Chapter 8

Arrays

Figure 8.1 The Eight Elements of Array x

```
double x[8];
```

Array x

x[0]	x[1]	x[2]	x[3]	x[4]	x[5]	x[6]	x[7]
16.0	12.0	6.0	8.0	2.5	12.0	14.0	–54.5

Figure 8.2 Arrays answer and score

answer[0]	T	score[monday]	9
answer[1]	F	score[tuesday]	7
answer[2]	F	score[wednesday]	5
	. . .	score[thursday]	3
answer[9]	T	score[friday]	1

Figure 8.3 Program to Print a Table of Differences

```
/*
 * Computes the mean and standard deviation of an array of data and displays
 * the difference between each value and the mean.
 */

#include <stdio.h>
#include <math.h>

#define MAX_ITEM  8  /* maximum number of items in list of data                */

int
main(void)
{
      double x[MAX_ITEM],     /* data list                                     */
             mean,            /* mean (average) of the data                    */
             st_dev,          /* standard deviation of the data                */
             sum,             /* sum of the data                               */
             sum_sqr;         /* sum of the squares of the data                */
      int    i;

      /* Gets the data                                                         */
      printf("Enter %d numbers separated by blanks or <return>s\n> ",
             MAX_ITEM);
      for (i = 0; i < MAX_ITEM; ++i)
          scanf("%lf", &x[i]);
                                                                  (continued)
```


Figure 8.3 Program to Print a Table of Differences (cont'd)

```
      /* Computes the sum and the sum of the squares of all data               */
      sum = 0;
      sum_sqr = 0;
      for (i = 0; i < MAX_ITEM; ++i) {
          sum += x[i];
          sum_sqr += x[i] * x[i];
      }

      /* Computes and prints the mean and standard deviation                   */
      mean = sum / MAX_ITEM;
      st_dev = sqrt(sum_sqr / MAX_ITEM - mean * mean);
      printf("The mean is %.2f.\n", mean);
      printf("The standard deviation is %.2f.\n\n", st_dev);

      /* Displays the difference between each item and the mean                */
      printf("\nTable of differences between data values and mean\n");
      printf("Index    Item     Difference\n");
      for (i = 0; i < MAX_ITEM; ++i)
          printf("%3d%6s%9.2f%5s%8.2f\n", i, " ", x[i], " ", x[i] - mean);

      return (0);
}
```

```
Enter 8 numbers separated by blanks or <return>s
> 16  12  6  8  2.5  12  14  -54.5
The mean is 2.00.
The standard deviation is 21.75.

Table of differences between data values and mean
Index      Item      Difference
  0        16.00        14.00
  1        12.00        10.00
  2         6.00         4.00
  3         8.00         6.00
  4         2.50         0.50
  5        12.00        10.00
  6        14.00        12.00
  7       -54.50       -56.50
```


Figure 8.4 Data Area for Calling Module and Function do_it

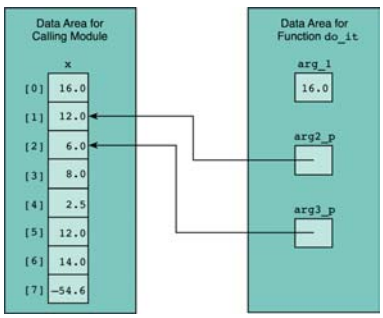

Figure 8.5 Function fill_array

```
1.  /*
2.   * Sets all elements of its array parameter to in_value.
3.   * Pre:  n and in_value are defined.
4.   * Post: list[i] = in_value, for 0 <= i < n.
5.   */
6.  void
7.  fill_array (int list[],    /* output - list of n integers                */
8.              int n,         /* input - number of list elements            */
9.              int in_value)  /* input - initial value                      */
10. {
11.
12.     int i;            /* array subscript and loop control               */
13.
14.     for  (i = 0;  i < n;  ++i)
15.         list[i] = in_value;
16. }
```


Figure 8.6 Data Areas Before Return from fill_array (x, 5, 1);

