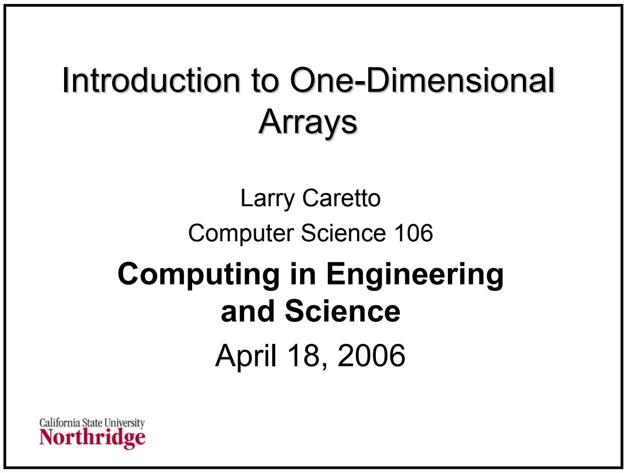


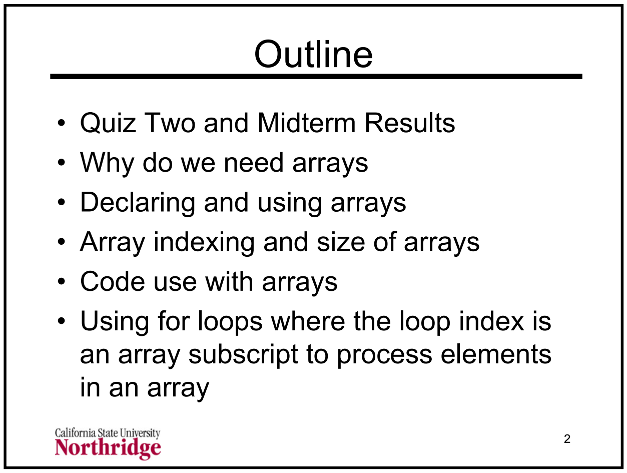
Introduction to One-Dimensional Arrays

Larry Caretto

Computer Science 106

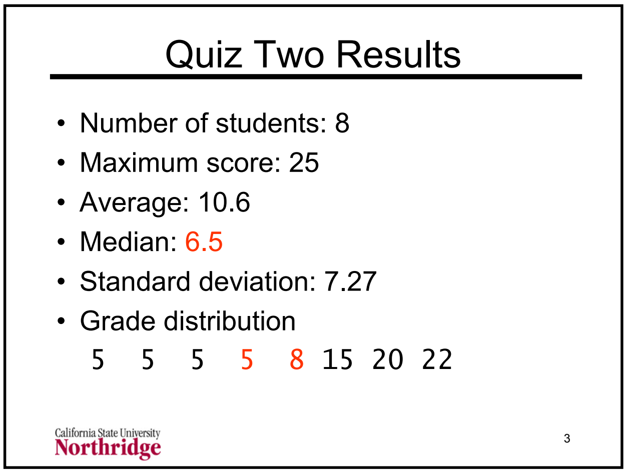
Computing in Engineering and Science

April 18, 2006



Outline

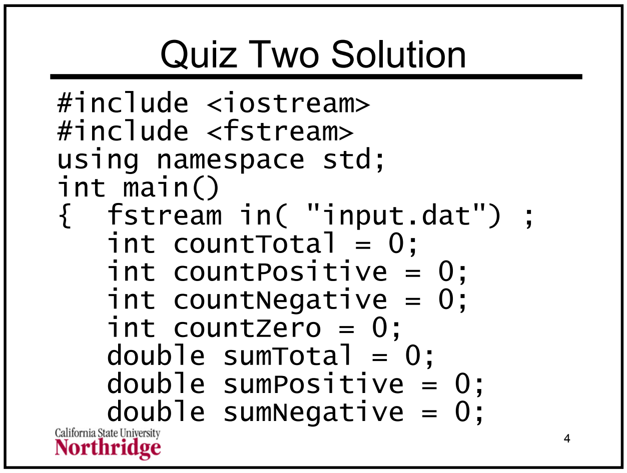
- Quiz Two and Midterm Results
- Why do we need arrays
- Declaring and using arrays
- Array indexing and size of arrays
- Code use with arrays
- Using for loops where the loop index is an array subscript to process elements in an array



