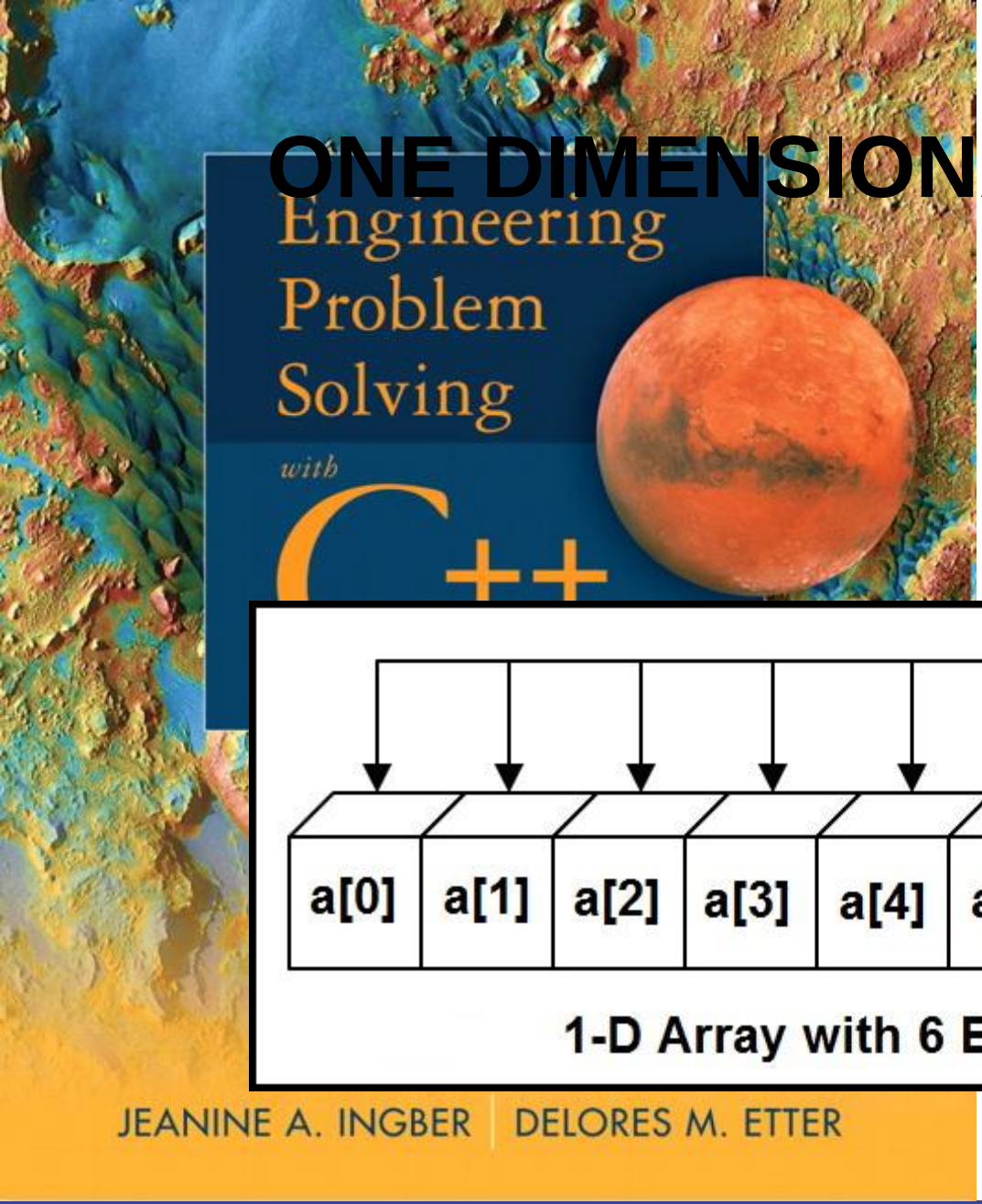
ONE DIMENSIONAL ARRAYS



Objectives THESE ARE THE ONLY SECTIONS TO READ AND STUDY

Be sure to do all “Pratice” problems in the sections below-answer in the back of the book.

1. Arrays
2. Problem Solving Applied: Hurricane Categories
3. Statistical Measurements
4. Problem Solving Applied: Speech Signal Analysis

Be sure to run youtube video(s) in the online syllabus

Objectives

Develop problem solutions in C++ containing:

- One-dimensional arrays and vectors
- Programmer-defined modules for statistical analysis of data
- Custom header files

Arrays Defined

- An ***array*** is a data structure for storing a contiguous block of data.
- All data elements in an array must be of the *same type*.
- Individual elements of the array are specified using the array **name** and an ***offset***. (also called **index**)
- In C++ the offset of the first element is **always 0 (zero).**
- **IN MATH AND MATLAB THE FIRST ELEMENT IS NUMBERED 1!**

Array Definition Syntax

Syntax:

data_type identifier**[** [array_size] **]** [= initialization_list];

//array_size must be an integer constant

Examples

```
int data[5]; //allocates consecutive memory for 5 integer values
char vowels[5] = {'a', 'e', 'i', 'o', 'u'}; //allocates and initializes
double t[100] = {0.0}; //allocates and initialized all values to 0.0
```

