Sorting

Last updated 10/29/20

Sorting

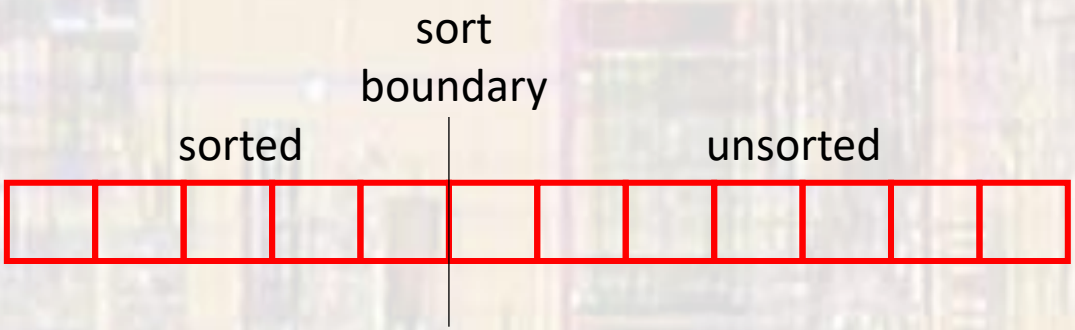
- These slides introduce the sorting application
- Upon completion: You should be able interpret and code using this application

Array Applications

- Sorting
  - Want to put the contents of an array in order
  - Selection Sort
  - Bubble Sort
  - Insertion Sort
  - →
  - Quicksort
  - Quickersort

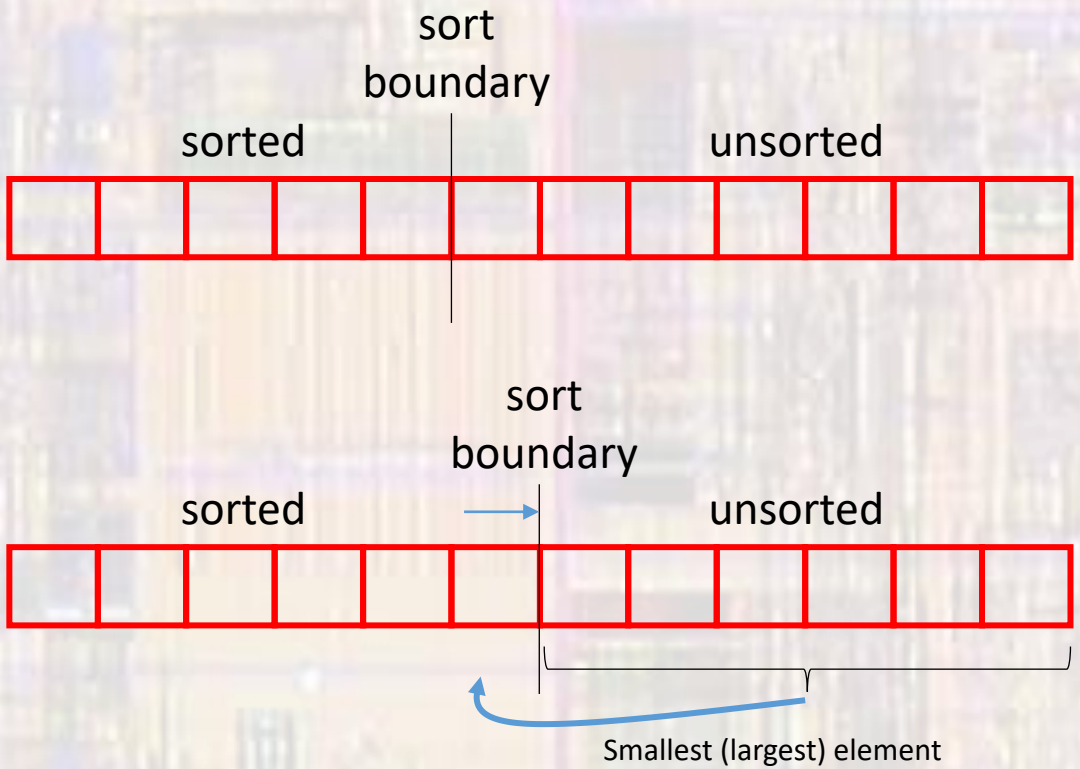
Array Applications

- Bubble Sort - conceptual
  - Sort an array of numbers into ascending or descending order
    - Split the list into 2 parts: sorted and unsorted
    - Find the smallest(largest) element in the unsorted part of the list
    - Move that element to the end of the sorted list
    - Move the sort boundary up by 1 element