Quiz Two Results

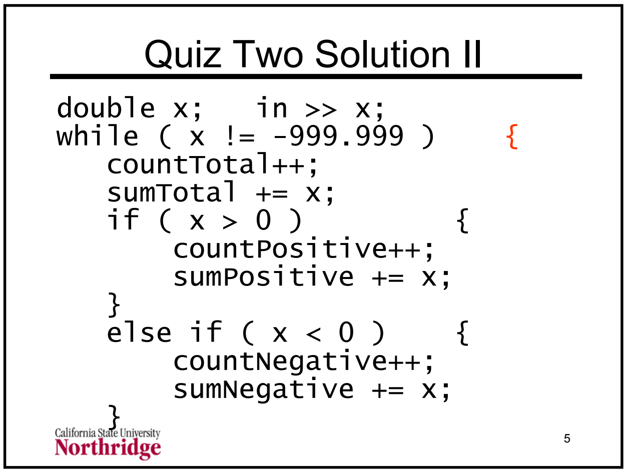
- Number of students: 8
- Maximum score: 25
- Average: 10.6
- Median: 6.5
- Standard deviation: 7.27
- Grade distribution

  5 5 5 5 8 15 20 22



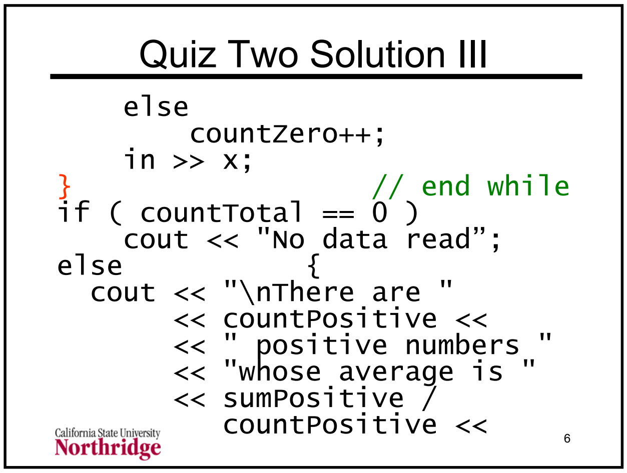
Quiz Two Solution

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{  fstream in( "input.dat") ;
   int countTotal = 0;
   int countPositive = 0;
   int countNegative = 0;
   int countZero = 0;
   double sumTotal = 0;
   double sumPositive = 0;
   double sumNegative = 0;
```



Quiz Two Solution II

```
double x;   in >> x;
while ( x != -999.999 )    {
   countTotal++;
   sumTotal += x;
   if ( x > 0 )        {
       countPositive++;
       sumPositive += x;
   }
   else if ( x < 0 )   {
       countNegative++;
       sumNegative += x;
   }
```



Quiz Two Solution III

```
    else
        countZero++;
    in >> x;
}                  // end while
if ( countTotal == 0 )
    cout << "No data read”;
else            {
  cout << "\nThere are "
       << countPositive <<
       << " positive numbers "
       << "whose average is "
       << sumPositive /
          countPositive <<
```

Quiz Two Solution IV

```
<< ".\nThere are "
<< countNegative
<< " negative numbers "
<< "whose average is “
<< sumNegative /
   countNegative
<< ".\nThere are " << countZero
<< " zeros " << ".\nThere are “
<< countTotal << " total data “
<< "points whose average is "
<< sumTotal / countTotal << ".\n";
```

Midterm

- Number of students: 9
- Maximum score: 110
- Average: 76.1
- Median: 74
- Standard deviation: 16.19
- Grade distribution

  52 58 65 70 74 85 91 93 97

First Problem

