Programming with Two-dimensional Arrays

Larry Caretto
Computer Science 106

Computing in Engineering and Science

May 9, 2006

Outline

- Review basics of 2D arrays
  - Contrast with 1D arrays
  - Notation and declaring array sizes
- Code applications with 2D arrays
- Passing 2D arrays to functions
- Higher-dimensional arrays
- Summary of array use

Two-dimensional Arrays

- One-dimensional arrays refer to a variable that has multiple entries with a single classification
- Two-dimensional arrays are used to represent data with two classifications
  - Example: an experiment on manufacturing productivity measures daily output of four machines with six operators

Two-dimensional Arrays

- One-dimensional variable
  - mathematical notation x_i
  - C++ array notation x[i]
- Two-dimensional
  - mathematical notation x_{ik}
  - C++ array notation x[i][k]
- One-way versus two-way classification

Two-Dimensional Array

[0][0]	[0][1]	[0][2]	[0][3]	[0][4]
[1][0]	[1][1]	[1][2]	[1][3]	[1][4]
[2][0]	[2][1]	[2][2]	[2][3]	[2][4]
[3][0]	[3][1]	[3][2]	[3][3]	[3][4]
[4][0]	[4][1]	[4][2]	[4][3]	[4][4]
[5][0]	[5][1]	[5][2]	[5][3]	[5][4]
[6][0]	[6][1]	[6][2]	[6][3]	[6][4]

- View two-dimensional arrays as a table with rows and columns of cells
- Row index
- Column

Two-dimensional Example

- In the example of a manufacturing process measuring the output of four machines with six operators
  - Array named output depending on integer subscripts machine and operator
  - First subscript is for operator and second is for machine

```
const int maxOp = 6, maxMach = 4;
int output[maxOp][maxMach];
cout << output[3][2];
```

Two-Dimensional Array Data

	M 0	M 1	M 2	M 3	Op tot
Op 0	34	53	43	31	161
Op 1	39	55	42	36	172
Op 2	33	52	45	40	170
Op 3	31	48	39	25	143
Op 4	38	59	48	42	187
Op 5	33	49	48	28	158
M tot	208	316	265	202	991

output[3][2]

Individual data plus totals for operators and machines

Array data

Two-dimensional array Code

```
const int maxOp = 6, maxMach = 4
int output[maxOp][maxMach];
for (int op = 0; op < maxOp; op++)
{
     for ( int mach = 0; mach <
               maxMach; mach++ )
        cout << output[op][mach] <<
       " units produced at machine "
       << mach << " with operator "
       << op;
}
```


Other Code

- How would you compute the total units produced by each machine?
- How would you compute the total units produced by each operator?
- How would you compute the average and standard deviation for all the units produced by the operators?

Units for Each Machine

- This sum is the total output of each machine from all operators (column sum)

```
int outMach[maxMach];
for (int mac = 0; mac < maxMach; mac++)
{
    outMach[mac] = 0;
    for (int op = 0; op < maxOp; op++)
    {outMach[mac] += output[op][mac];}
    cout << "Total machine " << mac <<
        << " output is "<<outMach[mac];
}
```


Units for Each Operator

- This sum is the total output of each operator from all machines (row sum)

```
int outOp[maxOp];
for ( int op = 0; op < maxOp; op++ )
{
   outOp[op] = 0;
   for ( int m = 0; m < maxMach; m++ )
    { outOp[op] += output[op][m]; }
   cout << "Total operator " << op
        << " output is " << outOp[op];
}
```


Comments on this Code

- Note that we use one-dimensional arrays to store row (operator) and column (machine) sums
- Note that order of subscripts is always [operator][machine]
- Conventional, but not required, to write tables as arrays with subscript ordered as [row][column]

Simultaneous Linear Equations

- Example of 3 equations (3 unknowns)
  - $3x + 7y - 3z = 8$
  - $2x - 4y + z = -3$
  - $8x + 6y - 2z = 14$
- How can we develop a general notation for N equations in N unknowns?
  - Call variables x_0, x_1, x_2, etc.
  - Call right hand side b_0, b_1, b_2, etc.
  - Call top row coefficients a_{00}, a_{01}, a_{02}, etc.

