COP 3223: C Programming Spring 2009

Functions In C – Part 5

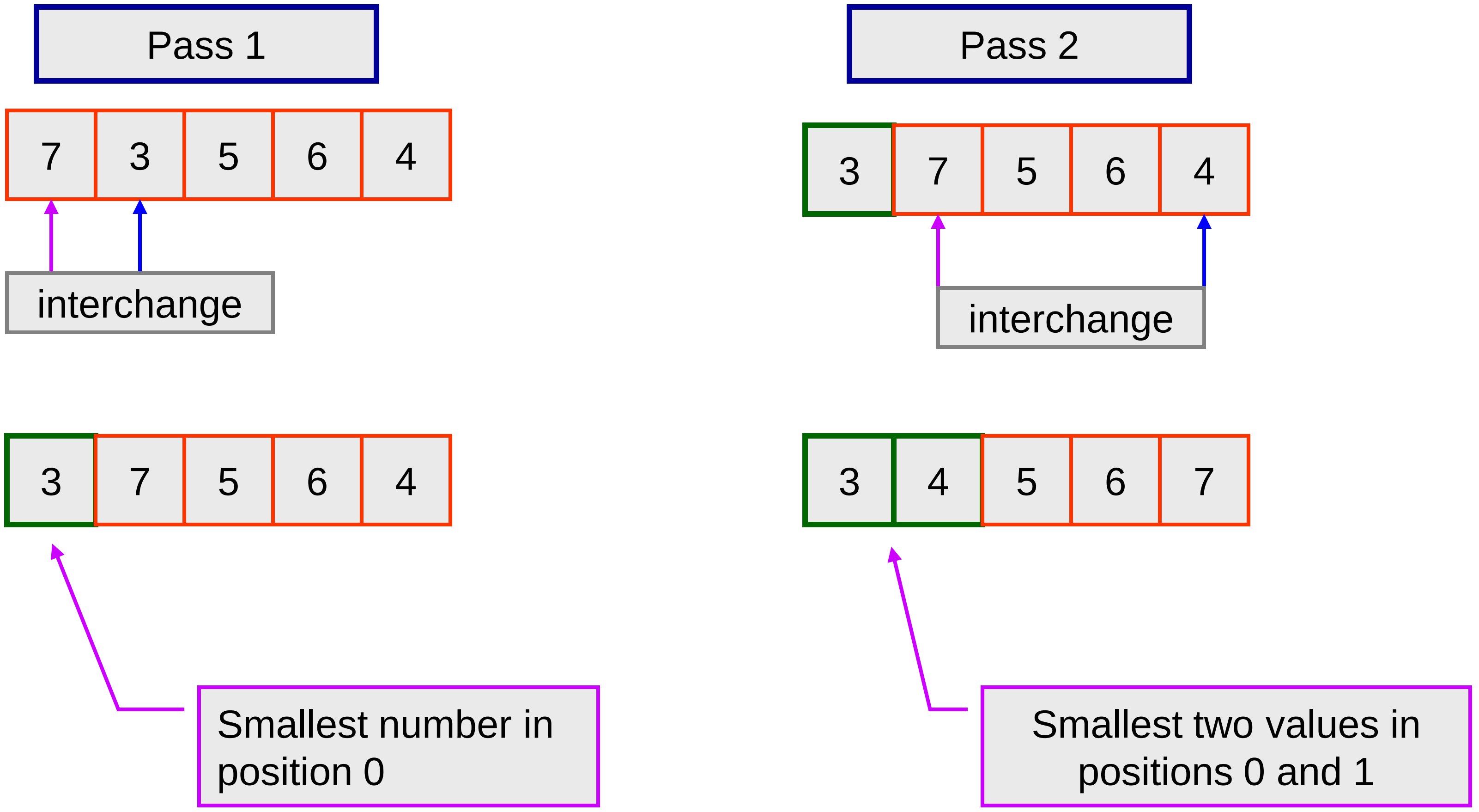
Instructor : Dr. Mark Llewellyn
markl@cs.ucf.edu
HEC 236, 407-823-2790
http://www.cs.ucf.edu/courses/cop3223/spr2009/section1

School of Electrical Engineering and Computer Science
University of Central Florida

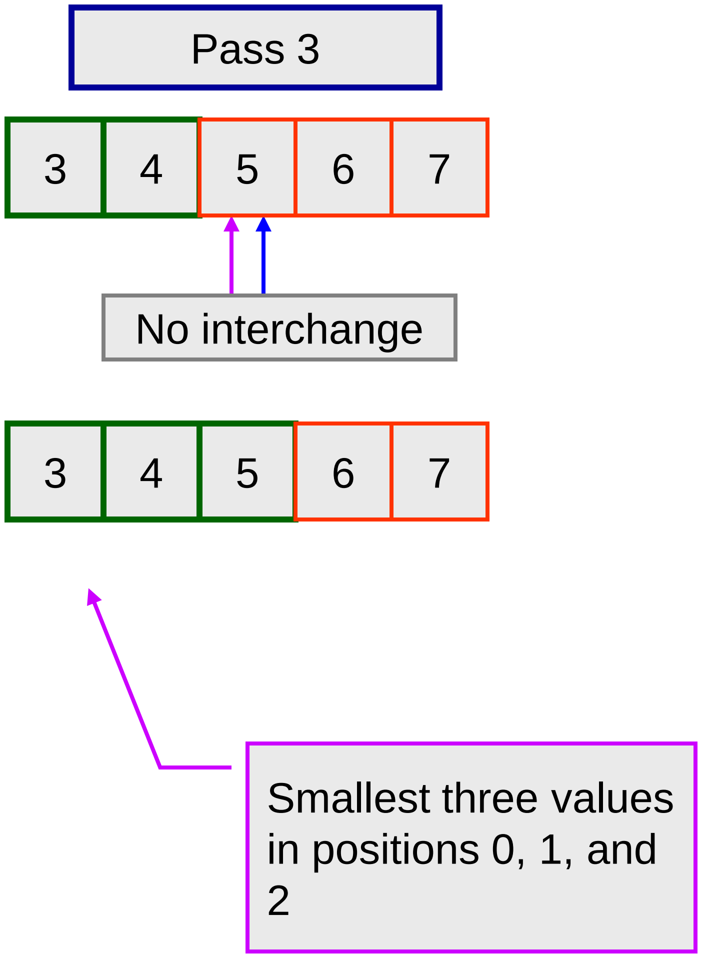
More On Using Functions In C

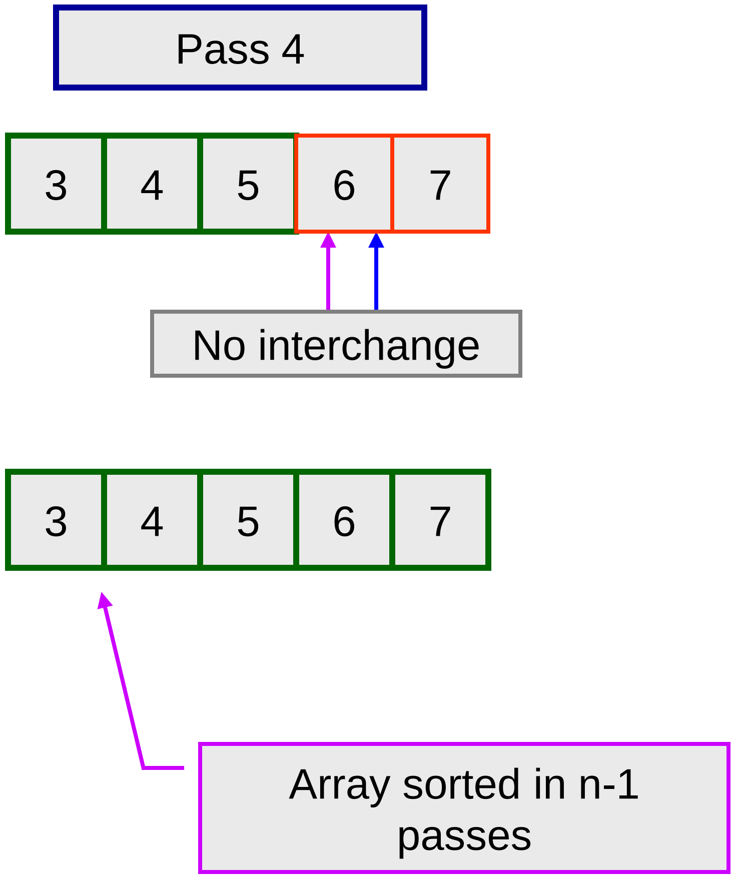
- Let's write another program in C that uses multiple functions to complete its task.
- In the last section of notes we wrote a program that used a bubble-sort technique to sort an array of integers. This time we'll write a program that does exactly the same thing, except it will use a technique known as a selection sort.
- A description of how a selection sort works is shown on the next two pages. In the bubble-sort, the function "bubbled" the smallest value in the array to the first position in the array on the first pass, the second smallest value to the second position on the second pass, and so on. The selection sort, "selects" the smallest value on the first pass and moves it to the first position in the array, then it "selects" the second smallest value on the second pass, and so on.