```
const double g = 9.808
double mDot, height, efficiency;
cout << “Enter the mass flow rate in kg/s: “;
cin >> mDot;
cout << “Enter the efficiency as a fraction:“;
cin >> efficiency;
cout << “Enter the water height in meters: “
cin >> height;
double power = efficiency * mDot * g * height
  * 1e-6;
cout << “\nFor your input data: “
     << “\n  Mass flow rate = “ << mDot
     << “kg/s” << “\n      Efficiency = “
     << efficiency” << “\n Height of water = “
     << height << “m\n” << “\nPower output = “
     << power << “MW.”
```

Second Problem

- Trace code through loop
- Read data from file, left to right to exhaust data on one line then move to next line
- Exit while loop when a = -9999
- Gives only three cases

  Two real roots: -1 and -1

  Two real roots: 3 and -2

  Complex roots

  Real part = 0,   Imaginary part = 2

Third Problem

```
cin >> n >> a >> b;
while ( n != -999.999 )  {
   cout << “\nFor your inputs a = “ << a
        << “, b = “ << b << “, and n = “
        << n << “, the integral is “ ;
   if ( n == -1 && ab > 0 )
       cout << log( b / a );
   else if ( n == -1 )
       cout << “not defined.”;
   else
       cout << ( pow( b, n – 1 ) –
         pow( a, n – 1 ) ) / ( n – 1 );
   cout << “Enter n a b: ”;
   cin >> n >> a >> b;
}
```

Representing Data

Run	Data
1	12.3
2	14.4
3	11.8
4	12.5
5	13.2
6	14.1

- Consider a set of experimental data with several runs
- How do we represent the data in such a way that we can process these data and similar data sets with more values?

Representing Data II

Run, i	x data
1	x_1
2	x_2
3	x_3
4	x_4
5	x_5
6	x_6

- x_i is mathematical notation for several cases of similar data
- Use this formula to find the mean of N data items

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

Representing Data III

- When we have a set of N data items like x_i it will occupy N memory locations
- A variable like x, declared as double x, occupies only one memory location
- For our data on x_i
  - We want to call is by its name, x
  - We want to have N memory locations
  - We want to compute formulas like average x
  - We want to refer to a specific x, say x_3

Arrays Represent Data

- An array is a way that we can represent the mathematical notation for x_i
- We use the programming notation x[i] to represent the general data element x_i
- When we declare a variable as an array, we reserve the memory locations that we will need for the data
  - Regular variable: double x;
  - Array variable: double x[200];

How to represent x_i

- An array has a single variable name, like x, augmented by a subscript to identify the particular data item
- Example x[3] or x[k]
- Power of array structure is use of variable subscript as loop index to refer to different elements
- Arrays must be declared with maximum size

One-dimensional C++ Array

Math	C++
x_0	x[0]
x_1	x[1]
x_2	x[2]
x_3	x[3]
x_4	x[4]
x_5	x[5]

- View one-dimensional arrays as a column (or row) of cells
- Start with zero subscript
- Array shown here has 6 elements with subscripts from 0 to 5

Maximum Array Subscript

```
double w[4];  // 4 elements
const int MAX_SIZE = 10;
double x[MAX_SIZE];  // 10 elements
```

- Minimum subscript is zero
- Maximum subscript is one less than the number of elements
- w[0], w[1], w[2], and w[3] are the four elements of the w array
- Note different meanings of w[N]

Array Size Declarations

- Can be any expression that evaluates to a constant

```
const int MAX_SIZE = 100;
int count[MAX_SIZE];
double current[2*MAX_SIZE];
```

- Symbolic constant is preferred style
  - Allows subsequent reference to array dimension in code with ability to change all occurrences of this value with one edit

Declaring and Using Arrays

- In a declaration statement like double x[20], the 20 is the maximum array size
  - This must be a literal or symbolic constant
- When using arrays in statements like z = x[3] or x[k] = w * q the “3” or “k” refers to a particular element of the x array
  - Here we can use a variable or a constant, but the variable must be assigned a value before it is used as a subscript

Initializing Arrays

- We can declare and initialize ordinary variables, *e. g.*, `int x = 10;`
- We can also do this for arrays by giving a value for each element

