

# COP 3223: C Programming Spring 2009

## Introduction To C - Part 1

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



# Introduction To C

- The C programming language was developed in the early 1970s by Dennis Ritchie and Ken Thompson in order to assist in their development of the Unix operating system.
- Since that time, the C language has been standardized according to both ANSI (American National Standards Institute) and ISO (International Standards Organization) standards. The current mostly widely used version of C is the ISO standard commonly referred to as C89.
  - The newer standard, known as C99, has not yet been universally adopted. So when most people refer to “standard C”, it is the C89 standard that is assumed. It will probably be some years before all C compilers will be C99 compliant.



# Introduction To C

- Many modern programming languages have been influenced by C.
- Java, Perl, C#, and C++, just to mention a few, have all been heavily influenced by the C language. Most of these languages have a syntax which is very C-like, and include many of the basic commands and data types that are defined in the C language.



# What Is C?

- **C is considered a low-level language.** It was primarily developed to write operating systems which requires many low-level instructions very close to an assembler language. (Assembler languages are very low-level languages, just one step up from machine language. They are cryptic languages but provide many direct machine level commands and are thus very efficient languages for low level programming.)
- **C is a small language** compared to many modern programming languages. This makes it a nice language to learn as a first programming language.
- **C is a permissive language**, which means that C (i.e., the compiler) assumes that you know what you are doing and allows you more latitude than many other programming languages. This is both good and bad, as we will see later. It is good because you can write working programs without being required to have extensive error checking , but bad because it is therefore easier to write incorrect programs.



# Strengths of C

- **C is efficient.** Since it was primarily developed for applications where assembly language had been traditionally used, it was crucial that C programs run quickly and in limited amounts of memory.
- **C is portable.** Although portability wasn't a primary goal of C, it has turned out to be one of the strengths of the language. Portability means that the same program can be compiled on different machines and still run correctly (provide the same functionality) on any machine on which it is compiled and executed. The standardization of the language has also enhanced its portability.
- **C is powerful.** C contains a large collection of data types and operators that combine to make the language powerful, meaning that you can accomplish quite a bit with a relatively small amount of code (C commands or instructions).
- **C is flexible.** Although originally designed for systems-level programming, it can be used for virtually any application.



# Weaknesses of C

- **C programs can be error-prone.** The flexibility of C makes it an error-prone language. Programming mistakes that would be caught by many other language compilers will not be detected by a C compiler. In this respect, C is similar to assembly language where logic errors will not be detectable until the program is in execution. As we move through the semester, we'll show you ways to make your C programs as “bullet-proof” as possible and avoid many of the pitfalls that can lead to errors in C programs.
- **C can be difficult to understand.** Although C is a small language, it has many features which are unique to C (features not found in many/any other programming languages). Many of these features can be combined in a great number of ways, which although obvious to the original developer of the code, may make the code hard to read or understand for others. This is why following certain standards and conventions can be helpful, so that anyone can understand your C program.



# The Use of C

- In this course, we'll show you how to effectively use the C language to write application programs that take advantage of the strengths of C and minimize its weaknesses.
- We'll do this by stressing the use of good coding conventions (how to write C programs in a proper style), taking advantage of existing code libraries, avoiding common pitfalls that many C programmers make, and using standard C code.

