Exercises in Array Programming

Larry Caretto
Computer Science 106

Computing in Engineering and Science

April 25, 2006

California State University Northridge

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

Run	Data	Math	C++
0	12.3	x_0	x[0]
1	14.4	x_1	x[1]
2	11.8	x_2	x[2]
3	12.5	x_3	x[3]
4	13.2	x_4	x[4]
5	14.1	x_5	x[5]

- C++ array, x[i] for math x_i
- 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

```
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 << "\nz[" << i << "] = "
         << z[i];
}
```


Console Input File Output

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

Defined Elements

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

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


Computing the Mean

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i = \frac{1}{N}\sum_{i=0}^{N-1} x_i$$

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

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

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

```
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

```
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

Average Power One

```
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

```
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

```
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

$$s = \sqrt{\frac{\sum_{i=0}^{N-1}(x_i - \bar{x})^2}{N-1}} = \sqrt{\frac{\left(\sum_{i=0}^{N-1} x_i^2\right) - N(\bar{x})^2}{N-1}} = \sqrt{\frac{\left(\sum_{i=0}^{N-1} x_i^2\right) - \frac{1}{N}\left(\sum_{i=0}^{N-1} x_i\right)^2}{N-1}}$$

- Use final equation to write code to compute standard deviation
- **Hint:** Compute two sums in a single for loop: $\sum_{i=0}^{N-1} x_i$ and $\sum_{i=0}^{N-1} x_i^2$

Standard Deviation Code

$$s = \sqrt{\frac{\left(\sum_{i=0}^{N-1} x_i^2\right) - \frac{1}{N}\left(\sum_{i=0}^{N-1} x_i\right)^2}{N-1}}$$

```
  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

x	f(x)
0	0
1	1
2	4
3	9
4	16
5	25

Even increments in x; no array required

x	f(x)
1	1
2	4
4	16
7	49
10	100
20	400

Uneven increments in x; array required for x

Two-way, Both Fixed Increment

Kinetic Energy Table (m = mass in kg, V = velocity in m/s and kinetic energy in joules)						
	m = 1	m = 2	m = 3	m = 4	m = 5	m = 6
V = 1	0.5	1.0	1.5	2.0	2.5	3.0
V = 2	2.0	4.0	6.0	8.0	10.0	12.0
V = 3	4.5	9.0	13.5	18.0	22.5	27.0

- 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

```
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

Kinetic Energy Table (m = mass in kg, V = velocity in m/s and kinetic energy in joules)					
	m = 1	m = 2	m = 5	m = 10	m = 20
V = 1	0.5	1.0	2.5	5.0	10.0
V = 2	2.0	4.0	10.0	20.0	40.0
V = 3	4.5	9.0	22.5	45.0	90.0

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

Variable Column Increments

```
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

Kinetic Energy Table (m = mass in kg, V = velocity in m/s and kinetic energy in joules)						
	m = 1	m = 2	m = 3	m = 4	m = 5	m = 6
V = 1	0.5	1.0	1.5	2.0	2.5	3.0
V = 2	2.0	4.0	6.0	8.0	10.0	12.0
V = 4	8.0	16.0	24.0	18.0	22.5	27.0

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

Variable Row Increments

```
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 m = 1; m <= 6; m++ )
        cout << setw(7) << 0.5 *
                    m * v[i] * v[i];
}
```

Two-way, Both Increments Vary

Kinetic Energy Table (m = mass in kg, V = velocity in m/s and kinetic energy in joules)					
	m = 1	m = 2	m = 5	m = 10	m = 20
V = 1	0.5	1.0	2.5	5.0	10.0
V = 2	2.0	4.0	10.0	20.0	40.0
V = 4	8.0	16.0	40.0	80.0	160.0

- This requires arrays for both the row variable (velocity and the column variable (mass)

Both Increments Variable

```
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 idea gas is found from the equation V = 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 = 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, and 10000 kPa
  - R = 8.3144 kPa·m^3/kmol·K

Exercise Solution

```
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, ρ, 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, and 1000 kPa
  - T = 300, 350, 400, 405.5, 450, and 500 K
- Formula: ρ = m/V = Mn/V = MP/RT where M = 17.03 kg/kmol is the molecular mass
  - (Universal) gas constant R = 8.3144 kPa·m³/kmol·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

```
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

```
    // 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/m^3) for
        P in kPa and T in kelvins

       T = 300.0 T = 350.0 T = 400.0 T = 405.5 T = 450.0
P =    1 0.0068275 0.0058522 0.0051206 0.0050512 0.0045517
P =    2 0.0136550 0.0117043 0.0102413 0.0101024 0.0091033
P =    5 0.0341376 0.0292608 0.0256032 0.0252559 0.0227584
P =   10 0.0682751 0.0585215 0.0512063 0.0505118 0.0455167
P =   20 0.1365502 0.1170431 0.1024127 0.1010236 0.0910335
P =   50 0.3413756 0.2926077 0.2560317 0.2525590 0.2275837
P =  100 0.6827512 0.5852153 0.5120634 0.5051180 0.4551675
P =  200 1.3655024 1.1704306 1.0241268 1.0102361 0.9103349
P =  500 3.4137561 2.9260766 2.5603170 2.5255902 2.2758374
P = 1000 6.8275121 5.8521532 5.1206341 5.0511804 4.5516747
```


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 Tuesday, April 25