Exercises in Array Programming

Larry Caretto
Computer Science 106
Computing in Engineering and Science
April 25, 2006

Outline

• Review introduction to arrays
• Review writing code with arrays and for loops
  – Array sums
  – Finding maximum and minimum elements in an array
  – Data processing with arrays
• Exercises in array programming

Representing Data

<table>
<thead>
<tr>
<th>Run</th>
<th>Data</th>
<th>Math</th>
<th>C++</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>12.3</td>
<td>x₀</td>
<td>x[0]</td>
</tr>
<tr>
<td>1</td>
<td>14.4</td>
<td>x₁</td>
<td>x[1]</td>
</tr>
<tr>
<td>2</td>
<td>11.8</td>
<td>x₂</td>
<td>x[2]</td>
</tr>
<tr>
<td>3</td>
<td>12.5</td>
<td>x₃</td>
<td>x[3]</td>
</tr>
<tr>
<td>4</td>
<td>13.2</td>
<td>x₄</td>
<td>x[4]</td>
</tr>
<tr>
<td>5</td>
<td>14.1</td>
<td>x₅</td>
<td>x[5]</td>
</tr>
</tbody>
</table>

• C++ array, x[i] for math xᵢ
• Maximum subscript = number of elements minus one

General Array Processing

• To process each element in an array with N elements, starting with the initial element, use a for loop with index k starting at zero and < N
  
  for ( int k = 0; k < N; k++)

• To process a subset of elements in the array starting at element F and ending with (and including) element L
  
  for ( int k = F; k <= L; k++)

File Input Screen Output

```cpp
const int MAX_SIZE = 100;
double z[MAX_SIZE];
ifstream inFile( "array.dat" );
for(int i = 0; i < MAX_SIZE; i++)
{
  inFile >> z[i];
  cout << "z[" << i << "] = " << z[i];
}
```

Console Input File Output

```cpp
const int MAX_SIZE = 100;
double z[MAX_SIZE];
ofstream outFile( "array.dat" );
for( int i = 0; i < MAX_SIZE; i++ )
{
  cout << "Enter z[" << i << "]: ";
  cin >> z[i];
  outFile << z[i] << endl;
}
```
**Defined Elements**

- The number of elements defined may be less than the array size
- You may declare an array to be the maximum size expected but actually specify a value for fewer elements

```c
double x[10];
for (int j = 1; j < 5; j++)
    x[j] = 1 / double(j);
```

**Computing the Mean**

\[
x = \frac{1}{N} \sum_{i=1}^{N} x_i = \frac{1}{N-1} \sum_{i=0}^{N-1} x_i
\]

- N data items to average
- Subscripts start at 0
- Last data item is element N-1 in array
- {} not needed

```c
sum = 0;
for (i = 0; i < N; i++)
    { sum += x[i]; }
average = sum / N;
```

**Finding the Maximum**

- Store the current maximum in a variable (e.g., `max`) and compare `max` to new array values
- If a new value is greater than `max` replace `max` by that value

```c
double max = x[0]; // initialize max
for (int i = 1; i < N; i++)
    { if (x[i] > max)
        { max = x[i]; }
    }
```

- Can omit both sets of braces

**Finding the Minimum**

- Store the current minimum in a variable (e.g., `min`) and compare `min` to new array values
- If a new value is greater than `min` replace `min` by that value

```c
double min = x[0]; // initialize min
for (int i = 1; i < N; i++)
    { if (x[i] < min)
        { min = x[i]; }
    }
```

- Can omit both sets of braces

**Array Processing Example**

- You have taken current and voltage data from a circuit
- There are N pairs of data
- Current is stored as the `amps[k]` array and voltage as the `volts[k]` array
- Write the code to compute the average power if arrays are already declared and data on N, `volts[]` and `amps[]` are already input

```c
double sum = 0;
for (int k = 0; k < N; k++)
    { power[k] = amps[k] * volts[k];
    sum += power[k];
    }
double averagePower = sum / N;
cout << "Power = " << averagePower << " watts";
```

*The power array could be used in subsequent (or the same) loops*
Average Power Two

```cpp
double sum = 0;
for (int k = 0; k < N; k++) {
    power = amps[k] * volts[k];
    sum += power;
}
double averagePower = sum / N;
cout << "Power = " << averagePower << " watts";
```

The power variable can be used in the same loop only.

Average Power Three

```cpp
double sum = 0;
for (int k = 0; k < N; k++) {
    sum += amps[k] * volts[k];
}
double averagePower = sum / N;
cout << "Power = " << averagePower << " watts";
```

The power calculation has to be redone if we want to use power[k] subsequently.

