array
- A matrix is a two-dimensional array
- A cell item in a matrix is specified with two index values, a row and a column

Declaring an APMatrix data type

- APMatrix is a “safe” two dimensional array.
- Declare by listing: apmatrix, the type of data you want for your basetype, the matrix name, and two indices (rows, columns).
- #include “apmatrix.h”; //for new compilers & .cpp for old
- General Form for declaring an apmatrix.

apmatrix <item type> <matrix name> (rows, columns);

apmatrix <item type> <matrix name> (rows, columns, initial_value);

- For example:
  - apmatrix <int> table (num_rows, num_cols);
  - apmatrix <int> table2 (num_rows, num_cols, initial_value);

Declaring a Matrix Variable

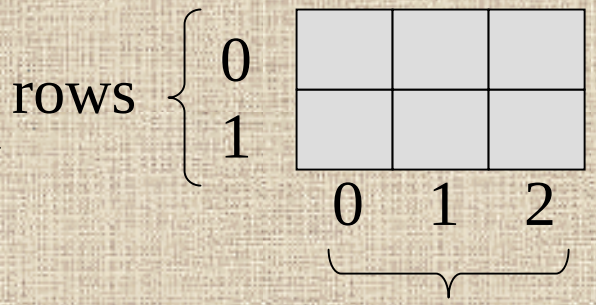
```
#include "apmatrix.h"
int main ()
{
apmatrix<int> numbers(2, 3);
```

number of columns

number of rows

matrix type item type matrix variable name

Item type can be any data type.
A matrix has no cells for items by default
Data items is cells are undefined



Visualizing the APMatrix: Indices

- In the apmatrix we just declared "table", we can visualize the indices of the matrix as follows: (Remember that matrix indices begin at zero, and they appear in ROW then COLUMN order.)

Columns

table

Rows				
	[0] [0]	**[0] [1]**	**[0] [2]**	**[0] [3]**
	[1] [0]	**[1] [1]**	**[1] [2]**	**[1] [3]**
	[2] [0]	**[2] [1]**	**[2] [2]**	**[2] [3]**

Visualizing the apmatrix: elements

- In the matrix we just declared: “table”, remember that the contents of each element is in an unpredictable state. (I will signify this with question marks.) The matrix will need to be initialized…

table

?	?	?	?	?
?	?	?	?	?
?	?	?	?	?

Initializing the apmatrix

- We can initialize this apmatrix when we declare it:

```
cout<<“Enter the number of rows: “;
cin>>num_rows;
cout<<“Enter the number of columns: “;
cin>>num_columns;
cout<<“Enter the initializing value: “;
cin>>initial_value;
apmatrix <int> table (num_rows, num_columns, initial_value);
```

table

-999	**-999**	**-999**	**-999**	**-999**
-999	**-999**	**-999**	**-999**	**-999**
-999	**-999**	**-999**	**-999**	**-999**

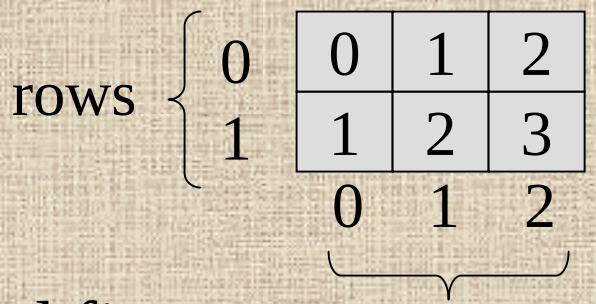
Operations on Data In An apmatrix

- Nested definite or indefinite loops are used to perform many operations with a matrix, including:
  - Entering the data
  - Printing the data
  - Searching the matrix for a key value (or max or min)
  - Sorting the data in the matrix
  - Performing calculations on the values in the matrix
    - calculations in a row or
    - calculations in a column
  - Matrix operations (Mathematical Operations):
    - symmetry, transposition, determinant, etc...

Using Nested Loops

```
#include "apmatrix.h"
int main ()
{
apmatrix<int> numbers(2, 3);

for (int row = 0; row < numbers.numrows(); ++row)
   for (int col = 0; row < numbers.numcols(); ++col)
      numbers[row][col] = row + col;
```



The row index is always specified first.

