```
// Fig. 6.16: fig06_16.c
// Survey data analysis with arrays;
// computing the mean, median and mode of the data.
#include <stdio.h>
#define SIZE 99

// function prototypes
void mean(const unsigned int answer[]);
void median(unsigned int answer[]);
void mode(unsigned int freq[], const unsigned int answer[]) ;
void bubbleSort(unsigned int a[]);
void printArray(const unsigned int a[]);

// function main begins program execution
int main(void)
{
   unsigned int frequency[10] = {0}; // initialize array frequency

   // initialize array response
   unsigned 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};

   // process responses
   mean(response);
   median(response);
   mode(frequency, response);
}

// calculate average of all response values
void mean(const unsigned int answer[])
{
   printf("%s\n%s\n%s\n", "********", "  Mean", "********");

   unsigned int total = 0; // variable to hold sum of array elements

   // total response values
   for (size_t j = 0; j < SIZE; ++j) {
      total += answer[j];
   }
```

```
   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 (%u). The mean value for\n"
          "this run is: %u / %u = %.4f\n\n",
          SIZE, total, SIZE, (double) total / SIZE);
}

// sort array and determine median element's value
void median(unsigned int answer[])
{
   printf("\n%s\n%s\n%s\n%s",
          "********", " Median", "********",
          "The unsorted array of responses is");

   printArray(answer); // output unsorted array

   bubbleSort(answer); // sort array

   printf("%s", "\n\nThe sorted array is");
   printArray(answer); // output sorted array

   // display median element
   printf("\n\nThe median is element %u of\n"
          "the sorted %u element array.\n"
          "For this run the median is %u\n\n",
          SIZE / 2, SIZE, answer[SIZE / 2]);
}

// determine most frequent response
void mode(unsigned int freq[], const unsigned int answer[])
{
   printf("\n%s\n%s\n%s\n", "********", "  Mode", "********");

   // initialize frequencies to 0
   for (size_t rating = 1; rating <= 9; ++rating) {
      freq[rating] = 0;
   }

   // summarize frequencies
   for (size_t j = 0; j < SIZE; ++j) {
      ++freq[answer[j]];
   }

   // output headers for result columns
   printf("%s%11s%19s\n\n%54s\n%54s\n\n",
          "Response", "Frequency", "Histogram",
```

```
            "1    1    2    2", "5    0    5    0    5");

   // output results
   unsigned int largest = 0; // represents largest frequency
   unsigned int modeValue = 0; // represents most frequent response

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

      // keep track of mode value and largest frequency value
      if (freq[rating] > largest) {
         largest = freq[rating];
         modeValue = rating;
      }

      // output histogram bar representing frequency value
      for (unsigned int h = 1; h <= freq[rating]; ++h) {
         printf("%s", "*");
      }

      puts(""); // being new line of output
   }

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

// function that sorts an array with bubble sort algorithm
void bubbleSort(unsigned int a[])
{
   // loop to control number of passes
   for (unsigned int pass = 1; pass < SIZE; ++pass) {

      // loop to control number of comparisons per pass
      for (size_t j = 0; j < SIZE - 1; ++j) {

         // swap elements if out of order
         if (a[j] > a[j + 1]) {
            unsigned int hold = a[j];
            a[j] = a[j + 1];
            a[j + 1] = hold;
         }
      }
   }
}
```

```
// output array contents (20 values per row)
void printArray(const unsigned int a[])
{
   // output array contents
   for (size_t j = 0; j < SIZE; ++j) {

      if (j % 20 == 0) { // begin new line every 20 values
         puts("");
      }

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