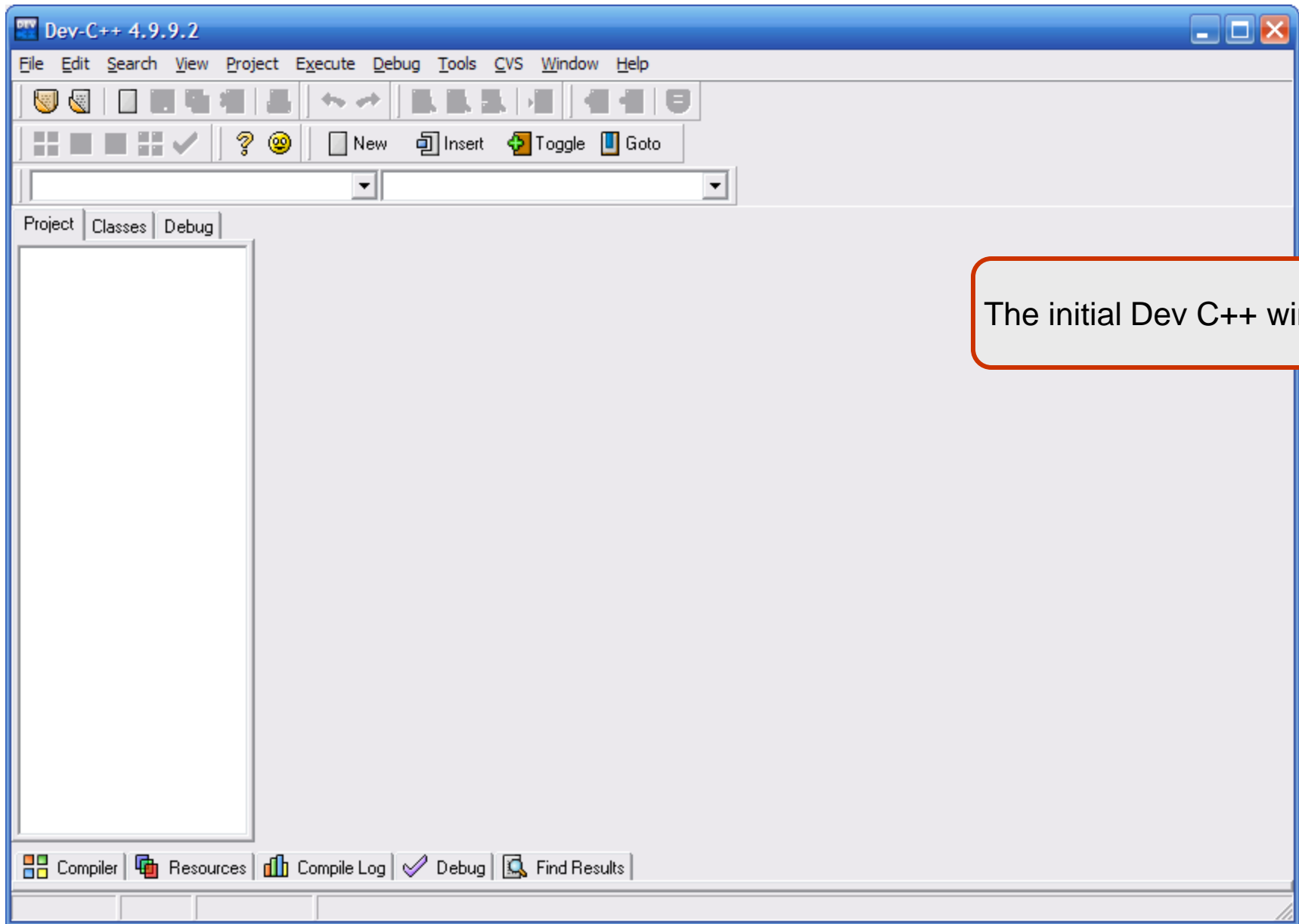


# Using Dev C++ (Your First C Program)

- Follow the instructions at [www.cs.ucf.edu/courses/cop3223/spr2009](http://www.cs.ucf.edu/courses/cop3223/spr2009) to download and install the Dev C++ compiler on your computer.
- Once you've done this you're ready to begin programming in C.
- As we said earlier, just as you learn natural languages by starting small and eventually increasing your vocabulary, so too with programming languages, you start by writing small programs and master the basics before you attempt to write more complex programs.
- So let's write a simple C program that will just print a message to the user.

