

Introduction to Data Structures

- A data structure is a grouping of related data items in memory.
- The items in a data structure can be processed individually, although some operations may be performed on the data structure as a whole.

Arrays

- An **array** is a data structure used for storage of a collection of data items that are all the same type (for example, the exam scores for a class).
- By using an array, we will be able to associate a single variable name (scores) with a group of related data items (exam scores).

Declaring and Referencing Arrays

```
double x[8] ;
```

- This declaration instructs the compiler to associate eight memory cells with the name x; these memory cells are adjacent to each other in memory.

- Each cell is 8 bytes in size (the size of a double), so the entire array occupies $8 \times 8 = 64$ bytes
- The subscripted variable $x[0]$ (read as "*x sub zero*" or "*x at zero*") may be used to reference the initial or 0th element of the array x ; $x[1]$ is the next element, and $x[7]$ is the last element.
- The integer enclosed in brackets is the array subscript, and its value must be in the range from zero to one less than the number of memory cells in the array.

Array Initialization

- An array can also be initialized in its declaration. The size of an array that is being fully initialized can optionally be omitted from the declaration since it can be deduced from the initialization list.
- For example, in the following statement, a 25-element array is initialized with prime numbers less than 100.

```
int prime_lt_100 [ 25 ] =  
{  
    2,  3,  5,  7, 11,  
    13, 17, 19, 23, 29,  
    31, 37, 41, 43, 47,  
    53, 59, 61, 67, 71,  
    73, 79, 83, 89, 97  
} ;
```

Using Indexed for Loops to Process Arrays

- In C, we can accomplish this processing using an indexed for loop, a counting loop whose control variable runs from zero to one less than the array size.
- Using the loop counter as an array index (*subscript*) gives access to each array element in turn.
- For example, store the squares of the integers 0 through 10 in an array.

```
const int SIZE = 11 ;
...

int square [ SIZE ] ;
int i ;

...

for ( i = 0 ; i < SIZE ; i++ )
{
    square[ i ] = i * i ;
}
...
```

Example: Statistical Computations Using Arrays

- One common use of arrays is for storage of a collection of related data values.
- Once the values are stored, some simple statistical computations may be performed.

```
/*  
 * Computes the mean and standard  
 * deviation of an array of data  
 * and prints the difference between  
 * each value and the mean.  
 */  
#include <iostream>  
#include <iomanip>  
#include <cmath>  
  
using namespace std ;  
  
// max. number of items in list  
  
const int MAX_ITEM = 8 ;  
  
int  
main( )  
{
```

```
double x [ MAX_ITEM ] ;    // data list
```

```

double mean ;    // mean (average) of
                 // the data
double st_dev ; // standard deviation
                 // of data
double sum ;    // sum of the data
double sum_sq ; // sum of the squares
                 // of data

int i ;         // loop counter

// Get the data

cout << "Enter "
      << MAX_ITEM
      << " numbers separated by "
      << "blanks or returns\n> " ;

for ( i = 0 ; i < MAX_ITEM ; i++ )
{
    cin >> x[ i ] ;
}

// Compute the sum and the sum of
// squares of data

sum = sum_sq = 0.0 ; // zero totals

```

```
for ( i = 0 ; i < MAX_ITEM ; i++ )
{
    sum += x[ i ] ;
    sum_sq += x[ i ] * x[ i ] ;
}

// Compute and print the mean and
// std. deviation

mean = sum / MAX_ITEM ;

st_dev = sqrt( sum_sq / MAX_ITEM -
               mean * mean ) ;

cout << "The mean is "
      << fixed << showpoint
      << setprecision( 2 ) << mean
      << "." << endl ;

cout << "The standard deviation is "
      << fixed << showpoint
      << setprecision( 2 ) << st_dev
      << "." << endl ;

// Display difference between each item
// and mean
```

```

cout << endl
    << "Table of differences between "
    << "data values and mean"
    << endl
    << "Index          "
    << "Item           "
    << "Difference"
    << endl ;

for ( i = 0 ; i < MAX_ITEM ; i++ )
{
    cout << setw( 3 ) << i << "      "
        << fixed << showpoint
        << setw( 9 )
        << setprecision( 2 )
        << x[ i ] << "      "
        << setw( 9 )
        << setprecision( 2 )
        << ( x[ i ] - mean )
        << endl ;
}
}

```