

CSI 333 – Programming at the Hardware-Software Interface

Some Examples of C Programs (continued)

Handout 3.3

Program Example 6: The purpose of this example is to show a program with several functions and illustrate call-by-value and call-by-reference for passing parameters.

```
#include <stdio.h>

#define SIZE 5

/* Prototypes for the functions called by main. */

void print_array(int[], int);
void increment(int [], int);
float stat_values(int [], int, int *, int *);

int main(void) {

    int values[SIZE] = {-17, 8, 9, 0, 44}; /* Initial values. */
    int num = SIZE; /* Number of values in the array. */

    int max, min; /* To obtain max and min values in the array. */
    float average; /* To obtain the average value. */

    /* In the following call to the stat_values function:
       (a) The parameter num is being passed by value.
       (b) Pointers to the parameters max and min are passed
           to achieve call-by-reference.
    */
    average = stat_values(values, num, &max, &min);

    printf("Max. value = %d\n", max); printf("Min. value = %d\n", min);
    printf("Average = %6.1f\n", average);

    /* Print the numbers stored in the values array before and
       after the call to increment.
    */

    printf("\nBefore the call to function increment:\n");
    print_array(values, num);
```

(over)

```

/* The next call changes all the numbers stored in the values array. */
increment(values, num);

printf("\nAfter the call to function increment:\n");
print_array(values, num);

return 0;
} /* End of main. */

```

```

void print_array(int x[], int n) {

/* Prints the values stored in the array x. The parameter */
/* n gives the number of elements in the array.          */

int j; /* Temporary. */

/* Loop to print all the values in the array. */
for (j = 0; j < n; j++)
    printf("%5d", x[j]);

/* Print a newline character after printing all the values. */
printf("\n");

} /* End of print_array */

```

```

void increment(int x[], int n) {

/* Increments each value stored in the array x by 1.      */
/* The parameter n gives the number of elements in the array. */
/* This function shows that arrays are passed by reference. */

int j; /* Temporary. */

/* Loop to increment all the values in the array. */
for (j = 0; j < n; j++)
    x[j]++;

} /* End of increment */

```

```

float stat_values (int x[], int n, int *max, int *min) {

/* The inputs to this function are the array x and the */
/* value n which gives the size of the array.          */

/* The function assumes that the value of n is strictly */
/* greater than zero.                                    */

```

```

/* The function computes the maximum and minimum values */
/* in the array and returns them through the two      */
/* parameters *max and *min.

/* The return value of the function is of type float; */
/* it is the average of the values stored in the array. */

int sum; /* Stores the sum; needed to compute the average. */
int j; /* Temporary. */

/* Initialization. */
*min = *max = x[0]; /* Note how the parameters are used. */
sum = x[0];

/* Loop through the array and compute the required values.
   We can skip element 0 of the array because of the way
   things were initialized.
*/
for (j = 1; j < n; j++) {

/* Update the maximum value. */
if (*max < x[j]) *max = x[j];

/* Update the minimum value. */
if (*min > x[j]) *min = x[j];

/* Update the sum. */
sum += x[j];

} /* End of for loop. */

/* The maximum and minimum values have been obtained and stored
   in the corresponding parameters. Compute and return the
   average value.
*/
return (float)sum/(float)n;

} /* End of stat_values */

```

Output:

```

Max. value = 44
Min. value = -17
Average    =   8.8

```

Before the call to function increment:

```
-17  8  9  0  44
```

After the call to function increment:

```
-16  9  10  1  45
```