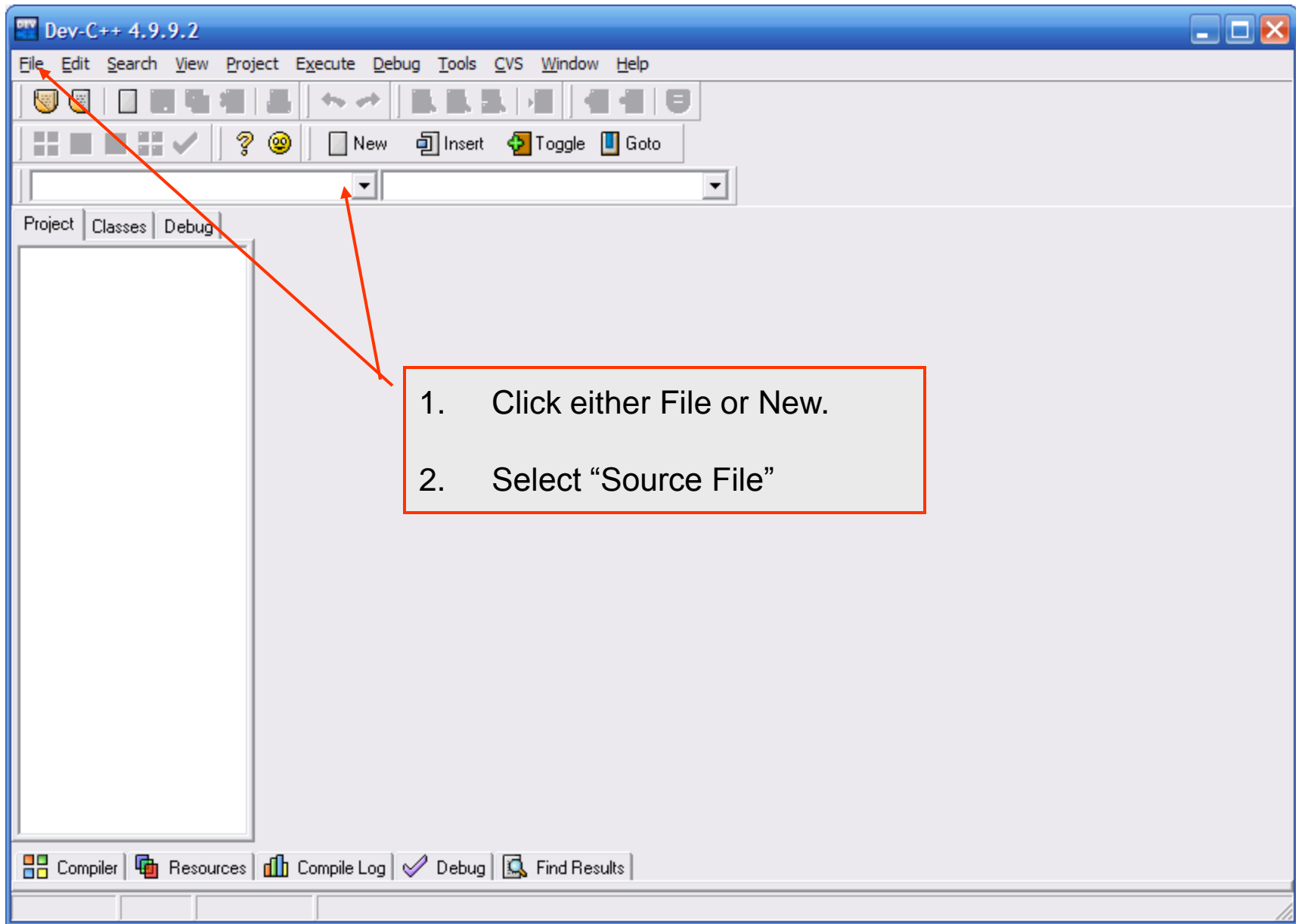


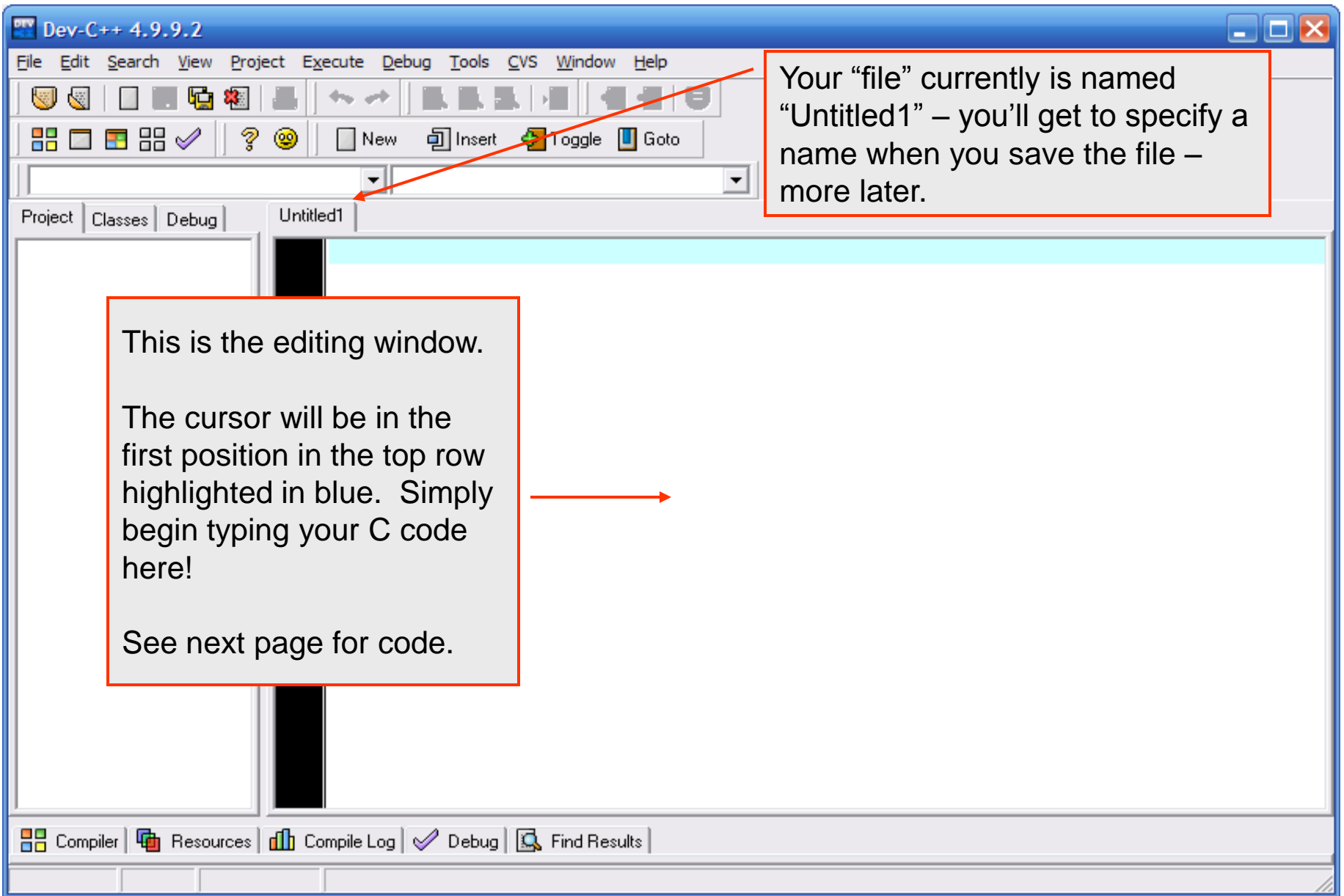




The initial Dev C++ window







Your “file” currently is named “Untitled1” – you’ll get to specify a name when you save the file – more later.

This is the editing window.

The cursor will be in the first position in the top row highlighted in blue. Simply begin typing your C code here!

See next page for code.