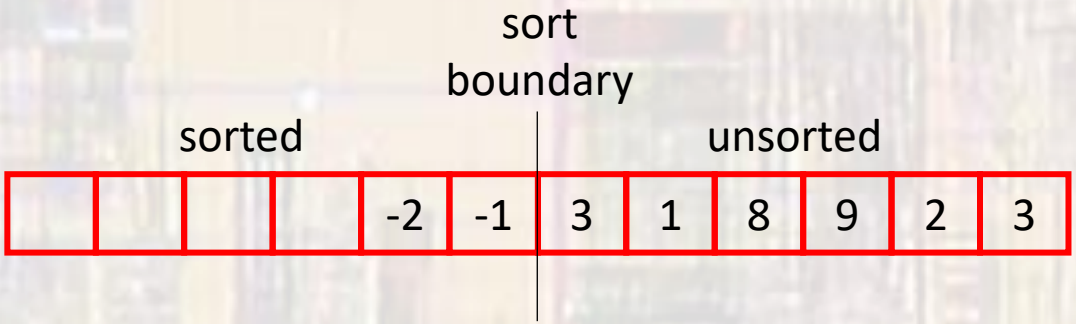
Array Applications

- Bubble Sort - conceptual



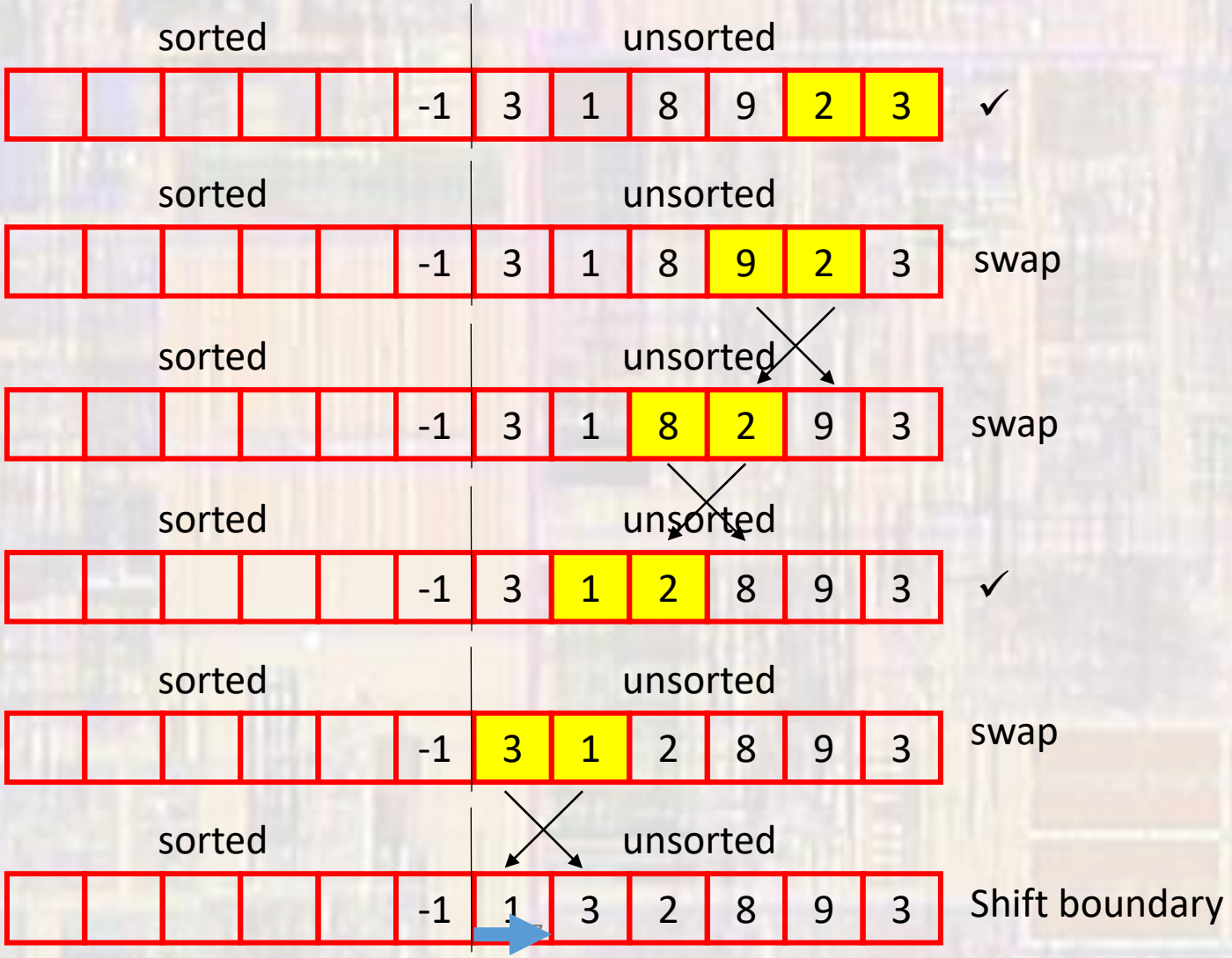
Array Applications

- Bubble Sort – implementation
  - How do we find the smallest(largest) element in the unsorted list?
    - Bubble it up to the beginning of the list
  - Compare 2 side by side elements
    - If in correct order, small → large(large → small), leave them alone
    - If not in correct order, swap them



