COP 3223: C Programming
Spring 2009

Functions In C – Part 4

Instructor : Dr. Mark Llewellyn
markl@cs.ucf.edu
HEC 236, 407-823-2790
http://www.cs.ucf.edu/courses/cop3223/spr2009/section1

School of Electrical Engineering and Computer Science
University of Central Florida

More Using Pass-by-Reference Parameters

- In the previous section of notes, we introduced the pass-by-reference mechanism for parameter passing and how this is simulated in C.
- Pass-by-value causes a copy of the value of the actual parameter to be copied (sent) to the formal parameter in the called function.
- Pass-by-reference essentially passes the address of the actual parameter to the formal parameter in the called function. In the called function, the formal parameter must be a pointer variable. When the address of the actual parameter is loaded into the formal parameter, the formal parameter now "points to" the same address in memory referenced by the actual parameter.

A pass-by-value example

A called function

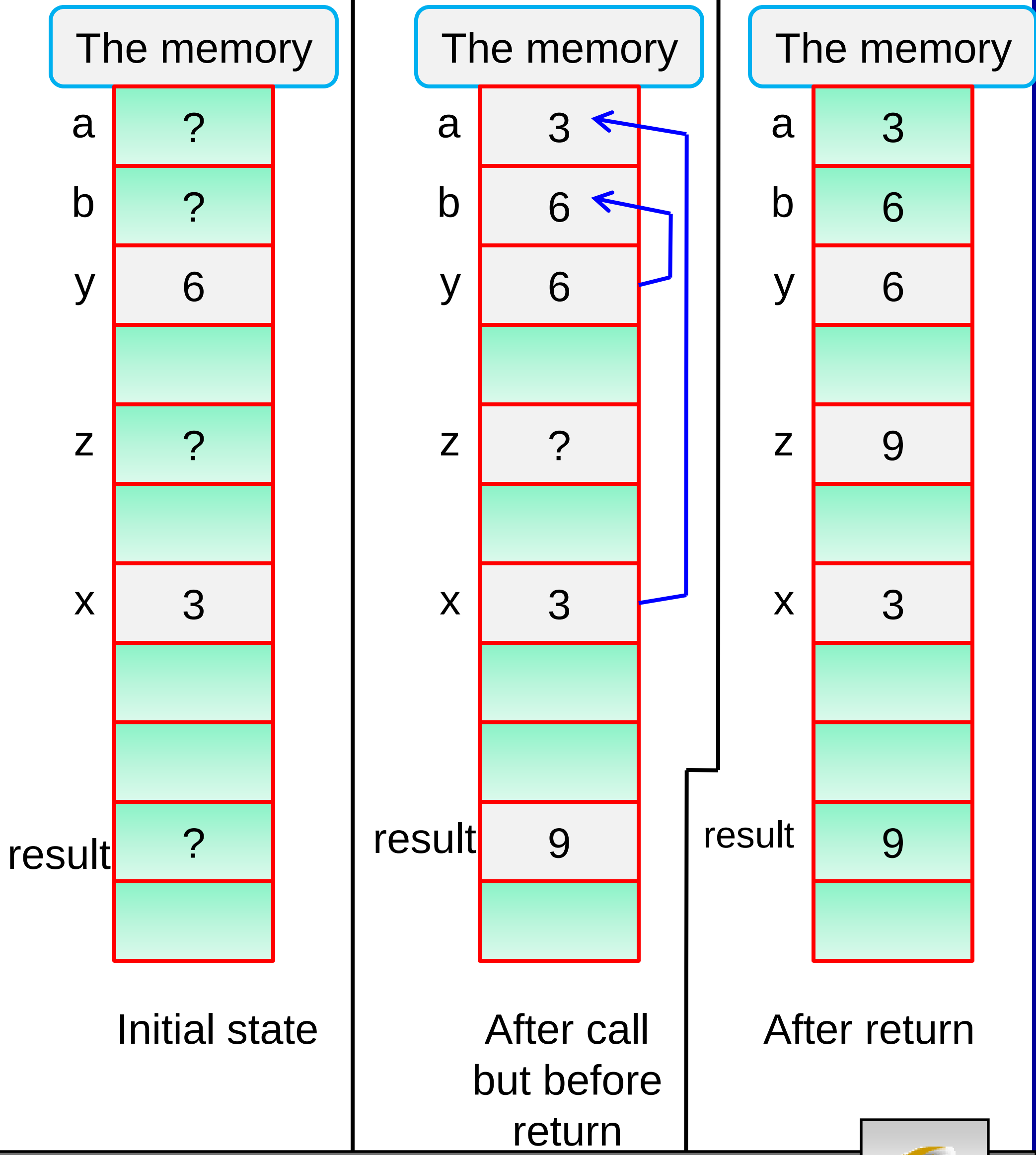
```
int  aFunction (int a, int b)
{
   int result;
   . . .
   return a+b;
}
```

Somewhere in a calling function

```
int x = 3, y = 6, z;
. . .
z =  aFunction (x, y);
printf("%d %d %d\n", x, y, z);
```

Prints "3 6 9"