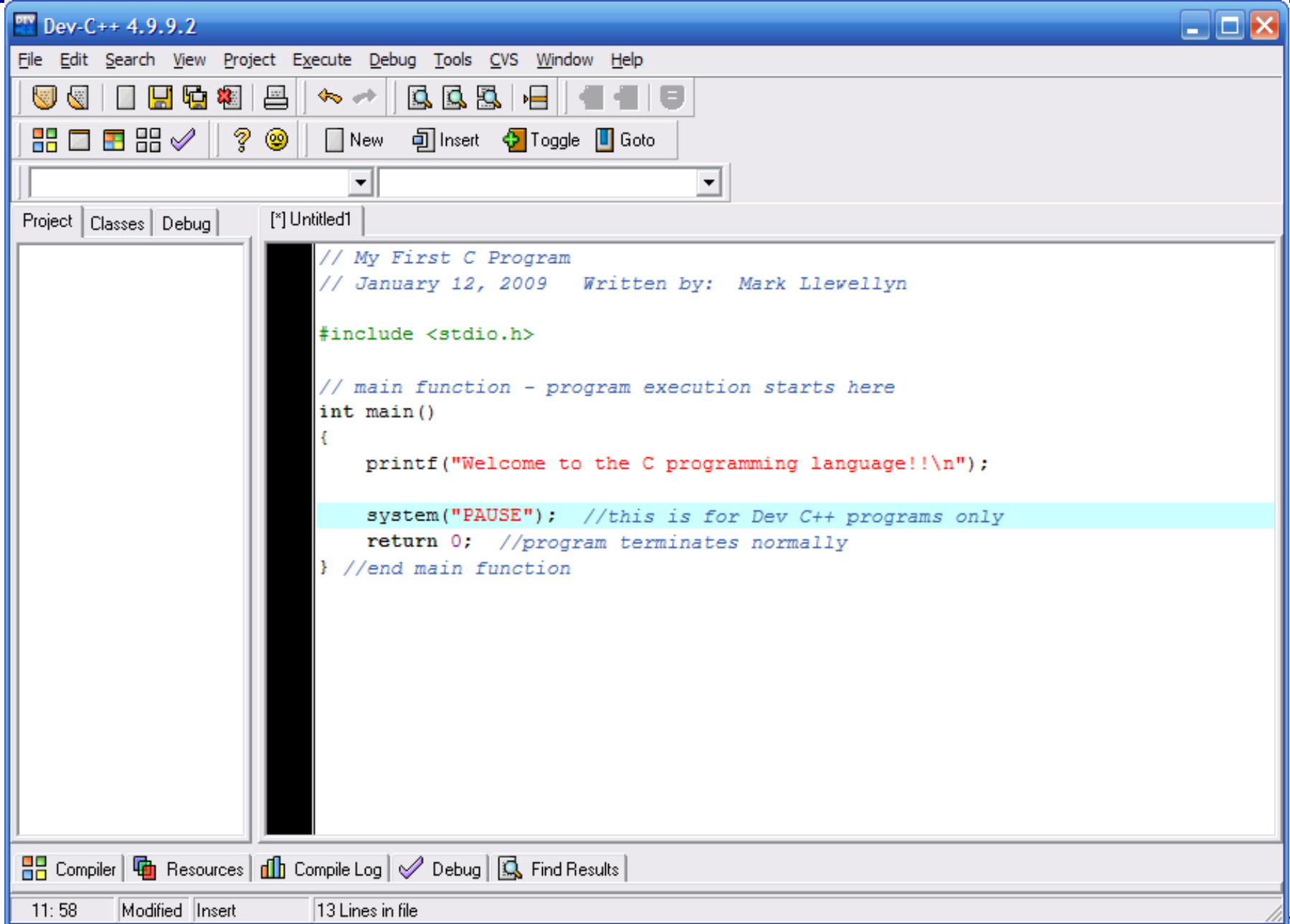


The line numbers are for discussion purposes only and are not part of the program!

# Your First C Program

```
1. // My First C Program
2. // January 12, 2009   Written by:  Mark Llewellyn
3. #include <stdio.h>
4. // main function - program execution starts here
5. int main()
6. {
7.     printf("Welcome to the C programming language!!\n");
8.
9.     system("PAUSE"); //this is for Dev C++ programs only
10.    return 0; //program terminates normally
11. } //end main function
```

