```c
/*
 * Author:   Justin Bieber
 * Date:     October 28, 2010
 * Purpose:  Sort an array of integers in increasing order
 *           using insertion sort algorithm
 */

#include <stdio.h>

void insertion_sort(int data[], int sz);
void insert(int val, int data[], int end_index);
int find_insertion_index(int val, int data[], int end_index);
void shift_right(int data[], int start_index, int end_index);
void print_array( int data[], int size );

int main(void) {
    int a[] = {1, 2, 3};
    insertion_sort(a, 3);
    print_array(a, 3); // should print 1 2 3

    int b[] = {5, 4, 6, 1, 19, 2, 1, 0, 3};
    insertion_sort(b, 9);
    print_array(b, 9); // should print 0 1 1 2 3 4 5 6 19

    return 0;
}

/*
 * Purpose: sort the array in increasing order using insertion method
 * Param: data - array of values to sort
 *        sz - size of the array
 */
void insertion_sort(int data[], int sz) {
    int i;

    for (i=1; i<sz; i++) {
        insert(data[i], data, i-1);
    }
}

/*
 * Purpose: insert val into correct position of array
 *          between 0 and end_index
 *          Assumes data is sorted from index 0 to end_index inclusive
 * Param:   val - value to be inserted
 *          data - array to insert val into
 *          end_index - the last index of sorted values in data
 */
void insert(int val, int data[], int end_index) {
    int insertion_index = find_insertion_index(val, data, end_index);

    if (insertion_index != (end_index + 1)) {
        shift_right(data, insertion_index, end_index);
        data[insertion_index] = val;
    }
}
```

```c
/*
 * Purpose: find index between 0 and end_index inclusive to insert val
 *          if val is bigger than all values in range, returns (end_index+1)
 *          Assumes data is in sorted order from 0 and end_index inclusive
 * Param:   val - value to be inserted
 *          data - array to insert val into
 *          end_index - the last index of sorted values in data
 */
int find_insertion_index(int val, int data[], int end_index) {
    int i;
    for (i = 0; i <= end_index; i++) {
        if (val < data[i])
            return i;
    }
    return i;
}

/*
 * Purpose: shift right values in data in the range specified
 *          and will overwrite value in end_index+1
 * Param:   data - array of values to be shifted right
 *          start_index - first value in the array to be shifted right
 *          end_index - last value in the array to be shifted right
 */
void shift_right(int data[], int start_index, int end_index){
    int i;

    for (i = end_index; i >= start_index; i--) {
        data[i+1] = data[i];
    }
}

/*
 * Purpose: print the values of an array to the screen
 * Param:   data – the array to be printeed
 *          size – the size of the array
 */
void print_array( int data[], int size ) {
    int index;

    for (index = 0; index<size; index++) {
        printf("%d ", data[index]);
    }
    printf("\n\n");
}
```