

Computer Programming 143 – Lecture 16

Arrays III

Electrical and Electronic Engineering Department
University of Stellenbosch

Prof Johan du Preez
Mr Callen Fisher
Dr Willem Jordaan
Dr Hannes Pretorius
Mr Willem Smit



Copyright

Copyright © 2020 Stellenbosch University
All rights reserved

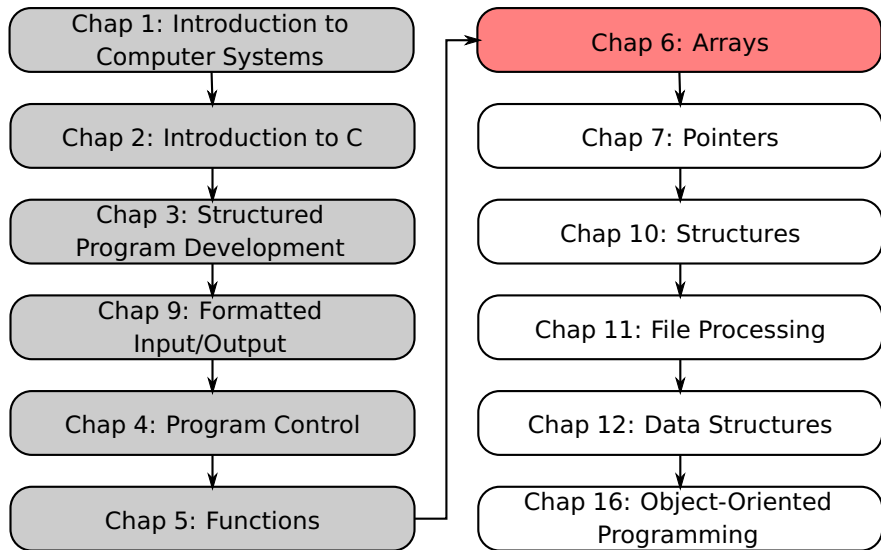
Disclaimer

This content is provided without warranty or representation of any kind. The use of the content is entirely at your own risk and Stellenbosch University (SU) will have no liability directly or indirectly as a result of this content.

The content must not be assumed to provide complete coverage of the particular study material. Content may be removed or changed without notice.

The video is of a recording with very limited post-recording editing. The video is intended for use only by SU students enrolled in the particular module.

Module Overview



Lecture Overview

1 Sorting Arrays (6.8)

2 Case Study: Computing Mean, Median and Mode (6.9)

6.8 Sorting Arrays

Sorting data

- Important computing application
- Virtually every organization must sort some data

Bubble sort

- Several passes through the array
- Successive pairs of elements are compared
 - If increasing order (or identical), no change
 - If decreasing order, elements are exchanged
- Repeat