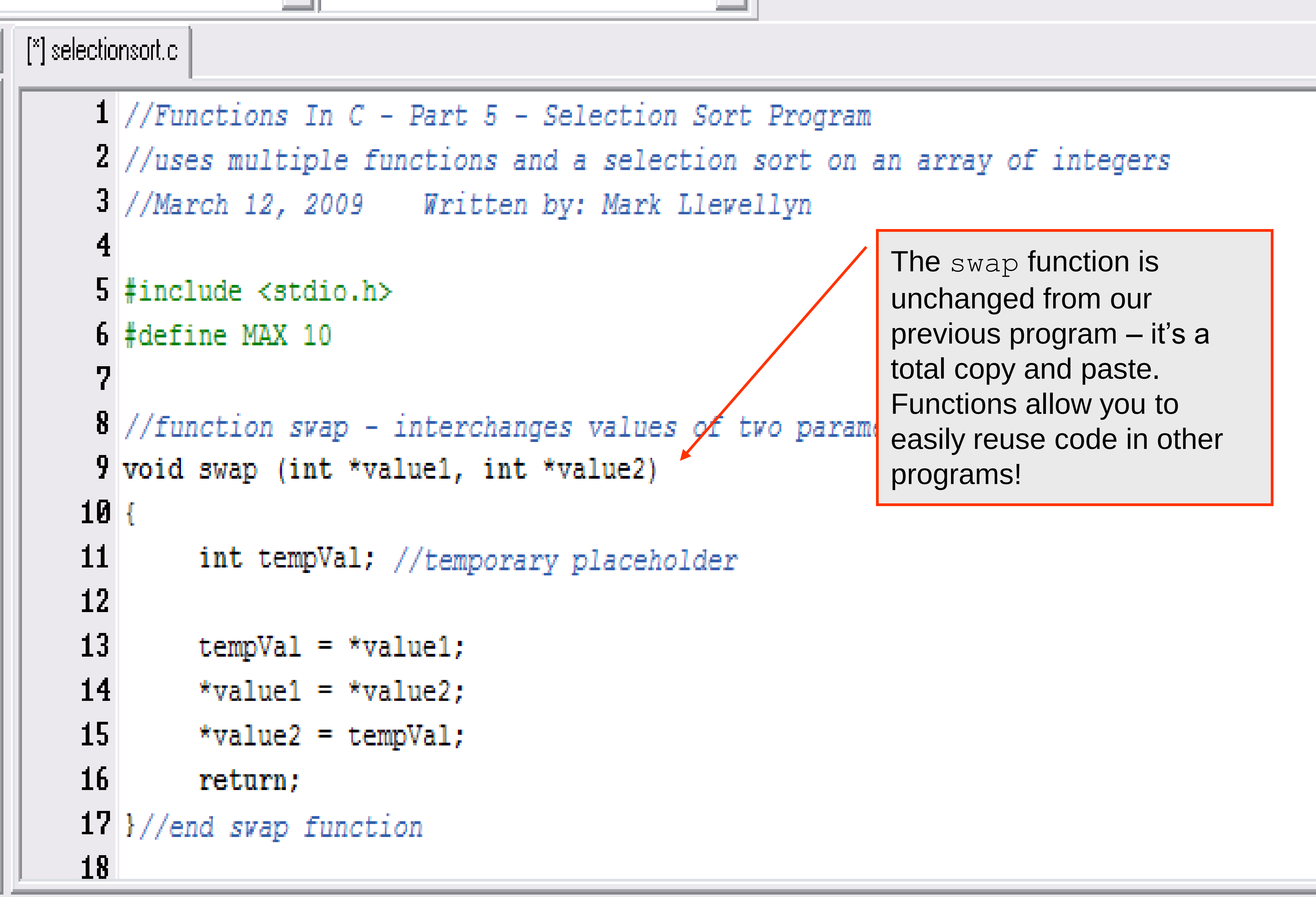
How The Selection Sort Works



How The Selection Sort Works



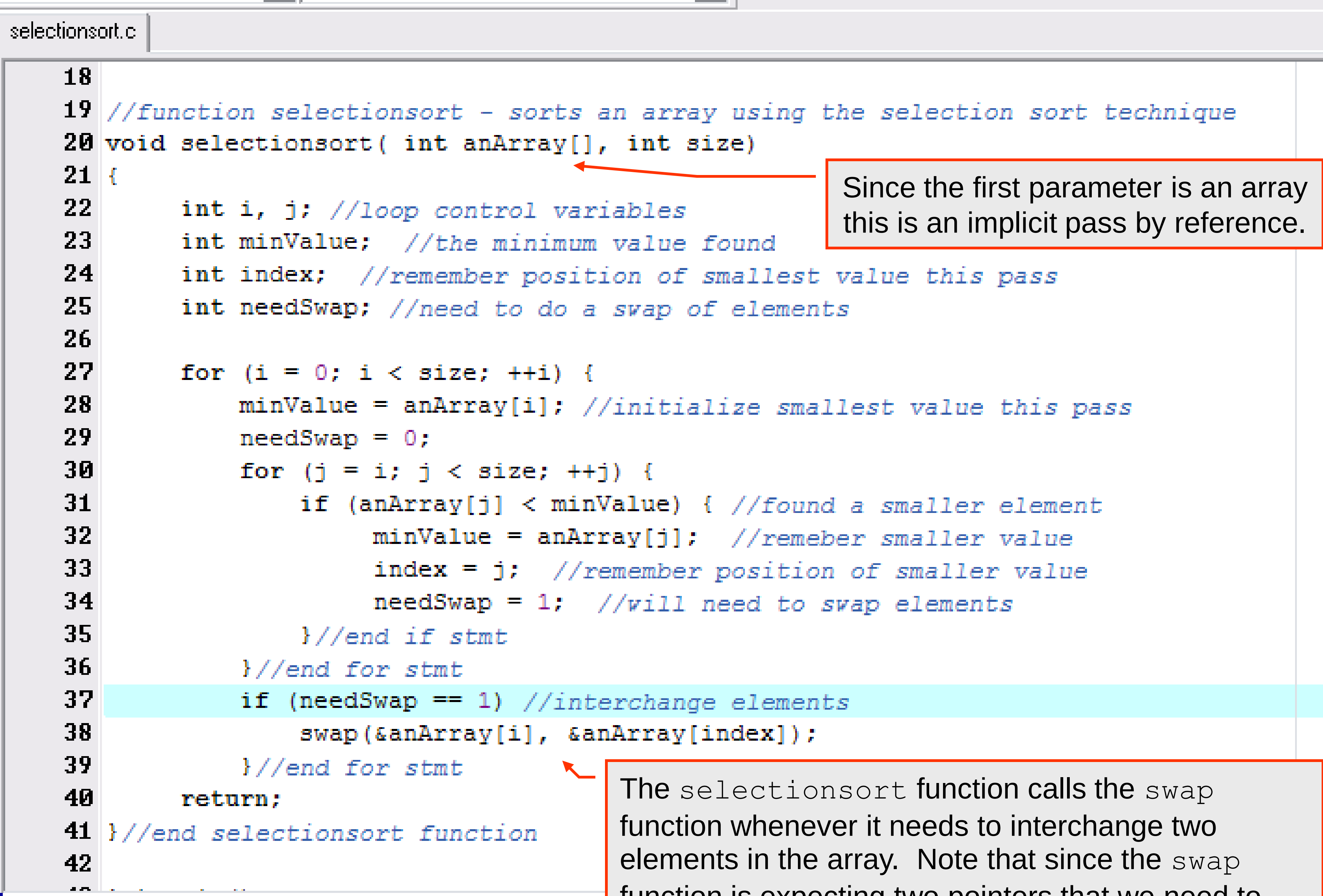




[*] selectionsort.c

```
//Functions In C - Part 5 - Selection Sort Program
//uses multiple functions and a selection sort on an array of integers
//March 12, 2009    Written by: Mark Llewellyn

#include <stdio.h>
#define MAX 10

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

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

```

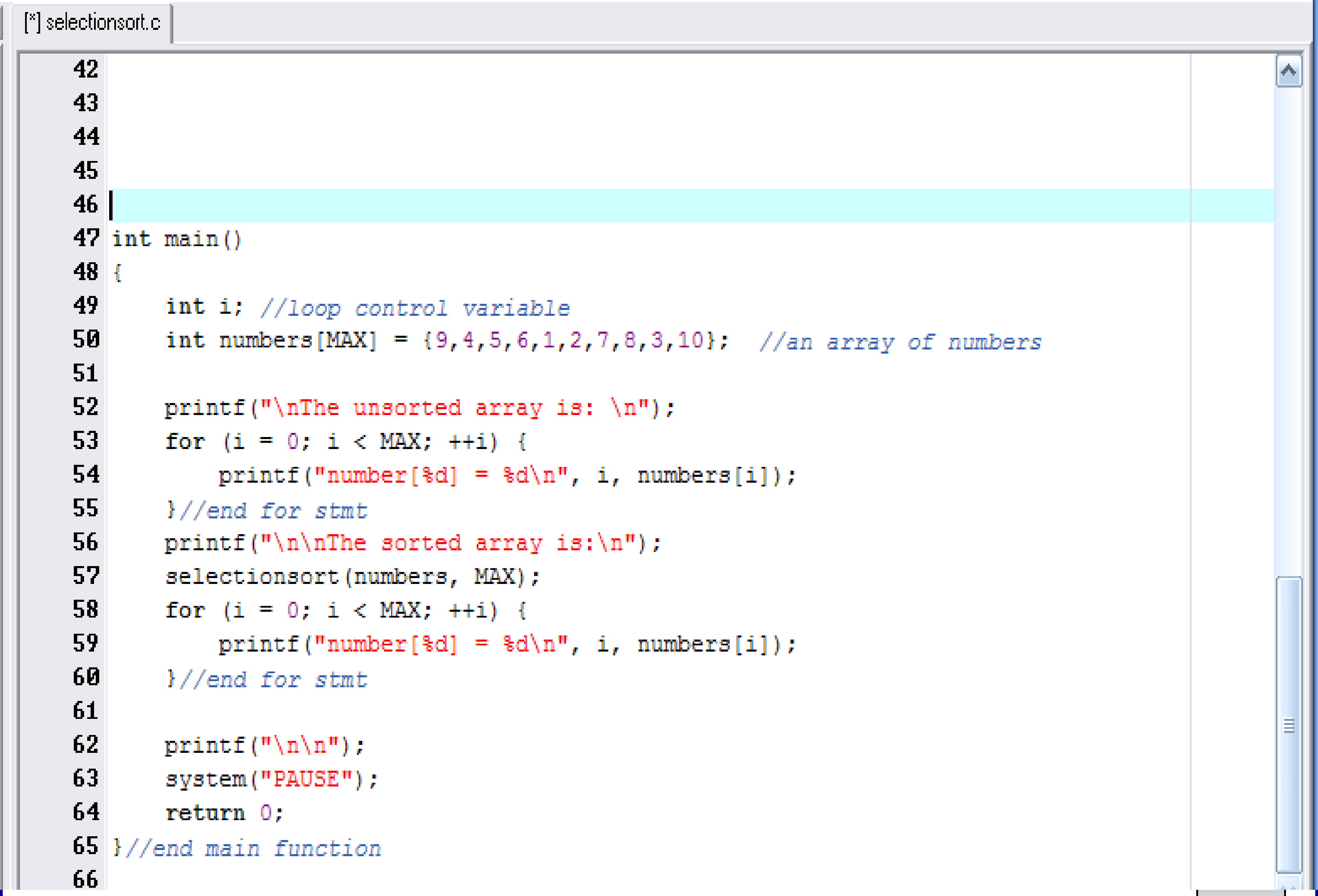
The `swap` function is unchanged from our previous program – it's a total copy and paste. Functions allow you to easily reuse code in other programs!


