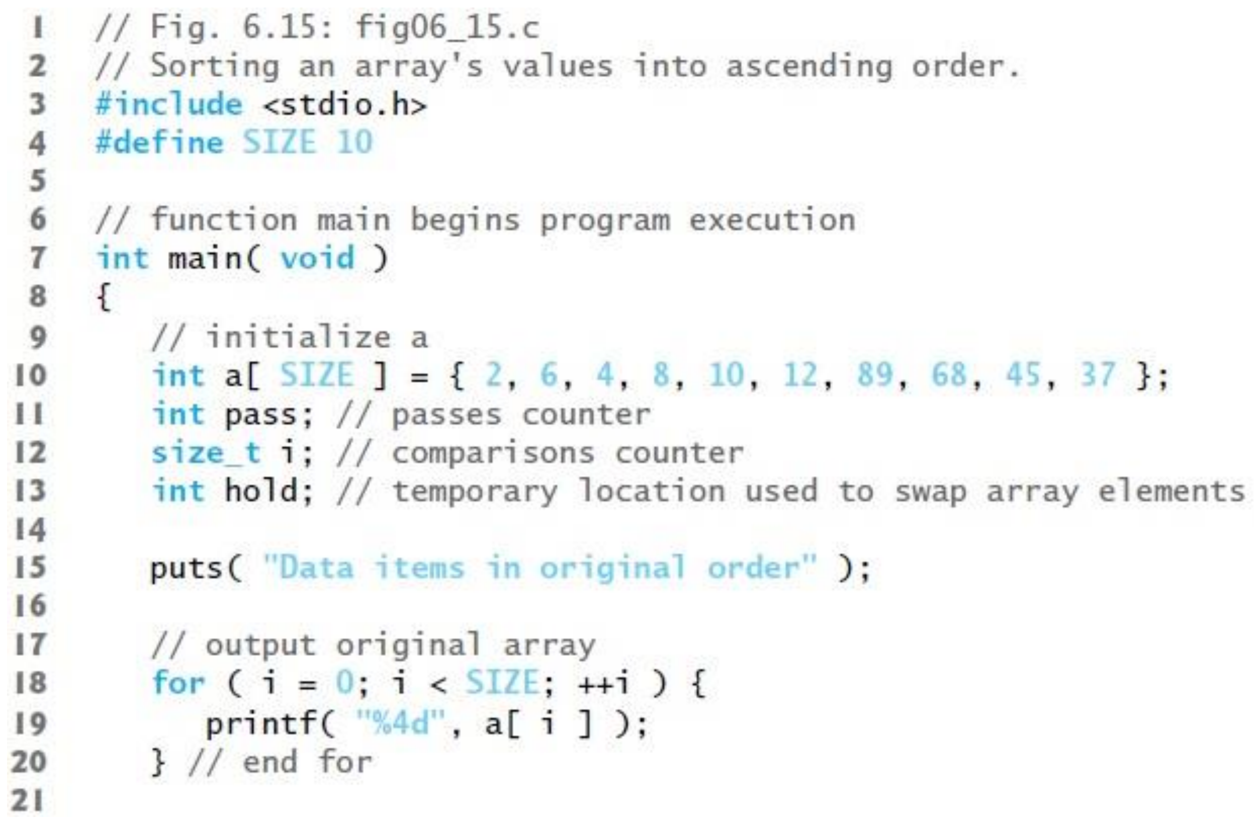
Lab # 14

January 05, 2017

Task 1: The bubble sort presented below is inefficient for large arrays. Make the following simple modifications to improve its performance. **(5 marks)**

a) After the first pass, the largest number is guaranteed to be in the highest-numbered element of the array; after the second pass, the two highest numbers are “in place,” and so on. Instead of making nine comparisons on every pass, modify the bubble sort to make eight comparisons on the second pass, seven on the third pass and so on.

b) The data in the array may already be in the proper or near-proper order, so why make nine passes if fewer will suffice? Modify the sort to check at the end of each pass whether any swaps have been made. If none has been made, then the data must already be in the proper order, so the program should terminate. If swaps have been made, then at least one more pass is needed.



```c
// Fig. 6.15: fig06_15.c
// Sorting an array's values into ascending order.
#include <stdio.h>
#define SIZE 10

// function main begins program execution
int main( void )
{
   // initialize a
   int a[ SIZE ] = { 2, 6, 4, 8, 10, 12, 89, 68, 45, 37 };
   int pass; // passes counter
   size_t i; // comparisons counter
   int hold; // temporary location used to swap array elements

   puts( "Data items in original order" );

   // output original array
   for ( i = 0; i < SIZE; ++i ) {
      printf( "%4d", a[ i ] );
   } // end for

   // bubble sort
   // 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 ( a[ i ] > a[ i + 1 ] ) {
            hold = a[ i ];
            a[ i ] = a[ i + 1 ];
            a[ i + 1 ] = hold;
         } // end if
      } // end inner for
   } // end outer for

   puts( "\nData items in ascending order" );

   // output sorted array
   for ( i = 0; i < SIZE; ++i ) {
      printf( "%4d", a[ i ] );
   } // end for

   puts( "" );
} // end main
```

```
Data items in original order
   2   6   4   8  10  12  89  68  45  37
Data items in ascending order
   2   4   6   8  10  12  37  45  68  89
```