Standard Form

$$a_{00}x_0 + a_{01}x_1 + a_{02}x_2 + \ldots + a_{0N-1}x_{N-1} + a_{0N}x_N = b_0$$
$$a_{10}x_0 + a_{11}x_1 + a_{12}x_2 + \ldots + a_{1N-1}x_{N-1} + a_{1N}x_N = b_1$$
$$a_{20}x_0 + a_{21}x_1 + a_{22}x_2 + \ldots + a_{2N-1}x_{N-1} + a_{2N}x_N = b_2$$
$$\ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots$$
$$a_{N-1,0}x_0 + a_{N-1,1}x_1 + \ldots \ldots \ldots \ldots \ldots + a_{N-1,N}x_N = b_{N-1}$$
$$a_{N0}x_0 + a_{N1}x_1 + a_{N2}x_2 + \ldots \ldots \ldots \ldots \ldots + a_{NN}x_N = b_N$$

- Note that subscripts on a are $a_{row,column}$ where row is equation and column is unknown

Compact Standard Form

$$\sum_{j=1}^{N} a_{ij}x_j = b_i \qquad i = 1, \ldots, N$$
$$\sum_{j=0}^{N-1} a_{ij}x_j = b_i \qquad i = 0, \ldots, N-1$$

- Set of equations defined by N and data on a_{ij} and b_i
- Functions to solve this problem use a two-dimensional a[i][j] array and one-dimensional arrays for b[i] and x[j]

Example in Standard Form

- Previous example 3 equations (N = 3)
  - $3x + 7y - 3z = 8$
  - $2x - 4y + z = -3$
  - $8x + 6y - 2z = 14$
- In standard form:
  - x is x_0, y is x_1, and z is x_2
  - $a_{00} = 3$, $a_{01} = 7$, $a_{02} = -3$, b_0, = 8
  - $a_{10} = 2$, $a_{11} = -4$, $a_{12} = 1$, b_1, = -3
  - $a_{20} = 8$, $a_{21} = 6$, $a_{22} = -2$, b_2, = 14

Standard Form in C++

- Equations represent unknowns as x_i, the right hand sides as b_i, and the left hand side coefficients as a_{ij}
- In C++ we use arrays x[col] for the unknowns, b[row] for the right hand sides, and a[row][col] for the coefficients on the left hand side
- Project three will use library program to solve this system of equations

Passing 2D Arrays to Functions

- Execution of array code based on computing memory location from address of first array member plus subscript for particular element
- For one-dimensional array we only need the address of the first element to find the location of x[i]
- What about two-dimensional arrays?

Passing 2D Arrays to Functions II

- Consider an array x with declared as x[totalFirst][totalSecond]
- The location of x[i][j] is computed as i + j*totalSecond locations from the start of the array
- We must know the second dimension to compute the location
- We must pass this second dimension to the function that has a two-dimensional array as a parameter

California State University Northridge 19

Passing 2D Arrays to Functions III

- Global constant: `const int maxSecond = 20`
- Function header
  `double getSum ( double x[][maxSecond],…`
- Function prototype (semicolon at end)
  `double getSum ( double x[][maxSecond],…`
  `double getSum ( double [][maxSecond],…`
- Calling program

```
const int maxFirst = 20;
double x[maxFirst][maxSecond];
// other code assigns values to x array
double result = getSum( x, ….
```

California State University Northridge 20

Passing 2D Arrays to Functions IV

- Global constant not required, but helpful to accommodate changes to size of second dimension
- The second dimension must be the same in the following three statements:
  - The function prototype
  - The function header
  - The declaration of the array passed to the function
- Final project uses two-dimensional arrays

California State University Northridge 21

Passing 2D Arrays to Functions V

- Example: write a function that accepts a two-dimensional array, output, used in the previous example and computes and returns the row sums and columns sums as well as the total
- How to pass information?
  - Pass 2D output array into function
  - Return 1D arrays with row and column sums
  - Return total in function name
  - Pass number of machines and operators, which can be less than the maximum array sizes, into function

California State University Northridge 22

Example of 2D Array Function

```
int getSums( int output[][maxMach],
             int opSum[], int machSum[],
             int Nop, int Nmach)
{
   int total = 0;
   for ( int op = 0; op < Nop; op++ )
   {
      opSum[op] = 0;
      for ( int m = 0; m < Nmach; m++ )
         opSum[op] += output[op][m];

      total += opSum[op];
   }
            // continues on next chart
```

California State University Northridge 23

2D Array Function Concluded

```
   for ( int m = 0; m < Nmach; m++ )
   {
      machSum[m] = 0;
      for ( int op = 0; op < Nop; op++ )
         machSum[op] += output[op][m];
   }
   return total;
}  // closes function opening brace
```

- How do we use this function?
- What is its prototype?