Entering Data into an apmatrix

- To simplify, we will use nested “for” loops instead of while loops with sentinels.

```
for(int row = 0; row < table.numrows(); ++row)
  for(int col = 0; col < table.numcols(); ++col)
  {
    cout<<“Enter score for row: <<row<<“ and column “<<col;
    cin>>table [row] [col];
  }
```

table

100	**80**	**99**	**92**	**88**
90	**70**	**85**	**78**	**77**
60	**45**	**67**	**98**	**65**

Displaying Data from an apmatrix

- A header for the data would be a good idea if you plan on displaying it in a table format, for now we will display the data in a list format…

```
for(int row = 0; row < table.numrows(); ++row)
  for(int col = 0; col < table.numcols(); ++col)
  {
    cout<<“Score for row: <<row<<“ and column “<<col<<“ “;
    cout<<table [row] [col]<<endl;
  }
```

Copying an apmatrix

- To copy an apmatrix:
  - The apmatrix class will resize the target matrix before it copies it.

```
matrix2 = matrix1;
```

Sum and Average of a Row

- To find the sum and average of one row (row index # 3 in this case) in a matrix:

```
row_sum = 0;
row_average = 0.0;
for(int col = 0; col < table.numcols( ); ++col)
    row_sum = row_sum + table[3] [col]; //row index is constant
row_average = double (row_sum) / double (table.numcols());
cout<<“Row sum: “<<row_sum<<endl;
cout<<“Row average: “<<row_average<<endl;
```

Sum and Average of a Column

- To find the sum and average of one column (column index # 3 in this case) in a matrix:

```
column_sum = 0;
column_average = 0.0;
for(int row = 0; row < table.numrows(); ++row)
    column _sum = column _sum + table[row] [3]; // column index
// is constant
column _average = double(column _sum) / double(table.numrows());
cout<<“Column sum: “<< column _sum<<endl;
cout<<“Column average: “<< column _average<<endl;
```

apmatrix functions: Resize

- Like vectors, a matrix can be resized.
  - This ability was designed into the apmatrix class.
  - Since this is a member function of the apmatrix class, we use the dot notation:

For Example:

table.resize (new_num_rows, new_num_columns);

apmatrix functions: numrows & numcols

- To determine the current size of the matrix, we can use two functions
  - numrows to determine the current number of rows in the matrix
  - numcols to determine the current number of columns in the matrix

```
num_rows = table.numrows( );
num_cols = table.numcols( );
```

Using apmatrix with Functions

- Pass matrices using reference parameters
  - Remember, when you pass a standard C++ array, it is always passed as a reference parameter (without you designating it as such with the **&** symbol).
  - With apmatrix, use the & symbol to pass the matrix as a reference parameter.

```
void display_table (apmatrix<int> &matrix);
```

  - This is very similar to passing a vector to a function using the apvector class.

MatrixDriver.cpp **MatrixDriver.txt**

Parallel Arrays: matrices & vectors

- Remember that arrays, whether one dimensional (vectors) or two-dimensional (matrices) can contain data of any type as long as every element of the array is of the same data type, called the array's basetype.
- Some situations would require more than one array to represent the data.
  - Consider the representation of Baseball statistics.
    - We need to represent the players names (apstring)
    - and the numeric data (integers or double).

Parallel Arrays: matrices & vectors

- In the data representation of the Baseball statistics:
  - the rows would represent all the information about one player. This is called a record in data base terminology.
  - the columns would represent a category of information. This is called a field in data base terminology.
    - The columns might represent:
      - The number of times a player was “at bat” (AB)
      - The number of hits a player had (HITS)
      - The players batting average(AVE), etc.

Visualizing Parallel Arrays

- We can visualize the Baseball Statistics Data as follows:
  - Note that the rows represent all the information about one player
  - Also Note that each column represents the same type of data.

Player		**AB**	**HITS**	**AVE**
Alomar		521	192	0.369
Brett		493	180	0.365
Canseco		451	160	0.355
Dribble		590	205	0.347
Hubble		501	167	0.333
Marachino		485	156	0.322
Noguchi		562	180	0.320

Parallel Arrays vs. Structs

- Before you get carried away with parallel arrays, C++ has a better way of dealing with "data base" type of situations.
  - Tune in to Chapter 9, and the lecture on Structs.
  - A struct is a way of representing all the information in a "record" (the rows in our example).
  - A vector of structs would be declared to represent the data in our previous Baseball statistics example. This is a much better way of representing the data.

Let’s look at the assigned program:

Prog10C.exe