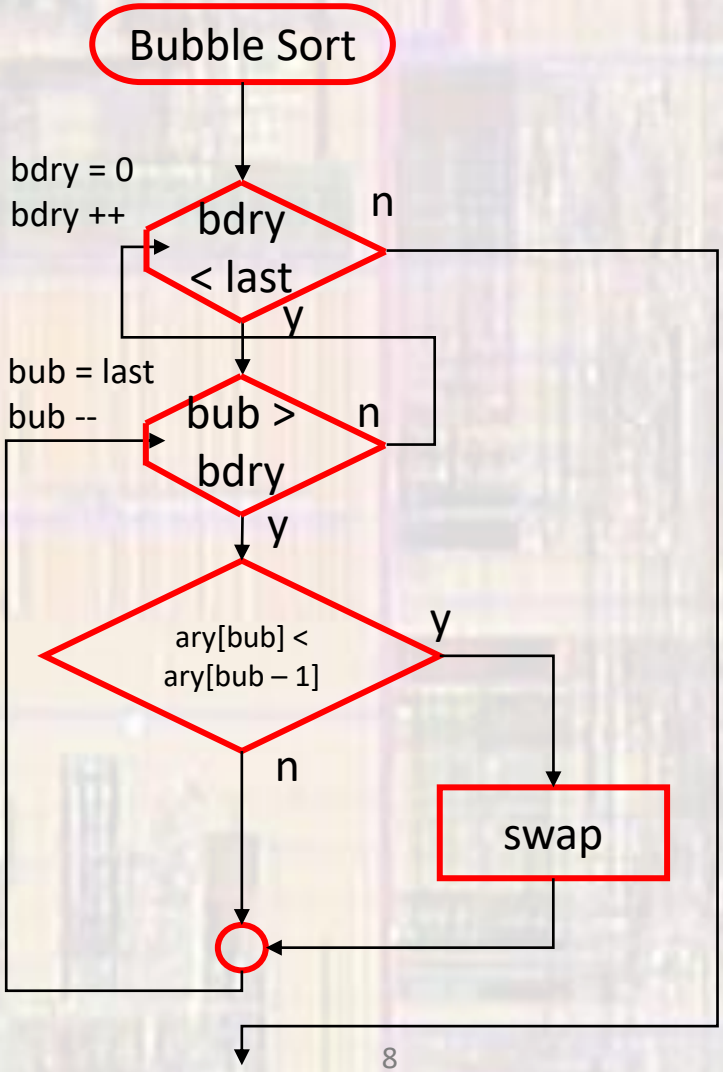
Array Applications

- Bubble Sort – implementation



Array Applications

- Bubble Sort – implementation

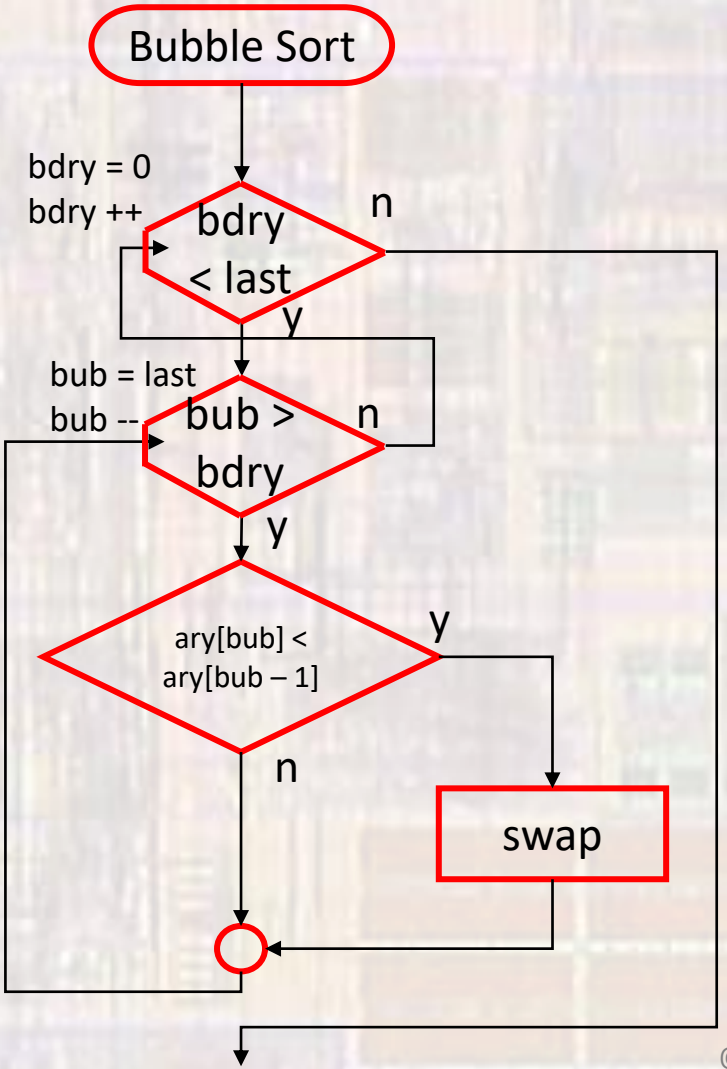


Array Applications

- Bubble Sort – implementation

```
void bubbleSort(int myArray[], int last){
  // local variables
  int tmp;
  int bdry;
  int bub;

  // outer loop
  for(bdry = 0; bdry < last; bdry++){
    // inner loop
    for(bub = last; bub > bdry; bub--){
      if(myArray[bub] < myArray[bub – 1]){
        tmp = myArray[bub];
        myArray[bub] = myArray[bub – 1];
        myArray[bub – 1] = tmp;
      } // end if
    } // end inner
  } // end outer
  return;
} // end bubbleSort
```



Array Applications

- Bubble Sort – usage

```c
/*
 * bubble_sort_example.c
 *
 *  Created on: Jan 23, 2019
 *      Author: johnsontimoj
 */

//////////////////////////////////////////////////
//
// Array example for lecture
//
// Bubble sort
//
//////////////////////////////////////////////////

// Includes
#include <stdio.h>

// Global Variables

// Function Prototypes
void bubbleSort(int myArray[], int last);
void print_array(int num_elements, const int the_array[]);

int main(void){
    //CC Composer I/O issue
    setbuf(stdout, NULL); // disable buffering

    // Local Variables
    int size;
    size = 8;
    int my_array[8] = { 9, 8, 7, 6, 5, 4, 3, 2 };
    print_array(8, my_array);
    printf("\n");
    printf("\n");

    bubbleSort(my_array, (size - 1));

    return 0;
} // end main
```

```c
void bubbleSort(int myArray[], int last){