Green filled locations not initialized or available



A pass-by-reference example

A called function

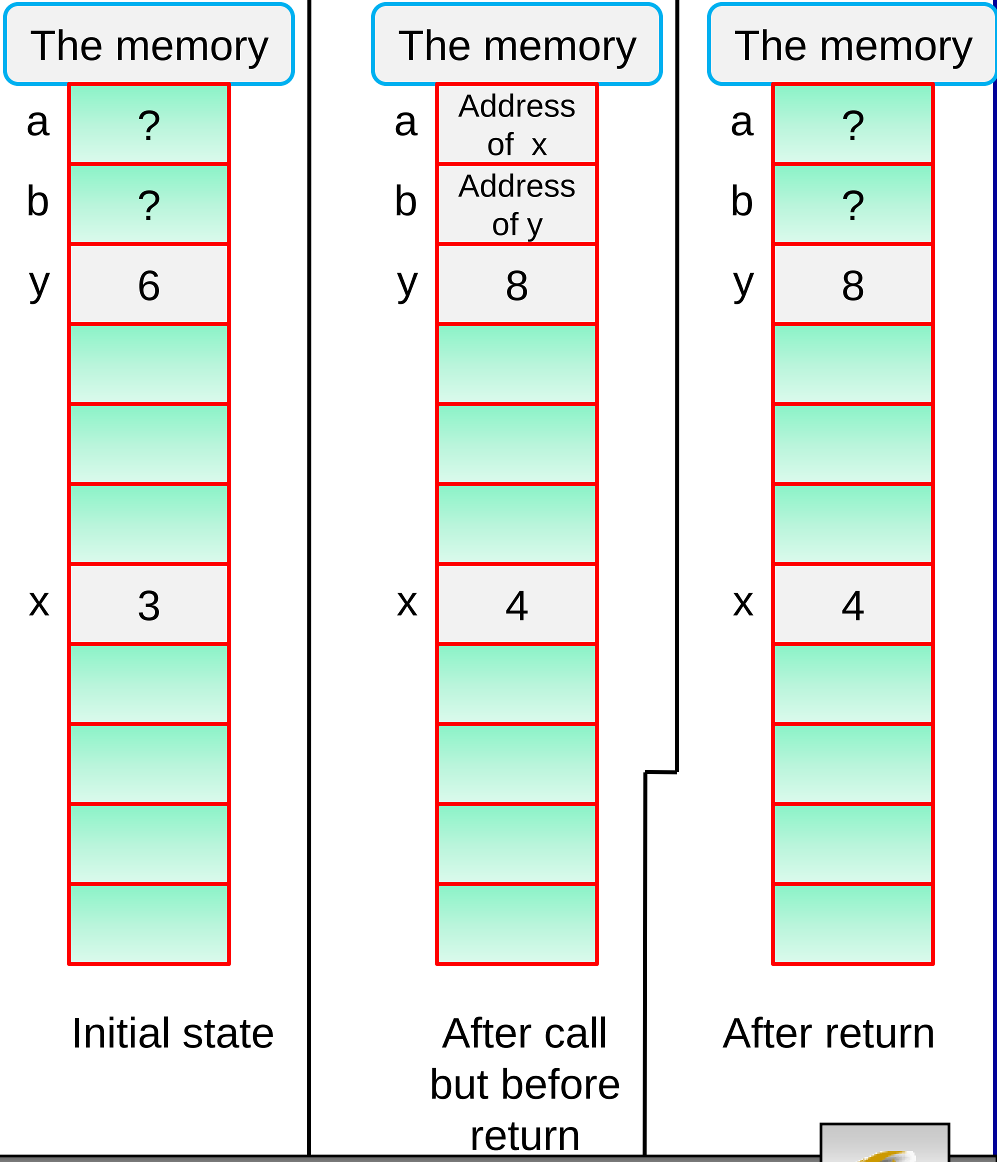
```
void  aFunction (int *a, int *b)
{
   *a = 4;  //de-ref ptr var
   *b = 8;  //de-ref ptr var
   return;
}
```

Somewhere in a calling function

```
int x = 3, y = 6;
. . .
aFunction (&x, &y);
printf("%d %d\n", x, y);
```

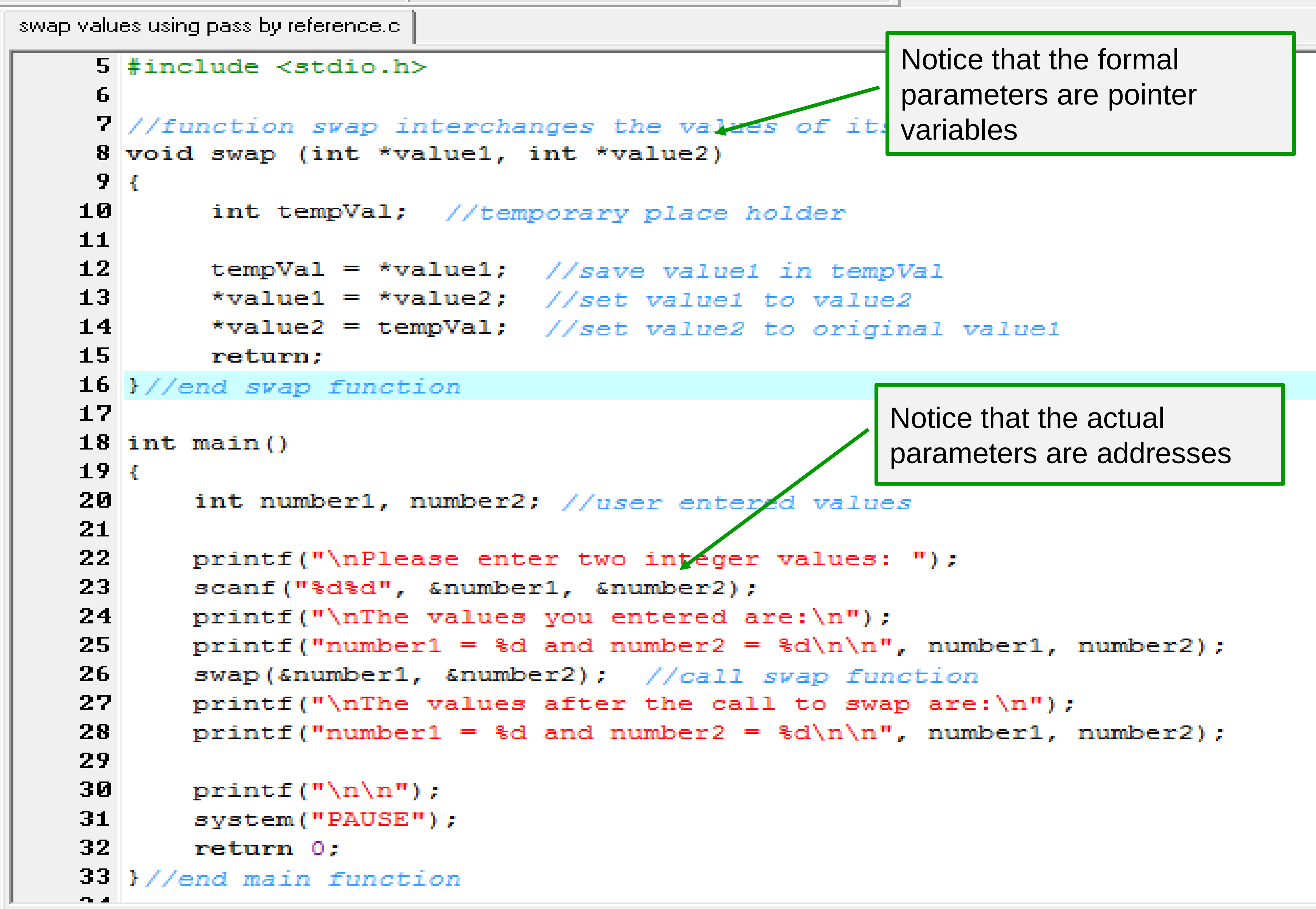
Prints "4 8"

Green filled locations not initialized or available



More With Pass-By-Reference Parameters

- Let's write another simple program that uses a function whose parameters are passed-by-reference.
- Will write a function that simply swaps the values of two parameters sent to it. We'll then later on use the same function in another program that will allow us to sort a set of numbers into ascending order.
- Basically, the `swap` function will take two parameters `x` and `y` and interchange their values. If before calling `swap` the value of `x` is 4 and the value of `y` is 6, after the call the value of `x` will be 6 and the value of `y` will be 4.