Standard Deviation Exercise

• Measure of spread around mean

\[ s = \sqrt{\frac{\sum (x_i - \bar{x})^2}{N-1}} = \sqrt{\frac{\sum x_i^2 - N \bar{x}^2}{N-1}} \]

• Use final equation to write code to compute standard deviation

• Hint: Compute two sums in a single for loop: \( \sum x_i \) and \( \sum x_i^2 \)

Standard Deviation Code

```cpp
double sum = 0, sum2 = 0;
for (int k = 0; k < N; k++) {
    sum += x[k];
    sum2 += x[k] * x[k];
}
return sqrt((sum2 - sum * sum) / N / (N - 1));
```

Calculating Tables with Arrays

• We have previously calculated tables where the step between variables was the same

• How can we construct a table with uneven step sizes in the variables?
  – We can define one or two arrays for a one-way or two-way table
  – Each array has all the values that we want in the independent value of the table

Calculating One-Way Tables

<table>
<thead>
<tr>
<th>x</th>
<th>f(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>25</td>
</tr>
</tbody>
</table>

Even increments in x; no array required

<table>
<thead>
<tr>
<th>x</th>
<th>f(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>20</td>
</tr>
</tbody>
</table>

Uneven increments in x; array required for x

<table>
<thead>
<tr>
<th>x</th>
<th>f(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>7</td>
<td>49</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>20</td>
<td>400</td>
</tr>
</tbody>
</table>
Two-way, Both Fixed Increment

<table>
<thead>
<tr>
<th>m = 1</th>
<th>m = 2</th>
<th>m = 3</th>
<th>m = 4</th>
<th>m = 5</th>
<th>m = 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>V = 1</td>
<td>0.5</td>
<td>1.0</td>
<td>1.5</td>
<td>2.0</td>
<td>2.5</td>
</tr>
<tr>
<td>V = 2</td>
<td>2.0</td>
<td>4.0</td>
<td>6.0</td>
<td>8.0</td>
<td>10.0</td>
</tr>
<tr>
<td>V = 3</td>
<td>4.5</td>
<td>9.0</td>
<td>13.5</td>
<td>18.0</td>
<td>22.5</td>
</tr>
</tbody>
</table>

• When both independent variables have fixed increments, no arrays are required
  – Exercise six and project two code

Even Increments In Both

• KE with even step sizes in both mass and velocity

```cpp
for ( int v = 1; v <= 3; v++ ) {
    cout << setprecision(0)
    << "\nv = " << v
    << setprecision(1);
    for ( int m = 1; m < 7; m++ ) {
        cout << setw(7)
        << 0.5 * m * v * v;
    }
}
```

Two-way, Variable Columns

<table>
<thead>
<tr>
<th>m = 1</th>
<th>m = 2</th>
<th>m = 5</th>
<th>m = 10</th>
<th>m = 20</th>
</tr>
</thead>
<tbody>
<tr>
<td>V = 1</td>
<td>0.5</td>
<td>1.0</td>
<td>2.5</td>
<td>5.0</td>
</tr>
<tr>
<td>V = 2</td>
<td>2.0</td>
<td>4.0</td>
<td>10.0</td>
<td>20.0</td>
</tr>
<tr>
<td>V = 3</td>
<td>4.5</td>
<td>9.0</td>
<td>22.5</td>
<td>45.0</td>
</tr>
</tbody>
</table>

This requires an array for the column variable (mass), but not for the row variable (velocity)

Variable Column Increments

```cpp
const int NM = 5;
int m[NM] = { 1, 2, 5, 10, 20 };
for ( int v = 1; v <= 3; v++ ) {
    cout << setprecision(0)
    << "\nv = " << v
    << setprecision(1);
    for ( int j = 1; j < NM; j++ )
    cout << setw(7)
    << 0.5 * m[j] * v * v;
}
```

Two-way, Variable Row Increment

<table>
<thead>
<tr>
<th>m = 1</th>
<th>m = 2</th>
<th>m = 3</th>
<th>m = 4</th>
<th>m = 5</th>
<th>m = 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>V = 1</td>
<td>0.5</td>
<td>1.0</td>
<td>1.5</td>
<td>2.0</td>
<td>2.5</td>
</tr>
<tr>
<td>V = 2</td>
<td>2.0</td>
<td>4.0</td>
<td>6.0</td>
<td>8.0</td>
<td>10.0</td>
</tr>
<tr>
<td>V = 4</td>
<td>8.0</td>
<td>16.0</td>
<td>24.0</td>
<td>18.0</td>
<td>22.5</td>
</tr>
</tbody>
</table>

• This requires an array for the row variable (velocity), but not for the column variable (mass)

Variable Row Increments

```cpp
const int NV = 3;
int v[NV] = { 1, 2, 4 };
for ( int i = 1; i < NV; i++ ) {
    cout << setprecision(0)
    << "\nv = " << v[i]
    << setprecision(1);
    for ( int j = 1; j < NV; j++ )
    cout << setw(7)
    << 0.5 * m[j] * v[i] * v[i];
}
```
**Two-way, Both Increments Vary**

<table>
<thead>
<tr>
<th>m</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>10</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V</td>
<td>4</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• This requires arrays for both the row variable (velocity in m/s) and the column variable (mass)

**Both Increments Variable**

```cpp
const int NM = 5, NV = 3;
int m[NM] = {1, 2, 5, 10, 20};
int v[NV] = {1, 2, 4};
for (int i = 1; i < NV; i++) {
    cout << setprecision(0) << "\nv = " << v[i] << setprecision(1);
    for (int j = 1; j < NM; j++)
        cout << setw(7) << 0.5 * m[j] * v[i] * v[i];
}
```