```
double x[5] = { 1, 3, 5, 18, 1 }
```

- Array size not required

```
double y[] = { 3, 8, 33, 7 }
```

- What is maximum subscript for y? 3
- What is the value of y[1]? 8

Two Uses of [] for C++ Arrays

Math	C++
x_0	x[0]
x_1	x[1]
x_2	x[2]
x_3	x[3]
x_4	x[4]
x_5	x[5]

- Declare this array for six elements total: `double x[6]`
- Refer to a particular element as in following examples

```
x[2] = 3.5;
x[k+2] = x[0] +
  x[1];
```

Must assign a value to k before this statement

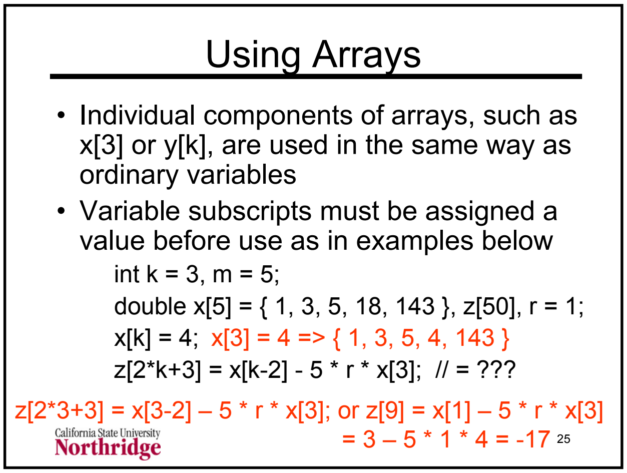
Maximum Array Subscript

- Array elements are stored in contiguous memory locations
- Program computes memory location from subscript
- C++ does not check to see if an array subscript is in bounds
- An incorrect subscript could affect some other memory location

Subscript out of Range

y
x[0]
x[1]
x[2]
x[3]
x[4]
z

- Cells show memory locations for y, x[] array, and z
- The x array has five elements stored in the locations shown
- x[-1] would give the same location as the variable y
- x[5] would give the same location as the variable z

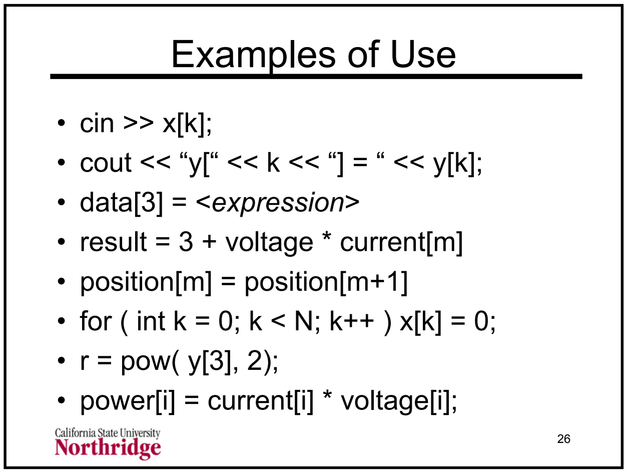


Using Arrays

- Individual components of arrays, such as x[3] or y[k], are used in the same way as ordinary variables
- Variable subscripts must be assigned a value before use as in examples below

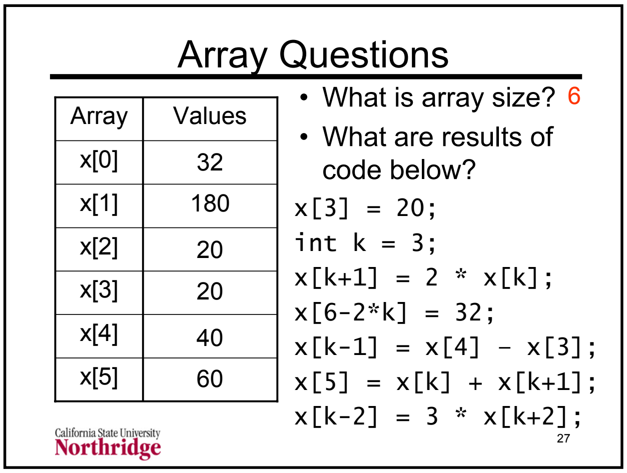
```
int k = 3, m = 5;
double x[5] = { 1, 3, 5, 18, 143 }, z[50], r = 1;
x[k] = 4;  x[3] = 4 => { 1, 3, 5, 4, 143 }
z[2*k+3] = x[k-2] - 5 * r * x[3];  // = ???
```

z[2*3+3] = x[3-2] – 5 * r * x[3]; or z[9] = x[1] – 5 * r * x[3]
= 3 – 5 * 1 * 4 = -17



Examples of Use

- cin >> x[k];
- cout << “y[“ << k << “] = “ << y[k];
- data[3] = <*expression*>
- result = 3 + voltage * current[m]
- position[m] = position[m+1]
- for (int k = 0; k < N; k++) x[k] = 0;
- r = pow(y[3], 2);
- power[i] = current[i] * voltage[i];

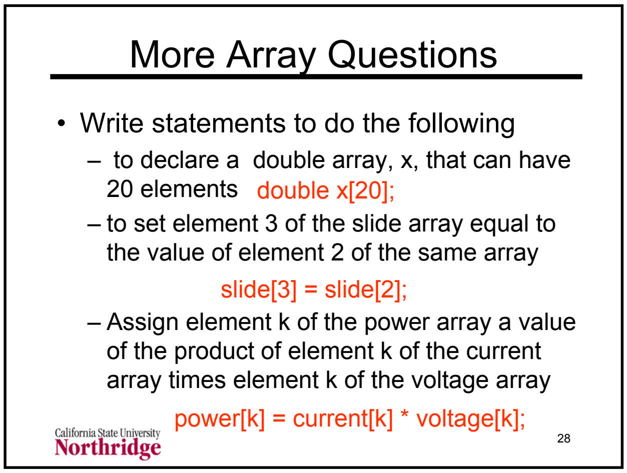


Array Questions

Array	Values
x[0]	32
x[1]	180
x[2]	20
x[3]	20
x[4]	40
x[5]	60

- What is array size? 6
- What are results of code below?