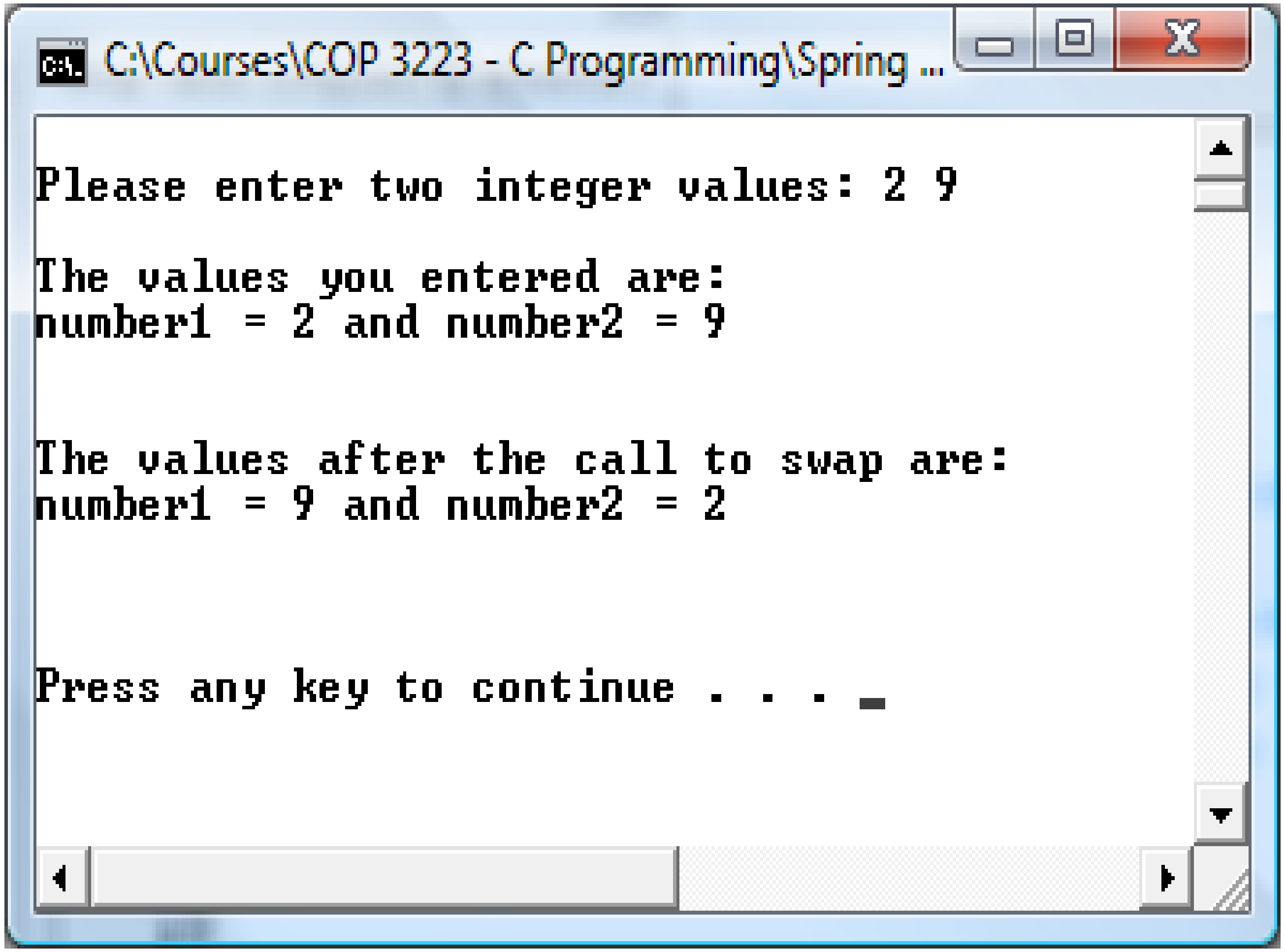


swap values using pass by reference.c

```c
#include <stdio.h>

//function swap interchanges the values of it
void swap (int *value1, int *value2)
{
    int tempVal;   //temporary place holder

    tempVal = *value1;   //save value1 in tempVal
    *value1 = *value2;   //set value1 to value2
    *value2 = tempVal;   //set value2 to original value1
    return;
}//end swap function

int main()
{
    int number1, number2; //user entered values

    printf("\nPlease enter two integer values: ");
    scanf("%d%d", &number1, &number2);
    printf("\nThe values you entered are:\n");
    printf("number1 = %d and number2 = %d\n\n", number1, number2);
    swap(&number1, &number2);  //call swap function
    printf("\nThe values after the call to swap are:\n");
    printf("number1 = %d and number2 = %d\n\n", number1, number2);

    printf("\n\n");
    system("PAUSE");
    return 0;
}//end main function
```