Valid References

```
cout << vowels[0];
cout << t[2];
```

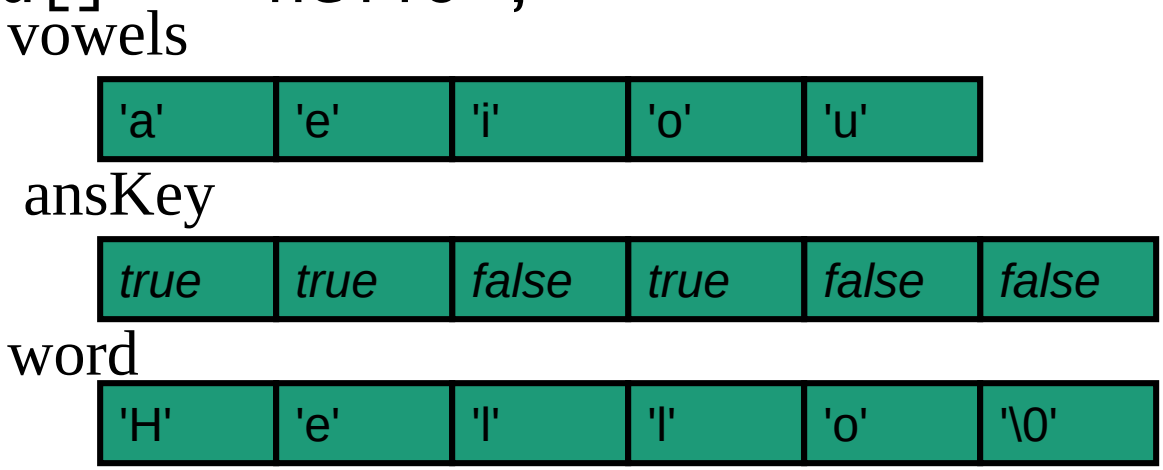
Invalid References

```
cout << vowels[5]; //invalid offset.
cout << t[-1]; //invalid offset
```

Initializing Arrays

Initializing array elements (initialization=>declaration)

```
char vowels[5] = {'a', 'e', 'i', 'o', 'u'};
bool ansKey[] ={true, true, false, true,
false, false};
char word[] = "Hello";
```



Accessing Array Elements

- Offsets are used to access individual elements of an array.
- General format:

  array_identifier[*offset*]
- Example

```
for (int i=0; i<=7; ++i)
  m[i] = double(i) + 0.5;
```
- Integer expressions may be used as offsets.
  - i.e. offset does not need to be constant

The "for loops" are used to specify ONE DIMENSIONAL Arrays (vectors) and Nested "for" loops for TWO DIMENSIONAL ARRAYS (Matrices) IN THE NEXT CHAPTER

E.g. 1 Dimensional array members could be vector components or list of data values for graphing!

```
double s[6]  = { 1,4,7,4.9 ,5.1 ,20.45} ;
```

so members of array ":s"!
s[0]=1 s[2]=7 s[6]=? s[?]=20.45 Class?
like a vector components but here 6 values rather than 3. We say we have N-Dimensional vectors
That is a 1 dimensional array.

```
for (int i=0; i<6; ++i)
        cout << s[i] << " ";     CLASS  OUTPUT?
cout << endl;

sum=0;
for (int i=0; i<6; ++i)
   sum =sum +s[i];            CLASS  OUTPUT?
cout << endl;
cout << sum<< " ";
```

Avoiding Bugs

- C++ does not enforce array bounds! BE EXTRA CAREFUL
  - Invalid references are NOT reported by the compiler.
  - May or may not result in run-time error.
    - E.g. segmentation fault, bus error, etc.
    - More often than not no run-time error occurs.
  - Such errors are often difficult to identify.
  - Results in unpredictable program behavior.

```
/* Program chapter7_1 our first array program*/
/* This program assigns a set of values to a */
/* one-dimensional array then prints a list of the array */
/* offsets and values to standard output. */
#include<iostream> //Required for cout.
#include<iomanip> //Required for setw().
using namespace std;
int main()
{
  //Declare varialbles.
  double t[21]; //The array. // how many members of t?
  int i; //The loop index.
  //Assign 21 value to array t.
  for(i=0; i<21; ++i) //i provides offset (indices)in array "t" and
  }                     //values for each member.note values
       t[i] = i*0.5;
  }
  //Print list of array offsets and values.
  //Print heading.
  cout << "21 values assigned to t"<< endl
      << "Offset Value" << endl;
  //Print list inside for loop.
  for(i=0; i<21; ++i)
  {
    cout << setw(6) << i << setw(10) << t[i] << endl;
  }
  return 0; // note t[i] for each value of t at offset i
}
```

```
/*  7_2 This program reads time and motion values from an input file and assigns the
 values to the arrays time and motion. Input values are printed to standard output TO VERIFY*/
#include<iostream> //required for cout
#include<fstream>  //required for ifstream
using namespace std;
int main()
{    // Declare objects.
  double time[10], motion[10];
  ifstream sensor3("sensor3.dat");   // Check for successful open
 if( !sensor3.fail() ) {
•        for (int k=0; k<10; ++k)  {
•              sensor3 >> time[k] >> motion[k];
•               cout << time[k] << '\t' << motion[k] << endl;
•          }
•     }
•     else
•     {      cout << "Could not open file sensor3.dat..goodbye." << endl;
•     }       return 0;
```

Computation with arrays

- Same as simple variables but the offset number has to be used to reference the value to be used in a mathematical expression.
- see and study in text PROGRAM 7_3
- // from 7_3 you will see the array y used to find the maximum value
- int k;
- for (k=0; k<numberOfValues; ++k)
- {
- lab >> y[k]; // here values are read from input file(object ->"lab") one at a time put in y
- sum += y[k]; //build a sum of each value.
- }
- // Compute average and count values that are greater than the average.
- yAve = sum/numberOfValues;
- for (int k=0; k<numberOfValues; ++k)
- {
- if (y[k] > yAve) // count all values greater than the average
- count++;
- }
- // Print count.
- cout << count << "values greater than the average \n";

Operator Precedence

Precedence	Operator	Associativity
1	() []	Innermost First
2	Unary operators: + - ++ -- ! (**type**)	Right to left
3	* / %	Left to right
4	+ -	Left to right
5	< <= > =>	Left to right
6	== !=	Left to right
7	&&	Left to right
8	||	Left to right
9	? :	Right to left
10	= += -= *= /= %=	Right to left
11	,	Left to right

1 Dimensional Arrays in Functions

- An array identifier, without subscripts(also called offsets or indices), references the starting address(first element) of the array.