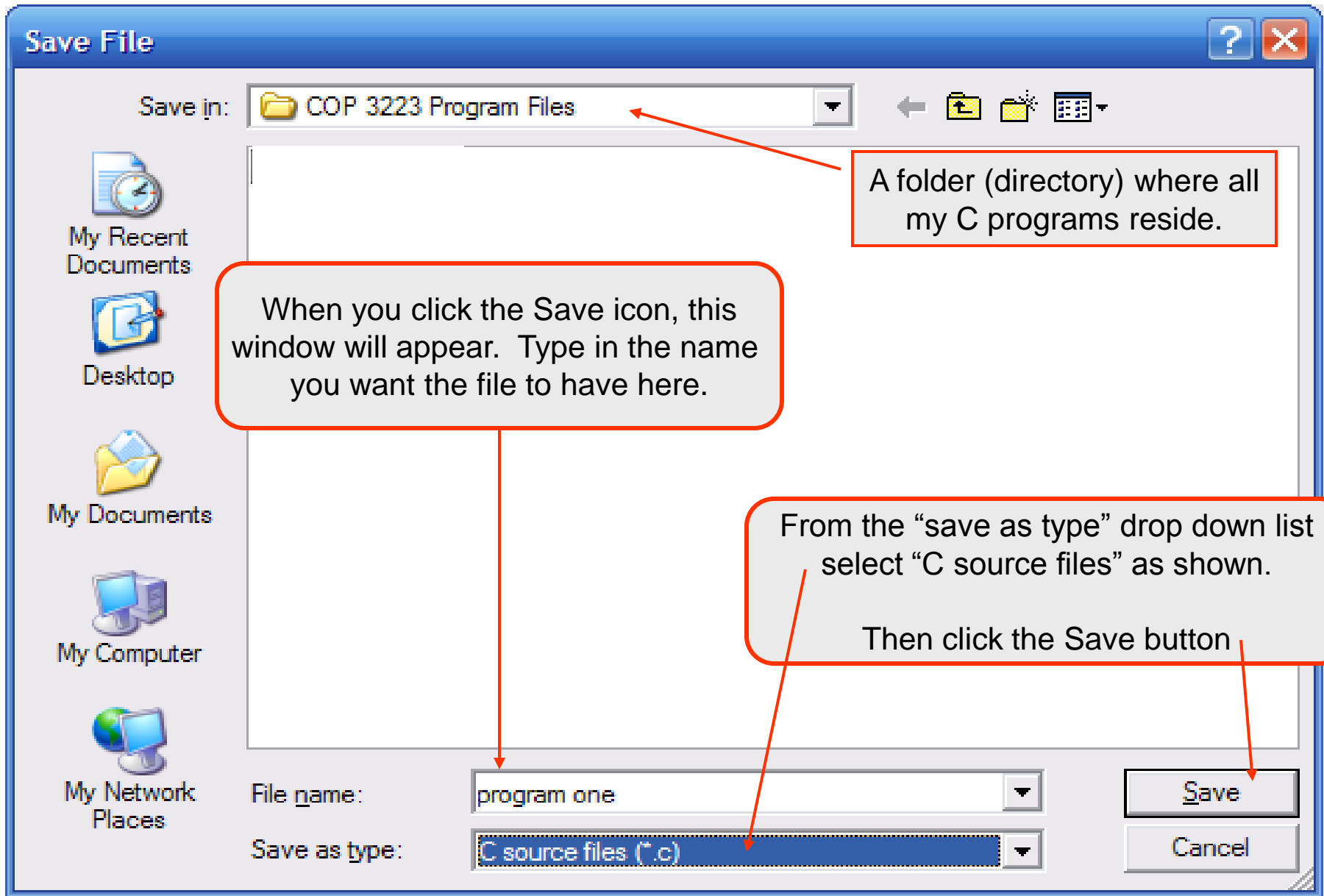


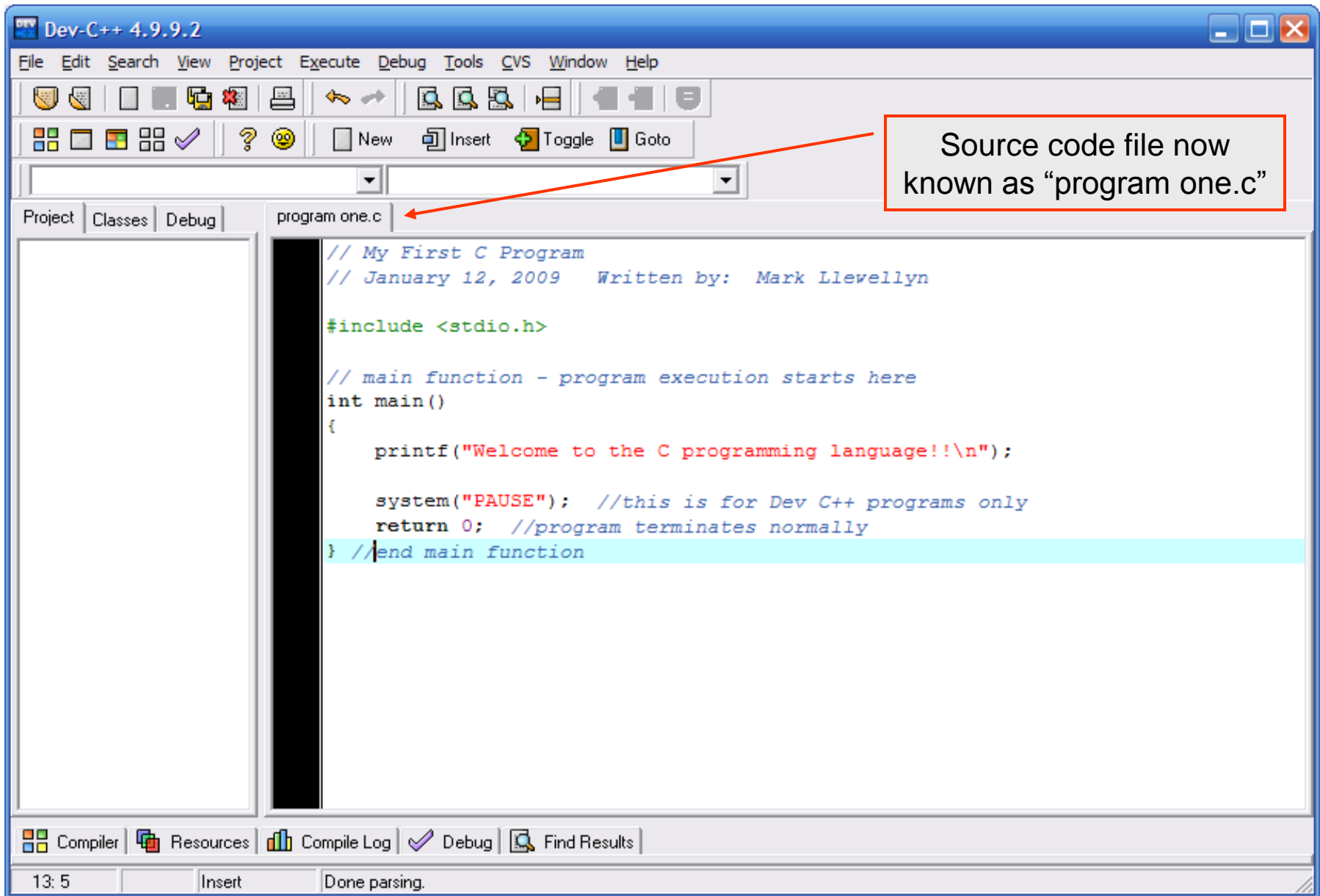


# Saving Your First C Program

- Once you've entered the code into the editing window, you need to save your program file.
- Always give your program files meaningful names.
- To save the file, click either the File option and then select Save or simply select the Save icon from the menu.
- Be sure to specify that the file type is a "C source file" as shown on the next screen shot.
- I suggest that you setup a directory where you can store all of the C programs you write in a single directory (folder).