Notice that the formal parameters are pointer variables

Notice that the actual parameters are addresses



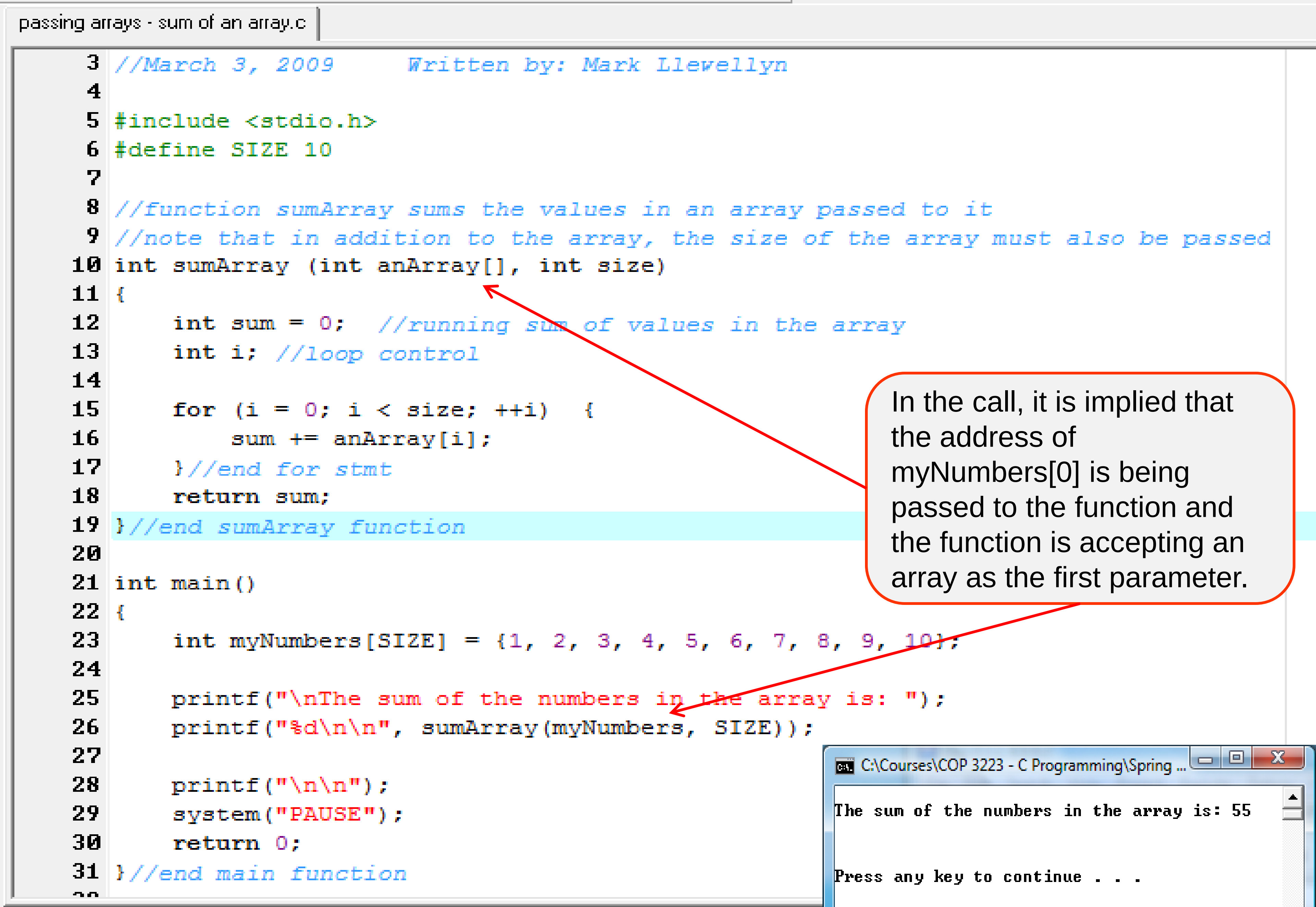
```
Please enter two integer values: 2 9

The values you entered are:
number1 = 2 and number2 = 9


The values after the call to swap are:
number1 = 9 and number2 = 2



Press any key to continue . . . _
```

Passing Arrays As Arguments

- In a function definition in C, a formal parameter that is declared as an array is, by default, a pointer to the array.
- More specifically, it is the address of the first location in the array. Thus, the name of the array is equivalent to `&arrayName[0]`.
- When an array is being passed to a function, its base address is passed-by-value to the function. The array elements themselves are not copied. Thus, through the base address (address of the first element of the array) access to all other locations in the array is available to the function. Thus, the array locations have been passed by reference and any changes made by the function to those array locations are made in the one and only copy of the array.

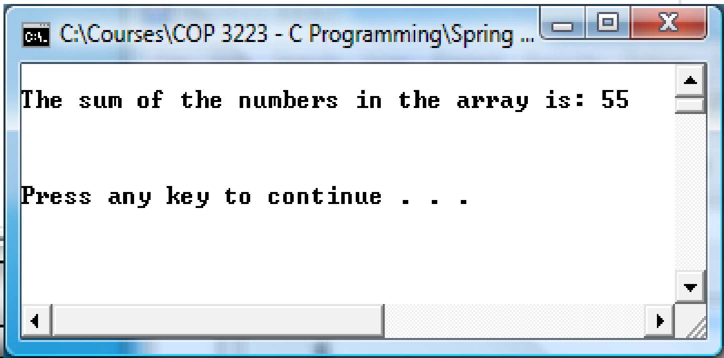
Passing Arrays As Arguments

- Standard C compilers allow array bracket notation to be included in the function parameter list for any array parameter, although it is not required that the brackets be included in the formal parameter description.
- From a program/code readability point of view, it is wise to include the brackets so that it is obvious to the reader that an array is being used as an argument to the function.
- Thus, if we have declared `int a[10];`, then in a function header `int a[]` is equivalent to `int *a`.
- Let's write a program that passes a 1-d array to a function and the function sums the values in the array and returns the result to the calling function.
- Notice the difference in the function declaration and the function call in the two versions of the program.