selectionsort.c

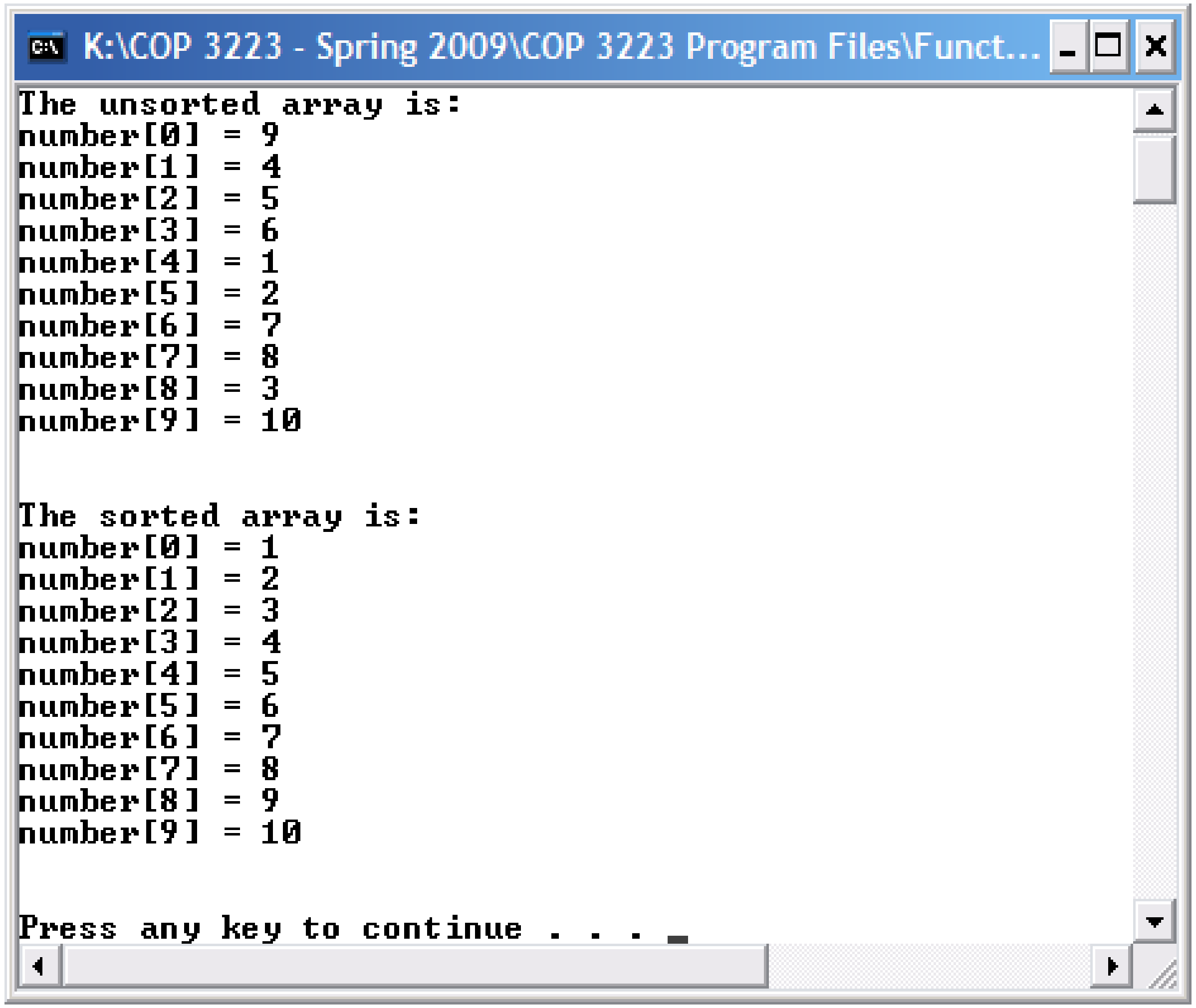
```c

//function selectionsort - sorts an array using the selection sort technique
void selectionsort( int anArray[], int size)
{
    int i, j; //loop control variables
    int minValue;  //the minimum value found
    int index;  //remember position of smallest value this pass
    int needSwap; //need to do a swap of elements

    for (i = 0; i < size; ++i) {
        minValue = anArray[i]; //initialize smallest value this pass
        needSwap = 0;
        for (j = i; j < size; ++j) {
            if (anArray[j] < minValue) { //found a smaller element
                minValue = anArray[j];  //remeber smaller value
                index = j;  //remember position of smaller value
                needSwap = 1;  //will need to swap elements
            }//end if stmt
        }//end for stmt
        if (needSwap == 1) //interchange elements
            swap(&anArray[i], &anArray[index]);
        }//end for stmt
    return;
}//end selectionsort function

```

Since the first parameter is an array this is an implicit pass by reference.

The `selectionsort` function calls the `swap` function whenever it needs to interchange two elements in the array. Note that since the `swap` function is expecting two pointers that we need to send the addresses of the array elements to `swap`.



```
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");
    selectionsort(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 3223 Program Files\Funct...

```
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 . . . _
```

More On Using Functions In C

- Let’s tackle another problem.
- We want to enter a series of integer numbers into an array from a file. The number of integers in the file is unknown in advance, but we’ll assume that the file will contain no more than 100 integers.
- Once the numbers are entered into the array – we’ll use a function to do this; we’ll ask the user to enter a value that would correspond to the smallest integer in the list, the next to the smallest, …, the next to the largest, or the largest value in the list and return to them this value.
- For example, if the list of numbers were: 4 6 19 23 5 77 14 and the user wanted to see the 3rd largest number in the list, the program should return the value 19 to them. Similarly, if they asked to see the next to the smallest the program should return 5.

More On Using Functions In C

- How would you approach solving this problem?
- Let’s do it this way:
  - One function will simply get the integers from the file and put them into the array. It will need the array as a parameter, which of course must be passed by reference (i.e., the address of the first element in the array). It should return the number of elements that were placed into the array.
  - Another function will handle the search for the element the user wants to view. This function will need the array and its actual size as parameters and should return the integer value the user specified.
- How should the searching being done? Think!!!

More On Using Functions In C

- What would you do if I gave you 1000 random integer numbers and asked you to give me back the 12th largest value of the 1000 integers?

- **Answer:** Sort the integers first, then simply look in the position 12 values from the largest element in the array and there is your 12th largest integer. Better yet…we've already solved this problem (i.e., sorting integers in an array) with either our `bubble sort` or `selection sort` programs. We won't even have to write any new code to solve the searching problem.

More On Using Functions In C

- The complete solution to this problem is on the course code page, I've only included new code here.
- The complete solution contains five functions (including the `main` function):

1. `main` – calls the function to fill the array, gets the user's search value, and prints the results.
2. `getValues` – handles the file I/O, fills the array, and returns the number of elements in the array.
3. `search` – handles the search by calling the selectionsort function.
4. `selectionsort` – uses a selection sort method to sort the array.
5. `swap` – used by selection sort to interchange element positions.

More On Using Functions In C

- The search function returns the xth largest value in the set of numbers.
- It does this by sorting the array and then returning the value in the xth position in the sorted array.

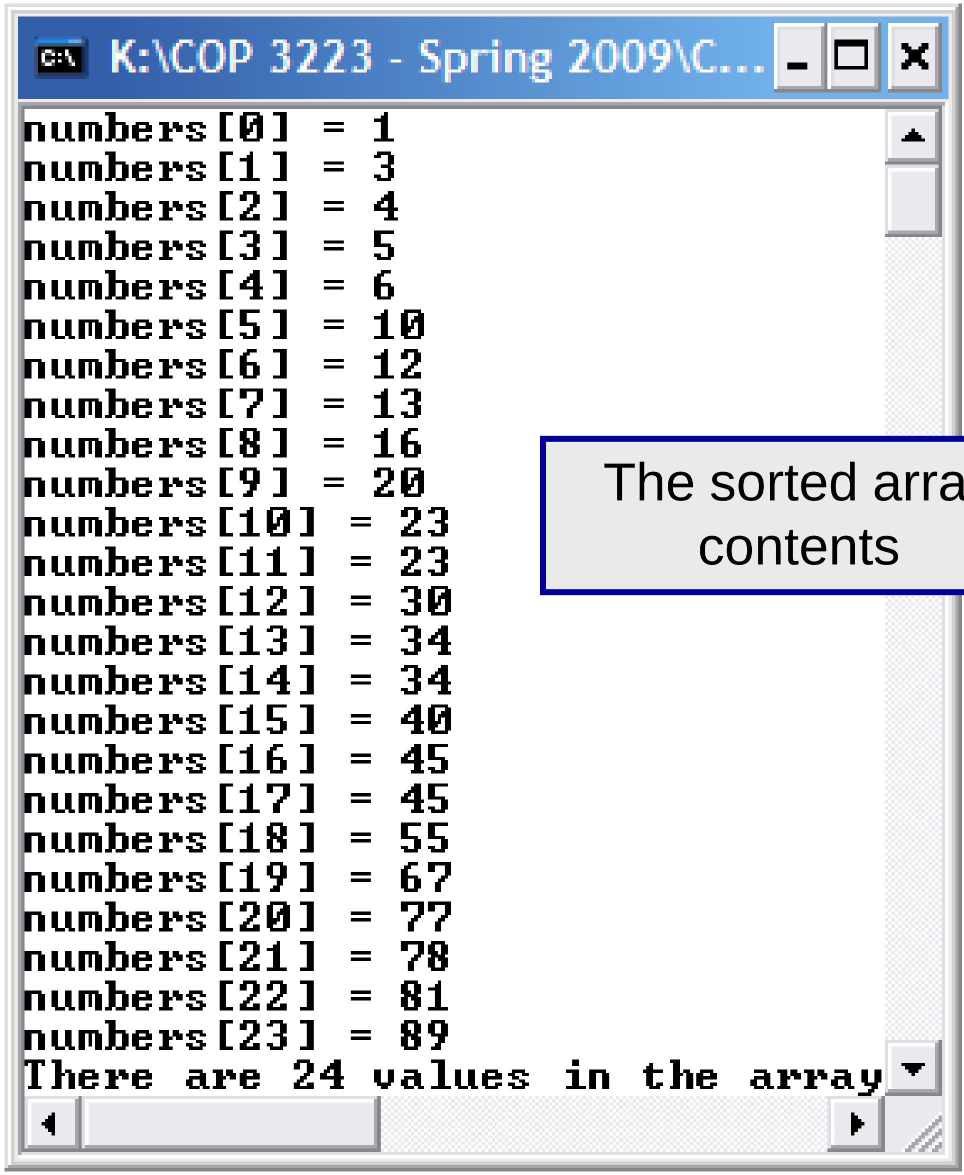
```
//function search - takes an array, the size of the array, the
//search value
int search (int anArray[], int size, int position)
{
    //sort the array
    selectionsort(anArray, size);  //sort the array
    return anArray[position]; //return element in desired position
}//end function search
```

More On Using Functions In C

```
//function to get the values into the array from the file
int getValues(int anArray[])
{
    int count;  //number of integers in the file put into the array
    FILE *inFilePtr; //ptr to input file

     if ( (inFilePtr = fopen("search numbers.dat", "r")) == NULL ) {
         printf("Sorry, could'nt open input file\n");
    }
    else {
         count = 0;
         fscanf(inFilePtr, "%d", &anArray[count]);
         while (!feof(inFilePtr)) {
               count++;
               fscanf(inFilePtr, "%d", &anArray[count]);
         }//end while stmt
         fclose(inFilePtr);
         return count--;
    }//end else stmt
    return 0;  //function failed
}//end getValues function
```

The file contents

