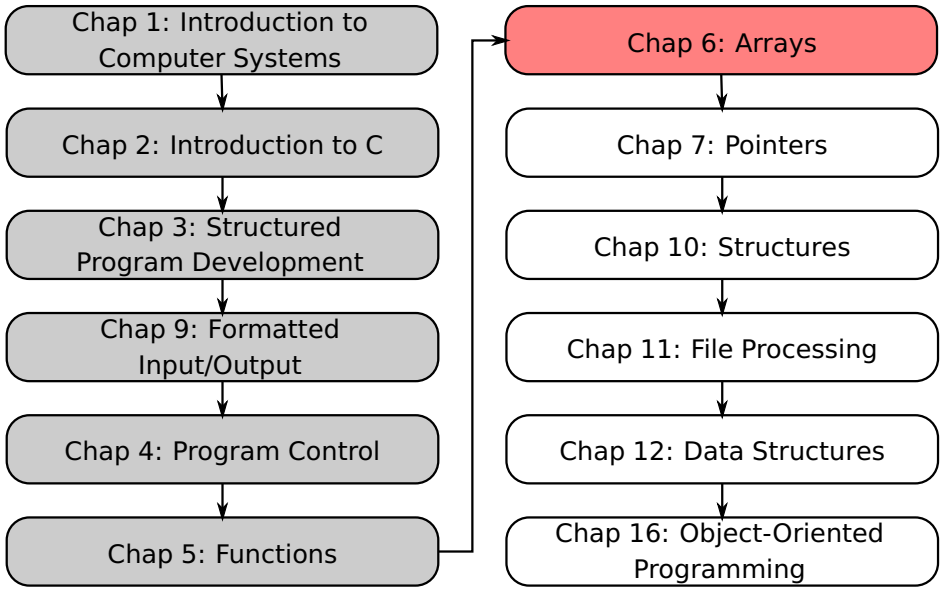
Computer Programming 143 – Lecture 16
Arrays III

Electrical and Electronic Engineering Department
University of Stellenbosch

Corné van Daalen
Thinus Booysen
Willie Smit
Willem Jordaan

Module Overview



Lecture Overview

6.8 Sorting Arrays

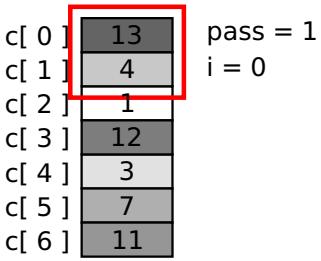
Sorting data

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

Bubble sort

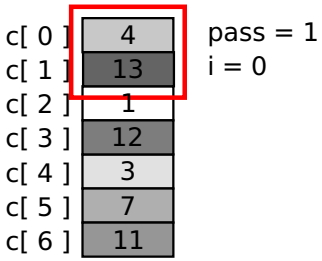
- Several passes through the array
- Successive pairs of elements are compared
  - If increasing order (or identical), no change
  - If decreasing order, elements are exchanged
- Repeat