```
x[3] = 20;
int k = 3;
x[k+1] = 2 * x[k];
x[6-2*k] = 32;
x[k-1] = x[4] - x[3];
x[5] = x[k] + x[k+1];
x[k-2] = 3 * x[k+2];
```

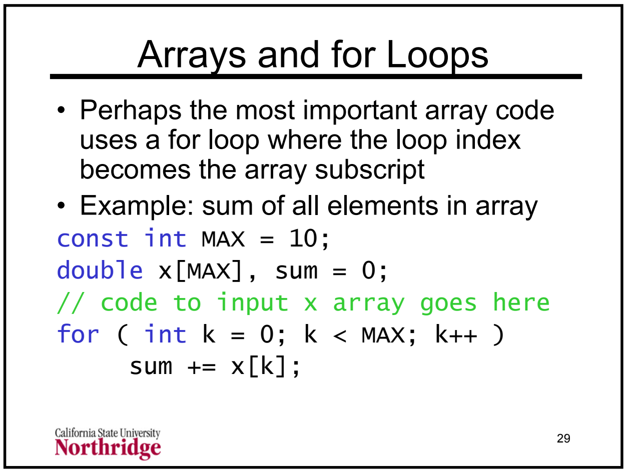


More Array Questions

- Write statements to do the following
  - to declare a double array, x, that can have 20 elements double x[20];
  - to set element 3 of the slide array equal to the value of element 2 of the same array

    slide[3] = slide[2];
  - Assign element k of the power array a value of the product of element k of the current array times element k of the voltage array

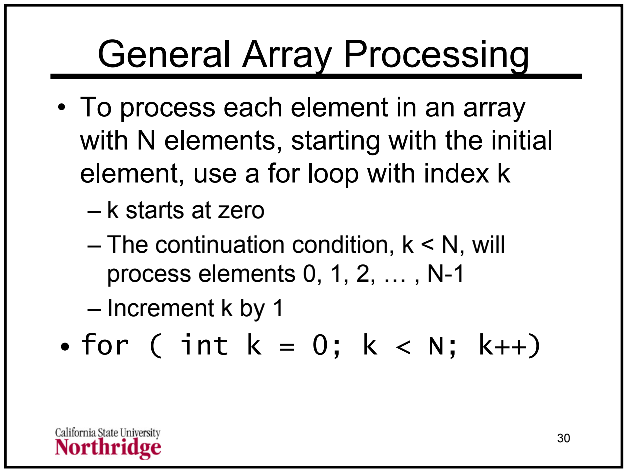
    power[k] = current[k] * voltage[k];



Arrays and for Loops

- Perhaps the most important array code uses a for loop where the loop index becomes the array subscript
- Example: sum of all elements in array