Task 2: Modify the following program so function `mode` is capable of handling a tie for the mode value. Also, modify function `median` so the two middle elements are averaged in an array with an even number of elements. **(5 marks)**

```c
/* Fig. 6.16: fig06_16.c
   This program introduces the topic of survey data analysis.
   It computes the mean, median, and  mode of the data */
#include <stdio.h>
#define SIZE 99

void mean( const int [] );
void median( int [] );
void mode( int [], const int [] ) ;
void bubbleSort( int [] );
void printArray( const int [] );

int main()
{
   int frequency[ 10 ] = { 0 };
   int response[ SIZE ] =
      { 6, 7, 8, 9, 8, 7, 8, 9, 8, 9,
        7, 8, 9, 5, 9, 8, 7, 8, 7, 8,
        6, 7, 8, 9, 3, 9, 8, 7, 8, 7,
        7, 8, 9, 8, 9, 8, 9, 7, 8, 9,
        6, 7, 8, 7, 8, 7, 9, 8, 9, 2,
        7, 8, 9, 8, 9, 8, 9, 7, 5, 3,
        5, 6, 7, 2, 5, 3, 9, 4, 6, 4,
        7, 8, 9, 6, 8, 7, 8, 9, 7, 8,
        7, 4, 4, 2, 5, 3, 8, 7, 5, 6,
        4, 5, 6, 1, 6, 5, 7, 8, 7 };

   mean( response );
   median( response );
   mode( frequency, response );
   return 0;
}
```

```

void mean( const int answer[] )
{
   int i, total = 0;

   printf( "%s\n%s\n%s\n", "********", "  Mean", "********"

   for ( i = 0; i < SIZE; i++ )
      total += answer[  i  ];

   printf( "The mean is the average value of the data\n"
           "items. The mean is equal to the total of\n"
           "all the data items divided by the number\n"
           "of data items ( %d ). The mean value for\n"
           "this run is: %d / %d = %.4f\n\n",
           SIZE, total, SIZE, (  double  ) total / SIZE );
}

void median( int answer[] )
{
   printf( "\n%s\n%s\n%s\n%s",
           "********", " Median", "********",
           "The unsorted array of responses is" );

   printArray( answer );
   bubbleSort( answer );
   printf( "\n\nThe sorted array is" );
   printArray( answer );
   printf( "\n\nThe median is element %d of\n"
           "the sorted %d element array.\n"
           "For this run the median is %d\n\n",
           SIZE / 2, SIZE, answer[ SIZE / 2 ] );
```

```
}

void mode( int freq[], const int answer[] )
{
   int rating, j, h, largest = 0, modeValue = 0;

   printf( "\n%s\n%s\n%s\n",
           "********", "  Mode", "********" );

   for ( rating = 1; rating <= 9; rating++ )
      freq[ rating ] = 0;

   for ( j = 0; j < SIZE; j++ )
      ++freq[ answer[ j ] ];

   printf( "%s%11s%19s\n\n%54s\n%54s\n\n",
           "Response", "Frequency", "Histogram",
           "1    1    2    2", "5    0    5    0    5" );

   for ( rating = 1; rating <= 9; rating++ ) {
      printf( "%8d%11d          ", rating, freq[ rating ] );

      if ( freq[ rating ] > largest ) {
         largest = freq[ rating ];
         modeValue = rating;
      }

      for ( h = 1; h <= freq[ rating ]; h++ )
         printf( "*" );

```

```
       printf( "\n" );
    }

    printf( "The mode is the most frequent value.\n"
            "For this run the mode is %d which occurred"
            " %d times.\n", modeValue, largest );
 }

void bubbleSort( int a[] )
  {
    int pass, j, hold;

    for ( pass = 1; pass < SIZE; pass++ )

       for ( j = 0; j < SIZE - 1; j++ )

          if ( a[ j ] > a[ j + 1 ] ) {
             hold = a[ j ];
             a[ j ] = a[ j + 1 ];
             a[ j + 1 ] = hold;
          }
}

void printArray( const int a[] )
  {
    int j;

    for ( j = 0; j < SIZE; j++ ) {

       if ( j % 20 == 0 )
          printf( "\n" );

       printf( "%2d", a[ j ] );
    }
}
```

Grading and LMS Submission

- Make sure that the lab engineer has graded your programs until 5 pm.
- You’ve uploaded the C source files in Zip format over LMS until 5:30 pm.