6.8 Bubble Sort

c[0]	13	pass = 1 i = 0
c[1]	4	
c[2]	1	
c[3]	12	
c[4]	3	
c[5]	7	
c[6]	11	

6.8 Bubble Sort

c[0]	4	pass = 1 i = 0
c[1]	13	
c[2]	1	
c[3]	12	
c[4]	3	
c[5]	7	
c[6]	11	

6.8 Bubble Sort

c[0]	4	pass = 1 i = 1
c[1]	13	
c[2]	1	
c[3]	12	
c[4]	3	
c[5]	7	
c[6]	11	

6.8 Bubble Sort

c[0]	4	pass = 1 i = 1
c[1]	1	
c[2]	13	
c[3]	12	
c[4]	3	
c[5]	7	
c[6]	11	

6.8 Bubble Sort

c[0]	4	pass = 1 i = 2
c[1]	1	
c[2]	13	
c[3]	12	
c[4]	3	
c[5]	7	
c[6]	11	

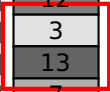
6.8 Bubble Sort

c[0]	4	pass = 1 i = 2
c[1]	1	
c[2]	12	
c[3]	13	
c[4]	3	
c[5]	7	
c[6]	11	

6.8 Bubble Sort

c[0]	4	pass = 1 i = 3
c[1]	1	
c[2]	12	
c[3]	13	
c[4]	3	
c[5]	7	
c[6]	11	

6.8 Bubble Sort

c[0]	4	pass = 1 i = 3
c[1]	1	
c[2]	12	
c[3]	3	
c[4]	13	
c[5]	7	
c[6]	11	

6.8 Bubble Sort

c[0]	4	pass = 1 i = 4
c[1]	1	
c[2]	12	
c[3]	3	
c[4]	13	
c[5]	7	
c[6]	11	

6.8 Bubble Sort

c[0]	4	pass = 1 i = 4
c[1]	1	
c[2]	12	
c[3]	3	
c[4]	7	
c[5]	13	
c[6]	11	

6.8 Bubble Sort

c[0]	4	pass = 1 i = 5
c[1]	1	
c[2]	12	
c[3]	3	
c[4]	7	
c[5]	13	
c[6]	11	

6.8 Bubble Sort

c[0]	4	pass = 1 i = 5
c[1]	1	
c[2]	12	
c[3]	3	
c[4]	7	
c[5]	11	
c[6]	13	

6.8 Bubble Sort

c[0]	4	pass = 2 i = 0
c[1]	1	
c[2]	12	
c[3]	3	
c[4]	7	
c[5]	11	
c[6]	13	

6.8 Bubble Sort

c[0]	1	pass = 2 i = 0
c[1]	4	
c[2]	12	
c[3]	3	
c[4]	7	
c[5]	11	
c[6]	13	

6.8 Bubble Sort

c[0]	1	pass = 2 i = 1
c[1]	4	
c[2]	12	
c[3]	3	
c[4]	7	
c[5]	11	
c[6]	13	

6.8 Bubble Sort

c[0]	1	pass = 2 i = 2
c[1]	4	
c[2]	12	
c[3]	3	
c[4]	7	
c[5]	11	
c[6]	13	

6.8 Bubble Sort

c[0]	1	pass = 2 i = 2
c[1]	4	
c[2]	3	
c[3]	12	
c[4]	7	
c[5]	11	
c[6]	13	

6.8 Bubble Sort

c[0]	1	pass = 2 i = 3
c[1]	4	
c[2]	3	
c[3]	12	
c[4]	7	
c[5]	11	
c[6]	13	

6.8 Bubble Sort

c[0]	1	pass = 2 i = 3
c[1]	4	
c[2]	3	
c[3]	7	
c[4]	12	
c[5]	11	
c[6]	13	

6.8 Bubble Sort

c[0]	1	pass = 2 i = 4
c[1]	4	
c[2]	3	
c[3]	7	
c[4]	12	
c[5]	11	
c[6]	13	

6.8 Bubble Sort

c[0]	1	pass = 2 i = 4
c[1]	4	
c[2]	3	
c[3]	7	
c[4]	11	
c[5]	12	
c[6]	13	

6.8 Bubble Sort

c[0]	1	pass = 2 i = 5
c[1]	4	
c[2]	3	
c[3]	7	
c[4]	11	
c[5]	12	
c[6]	13	

6.8 Bubble Sort

c[0]	1	pass = 3 i = 0
c[1]	4	
c[2]	3	
c[3]	7	
c[4]	11	
c[5]	12	
c[6]	13	

6.8 Bubble Sort

c[0]	1	pass = 3 i = 1
c[1]	4	
c[2]	3	
c[3]	7	
c[4]	11	
c[5]	12	
c[6]	13	

6.8 Bubble Sort

c[0]	1	pass = 3 i = 1
c[1]	3	
c[2]	4	
c[3]	7	
c[4]	11	
c[5]	12	
c[6]	13	

6.8 Bubble Sort

c[0]	1	pass = 3 i = 2
c[1]	3	
c[2]	4	
c[3]	7	
c[4]	11	
c[5]	12	
c[6]	13	

6.8 Bubble Sort

c[0]	1	pass = 3 i = 3
c[1]	3	
c[2]	4	
c[3]	7	
c[4]	11	
c[5]	12	
c[6]	13	

6.8 Bubble Sort

c[0]	1	pass = 3 i = 4
c[1]	3	
c[2]	4	
c[3]	7	
c[4]	11	
c[5]	12	
c[6]	13	

6.8 Bubble Sort

c[0]	1	pass = 3 i = 5
c[1]	3	
c[2]	4	
c[3]	7	
c[4]	11	
c[5]	12	
c[6]	13	

6.8 Bubble Sort: Example

Problem

- Write a C function to perform bubble sorting of an array

Pseudocode

```
begin bubbleSort(array, arraySize)  
  for pass from 1 to arraySize  
    for index from 0 to (arraySize - 1)  
      if array[index] > array[index + 1]  
        swap values in array[index] and array[index + 1]  
end
```

6.8 Bubble Sort: Example

```
#include <stdio.h>
#define SIZE 10
void bubbleSort( int b[], int size );// function prototype
int main()
{
    int a[ SIZE ] = { 2, 6, 4, 8, 10, 12, 89, 68, 45, 37 };//initialize a