- In C++, **arrays are passed by reference (i.e. the starting address is passed, no size information of the array)**

- **That means the function can change the original Array values!**

- For Arrays in C++ the compiler (build) does not know their size.
- So Generally we specify an additional parameter representing the number of elements in the array.

Example of using 1D array in functions note two functions here call an array.Also See and study program 7_4 in text.

```
#include <iostream>
using namespace std;
const int MAXSIZE=20;
void ReadArr(double a[], int& count, istream& in);
int FindMin(const double a[], int count);

int main() {
  double darr[MAXSIZE];
  int cnt=0, position=0;
  ReadArr(darr, cnt, cin);          // function call 1
  position = FindMin(darr, cnt); // function call 2
  cout << "The smallest value in the array is "
       << darr[position] << endl;
}   //functions “ReaArr()” and FindMin() follow
```

Function call ReadArray()

```
// This function inputs values into an array until EOF
// or array limit reached
void ReadArray(double a[], int& count, istream& in) {
  double temp;
  count = 0;
  in >> temp;
  while ( (count < MAXSIZE) && !in.eof() ) {
       a[count] = temp;
       ++count;
       in >> temp;
  }
}
```

Function call FindMin()

```
//This function returns the offset of the smallest
//value in an array

int FindMin(const double a[], int size) {
    int offsetOfMin = 0;
      for (int i=1; i<size; ++i) {
            if (a[i] < a[offsetOfMin] ) {
                  offsetOfMin = i;
          } //end if
    } //end for
      return offsetOfMin;
} //end FindMin
```

HAND IN HW # 15 3pts each 36pts

DO THE FOLLOWING Exam Practice problems at the end chapter 7 1 to 10!

_____11 Given the declaration **char Sentence[15];** and that the data entered from the keyboard is: **The game is today**

What is the value stored into array Sentence by the statement:

cin >> Sentence;

A. The game B. The game i C. The game is tod D. The game is today E. none of these

_____12. Using the following *function prototype* which statement about the argument in the call to the function passed to parameter A is true.

void F(const int A[ ], int Cnt);

a. The argument is modified when changes are made to parameter A in function F.

b. The argument passed to parameter A must always have the same number of elements, every time function F is invoked.

c. Changes can not be made to parameter A in function F.

d. Every element of the argument passed to parameter A must be initialized prior to invoking function F.

***Hurricane Categories

- The Saffir-Simpson scale of hurricane intensities is used to classify hurricanes according to the amount of damage that the storm is likely to generate if it hits a populated area.

Category	Wind Speed (mph)	Storm Surge (feet)	Expected Property Damage
1	74-95	4-5	Minimal
2	96-110	6-8	Moderate
3	111-130	9-12	Extensive
4	131-155	13-18	Extreme
5	155+	18+	Catastrophic

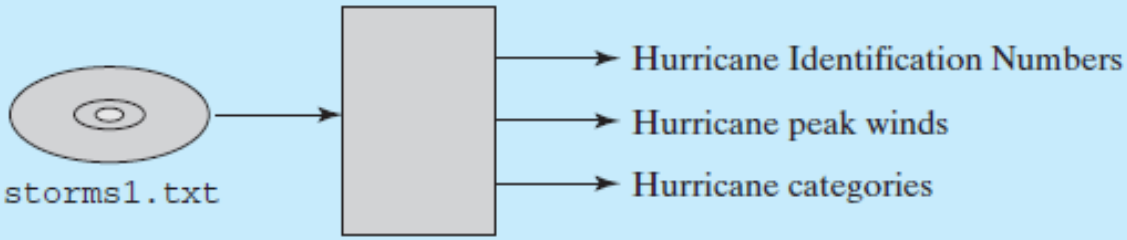
Problem Solving Applied: Hurricane Categories OR STRENGTH CATEGORY

1. PROBLEM STATEMENT

Determine the storms that are hurricanes using a data file of current storm information.

2. INPUT/OUTPUT DESCRIPTION

The I/O diagram shows the data file as the input and the hurricane information as output.



Problem Solving Applied: Hurricane Categories

3. HAND EXAMPLE

Assume that the data file contained the following five sets of data:

Identification	Peak Wind
142	38
153	135
162	59
177	76
181	63

The corresponding output would then be the following report:

Storms that Qualify as Hurricanes

REPORT->>

Identification	Peak Wind (mph)	Category
153*	135	4
177	76	1

Recall that the asterisk identifies the storm with the largest wind speed.

Problem Solving Applied: Hurricane Categories

4. ALGORITHM DEVELOPMENT

We first develop the decomposition outline because it divides the solution into a series of sequential steps. To print the information for storms that are hurricanes, we do not need an array. We could do that as we read through the file, since the hurricane status is dependent only on the wind speed. However, since we also need to use an asterisk to indicate the storm with the peak wind, we will need to store all the information in arrays. After we have determined the maximum, we can then go back through the data printing the hurricane information with the asterisk for the maximum wind speed.