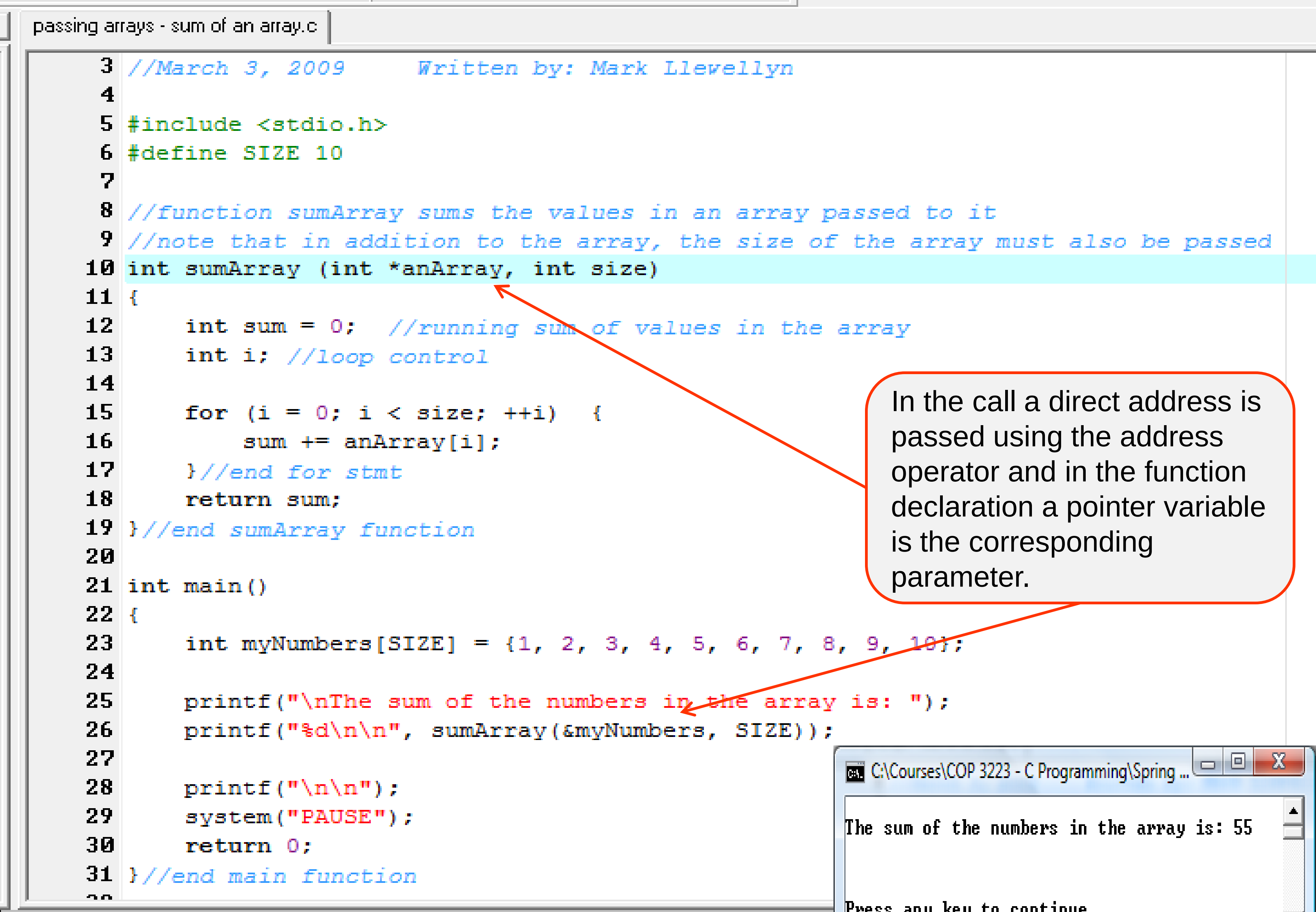

passing arrays - sum of an array.c

```c
//March 3, 2009     Written by: Mark Llewellyn

#include <stdio.h>
#define SIZE 10

//function sumArray sums the values in an array passed to it
//note that in addition to the array, the size of the array must also be passed
int sumArray (int anArray[], int size)
{
    int sum = 0;  //running sum of values in the array
    int i; //loop control

    for (i = 0; i < size; ++i)   {
        sum += anArray[i];
    }//end for stmt
    return sum;
}//end sumArray function

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

    printf("\nThe sum of the numbers in the array is: ");
    printf("%d\n\n", sumArray(myNumbers, SIZE));

    printf("\n\n");
    system("PAUSE");
    return 0;
}//end main function
```

In the call, it is implied that the address of myNumbers[0] is being passed to the function and the function is accepting an array as the first parameter.

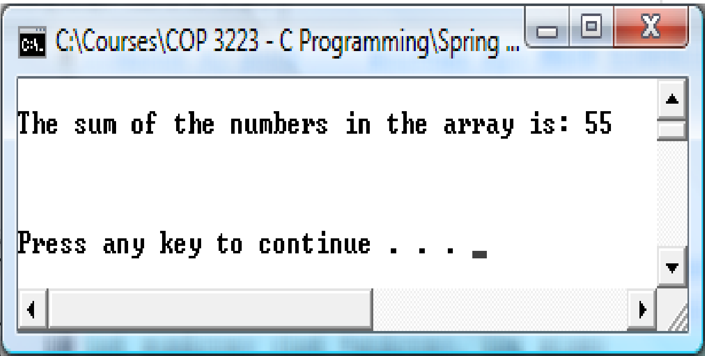

C:\Courses\COP 3223 - C Programming\Spring ...

```
The sum of the numbers in the array is: 55


Press any key to continue . . .
```



```c
//March 3, 2009     Written by: Mark Llewellyn

#include <stdio.h>
#define SIZE 10

//function sumArray sums the values in an array passed to it
//note that in addition to the array, the size of the array must also be passed
int sumArray (int *anArray, int size)
{
    int sum = 0;  //running sum of values in the array
    int i; //loop control

    for (i = 0; i < size; ++i)   {
        sum += anArray[i];
    }//end for stmt
    return sum;
}//end sumArray function

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

    printf("\nThe sum of the numbers in the array is: ");
    printf("%d\n\n", sumArray(&myNumbers, SIZE));

    printf("\n\n");
    system("PAUSE");
    return 0;
}//end main function
```

In the call a direct address is passed using the address operator and in the function declaration a pointer variable is the corresponding parameter.

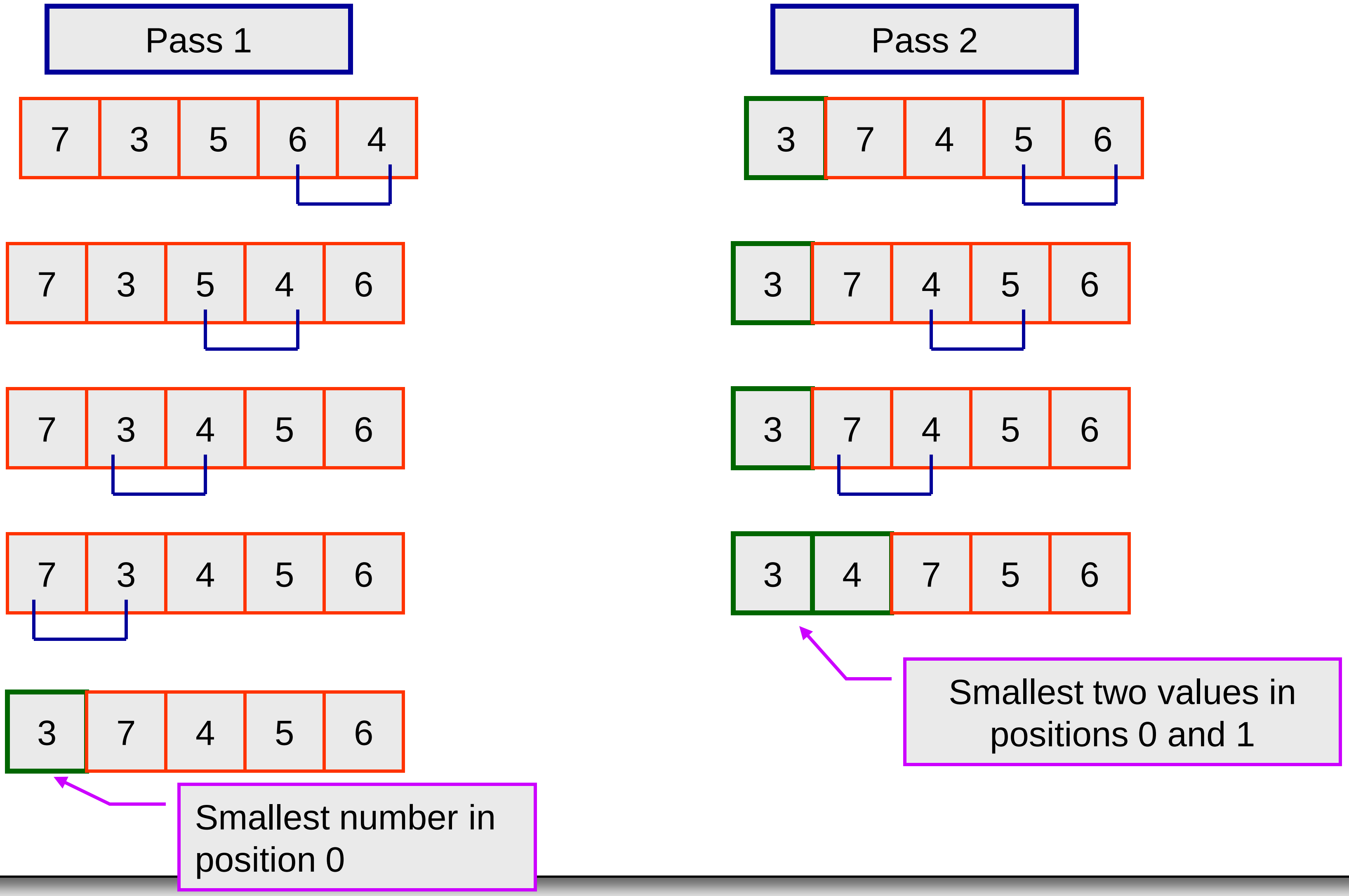


```
The sum of the numbers in the array is: 55


Press any key to continue . . .
```