```
const int MAX = 10;
double x[MAX], sum = 0;
// code to input x array goes here
for ( int k = 0; k < MAX; k++ )
     sum += x[k];
```



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
  - k starts at zero
  - The continuation condition, k < N, will process elements 0, 1, 2, … , N-1
  - Increment k by 1
- `for ( int k = 0; k < N; k++)`

General Array Processing II

- On the previous chart N means the number of elements defined, not the total number of elements that can be stored in the array
- Sometimes it is more convenient to refer to the subscripts than to the number of elements
- E. g., array whose first and last defined elements have subscripts F and L
- `for ( k = F, k <= L; k++ )`

General Array Processing III

- In the examples that follow, we will generally assume that an array has N elements, whose first subscript is zero
- The for loop command to process each element in such an array is
  `for ( k = 0; k < N; k++)`
- We can use different increments (e.g. k += 3) to skip elements

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 >> x[i]
    outFile << z[i] << endl;
}
```

Defined Elements

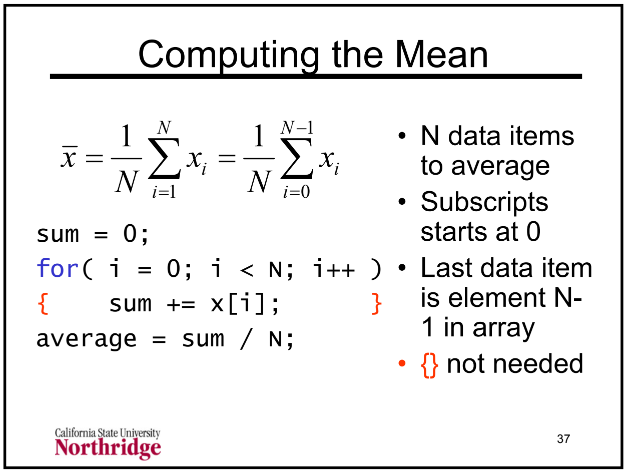
- The number of elements defined may be less then the array size
- You may declare an array to be the maximum size expected but actually specify a value for fewer elements

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

Computing Array Sums

- Use previous tool of running sum variable along with for loop to include all array elements in the sum
- Application to mean on next chart

```
sum = 0;
for( i = 0; i < N; i++ )  { } not
{     sum += x[i];      }  needed here
cout << “Array sum = “ << sum
```



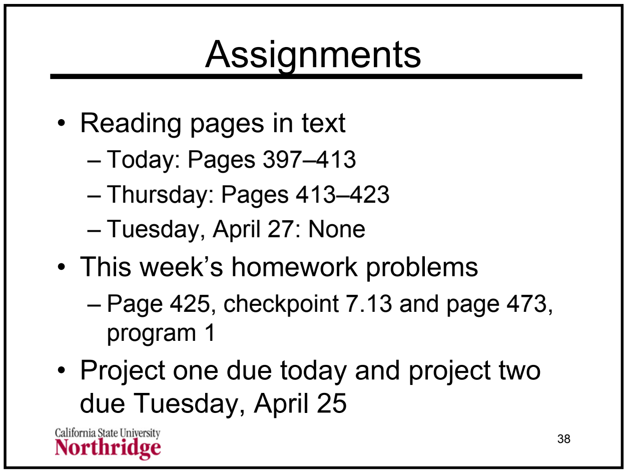
Computing the Mean

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

```
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

California State University Northridge



Assignments

- Reading pages in text
  - Today: Pages 397–413
  - Thursday: Pages 413–423
  - Tuesday, April 27: None
- This week’s homework problems
  - Page 425, checkpoint 7.13 and page 473, program 1
- Project one due today and project two due Tuesday, April 25

California State University Northridge