Decomposition Outline

1. Read the storm data into arrays, and determine the maximum wind speed.
2. Compute the intensity categories and print information for storms that are hurricanes, with an asterisk at the maximum.

```
#include<iostream> //Required for cin, cout, cerr. // This program reads
storm values from a data file
#include<fstream> //Required for fin. Program chapter7_5
using namespace std;
double category(double speed); //Function Prototypes.
int main()
{  const int MAX_SIZE = 500;    //Declare and initialize variables and
ARRAYS  id, mph
  int k(0), npts, id[MAX_SIZE];
  double mph [MAX_SIZE], max(0); //Function Prototypes.
  ifstream fin("storm1.txt");  get the data ID and Peak Wind
  if(fin.fail())               // test for file error
  {
    cerr << "Could not open file storm1.txt" << endl;
    exit(1);
  }
  fin >> id[k] >> mph[k]; //Read data and determine maximum mph. arrays
are loaded here
  while(!fin.fail())
  {
    if(mph[k] > max)
    {
        max = mph[k];
    }
    ++k;
    fin >> id[k] >> mph[k];
  }//end while
```

```
 npts = k;
  if(max >= 74) //Print hurricane report.
  {    cout << "Storms that Qualify as Hurricanes \n"
         << "Identification\t Peak Wind(mph)\t Category\n";
  }
  else
  {
    cout << "No hurricanes in the file \n";
  }
  for(k=0; k<npts; ++k)
  {
    if(mph[k] >= 74)
    {
      if(mph[k] == max)
      {
        cout << "\t" << id[k] << "*\t\t" << mph[k] << "\t"
            << category(mph[k]) << endl;      // note call to function
category
      }
      else
      {
        cout << "\t" << id[k] << "\t\t" <<mph[k] << "\t"
            << category(mph[k]) << endl; // note call to function
category
      }
    }//end if k
  }//end for
  fin.close();  return 0;} // end main
```

```
/*-------------------------------------------------------*/
/* This function determines the hurricane intensity */
/* category. */
double category(double speed)
{
  //Declare variables.
  int intensity(1);
  //Determine category.
  if(speed >= 155)
  {
    intensity=5;
  }
  else if(speed >= 131)
  {
    intensity = 4;
  }
  else if(speed >= 111)
  {
    intensity = 3;
  }
  else if(speed >= 96)
  {
    intensity = 2;
  }
    return intensity;
}
/*-------------------------------------------------------*/
```

```
#include<iostream> //Required for cin, cout, cerr. // PROGRAM CHAP 7_5
#include<fstream> //Required for fin.              // This program reads storm values from a data file
using namespace std;
double category(double speed); //Function Prototypes.
int main()
{ const int MAX_SIZE = 500; //Declare and initialize variables.
  int k(0), npts, id[MAX_SIZE];
  double mph [MAX_SIZE], max(0);
  ifstream fin("storm1.txt");
  if(fin.fail())
  {    cerr << "Could not open file storm1.txt" << endl;
    exit(1);
  }
  fin >> id[k] >> mph[k]; //Read data and determine maximum mph.
  while(!fin.fail()) {    // { PUT HERE TO GET FULL CODE ON ONE SLIDE
    if(mph[k] > max)
    {  max = mph[k];
    }
    ++k;
    fin >> id[k] >> mph[k];
  }//end while
  npts = k; //Print hurricane report.
  if(max >= 74)
  {    cout << "Storms that Qualify as Hurricanes \n"
         << "Identification\t Peak Wind(mph)\t Category\n";
  }
  else
  {    cout << "No hurricanes in the file \n";
  }
  for(k=0; k<npts; ++k)
   {    if(mph[k] >= 74)
    {    if(mph[k] == max)
      {        cout << "\t" << id[k] << "*\t\t" << mph[k] << "\t"<< category(mph[k]) << endl;
    }
      else
      {       cout << "\t" << id[k] << "\t\t" <<mph[k] << "\t"
            << category(mph[k]) << endl;
      }
    }//end if k
  }//end for
  fin.close();
  return 0;
 } // end main
  double category(double speed) //This function determines the hurricane intensity  category
  {   int intensity(1); //Declare variables.
     if(speed >= 155) //Determine category.
  {    intensity=5;
  }
  else if(speed >= 131)  {    intensity = 4;   // { intensity=4; put on this line for fitting code on slide
  }
  else if(speed >= 111)
  {    intensity = 3;
  }    else if(speed >= 96)
  {    intensity = 2;
  }
    return intensity;  }// end function
```

HAND IN LABORATORY TASK: LAB #26

1 Run the last program (**USE SOURCE CODE ONLINE** not from theses notes) after you create the ID and the wind info file **DO NOT HAND IN.**

2. Hand in the following modification to the last program and a modified original Data file

a. Be sure to input filename at the key board and not included in the input stream As the text did. Ie. no " name" used

b. Modify the DATA file to include AT LEAST two OF EACH hurricane TYPES as well as 3 non hurricane storms with numbers as ID

c. Your input data file will include a name for hurricane IDs and number for nonhurricane storms (like ALICE, BOB and 125) , Peak wind speed, storm surge Number that makes sense(1-3 for non-hurricane storms)

Then produce the
Output report.