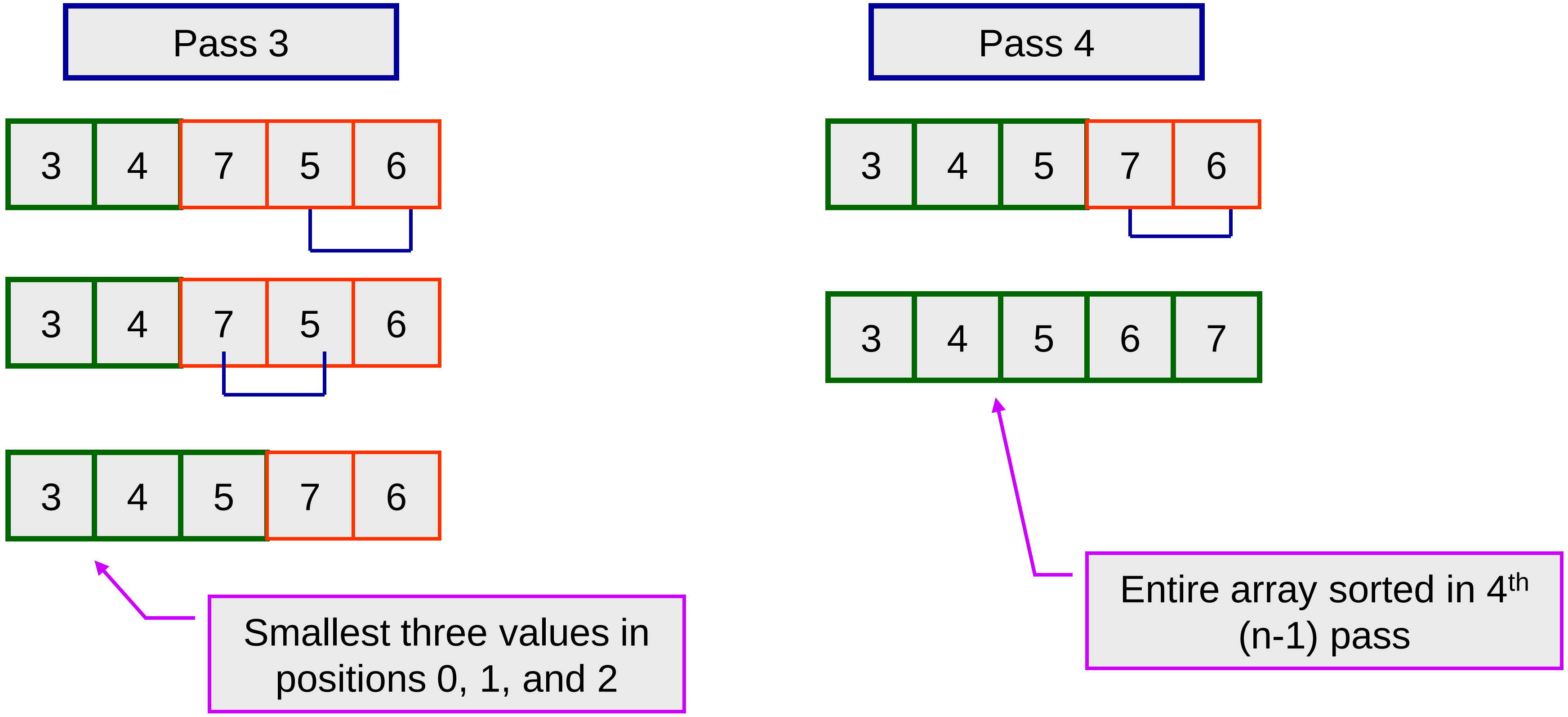
Sorting An Array Of Integer Values

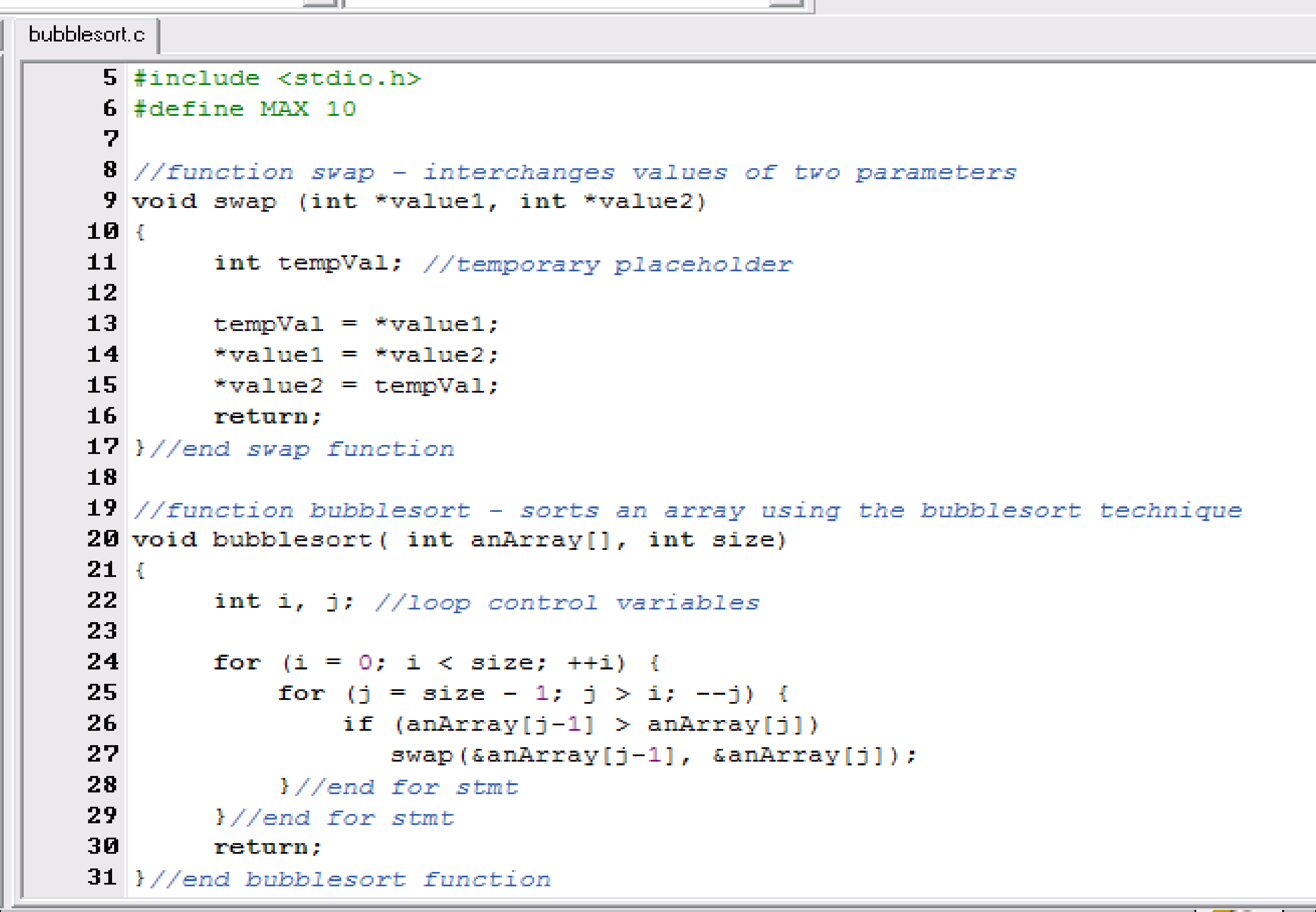
- Let's combine both parameter passing techniques and reuse the `swap` function we wrote on page 6 and develop a program that will sort an array of integer values.
- There are many different sorting algorithms available. The one we will use for this program is called a bubble sort. The technique it uses is to bubble the smallest value in the array to the first position in the array on the first pass through the array, bubble the second