```
34
6
78
12
3
1
67
89
55
34
16
23
45
81
23
30
40
20
10
5
4
77
13
45
```

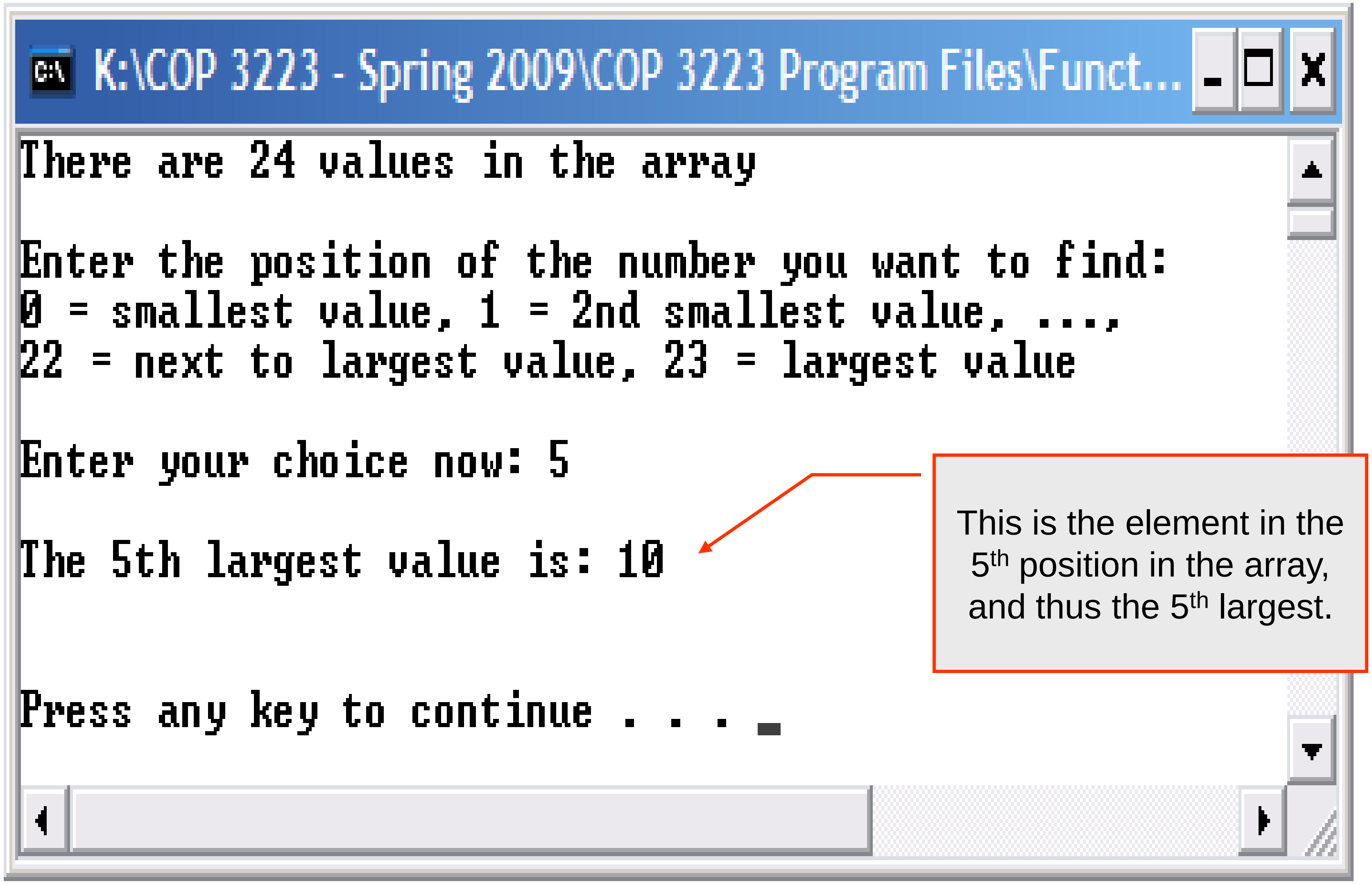


K:\COP 3223 - Spring 2009\C...

```
numbers[0] = 1
numbers[1] = 3
numbers[2] = 4
numbers[3] = 5
numbers[4] = 6
numbers[5] = 10
numbers[6] = 12
numbers[7] = 13
numbers[8] = 16
numbers[9] = 20
numbers[10] = 23
numbers[11] = 23
numbers[12] = 30
numbers[13] = 34
numbers[14] = 34
numbers[15] = 40
numbers[16] = 45
numbers[17] = 45
numbers[18] = 55
numbers[19] = 67
numbers[20] = 77
numbers[21] = 78
numbers[22] = 81
numbers[23] = 89
There are 24 values in the array
```

The sorted array contents

More On Using Functions In C



```
There are 24 values in the array

Enter the position of the number you want to find:
0 = smallest value, 1 = 2nd smallest value, ...,
22 = next to largest value, 23 = largest value