6.8 Bubble Sort



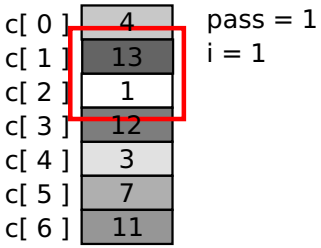
pass = 1
i = 0

6.8 Bubble Sort



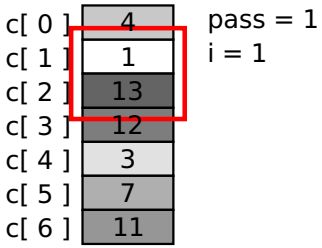
pass = 1
i = 0

6.8 Bubble Sort



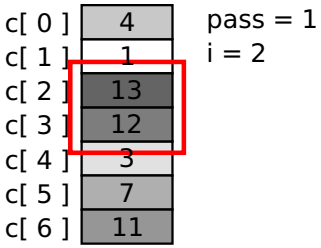
pass = 1
i = 1

6.8 Bubble Sort



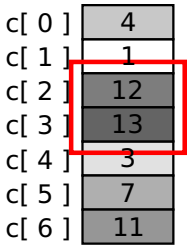
pass = 1
i = 1

6.8 Bubble Sort



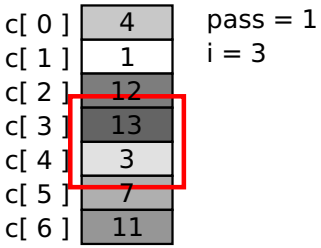
pass = 1
i = 2

6.8 Bubble Sort



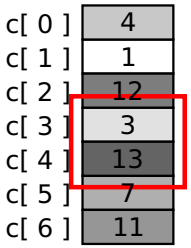
pass = 1
i = 2

6.8 Bubble Sort



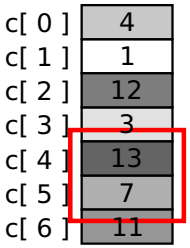
pass = 1
i = 3

6.8 Bubble Sort



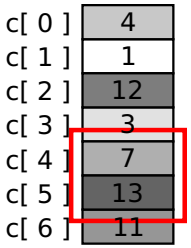
pass = 1
i = 3

6.8 Bubble Sort



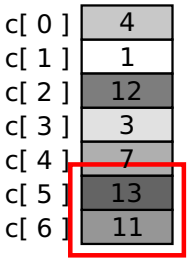
pass = 1
i = 4

6.8 Bubble Sort



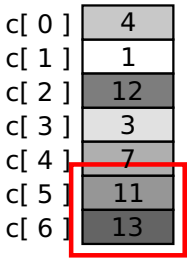
pass = 1
i = 4

6.8 Bubble Sort



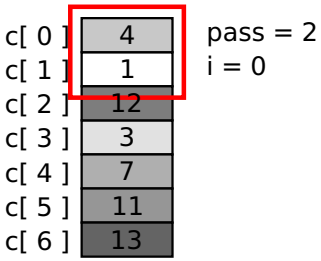
pass = 1
i = 5

6.8 Bubble Sort



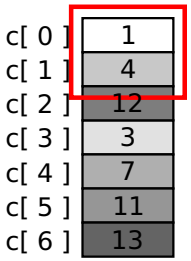
pass = 1
i = 5

6.8 Bubble Sort



pass = 2
i = 0

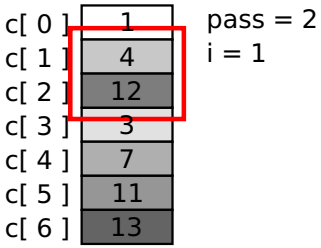
6.8 Bubble Sort



pass = 2
i = 0

(E&E Eng. Dept. US) CP143 Lecture 16 23 August 2016 5 / 15

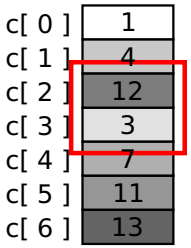
6.8 Bubble Sort



pass = 2
i = 1

(E&E Eng. Dept. US) CP143 Lecture 16 23 August 2016 5 / 15

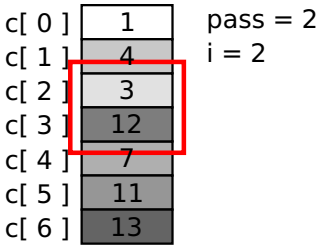
6.8 Bubble Sort



pass = 2
i = 2

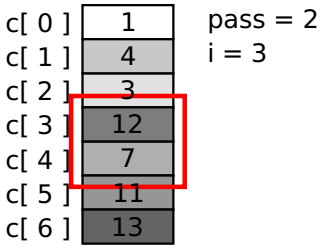
(E&E Eng. Dept. US) CP143 Lecture 16 23 August 2016 5 / 15