smallest value in the array to the second position in the array on the second pass through the array, and so on.
- We'll write a function called `bubblesort` that performs the sort and calls the function `swap` to change the order of any two elements in the array that are out of order according to the sort.

How The Bubble Sort Works



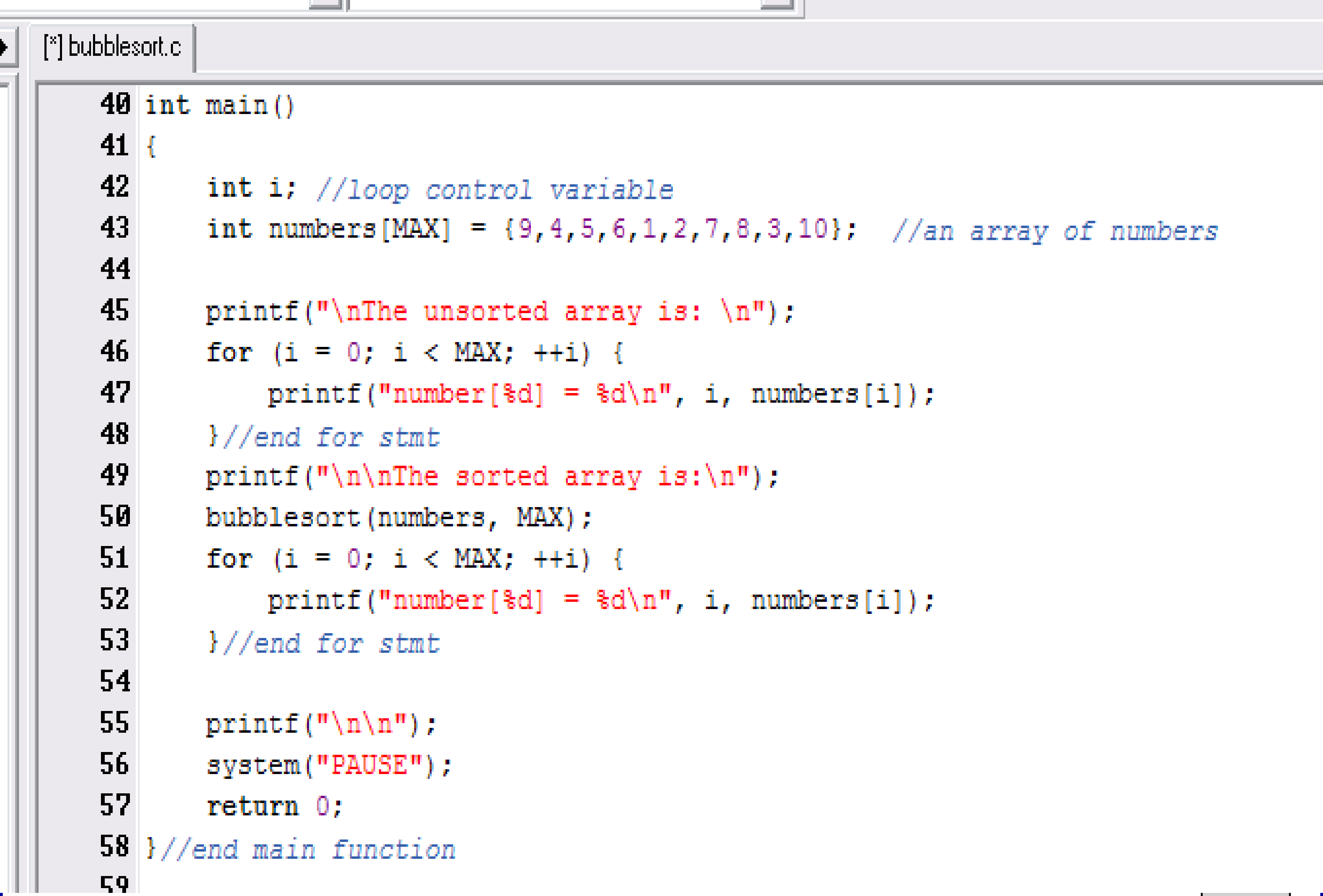
How The Bubble Sort Works


Pass 3
3 4 7 5 6
3 4 7 5 6
3 4 5 7 6
Smallest three values in positions 0, 1, and 2
Pass 4
3 4 5 7 6
3 4 5 6 7
Entire array sorted in 4th (n-1) pass