The program should generate a new data REPORT file containing

i. the final number of storms and number of hurricanes

ii. For each hurricane it finds the report should include information on it (ID NAME, PEAK WIND SPEED, CATERGORY, STORM SURGE NUMBER IN FEET CATEGORY, PROPERTY DAMAGE EXPECTED) in a table

Be sure to attach THE Input file, Report file and modified program

Skip to CHAPTER 8

Statistical Measurements DONE AFTER IF TIME PERMITS

- Data collected from engineering experiments often have statistical properties that change from one data set to another.
  - Examples; sin(60) is always the same, but the miles per gallon a vehicle consumes varies depending on type of driving, temperature, etc…
  - Experimental data over time (e.g. Temperature,)can be variable so Engineers are interested in the Maximum and Minimum values as well as how much the data spreads. In labs measurement of data is not very precise at times so what is needed is a way to express how well the data matches what one is looking.
  - Statistical Measurements that follow are used regularly be sure you understand these.

BASIC Statistical Analyses

- Basic statistical analyses often involve computing properties of a data set such as:
  - Minimum value
  - Maximum value
  - Mean (average) value
  - Median (middle) value
  - Variance
  - Standard deviation

  - So normally, when we get data and enter it into an array
  - We want the above values . we extract the above important Statistical analyses values in Engineering and Science by developing functions that give us our answers.

  We now develop the different functions that treat the array to get each of the above Values, as well as, understand what they mean for a given set of data. Some are obvious, like Minimum value, Maximum value and Mean the others have been developed to express the validity of data and will be explain as we go along.

Recall finding the minimum in an array

```
//This function returns the offset of the smallest
//value in an array  so we can find the value

int FindMin(const double a[], int size) {
  int offsetOfMin = 0;
  for (int i=1; i<size; ++i) {
    if (a[i] < a[offsetOfMin] ) {
      offsetOfMin = i;
    } //end if
  } //end for
  return offsetOfMin;
} //end FindMin
```

How can modify the last function to get the following?
Answers are new functions that you should study how they accomplish their goal.

- So now we will, given the previous algorithm for finding the offset of the minimum element of a data set, show how we can you modify it to find:
  - The offset of maximum element?
  - The value of the minimum element?
  - The value of the maximum element?

Offset of Maximum

```
//This function returns the offset of the smallest
//value in an array

int FindMin(const double a[], int size) {
  int offsetOfMin = 0;
  for (int i=1; i<size; ++i) {
    if (a[i] > a[offsetOfMin] ) {
      offsetOfMin = i;
    } //end if
  } //end for
  return offsetOfMin;
} //end FindMin
```

Simply changing the relational operator in the FindMin algorithm with yield an algorithm that finds the offset of the maximum value!
Note that identifiers are not meaningful to the compiler… identifiers are used only to uniquely identify (and type) variables.

Min and Max Offsets

```
//This function returns the
//offset of the smallest
//value in an array

int FindMin(const double a[],
            int size) {
  int offsetOfMin = 0;
  for (int i=1; i<size; ++i) {
    if (a[i] < a[offsetOfMin] ) {
      offsetOfMin = i;
    } //end if
  } //end for
  return offsetOfMin;
} //end FindMin
```

```
//This function returns the
//offset of the largest
//value in an array

int FindMax(const double a[],
            int size) {
  int offsetOfMax = 0;
  for (int i=1; i<size; ++i) {
    if (a[i] > a[offsetOfMax] ) {
      offsetOfMax = i;
    } //end if
  } //end for
  return offsetOfMax;
} //end FindMax
```

Min and Max Values

```
//This function returns the
//value of the smallest
//element in an array

double FindMinVal(const double a[],
                  int size) {
  int offsetOfMin = 0;
  for (int i=1; i<size; ++i) {
    if (a[i] < a[offsetOfMin] ) {
      offsetOfMin = i;
    } //end if
  } //end for
  return a[offsetOfMin];
} //end FindMinVal
```

```
//This function returns the
//value of the largest
//element in an array

double FindMaxVal(const double a[],
                  int size) {
  int offsetOfMax = 0;
  for (int i=1; i<size; ++i) {
    if (a[i] > a[offsetOfMax] ) {
      offsetOfMax = i;
    } //end if
  } //end for
  return a[offsetOfMax];
} //end FindMaxVal
```

Computing the Mean (ALSO CALLED THE AVERAGE)

```
//This function computes the mean of a dataset
//contained in an array.
// Mean = Sum(all array elements)/size
double Mean(const double a[], int size) {
  double sum(0);

  //compute the sum
  for (int i=0; i<size; ++i)
    sum += a[i];

  //return the mean
  return sum/size;
} //end Mean
```

Median value of an array of data

Given a set of data we are sometimes interested in the middle value after we sort the values in order.

so given an odd number of values, like 8, 4, 3,9,12

We sort them to 3,4,8,9,12

There are simple routines to sort data that is unsorted

See section 7.5 if interested.