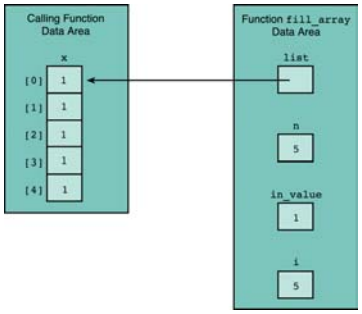

Figure 8.7 Function to Find the Largest Element in an Array

```
1.  /*
2.   *  Returns the largest of the first n values in array list
3.   *  Pre:  First n elements of array list are defined and n > 0
4.   */
5.  int
6.  get_max(const int list[], /* input - list of n integers                  */
7.          int       n)      /* input - number of list elements to examine */
8.  {
9.      int i,
10.         cur_large;      /* largest value so far                         */
11.
12.     /*  Initial array element is largest so far.                        */
13.     cur_large = list[0];
14.
15.     /*  Compare each remaining list element to the largest so far;
16.         save the larger                                                 */
17.     for  (i = 1;  i < n;  ++i)
18.         if (list[i] > cur_large)
19.             cur_large = list[i];
20.
21.     return (cur_large);
22. }
```


Figure 8.8 Diagram of a Function That Computes an Array Result

Figure 8.9 Function to Add Two Arrays

```
1.  /*
2.   *  Adds corresponding elements of arrays ar1 and ar2, storing  the result in
3.   *  arsum. Processes first n elements only.
4.   *  Pre:  First n elements of ar1 and ar2 are defined. arsum's corresponding
5.   *       actual argument has a declared size >= n (n >= 0)
6.   */
7.  void
8.  add_arrays(const double ar1[],   /* input -                                */
9.             const double ar2[],   /*       arrays being added               */
10.            double       arsum[], /* output - sum of corresponding
11.                                      elements of ar1 and ar2               */
12.            int          n)       /* input -  number of element
13.                                      pairs summed                          */
14. {
15.     int i;
16.
17.     /* Adds corresponding elements of ar1 and ar2                         */
18.     for  (i = 0;  i < n;  ++i)
19.         arsum[i] = ar1[i] + ar2[i];
20. }
```


Figure 8.10 Function Data Areas for add_arrays(x, y, x_plus_y, 5);

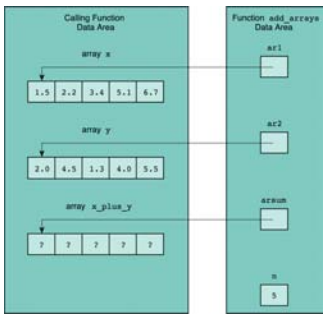

Figure 8.11 Diagram of Function fill_to_sentinel

Figure 8.12 Function Using a Sentinel-Controlled Loop to Store Input Data in an Array

```
/*
 * Gets data to place in dbl_arr until value of sentinel is encountered in
 * the input.
 * Returns number of values stored through dbl_sizep.
 * Stops input prematurely if there are more than dbl_max data values before
 * the sentinel or if invalid data is encountered.
 * Pre:  sentinel and dbl_max are defined and dbl_max is the declared size
 *       of dbl_arr
 */
void
fill_to_sentinel(int    dbl_max,    /* input - declared size of dbl_arr      */
                 double sentinel,   /* input - end of data value in
                                          input list                        */
                 double dbl_arr[],  /* output - array of data                */
                 int    *dbl_sizep) /* output - number of data values
                                          stored in dbl_arr                 */
{
    double data;
    int    i, status;

    /* Sentinel input loop                                                  */
    i = 0;
    status = scanf("%lf", &data);
    while (status == 1  &&  data != sentinel  &&  i < dbl_max) {
        dbl_arr[i] = data;
        ++i;
        status = scanf("%lf", &data);
    }

    /* Issues error message on premature exit                               */
    if (status != 1) {
        printf("\n*** Error in data format ***\n");
        printf("*** Using first %d data values ***\n", i);
    } else if (data != sentinel) {
        printf("\n*** Error: too much data before sentinel ***\n");
        printf("*** Using first %d data values ***\n", i);
    }

    /* Sends back size of used portion of array                             */
    *dbl_sizep = i;
}
```


Figure 8.13 Driver for Testing fill_to_sentinel

```
/* Driver to test fill_to_sentinel function */

#define A_SIZE  20
#define SENT   -1.0

int
main(void)
{
    double arr[A_SIZE];
    int    in_use,     /* number of elements of arr in use */
           i;

    fill_to_sentinel(A_SIZE, SENT, arr, &in_use);

    printf("List of data values\n");
    for  (i = 0;  i < in_use;  ++i)
        printf("%13.3f\n", arr[i]);

    return (0);
}
```