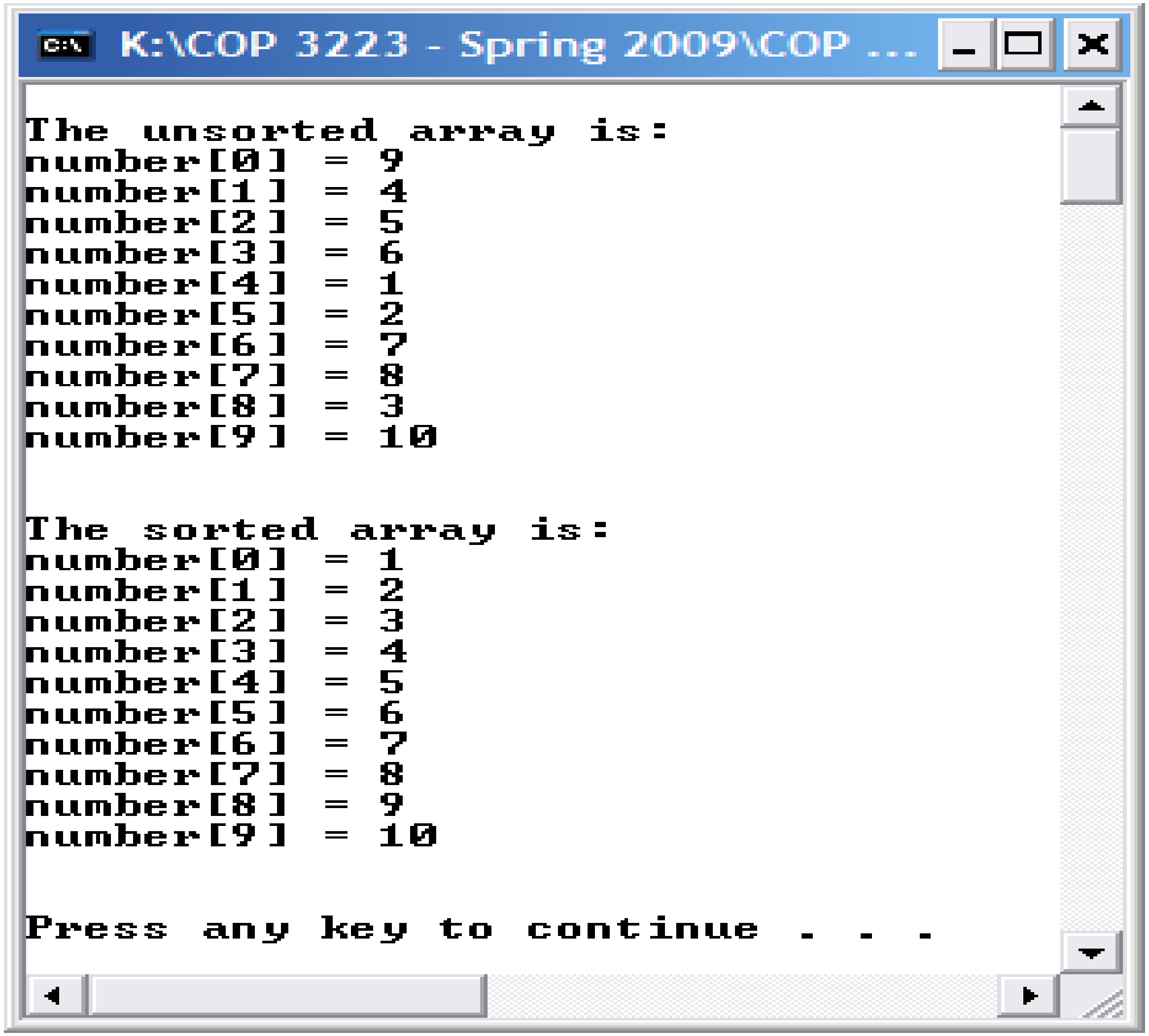
bubblesort.c

```
#include <stdio.h>
#define MAX 10

//function swap - interchanges values of two parameters
void swap (int *value1, int *value2)
{
      int tempVal; //temporary placeholder

      tempVal = *value1;
      *value1 = *value2;
      *value2 = tempVal;
      return;
}//end swap function

//function bubblesort - sorts an array using the bubblesort technique
void bubblesort( int anArray[], int size)
{
      int i, j; //loop control variables

      for (i = 0; i < size; ++i) {
          for (j = size - 1; j > i; --j) {
              if (anArray[j-1] > anArray[j])
                 swap(&anArray[j-1], &anArray[j]);
          }//end for stmt
      }//end for stmt
      return;
}//end bubblesort function
```



[*] bubblesort.c

```c
int main()
{
    int i; //loop control variable
    int numbers[MAX] = {9,4,5,6,1,2,7,8,3,10};  //an array of numbers

    printf("\nThe unsorted array is: \n");
    for (i = 0; i < MAX; ++i) {
        printf("number[%d] = %d\n", i, numbers[i]);
    }//end for stmt
    printf("\n\nThe sorted array is:\n");
    bubblesort(numbers, MAX);
    for (i = 0; i < MAX; ++i) {
        printf("number[%d] = %d\n", i, numbers[i]);
    }//end for stmt

    printf("\n\n");
    system("PAUSE");
    return 0;
}//end main function
```


K:\COP 3223 - Spring 2009\COP ...

```
The unsorted array is:
number[0] = 9
number[1] = 4
number[2] = 5
number[3] = 6
number[4] = 1
number[5] = 2
number[6] = 7
number[7] = 8
number[8] = 3
number[9] = 10


The sorted array is:
number[0] = 1
number[1] = 2
number[2] = 3
number[3] = 4
number[4] = 5
number[5] = 6
number[6] = 7
number[7] = 8
number[8] = 9
number[9] = 10


Press any key to continue . . .
```

Practice Problems

1. Trace the execution of the bubble sort program on pages 15 & 16 assuming that the array initially contains the values 4, 12, 2, 5, 1. Assume that MAX is also changed to have a value of 5.

Practice Problems

2. Write a C program that uses a function to multiply every value in an array which is passed to the function by some constant amount. Then write a second function that prints the values that appear in the array whenever it is called. Have the main function read the values into the array from an input file named `input.dat`, where it is unknown in advance how many values will appear in the file (assume it will be less than 100 values). Once the values are read into the array have the main function call the array print function and print out the contents of the array. Then the main function should call the function that will modify the array values and finally have the main function once again call the array print function to print out the values after they have been modified.

Practice Problems

3. Re-write the bubble sort program so that rather than producing an ascending sort order in the array it produces a descending sort order (i.e., the largest number in the array will be in the 0 position). Hint: Think for a minute, this is much easier than it might seem.