Then the Median is 8 in this simple case

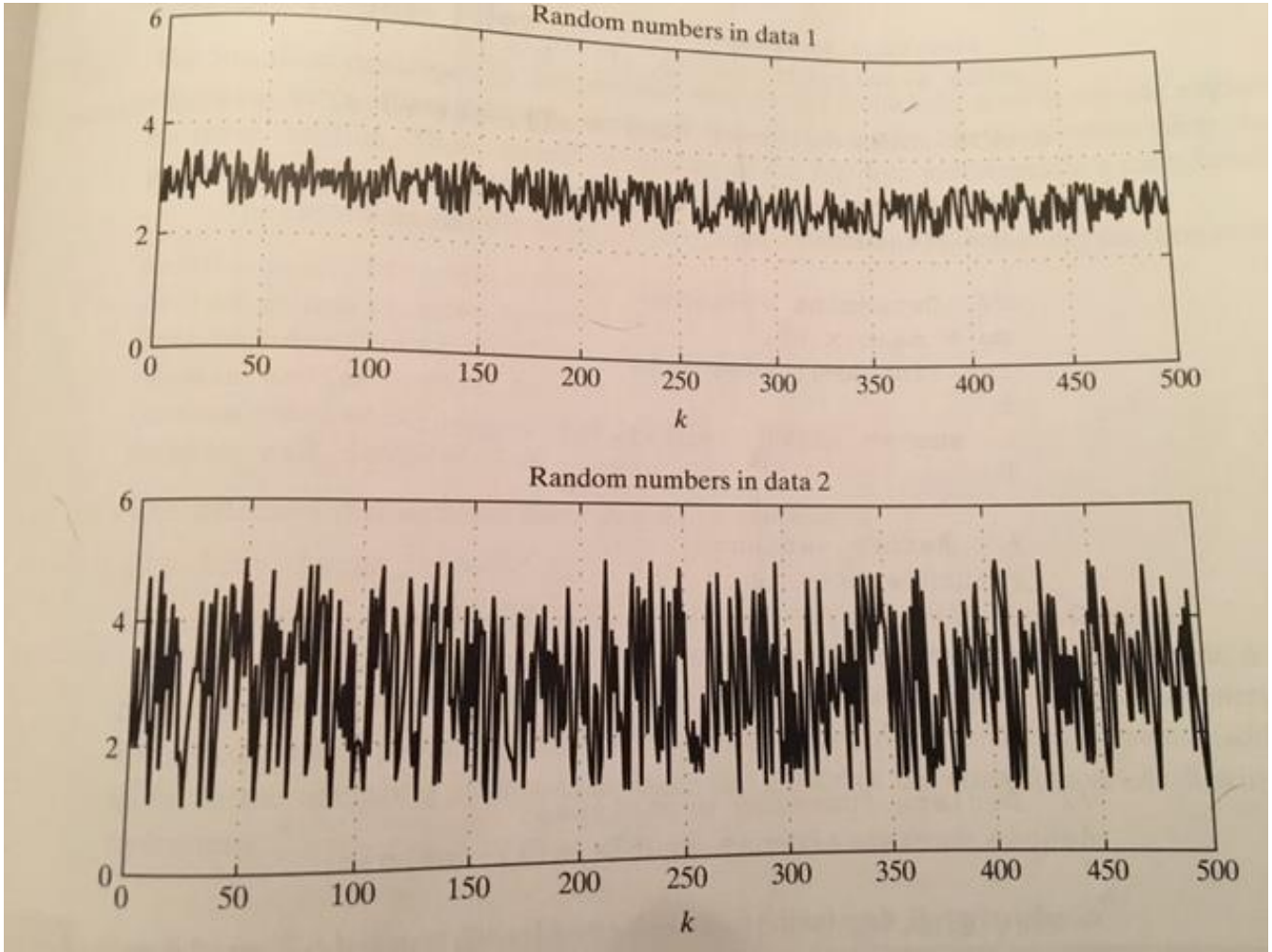
If we have an even number like 3,4,8,9,12,20 then the two middle values are averaged (8+9)/2

Median is 8.5

We will not explore this further.

Check the text for more details.

The variance and standard deviation give us a marker for the spread or dispersion or the amount of variation in a data set. Its easy to see how the following differ but The variance and standard deviation gives us a numerical values to show the spread. Here the mean is the same(~3) but the variation from the mean are different. We build the variance from the mean, summing up all the squares of the differences of each value from the mean. Dividing by one less than the number of points Taking the square root of the variance gives us the standard deviation of our data set.



Variance and Standard deviation mathematically

- Commonly used in Engineering and science is the statistical measurements for a data set is variance.

- Variance = $\sigma^2 = \frac{\sum_{k=0}^{n-1}(x_k - \mu)^2}{n-1}$

where σ^2 is the variance and μ is the mean of all values x_0, x_1, … , x_{n-1}.

The standard deviation is just σ. Is the Square root of the variance which shows the dispersion of the Data set.

For a deeper discussion see the text and

https://en.wikipedia.org/wiki/Standard_deviation

Computing the Variance
(once variance is returned we can just take the square root of it for the standard deviation

```
//This function computes the variance of a
dataset
//contained in an array.

double variance(const double a[], int n) {
  double sum(0), mu = mean(a,n);

  //compute the sum
  for (int i=0; i<n; ++i)
    sum += (a[i]-mu)*(a[i]-mu);

  //return the mean
  return sum/(n-1);
} //end variance
```

This file called **stat_lib.cpp** siting in the same place as the original program and the header file for follow has the functions mean,variance and standard deviation.

```
/* This function returns the average or mean value of an array with n elements. */
double mean(const double x[], int n)
 {
.   double sum(0); // Declare and initialize objects
    for (int k=0; k<n; ++k) // Determine mean value.
        {
            sum += x[k];
        }
.        return sum/n; // Return mean value
 } // end mean
/* This function returns the variance of an array with n elements. */
double variance(const double x[], int n) // Function header.
 {
    double sum(0), mu; // Declare objects.
      mu = mean(x,n); // Determine variance.
      for (int k=0; k<n; ++k)
         {
            sum += (x[k] - mu)*(x[k] - mu);
         }
       return sum/(n-1); // Return variance
} // end variance
/* This function returns the standard deviation of an array with n elements. */
    double std_dev(const double x[], int n) // Function header.
    {
       return sqrt(variance(x,n)); // Return standard deviation.
     }
```

Custom Header Files (.h)

Frequently used classes and functions may be stored in separate files to easily include them in other programs. Recall in VS we have a choice to create a cpp or h file!