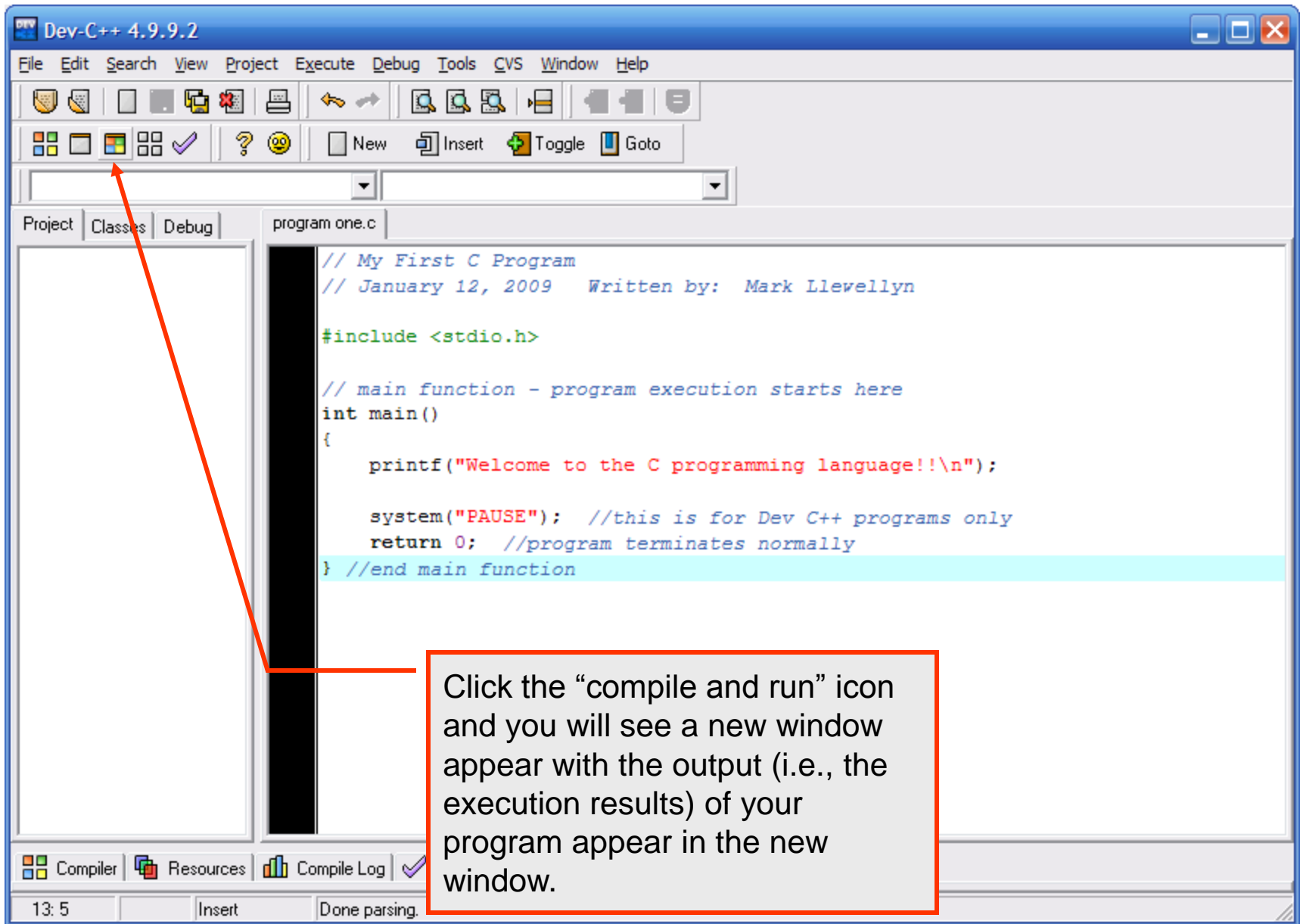


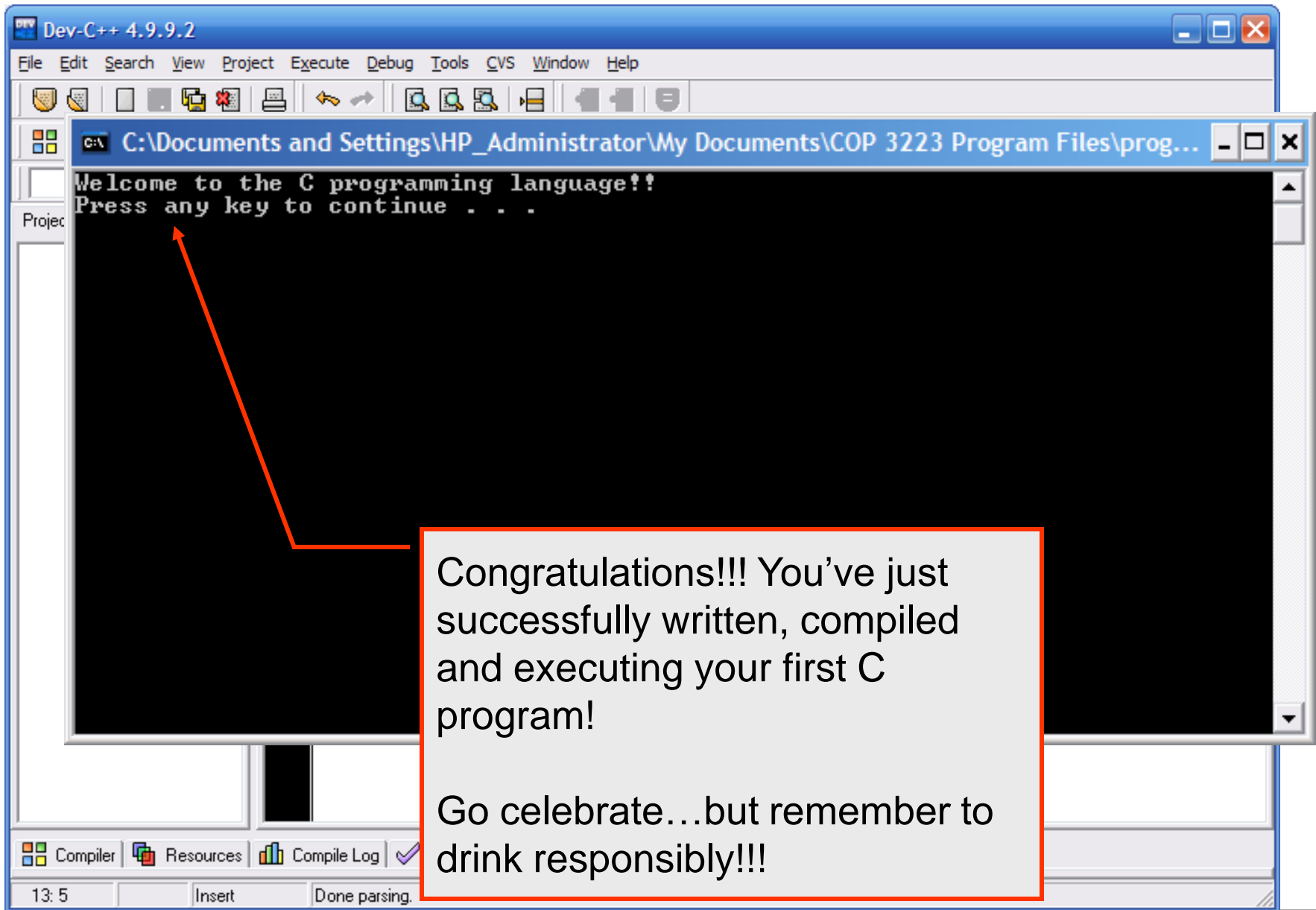


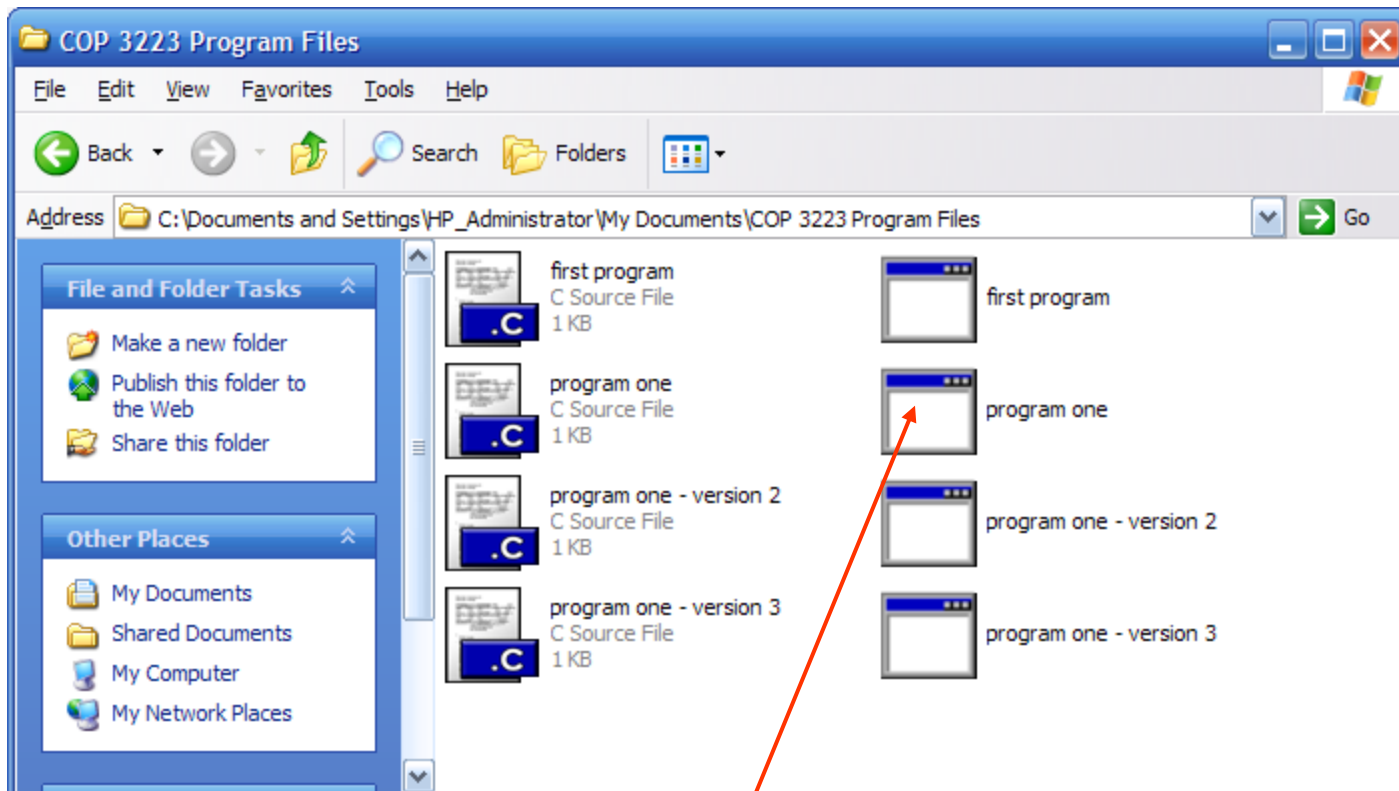
# Compiling and Executing Your Program

- Once you've saved your source file (your C program file), you need to compile the code into the machine language code that will be executed on your computer. (Remember that what the compiler does is converts your source code into machine readable code that is executable on your machine.)
- Dev C++ provides several different methods for compiling and executing your C programs. We'll start with a fairly simple technique that will compile and execute your program in a single step. (see next screen shot.)

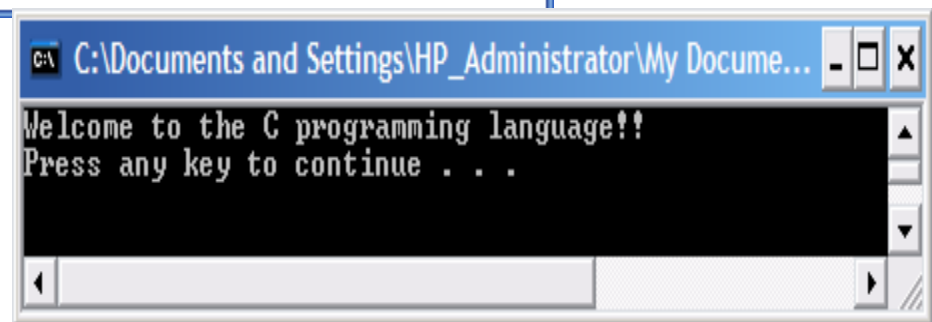








An alternate way, and typically easier way to run your C applications is to switch to the directory where the compiled source code is maintained and simply double click the application file you want to execute.



# A Detailed Look At The Program

- Even though this is a simple C program, it illustrates several important features of the C language.
- Referring to the line numbers in the code on page 12, lines 1 and 2 are comments.
- **Comments** are inserted by the programmer to document the code and improve its readability.
- Comments are ignored by the C compiler and do not cause the computer to perform any action when the program is executed (run). Since the compiler ignores comments, it generates no machine code for them.

**GOOD PROGRAMMING PRACTICE:** A common programming convention is to include comments at the beginning of the source code file that identifies the name of the program, with perhaps a brief description of the application (what is its purpose), the date the program was developed and the name of the person who created the code.



# A Detailed Look At The Program

- The C99 standard introduced the style of comments used in our first program.
- This type of comment begins with a double backslash and the comment continues to the end of the line. Thus each comment must begin with the double backslash characters.
- The C89 standard as well as all earlier versions of C, used a slightly different comment form. In this style a comment begins with a `/*` and ends with a `*/`.
- The advantage of this older style comment is that you can write a comment that covers many lines in the file and are not required to place a comment symbol at the start of each line. The disadvantages are (1) it makes the program somewhat less readable since long comments do not really stand out from the code, and (2) it is a common programmer error to forget to place the ending comment delimiter (`*/`) thus failing to end the comment and causing compilation or run-time errors.
- Since the newer style comment is now widely incorporated into many current C compilers, I would suggest using the newer style comment whenever possible.



# A Detailed Look At The Program

- Line 3 of the program, `#include <stdio.h>`, is a **directive** to the C preprocessor.
- All lines of code that begin with `#` are processed by the preprocessor which is done before the program is compiled.
- In this case, the line of code tells the preprocessor to include the contents of the standard input/output header (`stdio.h`) in the program.
- This header contains information used by the compiler when compiling calls to standard input/output library functions such as `printf`.
- We'll look more closely at the contents of headers later on.