Enter your choice now: 5

The 5th largest value is: 10


Press any key to continue . . .
```

This is the element in the 5th position in the array, and thus the 5th largest.

Large Programs – Creating Your Own Libraries

- As we've now become aware, creating functions allows us to solve small components of a larger problem in isolation and them combine the small components together in order to solve the large problem. Code reuse also factors into the mix when developing large-scale programs.
- A common technique when developing large programs is to divide the program into separate modules stored as separate source files. From a code reuse point of view, we want to group similar functions into libraries and then include the libraries as needed in the larger programs. You've done this from the very first C program you wrote, when you placed the directive `#include <stdio.h>` into your program so that you could use the `printf` function that was defined somewhere other than in your program.

Large Programs – Creating Your Own Libraries

- Convention in C is that such libraries are referenced in source code using header files.
- In C, header files always have a `.h` extension and contain only definitions of data types, function prototypes (the function header, hence the name header file), and C preprocessor commands, and nothing else (although some comments are sometimes included for readability).
- You know how to include header files that refer to functions defined in standard C libraries using the preprocessor directive #include. The same technique is used to refer to user defined header files and libraries, although there is a slight syntax change for user defined header files, as you will shortly see.

Large Programs – Creating Your Own Libraries

- Exactly how header files and their corresponding source files are linked into source code that references them is handled somewhat differently on various platforms and operating systems. For example, the creation/linking process is a bit different on a Unix/Linux based system than it is on a Windows system.
- In addition, certain platforms (namely Unix/Linux) have a number of utility programs that can assist in the creating and linking of libraries and header files.
- I'll show you how to do this using the DevC++ environment on Window with which most of you will be most familiar. Those of you using Macs will follow a similar process inside DevC++, but the external command prompts will undoubtedly be somewhat different.

Large Programs – Creating Your Own Libraries

- Large programs and those that can easily access user-defined libraries are most easily created in the DevC++ environment as a Project.
- In DevC++ go to File, New, Project and select Empty Project.
- Enter a name for the project and then click the radio button for a C project.
- Click OK, and then select the directory in which to store the project.
- If you have not already created the source file, the create it as you normally do and then save it and add it to the project. If the source file is already created, then simply click on Project and select Add to Project.

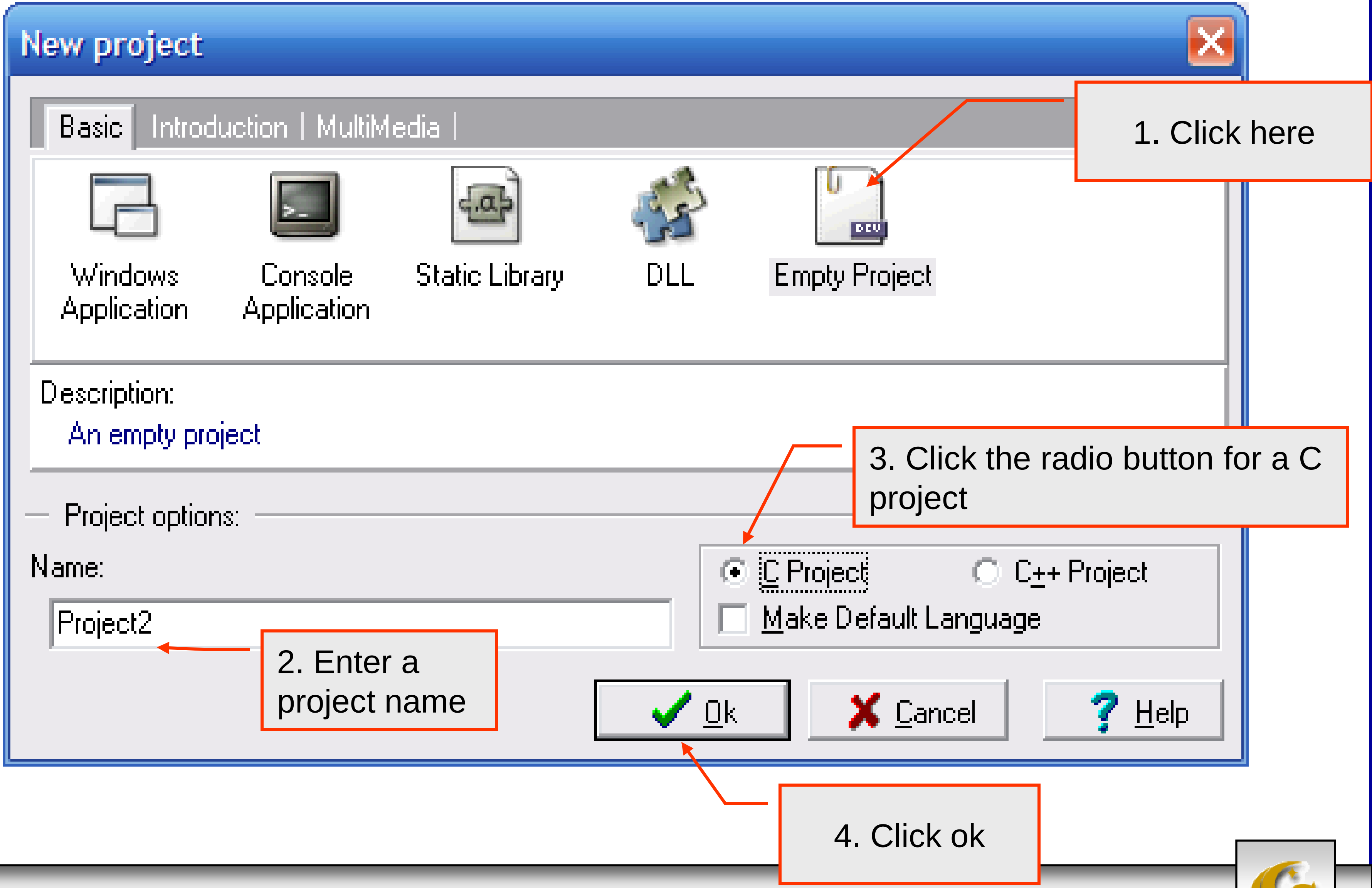
Large Programs – Creating Your Own Libraries

- To add header files to the project, there are several possible techniques that are commonly used. One of the simplest is to make a copy of the original library source file, then edit out all but the function headers from this file (and comments if you choose). Place a semi-colon at the end of every function header and then save the file with a `.h` extension.
- Note that it is important that the header file contain only function headers and not code. Different environments handle this in various ways, but most will generate an error if code is present in a header file.
- Once you have all the files that comprise the project together, the go to Execute and select Compile. This will compile every file within the project (you'll see the compile window and it moves through each of the files). If you want to only compile a single file, the go to Execute and select Compile Current File.

Large Programs – Creating Your Own Libraries

- To run the source file that contains the main function:
- In Windows, you'll need to open a Command Prompt window and switch to the directory that contains the project files.
- Once there, simply type the name of the `.exe` file, which should be the name you gave the project. Note that it is not the name of the directory that contains the project. In my case, I named the folder Project 2, but within DevC++ the project was named Project2 (no space).
- This will execute the object file that contains the main function. In the example this will be the file named `program using libraries.o`.
- The following several pages illustrate how to set up a project in DevC++, create header files, and execute the program.

Large Programs – Creating Your Own Libraries



Description:
An empty project

Project options:
Name: Project2

[✓ C Project] C++ Project
Make Default Language

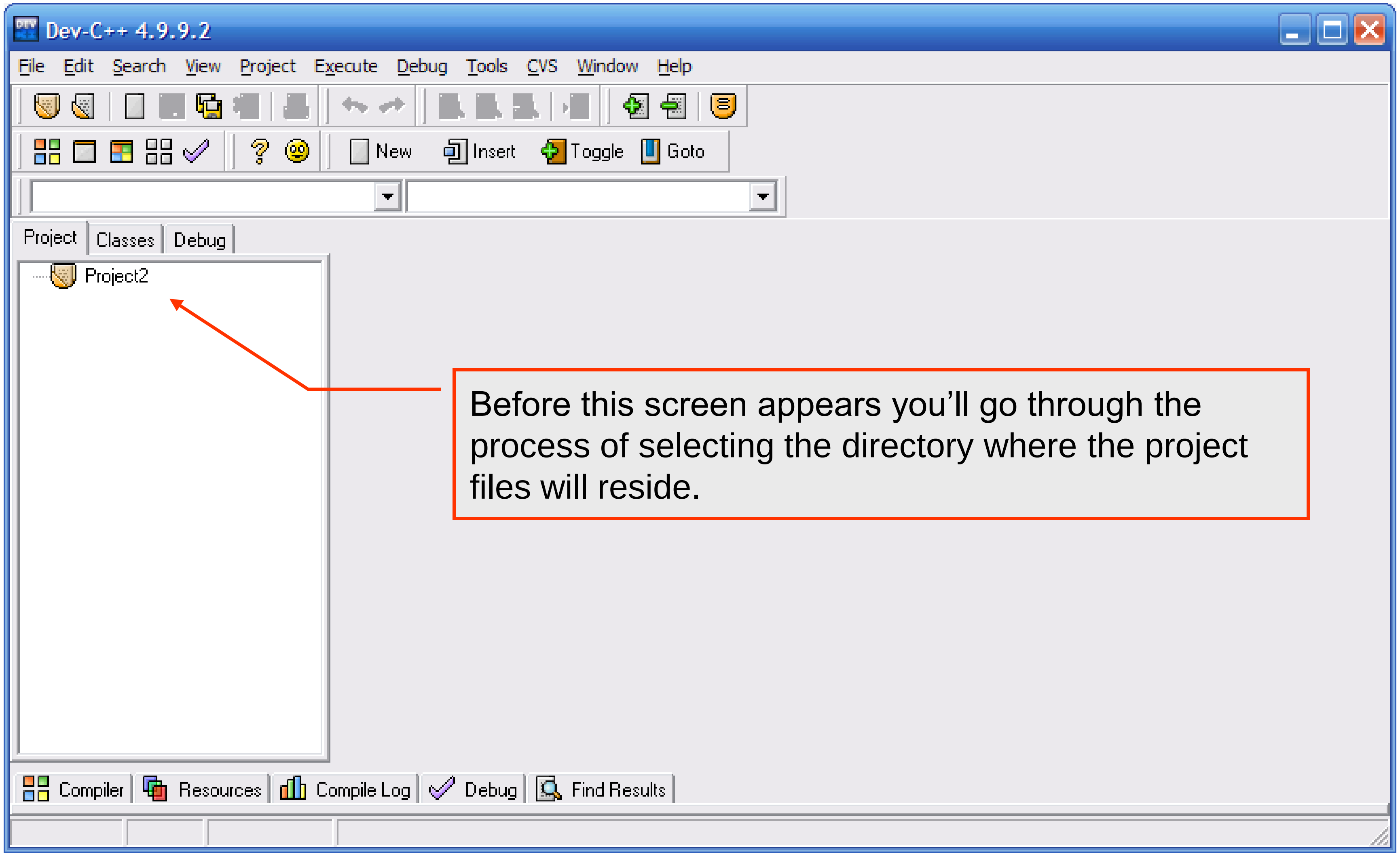
1. Click here

2. Enter a project name

3. Click the radio button for a C project

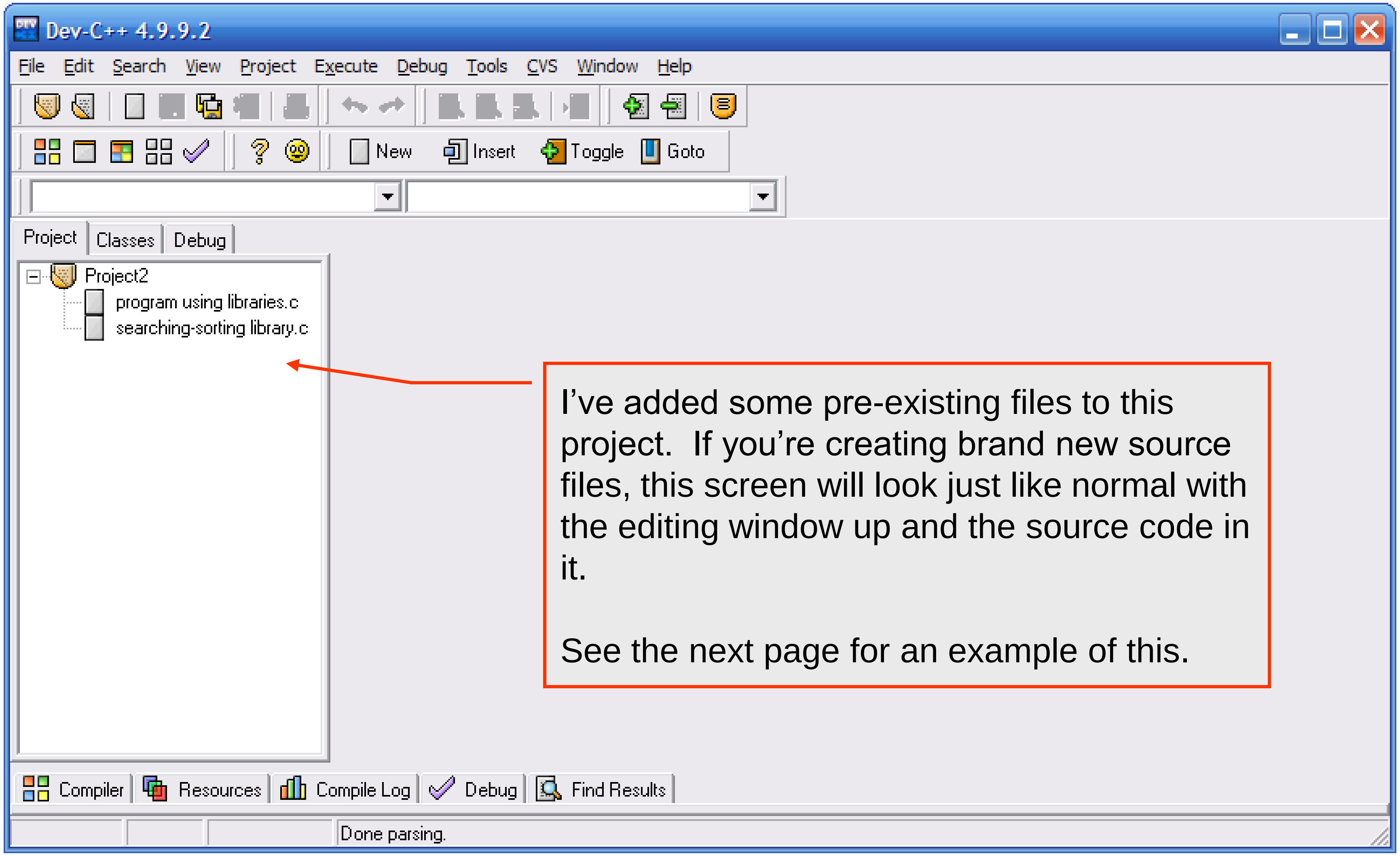
4. Click ok

Large Programs – Creating Your Own Libraries


Dev-C++ 4.9.9.2
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
Project Classes Debug
Project2
Compiler Resources Compile Log Debug Find Results

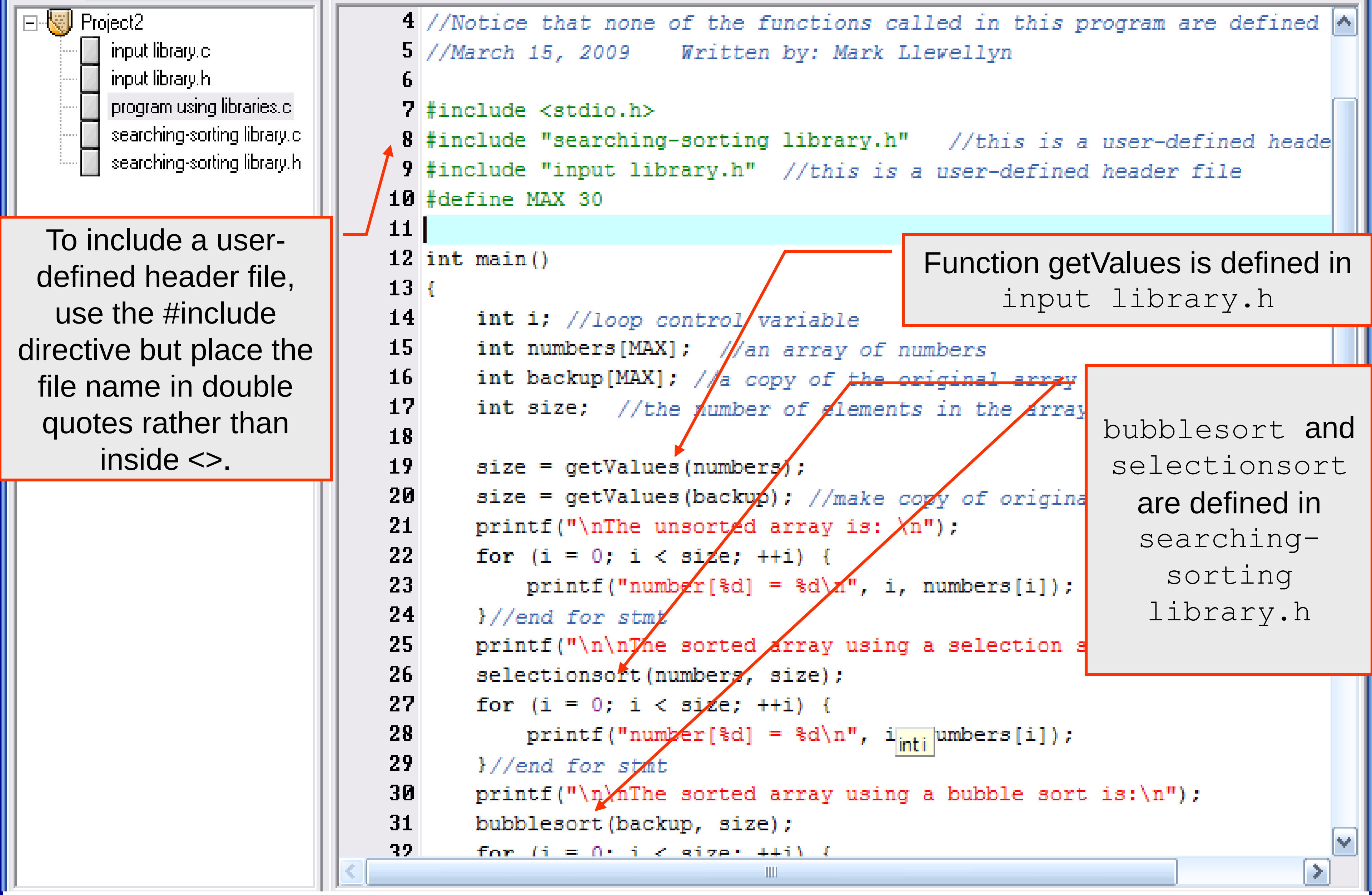
Before this screen appears you'll go through the process of selecting the directory where the project files will reside.

Large Programs – Creating Your Own Libraries


Dev-C++ 4.9.9.2
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
Project Classes Debug
Project2
program using libraries.c
searching-sorting library.c
Compiler Resources Compile Log Debug Find Results
Done parsing.

I've added some pre-existing files to this project. If you're creating brand new source files, this screen will look just like normal with the editing window up and the source code in it.

See the next page for an example of this.



Project2
- input library.c
- input library.h
- program using libraries.c
- searching-sorting library.c
- searching-sorting library.h

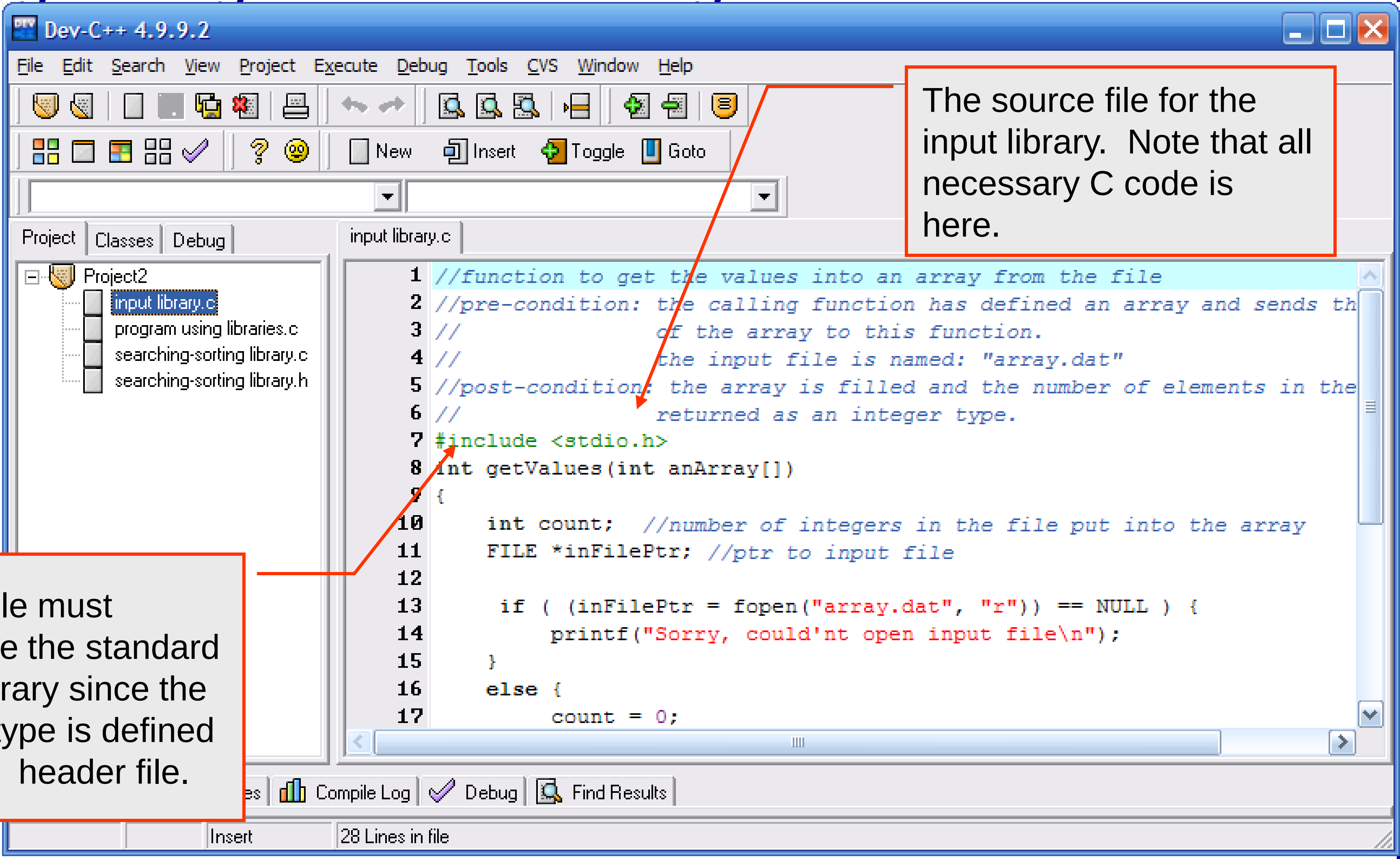
```
//Notice that none of the functions called in this program are defined
//March 15, 2009    Written by: Mark Llewellyn