The last slide had the file "stat_lib.cpp" containing functions
Now we can create a "name.h" a header file that contains function prototypes linking to a cpp (stat_lib.cpp) containing the actual functions which now **do not have to be included in our main program file**. we can use this approach for other applications when these functions are needed so we say we Define code 'libraries' of functions

For example: a Header file named "library_name.h"

Has the function prototypes and refers to the file called the

Implementation file named "library_name.cpp" con

To use the library, #include "library_name.h" and be sure the linker can find the implementation file by placing it with your main.cpp source file like the data files we used.

Problem Solving Applied: Speech Signal Analysis

We will use a header file "stat_lib.h" in this problem

To call the mean, variance and standard deviation of the data set to be presented. Others like the minimum and maximum could be added in the program but not needed for this problem to come. i.e. stat_lib.h file online contains the following **and more**!

```
//This header file defines commonly used statistical
functions
double Mean(const double [], int);
double Variance(const double [], int);
double std_dev(const double [], int);
```

We have to construct the file stat_lib.cpp containing the functions and place it with our main program and any data files needed. As follows!

The actual header file at "source doc contains a lot more as follows.

The actual header file containing the function prototypes in the source code online follows starting with #ifndef…..
The header file is called from the program with the statement
#include “stat_lib.h”

(we are only using it now for mean(),variance() and std-dev() function calls)

```
#ifndef STAT_LIB
#define STAT_LIB
#include<iostream>
#include<cmath>
double maxval(const double x[], int n);
 double minval(const double x[], int n);
double mean(const double x[], int n);
double median(const double x[], int n);
double variance(const double x[], int n);
 double std_dev(const double x[], int n);
#endif
```

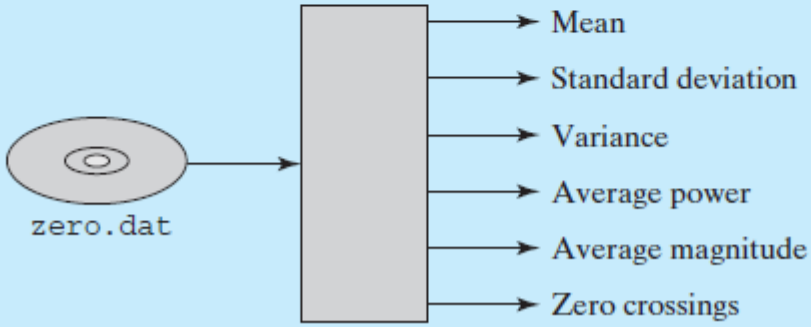
Problem Solving Applied: Speech Signal Analysis

1. PROBLEM STATEMENT

Compute the following statistical measurements for a speech utterance: mean, standard deviation, variance, average power, average magnitude, and number of zero crossings.

2. INPUT/OUTPUT DESCRIPTION

The I/O diagram shows the data file as the input and the statistical measurements as output.



3. HAND EXAMPLE

For a hand example, assume that the file contains the following values:

```
2.5   8.2   -1.1   -0.2   1.5
```

Using a calculator, we can compute the following values:

$$\text{Mean} = \mu = \frac{(2.5 + 8.2 - 1.1 - 0.2 + 1.5)}{5}$$
$$= 2.18;$$

$$\text{Variance} = [(2.5 - \mu)^2 + (8.2 - \mu)^2 + (-1.1 - \mu)^2 + (-0.2 - \mu)^2 + (1.5 - \mu)^2]/4$$
$$= 13.307;$$

$$\text{Standard deviation} = \sqrt{13.307}$$
$$= 3.648;$$

$$\text{Average power} = \frac{[(2.5)^2 + (8.2)^2 + (-1.1)^2 + (-0.2)^2 + (1.5)^2]}{5}$$
$$= 15.398;$$

$$\text{Average magnitude} = \frac{(|2.5| + |8.2| + |-1.1| + |-0.2| + |1.5|)}{5}$$
$$= 2.7;$$

$$\text{Number of zero crossings} = 2.$$

Problem Solving Applied: Speech Signal Analysis

4. ALGORITHM DEVELOPMENT

We first develop the decomposition outline, because it divides the solution into a series of sequential steps.

Decomposition Outline

1. Read the speech signal into an array.
2. Compute and print statistical measurements.

Step 1 involves reading the data file and determining the number of data points. Step 2 involves computing and printing the statistical k measurements, using the functions already developed when possible. The refinement in pseudocode for the `main` function and for the additional statistical functions needed is shown next; the structure chart was shown in Figure 5.1 to illustrate an example of a main function that references several programmer-defined functions.