# A Detailed Look At The Program

- Line 5 of the program, `int main()`, is a part of every C program.
- The parentheses after `main` indicate that `main` is a program building block called a **function**. C programs contain one or more functions, one of which must be named `main`.
- Every program in C begins executing at the function `main`.
- The left brace (line 7), `{`, must begin the **body** of every function. A corresponding right brace (line 11) must end each function.
- This pair of braces and the portion of the code between the braces is called a **block**. The block is an important program unit in C.





# A Detailed Look At The Program

**GOOD PROGRAMMING PRACTICE:** Each function in your program should be preceded by a comment that describes the purpose of the function. Sometimes this comment may also specify ranges of acceptable input values on which the function will correctly operate.

- Line 7 of the program, `printf(...)`, instructs the computer to perform an action, namely to print on the terminal screen (more precisely whatever is the default standard output, which is typically your screen), the string of characters marked by the quotation marks inside the parentheses.
- The entire line, including `printf`, its argument (the things inside the parentheses), and the semicolon is called a **statement**.
- Every statement must end with a semi-colon in C.



# A Detailed Look At The Program

- Notice that the “\n” in the argument did not appear in the output produced by the program (see screen shot on page 19).
- The “\n” is called an **escape character**. It indicates that the `printf` is supposed to do something out of the ordinary. When encountering a backslash in a string, the compiler looks ahead at the next character and combines it with the backslash to form an **escape sequence**.
- The escape sequence `\n` means newline. The table below lists some common C escape sequences.

Escape Sequence	Description
<code>\n</code>	Newline. Position the cursor at the beginning of the next line
<code>\t</code>	Horizontal tab. Move the cursor to the next tab stop.
<code>\a</code>	Alert. Sound the system bell.
<code>\\</code>	Backslash. Insert a backslash character in a string.
<code>\"</code>	Double quote. Insert a double quote character in a string.



# A Detailed Look At The Program

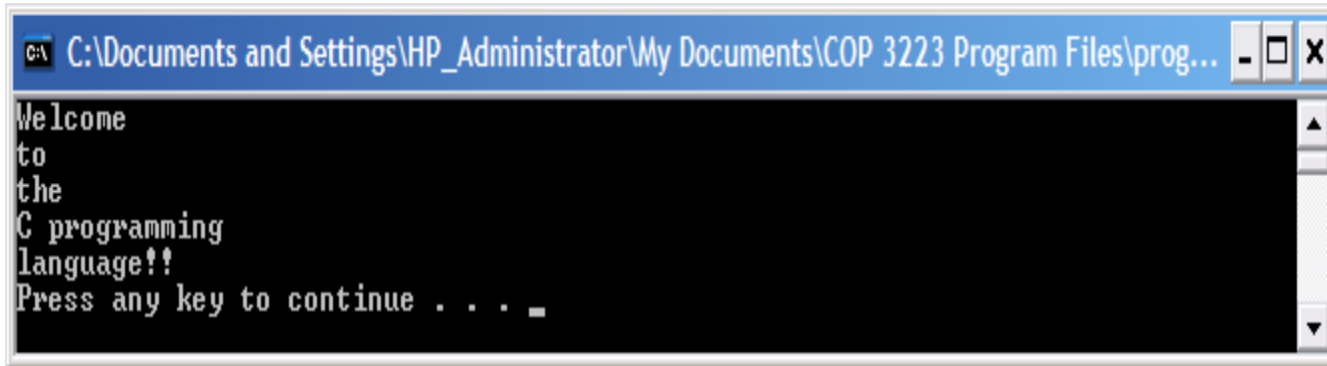
- Line 10 of the program is the last statement in `main()`, and is included at the end of every `main` function.
- The keyword `return` is one of several ways that we will use to exit a function.
- When the `return` statement is used at the end of `main`, the value `0` indicates that the program has terminated successfully.
- When we look at functions more closely later on, the use of this statement will make more sense. For now, just be sure to include it at the end of every `main` function.

**GOOD PROGRAMMING PRACTICE:** Add a comment to the line containing the right brace, `}`, that closes every function, including `main`. This will again enhance the readability of your code.



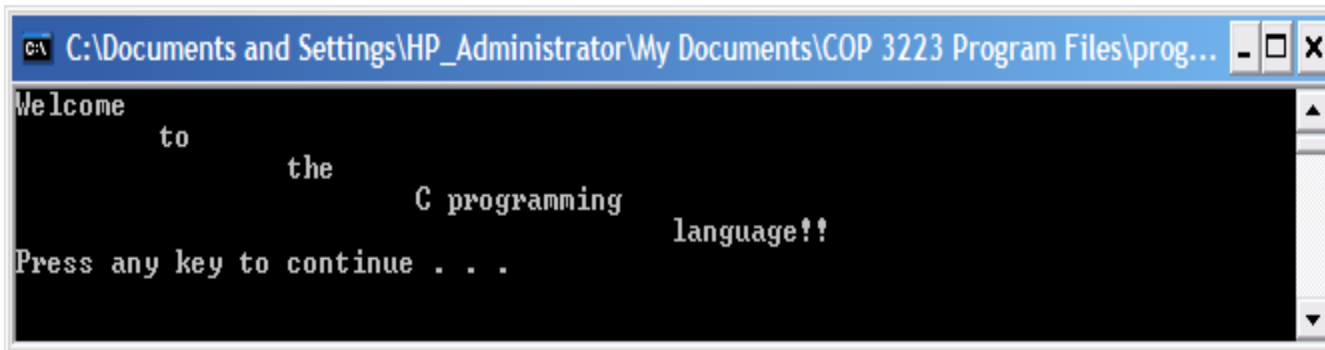
# Some Practice For You

1. Modify the first C program so that the output appears as shown below.



```
C:\Documents and Settings\HP_Administrator\My Documents\COP 3223 Program Files\prog... - □ X
Welcome
to
the
C programming
language!!
Press any key to continue . . . .
```

2. Modify the first C program so that the output appears as shown below.



```
C:\Documents and Settings\HP_Administrator\My Documents\COP 3223 Program Files\prog... - □ X
Welcome
    to
        the
            C programming
                language!!
Press any key to continue . . .
```