6.8 Bubble Sort



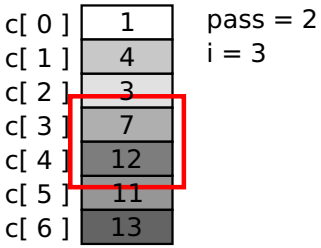
pass = 2
i = 2

6.8 Bubble Sort



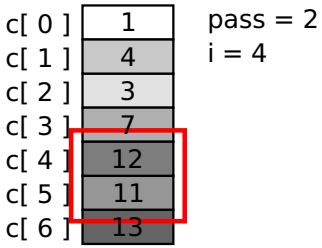
pass = 2
i = 3

6.8 Bubble Sort



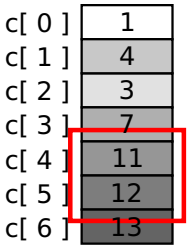
pass = 2
i = 3

6.8 Bubble Sort



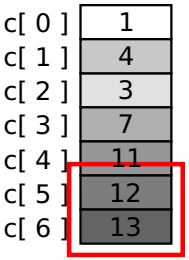
pass = 2
i = 4

6.8 Bubble Sort



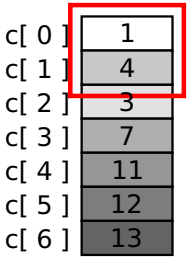
pass = 2
i = 4

6.8 Bubble Sort



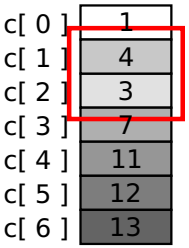
pass = 2
i = 5

6.8 Bubble Sort



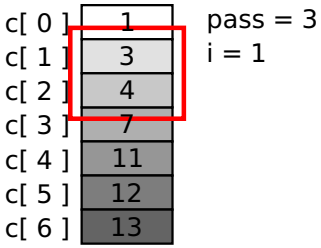
pass = 3
i = 0

6.8 Bubble Sort



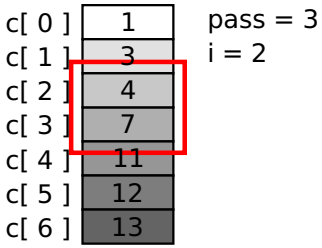
pass = 3
i = 1

6.8 Bubble Sort



pass = 3
i = 1

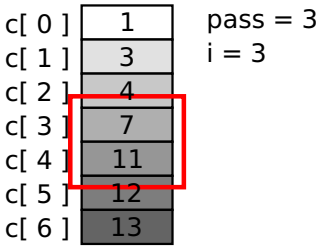
6.8 Bubble Sort



pass = 3
i = 2

(E&E Eng. Dept. US) CP143 Lecture 16 23 August 2016 5 / 15

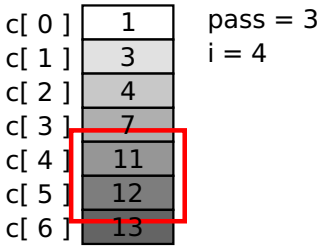
6.8 Bubble Sort



pass = 3
i = 3

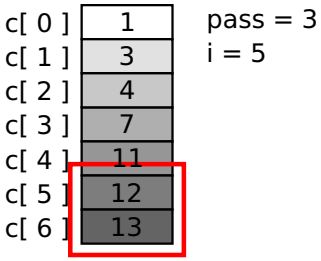
(E&E Eng. Dept. US) CP143 Lecture 16 23 August 2016 5 / 15

6.8 Bubble Sort



pass = 3
i = 4

(E&E Eng. Dept. US) CP143 Lecture 16 23 August 2016 5 / 15



Problem

- Write a C function to perform bubble sorting in any 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
```


Example

```c
#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;
}
```

```c
/* 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;
         }
      }
   }
}
```

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

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.

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
```


Pseudocode

```
begin mode(array)
    initialize variables largest, modeValue and freqArray elements to 0
    count the number of occurrences of each number
    for each occurance rating from 1 to 9
        if freqArray[rating] > largest
        swap values in freqArray[rating] and 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
```

Case Study

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

Perspective

Today

Arrays III

- Sorting arrays
- Example: mean, median and mode

Next lecture

- Multidimensional Arrays

Homework

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