Solution of this problem is multifaceted
code is online Chapter7_6.cpp
but so also is the stat_lib.h
and the stat_lib.cpp I created for you to run this program as illustrated above!
three data files are also there based on a recording of voices saying the word "two" ie two1.dat, two2.dat, two3.dat all have to be loaded in the same directory as main 7_6.cpp for the linker step can find everything.

```
/* Program chapter7_7  This gram computes a set of statistical
measurements from a speech signal.                              */
#include<iostream> //Required for cin, cout.
#include<fstream> //Required for ifstream.
#include<string> //Required for string
#include<cmath> //Required for abs()
#include "stat_lib.h" //Required for mean(),variance(), std-dev()
using namespace std;
// Declare additional function prototypes and define constants.
double ave_power(double x[], int n);
double ave_magn(double x[], int n);
int crossings(double x[], int n);
int main()
{ const int MAXIMUM = 2500; // Declare objects.
  int npts(0);  double speech[MAXIMUM];
  string filename;
  ifstream file_1;
  // Prompt user for file name and  open file.
  cout << "Enter filename ";
  cin >> filename;
  file_1.open(filename.c_str());
  if( file_1.fail() )
  {     cout << "error opening file " << filename  << endl;
    return 0;
  }
```

```
// Read information from a data file. *
  while (npts <= MAXIMUM-1 && file_1 >> speech[npts])
  {
    npts++;
  } //end while
  // Compute and print statistics.
  cout << "Digit Statistics \n";
  cout << "\tmean: " << mean(speech,npts) << endl;
  cout << "\tstandard deviation: "
      << std_dev(speech,npts) << endl;
  cout << "\tvariance: " << variance(speech,npts)
      << endl;
  cout << "\taverage power: " << ave_power(speech,npts)
      << endl;
  cout << "\taverage magnitude: "
      << ave_magn(speech,npts) << endl;
  cout << "\tzero crossings: " << crossings(speech,npts)
      << endl;
  // Close file and exit program.
  file_1.close();
  return 0;
}
// additional functions needed follow in the code
```

```
/* This function returns the average power  of an array x with n elements. */
double ave_power(double x[], int n)
{  double sum(0); // Declare and initialize objects.
   for (int k=0; k<=n-1; ++k) // Determine average power.
  {
      sum += x[k]*x[k];
  }
     return sum/n; // Return average power.
}
/* This function returns the average magnitude of an array x with n values. */
double ave_magn(double x[], int n)
{
  double sum(0); // Declare and initialize objects.
  for (int k=0; k<=n-1; ++k) // Determine average magnitude.
  {
    sum += abs(x[k]);
  }
  return sum/n; // Return average magnitude.
}
/* This function returns a count of the number of zero crossings in an array x(n). */
int crossings(double x[], int n)
{
   int count(0); // Declare and initialize objects.

    for (int k=0; k<=n-2; ++k) // Determine number of zero crossings.
  {
    if (x[k]*x[k+1] < 0)
    count++;
  }
  // Return number of zero crossings.
  return count;
}
```

```
/*-------------------------------------------------------------*/
/* Program chapter7_7                                          */
/*                                                             */
/* This program computes a set of statistical                  */
/* measurements from a speech signal.                          */
#include<iostream> //Required for cin, cout.
#include<fstream> //Required for ifstream.
#include<string> //Required for string
#include<cmath> //Required for abs()
#include "stat_lib.h" //Required for mean(), variance(), std-dev()
using namespace std;
// Declare function prototypes and define constants.
double ave_power(double x[], int n);
double ave_magn(double x[], int n);
int crossings(double x[], int n);
int main()
{
  // Declare objects.
  const int MAXIMUM = 2500;
  int npts(0);
  double speech[MAXIMUM];
  string filename;
  ifstream file_1;
  // Prompt user for file name and
  // open file.
  cout << "Enter filename ";
  cin >> filename;
  file_1.open(filename.c_str());
  if( file_1.fail() )
  {
    cout << "error opening file " << filename
         << endl;
    return 0;
  }
  // Read information from a data file. *
  while (npts <= MAXIMUM-1 && file_1 >> speech[npts])
  {
    npts++;
  } //end while
  // Compute and print statistics.
  cout << "Digit Statistics \n";
  cout << "\tmean: " << mean(speech,npts) << endl;
  cout << "\tstandard deviation: "
      << std_dev(speech,npts) << endl;
  cout << "\tvariance: " << variance(speech,npts)
      << endl;
  cout << "\taverage power: " << ave_power(speech,npts)
      << endl;
  cout << "\taverage magnitude: "
      << ave_magn(speech,npts) << endl;
  cout << "\tzero crossings: " << crossings(speech,npts)
      << endl;
  // Close file and exit program.
  file_1.close();
  return 0;
}
```

HAND IN LABORATORY TASK: LAB #27

some signal processing of voice files

ALL FILES ONLINE

1. . Load the source chapter7_6.cpp via the usual choice of cpp or .h
(note what directory this file will go into)
2. Then load the cpp file containing a library of functions stat_lib.cpp **in MSVS**
By picking cpp in the choice as in 1. and copying content from online file to MSVS
3. And also load the ".h" file. stat_lib.h from the choice of cpp and .h same way!
NOTE:all three of the latter files have to be setup in MSVS and be sure to right click
In the explorer window change names to be sure names called stat_lib.h & stat_lib.cpp
your source.cpp now "Build" main "source" program which will "build" all at once.
4. copy the three voice data files from the source online via two1.txt, two2.txt and
two3.txt in to the directory where The main source file is. Use notepad to get data &save
Build the cpp files. (Use **select all** when copying these large data files)
5. Run the program for each of the Data files copying the output as comments
As usual. Compare the output in your own words of these three voice
Data files and comment on the difficulty of designing speech-recognition
6. Hand in copies of the chapt7_6.cpp file with the outputs of the three data files
Attached as comments as usual.
7. Your hand written comparison should be attached
STAPLE ALL! DO NOT COPY TO HAND IN THE DATA FILES