#include <stdio.h>
#include "searching-sorting library.h"   //this is a user-defined heade
#include "input library.h"  //this is a user-defined header file
#define MAX 30

int main()
{
    int i; //loop control variable
    int numbers[MAX];  //an array of numbers
    int backup[MAX]; //a copy of the original array
    int size;  //the number of elements in the array

    size = getValues(numbers);
    size = getValues(backup); //make copy of origina
    printf("\nThe unsorted array is: \n");
    for (i = 0; i < size; ++i) {
        printf("number[%d] = %d\n", i, numbers[i]);
    }//end for stmt
    printf("\n\nThe sorted array using a selection s
    selectionsort(numbers, size);
    for (i = 0; i < size; ++i) {
        printf("number[%d] = %d\n", i int i umbers[i]);
    }//end for stmt
    printf("\n\nThe sorted array using a bubble sort is:\n");
    bubblesort(backup, size);
    for (i = 0; i < size; ++i) {
```

To include a user-defined header file, use the #include directive but place the file name in double quotes rather than inside <>.

Function getValues is defined in `input library.h`

`bubblesort` and `selectionsort` are defined in `searching-sorting library.h`

Large Programs – Creating Your Own Libraries

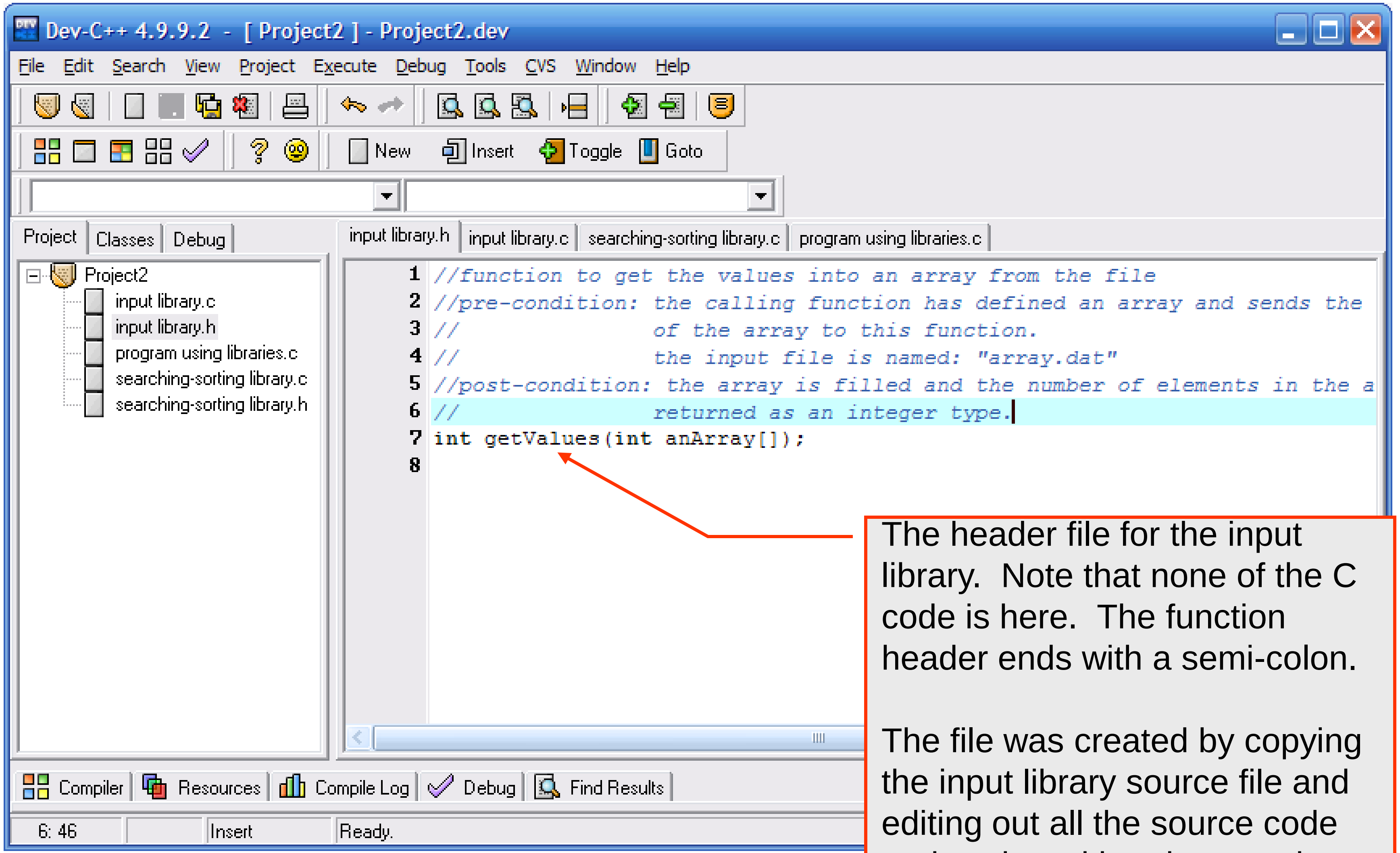

Dev-C++ 4.9.9.2
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
Project Classes Debug
Project2
input library.c
program using libraries.c
searching-sorting library.c
searching-sorting library.h
input library.c
Compile Log Debug Find Results
Insert
28 Lines in file

```c
//function to get the values into an array from the file
//pre-condition: the calling function has defined an array and sends th
//                of the array to this function.
//                the input file is named: "array.dat"
//post-condition: the array is filled and the number of elements in the
//                returned as an integer type.
#include <stdio.h>
int getValues(int anArray[])
{
    int count;  //number of integers in the file put into the array
    FILE *inFilePtr; //ptr to input file