California State University Northridge 24

Using the 2D Array Function

- Start with global constants for common array dimensions in various locations

```
const int maxMach = 10, maxOp = 10;
```

- Prototype is just header with a semicolon

```
int getSums( int output[][maxMach],
             int opSum[], int machSum[],
             int Nop, int Nmach);
```

- Use global constants as array dimensions in calling program

```
int output[maxOp][maxMach],
  opSum[maxOp], machSum[maxMach];
```


Using the 2D Array Function

- Get data in calling program (usually from file)

```
ifstream inFile( "production.dat" );
inFile >> Nop >> Nmach;
for (op = 0; op < Nop; op++ )
    for ( m = 0; m < Nmach; m++ )
        infile >> output[op][m];
```

- Call function

```
int total = getSums( output, opSum,
 machSum, Nop, Nmach);
```

- Output results

Array call has only array names

Input Data Files for Arrays

- Must match input statements in code

```
for (i = 0; i < N; i++) inFil >> x[i];
for (i = 0; i < N; i++) inFil >> y[i];
```

- Compare above statements with code below

```
for (i = 0; i < N; i++)
{  inFil >> x[i] >> y[i]; }
```

- First example read all x data then all y data. Second reads x and y data in pairs
- Usually write code to determine number of array elements by testing for end of file

Input Data File for 1D Arrays

```
12
20
32
55
43
19
27
88
```

```
12 20 32 55 43 19 27 88
```

```
12 20
32 55
43 19
27 88
```

- How does the code below read x and y from each file on this page?

```
for (i = 0; i < 3; i++)
  cin >> x[i] >> y[i];
```

- What about this code?

```
for (i = 0; i < 3; i++)
  cin >> x[i];
for (i = 0; i < 3; i++)
  cin >> y[i];
```


Input Data Files for 2D Arrays

- Recall input code from example of passing 2D arrays to functions

```
ifstream inFile("production.dat");
inFile >> Nop >> Nmach;
for (op = 0; op < Nop; op++ )
{   for ( m = 0; m < Nmach; m++ )
    {   inFile >> output[op][m]; }
}
```

- How would you prepare the data file?

Braces not needed

Input Data File for 2D Arrays

- Usually prepare data file for 2D arrays to look like row and column data

```
6    4
34   53   43   31
39   55   42   36
33   52   45   40
31   48   39   25
38   59   48   42
33   49   48   28
```

Is There Life After 2D Arrays

- Yes, we can have arrays with three or more dimensions
- A program to compute emissions of different species, different vehicle types, different model years could use

emissions[species][vehType][modelYear]

- Code structures are similar with use of nested for loops on array subscripts
- Will not cover in this course

Summary of Arrays

- Used to represent data of one kind with multiple occurrences
- Can have one-way, two-way, etc., classifications of the data
- Math symbols a_{ij} and x_j become C++ arrays a[i][j] and x[i]
- Declaring array size; maximum subscript; no subscript checking

Array Summary Continued

- Use for loops where loop index is array subscript to access array elements
- Array elements like ordinary variables
- Passing whole arrays to functions (header, prototype, call, 1D vs. 2D)
- Nested loops for 2D array code
- Input files for arrays must match input statements

Array Quiz Thursday

- Program to do calculations with arrays
  - Will have simple operations on one, two, or three one-dimensional arrays
- Use of functions in code you write will give extra credit
- Use output routine from exercise eight, task one
  - Be prepared to modify it to read one or three arrays

Assignments

- Reading pages in text
  - Today: Pages 447–454
  - Thursday: Pages 625–645
  - Tuesday, May 9: Pages 775–799
- This week's homework problems
  - Page 474, program 5
- Exercise eight due this Thursday
- Lab quiz on exercise eight on Thursday, May 11

Sample Quiz

- Read one-dimensional array data on second-by-second velocity, v_i, in mph
  - *I.e.*, v_0 is velocity at t = 0 s, v_1 is velocity at t = 1 s, v_2 is velocity at t = 2 s, *etc*.
- Compute acceleration, a_i, (mph/s) for each i by the following equations (Δt = 1 s)
  - General formula for $1 \le i \le N - 2$ and special formulas for a_0 and a_{N-1}

$$a_0 = \frac{-v_2 + 4v_1 - 3v_0}{2\Delta t} \qquad a_i = \frac{v_{i+1} - v_{i-1}}{2\Delta t} \qquad a_{N-1} = \frac{v_{N-3} - 4v_{N-2} + 3v_{N-1}}{2\Delta t}$$