Designing Software with Arrays and Functions

Larry Caretto
Computer Science 106

Computing in Engineering and Science

May 2, 2006

California State University Northridge

Outline

- Review passing arrays to functions
  - Use array notation x[] in header and prototype for arrays
  - By default, arrays are always passed by reference
    - No ampersand (&) required
- Constructing functions with arrays
- Restructuring arrays

California State University Northridge 2

Passing Arrays to Functions

- We can pass an array element to a function as we pass any variable
- y = pow(x[k], 3);
- Here the pow function returns the cube of element k of the x array
- This is no different from passing a single variable to a function
- We can also pass whole arrays, like x, to functions: getAverage(x, first, last)

California State University Northridge 3

getAverage

- When we pass a whole array to a function,
  - the function can use any element of the array
  - the array is ***always*** passed by reference
- Header: double getAverage (double x[], int first, int last)
- Prototypes:
  - double getAverage (double x[], int first, int last);
  - double getAverage (double [], int, int);
- Note use of [] to specify an array as a function argument

California State University Northridge 4

getAverage

```
double getAverage ( double x[],
         int first, int last )
{
    double sum = 0;
    for ( int i = first;
           i <= last; i++ )
        sum += x[i];
    return sum /
          ( last – first + 1 );
}
```

California State University Northridge 5

Use of getAverage

- double x[22], power[50], density[30];
- // code to get input data on x and power
- double mean = getAverage(x, 0, 10);
- double average = getAverage(power, 12, 24);
- How would you compute the average of all elements of the density array?

```
getAverage( density, 0, 29);
```

California State University Northridge 6

Standard Deviation

- Measure of spread around mean

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

- First term is definition; others are computational forms
- How would we write a function to compute s for all the elements in an N-element array?

getStdDev

```
double getStdDev(double x[],int N)
{
    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 ) );
}
```


Arrays Passed by Reference

```
void setArray( double x[],
          int N, double value )
{
   for ( int k = 0, k < N; k++ )
        x[k] = value;
}
```

- A call, setArray(c, M) would zero the first M elements of the c array
- For arrays, pass by reference occurs by default without the need for an &

Array Function Exercise

- Write a function to compute a power array from current and voltage arrays
- What information do you pass to the function?
  - The array names for current and voltage and the number of elements (same for both arrays)
- How would you return the power array to the calling function
  - As a function parameter (arrays are always passed by reference)

Array Function Exercise II

- The function uses the arrays power[k], amps[k], and volts[k], for power, current, and voltage, respectively
- Write the function header

```
void getPower( double amps[], double
  volts[], int N, double power[] )
```

- Write two prototypes for this function

```
void getPower( double amps[], double
  volts[], int N, double power[]);
```


Array Function Exercise III

```
void getPower( double [],
 double [], int , double [] );
```

- Write a statement that calls `getPower` from a function with the declaration: `double curr[250], volt[250], pow[250];`

```
getPower( curr, volt, 250, pow )
```

Array Function Exercise IV

- For the same arrays, `double curr[250], volt[250], pow[250];` What would the result of the following statement be? `getPower ( curr, volt, 100, pow )`
- It would compute 100 elements of the power array, from power[0] to power[99]

California State University Northridge 13

Array Function Exercise V

- For the same arrays, `double curr[250], volt[250], pow[250];` What would the result of the following statement be? `getPower ( curr, volt, 300, pow )`
- It would use 50 storage locations at the ends of the curr and volt arrays to compute 50 extra values of "power" that would replace 50 storage locations following the power array

California State University Northridge 14

Array Function Exercise VI

- Write the code for the `getPower` function

```
void getPower( double amps[],
               double volts[], int N,
               double power[] )
{
    for ( int k = 1; k < N; k++ )
        power[k] = amps[k] *
                          volts[k];
}
```

California State University Northridge 15

Regression Function

- Calculation function from exercise eight
  - pass arrays x[i] and y[i] and size n into function
    - n is number of elements in array not necessarily the maximum possible number of elements
  - return yHat array and computed variables of slope, intercept, sxy, and R_squared to calling function by reference
- Shows typical array function calculations

California State University Northridge 16

Regression Calculations

- Evaluate the following equations for slope, b, and intercept, a

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

$$a = \frac{1}{N}\left[\left(\sum_{i=0}^{N-1} y_i\right) - b\left(\sum_{i=0}^{N-1} x_i\right)\right]$$

California State University Northridge 17

Regression Calculations II

- After a and b are found we find

$$\hat{y}_i = a + bx_i \qquad i = 0, \cdots N-1 \qquad s_{y|x} = \sqrt{\frac{\sum_{i=0}^{N-1}(y_i - \hat{y}_i)^2}{N-2}}$$

$$R^2 = 1 - \frac{(N-2)s_{y|x}^2}{\sum_{i=0}^{N-1} y_i^2 - \frac{1}{N}\left(\sum_{i=0}^{N-1} y_i\right)^2} = 1 - \frac{\sum_{i=0}^{N-1}(y_i - \hat{y}_i)^2}{\sum_{i=0}^{N-1} y_i^2 - \frac{1}{N}\left(\sum_{i=0}^{N-1} y_i\right)^2}$$

California State University Northridge 18

Function Design

- Write a function to do these calculations
- Must pass x_i and y_i arrays and N to function
- Function returns $\hat{y}_i$ array and slope, b, intercept, a, $s_{y|x}$, and R^2
  - Non-array return values must use pass by reference
  - Arrays always use pass by reference as default