     if ( (inFilePtr = fopen("array.dat", "r")) == NULL ) {
         printf("Sorry, could'nt open input file\n");
     }
     else {
         count = 0;
```

The source file for the input library. Note that all necessary C code is here.

This file must include the standard I/O library since the FILE type is defined in this header file.

Large Programs – Creating Your Own Libraries

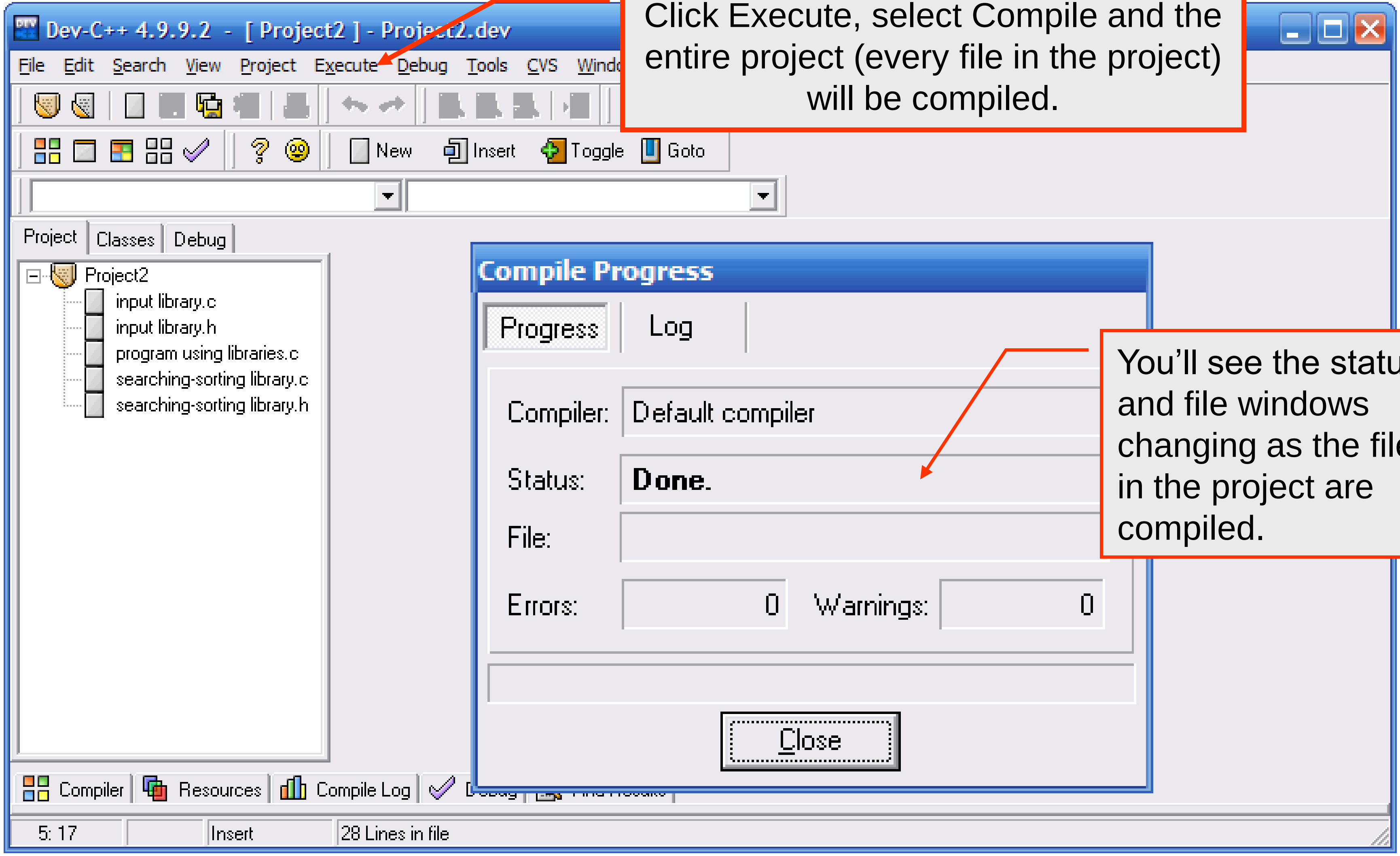

Dev-C++ 4.9.9.2 - [Project2] - Project2.dev
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
Project Classes Debug
input library.h input library.c searching-sorting library.c program using libraries.c
Project2
input library.c
input library.h
program using libraries.c
searching-sorting library.c
searching-sorting library.h
Compiler Resources Compile Log Debug Find Results
6: 46 Insert Ready.

```
//function to get the values into an array from the file
//pre-condition: the calling function has defined an array and sends the
//                of the array to this function.
//                the input file is named: "array.dat"
//post-condition: the array is filled and the number of elements in the a
//                returned as an integer type.
int getValues(int anArray[]);
```

The header file for the input library. Note that none of the C code is here. The function header ends with a semi-colon.

The file was created by copying the input library source file and editing out all the source code and saving with a .h extension.

Large Programs – Creating Your Own Libraries


Dev-C++ 4.9.9.2 - [Project2] - Project2.dev
File Edit Search View Project Execute Debug Tools CVS Windo
New Insert Toggle Goto
Project Classes Debug
Project2
input library.c
input library.h
program using libraries.c
searching-sorting library.c
searching-sorting library.h
Compiler Resources Compile Log Debug Find Results
5: 17 Insert 28 Lines in file

Compile Progress

Progress Log

Compiler: Default compiler

Status: Done.

File:

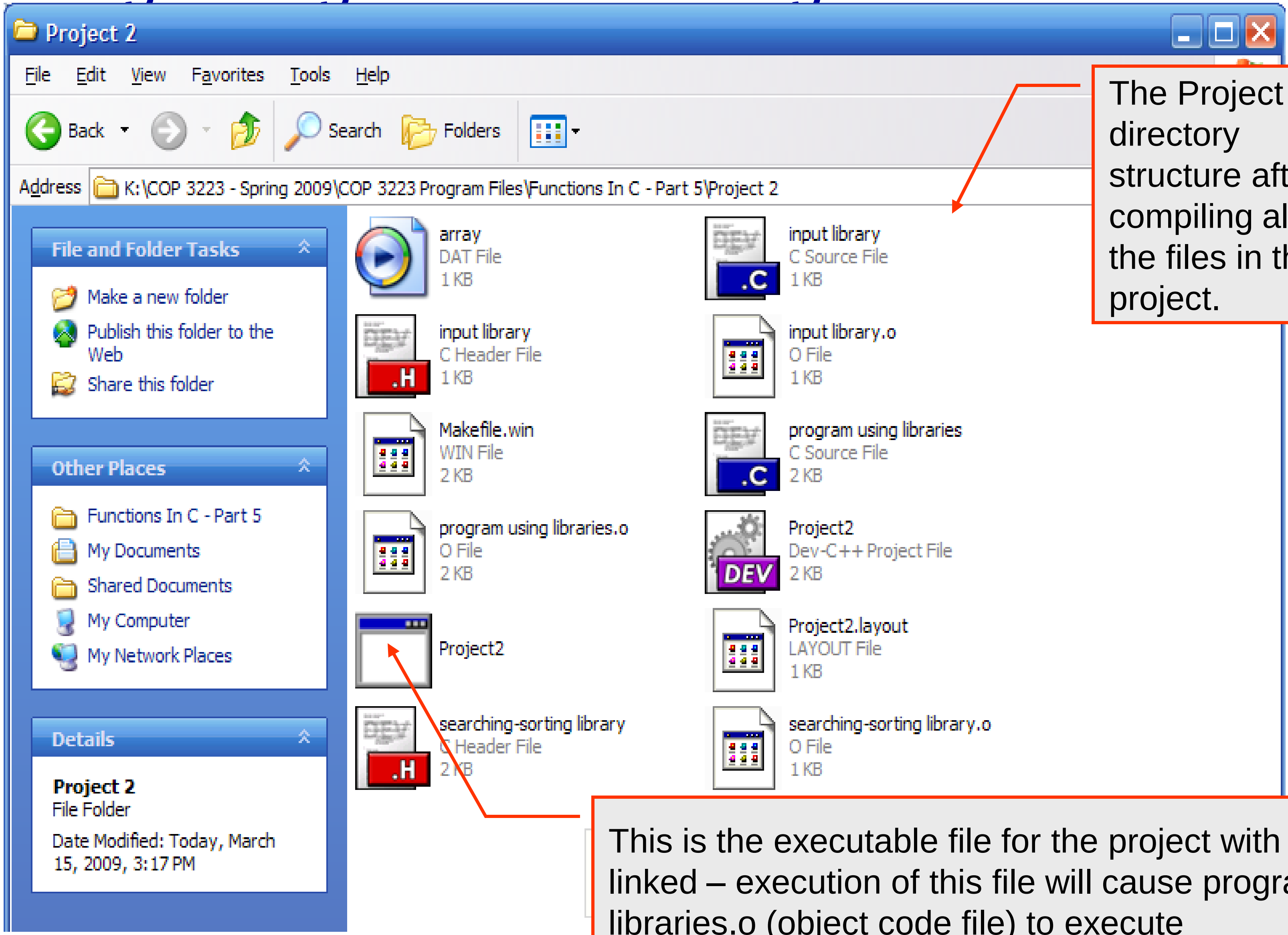
Errors: 0 Warnings: 0

Close

Click Execute, select Compile and the entire project (every file in the project) will be compiled.

You'll see the status and file windows changing as the files in the project are compiled.

Large Programs – Creating Your Own Libraries



Project 2

File Edit View Favorites Tools Help

Back Search Folders

Address K:\COP 3223 - Spring 2009\COP 3223 Program Files\Functions In C - Part 5\Project 2

File and Folder Tasks

Make a new folder

Publish this folder to the Web

Share this folder

Other Places

Functions In C - Part 5

My Documents

Shared Documents

My Computer

My Network Places

Details

Project 2
File Folder

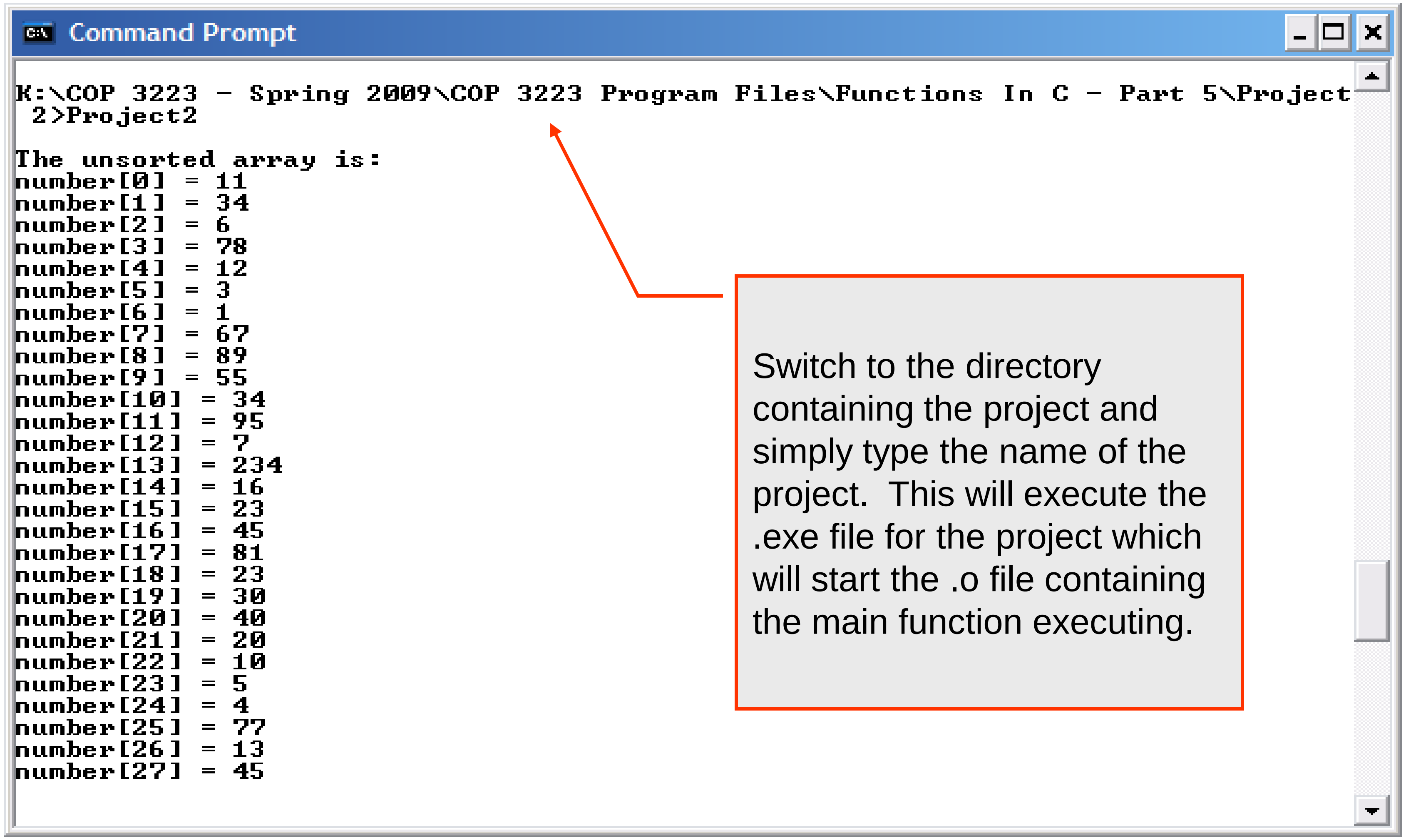
Date Modified: Today, March 15, 2009, 3:17 PM

Name	Type	Size
array	DAT File	1 KB
input library	C Source File	1 KB
input library	C Header File	1 KB
input library.o	O File	1 KB
Makefile.win	WIN File	2 KB
program using libraries	C Source File	2 KB
program using libraries.o	O File	2 KB
Project2	Dev-C++ Project File	2 KB
Project2		
Project2.layout	LAYOUT File	1 KB
searching-sorting library	C Header File	2 KB
searching-sorting library.o	O File	1 KB

The Project 2 directory structure after compiling all of the files in the project.

This is the executable file for the project with all libraries linked – execution of this file will cause program using libraries.o (object code file) to execute

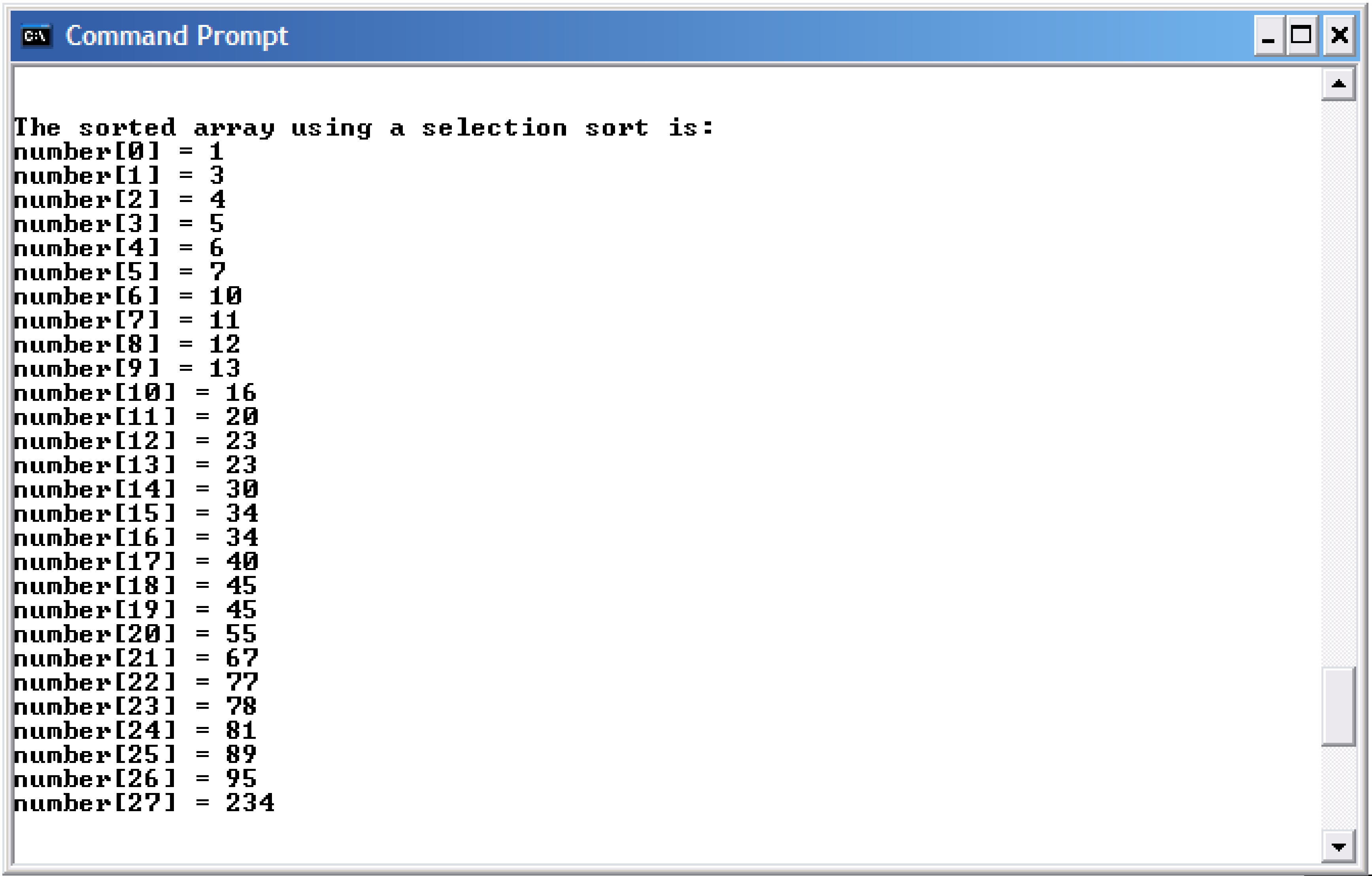
Large Programs – Creating Your Own Libraries