   // local variables
   int tmp;
   int bdry;
   int bub;

   // outer loop
   for(bdry = 0; bdry < last; bdry++){
      // inner loop
      for(bub = last; bub > bdry; bub--){
         if(myArray[bub] < myArray[bub - 1]){
            tmp = myArray[bub];
            myArray[bub] = myArray[bub - 1];
            myArray[bub - 1] = tmp;
         } // end if
         print_array(8, myArray);
         printf("\n");
      } // end inner
      printf("---------------\n");
      //print_array(8, myArray);
   } // end outer
   return;
} // end bubbleSort

void print_array(int num_elements, const int the_array[]){
    int i;
    for(i=0; i<num_elements; i++){
        printf("%i ", the_array[i]);
    }
} // end print_array
```

```
<terminated> (exit v
9 8 7 6 5 4 3 2

9 8 7 6 5 4 2 3
9 8 7 6 5 2 4 3
9 8 7 6 2 5 4 3
9 8 7 2 6 5 4 3
9 8 2 7 6 5 4 3
9 2 8 7 6 5 4 3
2 9 8 7 6 5 4 3
---------------
2 9 8 7 6 5 3 4
2 9 8 7 6 3 5 4
2 9 8 7 3 6 5 4
2 9 8 3 7 6 5 4
2 9 3 8 7 6 5 4
2 3 9 8 7 6 5 4
---------------
2 3 9 8 7 6 4 5
2 3 9 8 7 4 6 5
2 3 9 8 4 7 6 5
2 3 9 4 8 7 6 5
2 3 4 9 8 7 6 5
---------------
2 3 4 9 8 7 5 6
2 3 4 9 8 5 7 6
2 3 4 9 5 8 7 6
2 3 4 5 9 8 7 6
---------------
2 3 4 5 9 8 6 7
2 3 4 5 9 6 8 7
2 3 4 5 6 9 8 7
---------------
2 3 4 5 6 9 7 8
2 3 4 5 6 7 9 8
---------------
2 3 4 5 6 7 8 9
---------------
```

inner

outer

Array Applications

- Bubble Sort
  - Efficiency - Bubble sort takes
    - N outer loops
    - (N(N-1))/2 inner loops
    - and a maximum of (N(N-1))/2 exchanges