Function Design II

- Function code must compute all of the following sums from input data

$$\sum_{i=0}^{N-1} x_i y_i \quad \sum_{i=0}^{N-1} x_i \quad \sum_{i=0}^{N-1} y_i \quad \sum_{i=0}^{N-1} x_i^2 \quad \sum_{i=0}^{N-1} y_i^2$$

- Calculation of R^2 and $s_{y|x}$ requires calculation of another sum that can only be done after a and b are found

$$\hat{y}_i = a + bx_i \qquad i = 0, \cdots N-1 \qquad s_{y|x} = \sqrt{\frac{\sum_{i=0}^{N-1}(y_i - \hat{y}_i)^2}{N-2}}$$

Function Design III

- Steps for regression function
  - Compute sums below in a single loop

$$\sum_{i=0}^{N-1} x_i y_i \quad \sum_{i=0}^{N-1} x_i \quad \sum_{i=0}^{N-1} y_i \quad \sum_{i=0}^{N-1} x_i^2 \quad \sum_{i=0}^{N-1} y_i^2$$

  - Compute b and a
  - Compute $\hat{y}_i = a + bx_i$ and $\sum_{i=0}^{N-1}(y_i - \hat{y}_i)^2$
    - This can be done in the same loop
  - Compute $s_{y|x}$ and R^2

Regression Code

Passed into function

Returned from function

```
void doCalculations( double x[],
    double y[], double y_hat[],
    int n, double& slope,
    double& intercept, double& syx,
    double& R_squared )
{
double    sumx  = 0, // sum of x
          sumxx = 0, // sum of x^2
          sumxy = 0, // sum of x * y
          sumy  = 0, // sum of y
          sumyy = 0; // sum of y^2
```

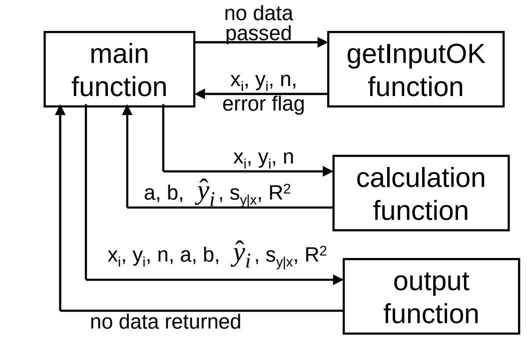

Regression Code II

```
for ( int i = 0; i < n; i++ ) {
    sumx  += x[i];
    sumy  += y[i];
    sumxx += x[i] * x[i];
    sumyy += y[i] * y[i];
    sumxy += x[i] * y[i];
}
slope = ( n * sumxy
        - sumx * sumy ) /
        ( n * sumxx - sumx * sumx );
intercept = ( sumy
            - slope * sumx ) / n;
```


Regression Code III

```
double sumyyhat2 = 0;
for ( i = 0; i < n; i++ ) {
    y_hat[i] = intercept
             + slope * x[i];
    sumyyhat2 += pow( y[i]
                  - y_hat[i], 2 );
}
syx = sqrt( sumyyhat2
          / ( n - 2 ) );
R_squared = 1 - ( n - 2 )
          * syx * syx /
          ( sumyy –
            sumy * sumy / n );
}
```

Overall Program Design



25

Code for main

```
const int MAX_DATA = 1000  // global
int main()
{
   int n;
   double slope, intercept, syx,
          R_squared, x[MAX_DATA],
          y[MAX_DATA],y_hat[MAX_DATA];
   if ( !getInputOK( x, y, n) )
      return EXIT_FAILURE;
   doCalculations( x, y, y_hat, n,
   slope, intercept, syx, R_squared );
   doOutput( slope, intercept, syx,
          R_squared, x, y, y_hat, n );
   return EXIT_SUCCESS;
}
```

26

Assignments

- Reading pages in text
  - Today: None
  - Thursday: Pages 435–445
  - Tuesday, May 9: Pages 447–454
- This week's homework problems
  - Page 434, checkpoints 7.17 and 7.18
- Exercise eight due Thursday
- Lab quiz on exercise eight on Thursday, May 11

27

A Note about One Sum

- The separate calculation of $\sum_{i=0}^{N-1}(y_i-\hat{y}_i)^2$ is not required
  - It can be found from other sums already computed and the definition of $\hat{y}_i$

$$\sum_{i=0}^{N-1}(y_i-\hat{y}_i)^2 = \sum_{i=0}^{N-1}(y_i-a-bx_i)^2 =$$

$$\sum_{i=0}^{N-1}(y_i^2+a^2+b^2x_i^2-2y_ia-2y_ibx_i+2abx_i) =$$

$$\sum_{i=0}^{N-1}y_i^2+Na^2+b^2\sum_{i=0}^{N-1}x_i^2-2a\sum_{i=0}^{N-1}y_i-2b\sum_{i=0}^{N-1}x_iy_i+2ab\sum_{i=0}^{N-1}x_i$$

28

A Note about One Sum II

- With this result we can compute $\sum_{i=0}^{N-1}(y_i-\hat{y}_i)^2$ from the following C++ variables representing quantities already found

$$\sum_{i=0}^{N-1}(y_i-\hat{y}_i)^2 = \sum_{i=0}^{N-1}y_i^2+Na^2+b^2\sum_{i=0}^{N-1}x_i^2-2a\sum_{i=0}^{N-1}y_i-2b\sum_{i=0}^{N-1}x_iy_i+2ab\sum_{i=0}^{N-1}x_i$$

```
sumyyhat2 = sumyy + n * pow(
intercept, 2) + pow(slope,2) *
sumxx – 2 * intercept * sumy – 2 *
slope * sumxy + 2 * slope *
intercept * sumx
```

29