```
K:\COP 3223 - Spring 2009\COP 3223 Program Files\Functions In C - Part 5\Project
 2>Project2

The unsorted array is:
number[0] = 11
number[1] = 34
number[2] = 6
number[3] = 78
number[4] = 12
number[5] = 3
number[6] = 1
number[7] = 67
number[8] = 89
number[9] = 55
number[10] = 34
number[11] = 95
number[12] = 7
number[13] = 234
number[14] = 16
number[15] = 23
number[16] = 45
number[17] = 81
number[18] = 23
number[19] = 30
number[20] = 40
number[21] = 20
number[22] = 10
number[23] = 5
number[24] = 4
number[25] = 77
number[26] = 13
number[27] = 45
```

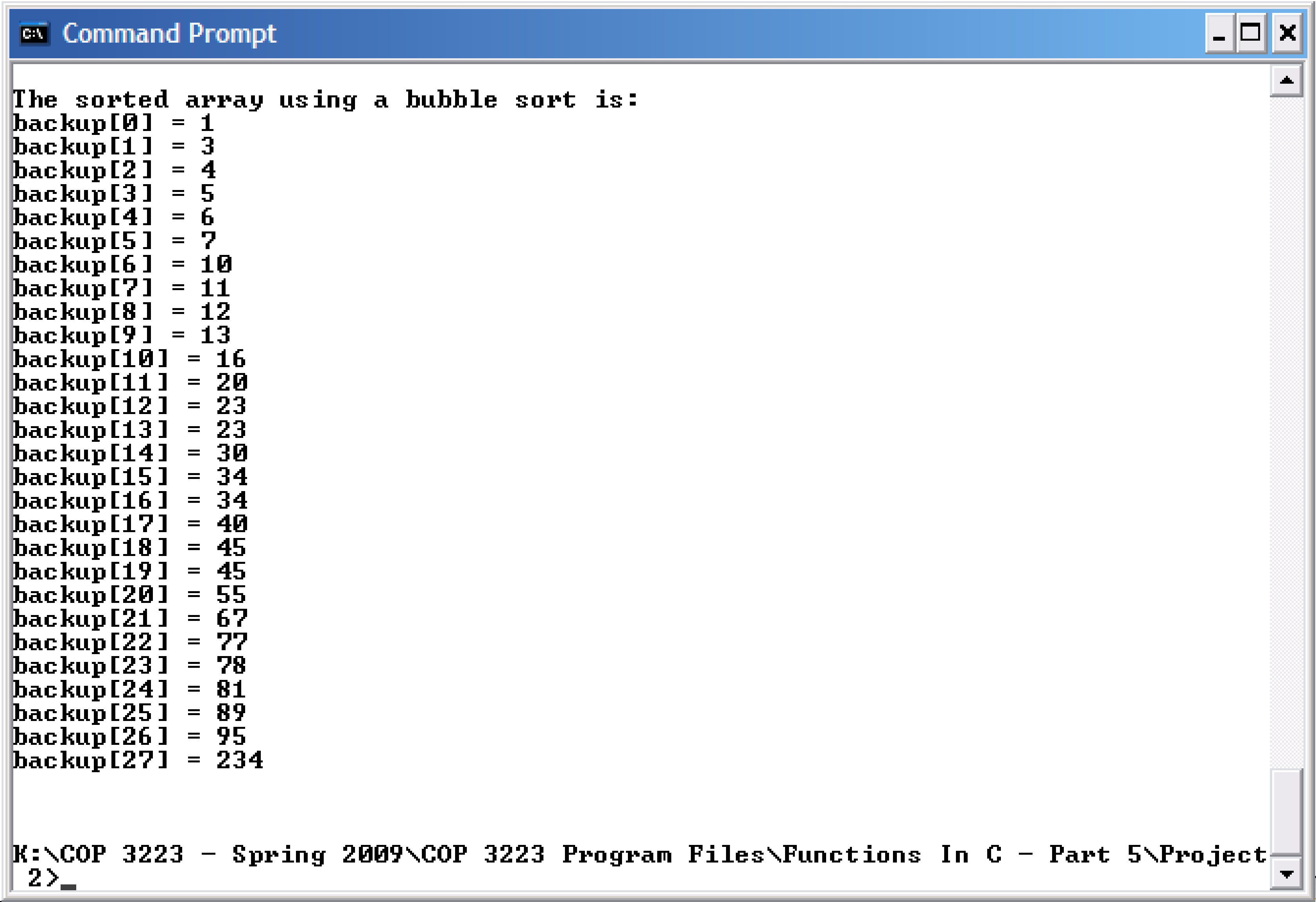
Switch to the directory containing the project and simply type the name of the project. This will execute the .exe file for the project which will start the .o file containing the main function executing.

Large Programs – Creating Your Own Libraries



```
The sorted array using a selection sort is:
number[0] = 1
number[1] = 3
number[2] = 4
number[3] = 5
number[4] = 6
number[5] = 7
number[6] = 10
number[7] = 11
number[8] = 12
number[9] = 13
number[10] = 16
number[11] = 20
number[12] = 23
number[13] = 23
number[14] = 30
number[15] = 34
number[16] = 34
number[17] = 40
number[18] = 45
number[19] = 45
number[20] = 55
number[21] = 67
number[22] = 77
number[23] = 78
number[24] = 81
number[25] = 89
number[26] = 95
number[27] = 234
```

Large Programs – Creating Your Own Libraries


Command Prompt

```
The sorted array using a bubble sort is:
backup[0] = 1
backup[1] = 3
backup[2] = 4
backup[3] = 5
backup[4] = 6
backup[5] = 7
backup[6] = 10
backup[7] = 11
backup[8] = 12
backup[9] = 13
backup[10] = 16
backup[11] = 20
backup[12] = 23
backup[13] = 23
backup[14] = 30
backup[15] = 34
backup[16] = 34
backup[17] = 40
backup[18] = 45
backup[19] = 45
backup[20] = 55
backup[21] = 67
backup[22] = 77
backup[23] = 78
backup[24] = 81
backup[25] = 89
backup[26] = 95
backup[27] = 234


K:\COP 3223 - Spring 2009\COP 3223 Program Files\Functions In C - Part 5\Project
 2>_
```

Practice Problems

1. Go back to the program you wrote for Assignment #3 leap year problem and rewrite the program using a function that will return the number of days that are in the year passed to it as a parameter.

2. Go back to the program you wrote for Assignment #3, the where to study problem (the shortest distance problem) and rewrite the program so that it uses a function to determine the distance between two locations passed to it.

Practice Problems

3. Go back to Assignment #4 and rewrite your program so that it includes two functions in addition to the main function. One function reads in all the temperature data from the file and determines the percentage of time the air conditioner is on each hour and puts these values into an array. The second function prints the hourly utilization graph. The main function will print the total percentage of time the air conditioner was on during the 24 hour period.