**Exercise**

• The volume occupied by an ideal gas is found from the equation $V = \frac{nRT}{P}$
  – $V$ is the volume in m$^3$
  – $n$ is the number of kmols (kgmoles)
  – $R = 8.3144$ kPa·m$^3$/kmol·K is the (universal) gas constant
  – $T$ is the temperature in kelvins
  – $P$ is the pressure in kPa

• Write a program that computes $V$ for fixed $n$ and $T$ using several $P$ values

**Exercise II**

• Write a program that computes and prints $V = \frac{nRT}{P}$ for a series of pressures using $n = 1$ kmol and $T = 273.15$ K
  – $P = 1, 2, 5, 10, 20, 50, 100, 200, 500, 1000, 2000, 5000, \text{ and } 10000$ kPa
  – $R = 8.3144$ kPa·m$^3$/kmol·K

**Exercise Solution**

```cpp
const double R = 8.3144;
const int NP = 13;
double n = 1, T = 273.15, P[NP] = {1, 2, 5, 10, 20, 50, 100, 200, 500, 1000, 2000, 5000, 10000};
for (int k = 0; k < NP; k++) {
    cout << "\nFor P = " << P[k] << " kPa, V = " << n * R * T / P[k] << " m^3";
}
```

**Calculating Two-way Tables**

• The previous exercise used an array to compute a one way table with uneven increments in pressure
  – Found $V$ as a function of $P$ for one kmol of an ideal gas at $T = 273.15$ k

• What if we wanted a two-way table with uneven increments in both tables?
  – Define arrays for both the row and the column variables
Two-way Table Exercise

- Compute the density, $\rho$, of ammonia from the ideal gas law for each possible combination of pressure and temperature
  - $P = 1, 2, 5, 10, 20, 50, 100, 200, 500, \text{ and } 1000 \text{ kPa}$
  - $T = 300, 350, 400, 405.5, 450, \text{ and } 500 \text{ K}$
- Formula: $\rho = \frac{m}{V} = \frac{Mn}{V} = \frac{MP}{RT}$ where $M = 17.03 \text{ kg/kmol}$ is the molecular mass
- (Universal) gas constant $R = 8.3144 \text{ kPa} \cdot \text{m}^3/\text{kmol} \cdot \text{K}$

Two-way Table Exercise II

- How to design table
  - Temperatures in rows and pressures in columns or vice versa
  - Arrays for individual $T$ and $P$ values
- Code is same as in exercise six
  - Separate loop to print column headers
  - Outer loop over row variable
    - Inner loop to compute each row entry for different values of the column variable

Two-way Table Exercise III

```cpp
const double R = 8.3144, M = 17.03;
const int NP = 10, NT = 6;
double n = 1, P[NP] = {1, 2, 5, 10, 20, 50, 100, 200, 500, 1000 },
T[NT] = { 300, 350, 400, 405.5, 450, 500 };
// print column headers
cout << "        " << fixed
<< setprecision(1);
for ( int k = 0; k < NT; k++ )
cout << " T = " << setw(5) << T[k];
```

Two-Way Table Exercise IV

```cpp
// outer loop to do all rows
for ( int m = 0; m < NP; m++ )  {
  cout << setprecision(0)
  << "\nP = " << setw(4) << P[m] << setprecision(7);
  // inner (column) loop
  for ( int k = 0; k < NT; k++ )
    cout << setw(10) << M * P[m] / ( R * T[k] );
}
```

Two-Way Table Exercise V

Table of ammonia density (kg/m³) for $P$ in kPa and $T$ in kelvins

<table>
<thead>
<tr>
<th>$P$ (kPa)</th>
<th>$T$ (K)</th>
<th>Density (kg/m³)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>300.0</td>
<td>0.0068275</td>
</tr>
<tr>
<td>2</td>
<td>300.0</td>
<td>0.0136550</td>
</tr>
<tr>
<td>5</td>
<td>300.0</td>
<td>0.0341376</td>
</tr>
<tr>
<td>10</td>
<td>300.0</td>
<td>0.0682751</td>
</tr>
<tr>
<td>20</td>
<td>300.0</td>
<td>0.1365502</td>
</tr>
<tr>
<td>50</td>
<td>300.0</td>
<td>0.3413756</td>
</tr>
<tr>
<td>100</td>
<td>300.0</td>
<td>0.6827512</td>
</tr>
<tr>
<td>200</td>
<td>300.0</td>
<td>1.3655024</td>
</tr>
<tr>
<td>500</td>
<td>300.0</td>
<td>3.4137561</td>
</tr>
<tr>
<td>1000</td>
<td>300.0</td>
<td>6.8275121</td>
</tr>
</tbody>
</table>

Assignments

- Reading pages in text
  - Today: None
  - Thursday: Pages 425–433
  - Tuesday, May 2: None
- This week’s homework problems
  - Page 474, program 6
- Project two due April, Tuesday 25