    int j; //array index

    printf( "Data items in original order\n" );
    for (j = 0; j < SIZE; j++) {
        printf("%d ",a[j]);
    }
    bubbleSort( a, SIZE );//sorting the array

    printf( "\nData items after sort\n" );
    for (j = 0; j < SIZE; j++) {
        printf("%d ",a[j]);
    }
    return 0;
}
```

```

/* bubble sort */
void bubbleSort( int b[], int size )
{
    int i;    /* inner counter */
    int pass; /* outer counter */
    int hold; /* temporary location used to swap elements */

    /* loop to control number of passes */
    for ( pass = 1; pass < size; pass++ ) {
        /* loop to control number of comparisons per pass */
        for ( i = 0; i < size - 1; i++ ) {
            /* compare adjacent elements and swap them if first
               element is greater than second element */
            if ( b[ i ] > b[ i + 1 ] ) {
                hold = b[ i ];
                b[ i ] = b[ i + 1 ];
                b[ i + 1 ] = hold;
            }
        }
    }
}

```

6.9 Case Study: Computing Mean, Median and Mode

Computing mean, median and mode

- Given an unsorted array of integers
- The **mean** is the average value of the array
 - Determined by dividing the sum of the elements by the number of elements
- The **median** is the “middle value” of the array
 - Determined by sorting the array and picking the element at the middle index
- The **mode** is the value that occurs most frequently
 - Determined by counting the number of times each value occurs
 - The result is also stored in an array
 - The mode is given by the maximum value of this array of results

6.9 Case Study: Computing Mean, Median and Mode

Problem

- Ninety nine responses were collected in a survey. Each response is a number between 1 and 9. Place the responses in an array and summarize the results by giving the mean, median and mode of the survey data.

6.9 Case Study: Computing Mean, Median and Mode

Pseudocode

```
begin mean(array)  
    initialize total to 0  
    for index from 0 to arraySize  
        add array[index] to total  
    calculate mean as total/arraySize  
end  
  
begin median (array)  
    display unsorted array  
    use function bubbleSort to sort array  
    display sorted array  
    display median element  
end
```


6.9 Case Study: Computing Mean, Median and Mode

Pseudocode

begin mode(array)

initialize variables largest, modeValue and freqArray elements to 0

count the number of occurrences of each number → freqArray

for each occurrence rating from 1 to 9

if freqArray[rating] > largest

assign freqArray[rating] to largest

set modeValue to rating

Display the modeValue and it number of occurrences

end

begin main()

initialize response array

call function mean[response array]

call function median[response array]

call function mode[response array]

end

6.9 Case Study: Computing Mean, Median and Mode

Refer to Fig. 6.16 in Deitel & Deitel for the full program listing

Today

Arrays III

- Sorting arrays
- Case study: mean, median and mode

Next lecture

Arrays IV

- Multidimensional Arrays

Homework

- 1 Study Sections 6.8-6.9 in Deitel & Deitel
- 2 Do Self Review Exercise 6.1(d) in Deitel & Deitel
- 3 Do Exercises 6.6(g), 6.8(f), 6.15 in Deitel & Deitel