Where To Go from Here?

WEEK
3

You have been presented with a multitude of advanced topics ranging from linked lists to dynamic memory allocation, with step-by-step detail on developing your own complete applications from scratch. Today you will learn:

- ☐ What next steps are available for your use of the C programming language.
- ☐ What other areas are available for C programming.
- ☐ What C++ is (an overview).
- ☐ Why you shouldn't reinvent the wheel.
- ☐ What canned tools are available that can aid in your C development.

What's Next?

You may now be wondering what you can do with your knowledge of the C language and application development. You may also be wondering what else there is to learn about programming in C. With hundreds of books available on the C language, you may be thinking that there is still a lot to learn.

You should now have a solid foundation for using the C language. Where you go from here is your choice. Most people choose to go in one of three directions:

- ☐ Specialized programming
- ☐ Other languages
- ☐ Other platforms

Specialized Programming

The C language includes constructs that enable you to take advantage of specific machines and specific operating systems. As you incorporate more of its power, you begin to develop applications that may be more machine specific. Many people who do advanced professional programming begin to focus on a specific area. These specific areas can range from graphics programming to writing compilers. Depending on the area of expertise, different constructs are needed. For example, in graphics programming, you may begin working with video memory, advanced math, and more. If you decide to concentrate on programming networks, then graphics are not as important.

You'll find that there are several C books available that cover specialized topics. The most common areas are graphics programming, compilers, and network programming.

Graphics Programming

Graphics programming is an area that many "at home" programmers are interested in. Graphics programs involve working with the video at a pixel-by-pixel level and many functions different than those used by the text graphics presented in this book are needed. In addition, it is often necessary to understand working with dynamic memory and mathematical formulas. It is possible to find books on topics such as graphics, advanced graphics, developing graphical displays, developing games, and more.

Note: The following are just a few books that are available:

Graphics Programming in C by Roger T. Stevens, M&T Books.

High-Performance Graphics in C (Animation and Simulation) by Lee Adams, Wincrest.

Compilers

Some programmers like to understand everything. To understand the full function of the C language, you could attempt to write your own compiler. There are a few books available on writing your own compiler; however, they are very technical in nature. This is an area that you would definitely need to be interested in before attempting to pursue it.

Network Programming

C can be used to create network programs. These can be programs such as the *Record of Records!* program presented throughout this book. You can also write network software using C. In addition, you could even write your own operating system. UNIX is an operating system that was developed with the C language. Again, like writing your own compiler, this is a very technical area to pursue. Books for this area of C development are a little harder to find, but they do exist.

Note: The following are just a few books that are available:

C/C++ For Expert Systems by David Hu, MIS: Press.

C Programmer's Guide to NetBIOS by W. David Schwaderer, Howard W. Sams & Company.

Advanced Application Development

In this book, you created an application with the help of several functions, which you created also. There are several books available that will help you expand your library of functions. In addition, there are several books that offer different approaches to developing applications. Some books are generic; others are extremely specific. Many of these books don't provide you with anything more than what was presented here. Other books, such as Jack Purdum's *C Programmer's Toolkit*, provide you with a multitude of additional useful functions.

Note: The following are just a few books that are available:

User Interfaces in C by Mark Goodwin, MIS Press.

C Programmer's Toolkit by Jack Purdum, QUE Corporation.

Advanced C Programming for Displays by Marc J. Rochkind, Prentice Hall.

Other Languages

In addition to moving into specialized areas, you may also choose to expand into other languages. C programmers will most often move into one of two other languages. These are assembler and C++.

Assembler

Assembler is a language that is much more technical than C. In addition, it is more cryptic. The reason for this is the assembler is one step closer than C to talking to the computer in its own language (ones and zeros). Listing 21.1 is an example of assembler code.

Listing 21.1. Some assembler code.

```
;Assembler code to get a key.
;
waitkey:    move     ah,01h
            int      16h
            jz       waitkey
            mov      ah,0
            int      16h
            or       al,al
            jz       waitkey1
            xor      ah,ah
            jmp      waitkey2
waitkey1    xchg     ah,al
            inc      ah
waitkey2    ret
```

As you can see, this is much harder to follow than a C listing; however, there are benefits to assembler. The main benefit is control. Most C commands break down into several assembler commands. Because of this, the precision of control is greater in assembler. Additionally, because assembler is closer to the ones and zeros that the machine understands, it can be written more efficiently.

The downside of assembler comes from one of the conditions that is also considered a benefit. The downside is that because you are writing so much closer to the machine level, assembler code is not as portable as C code.

If you're going to move into technical programming, then you may need to use assembler. Operating systems or network programming would be considered technical programming. Additionally, if speed is important, you'll be able to write slightly faster code using assembler instead of C.

Expert Tip: Use assembler to write those functions that are speed critical. Use C for the rest of the code.

Note: Assembler functions can be linked with C programs.

C++

Many C programmers are moving into C++. C is a subset of C++. This means that a C program could be compiled under C++; however, this would not make the C program a C++ program. C++ is more than just an enhancement to the C language. It is also a different way of developing.

An object-oriented language differs from a hierarchical or procedure language. C++ is an object-oriented language. The C language can be considered a procedural language. A C program is developed by creating functions one after another. Usually the `main()` function is created and then each of the subsequent functions are created. When the program runs, each function can be called in order. When a function completes, it returns to the previous function. In this manner, you can follow a C program one line at a time. An object-oriented program is developed differently than this.

An object-oriented programming language such as C++ differs from a procedural language such as C because of three concepts that are incorporated. These concepts are encapsulation, polymorphism, and inheritance.

Note: An object-oriented language is identified by its capability to encapsulate, polymorph, and inherit.

Encapsulation

Encapsulation is the concept of making packages that contain everything you need. These packages are called objects. An object contains everything you need to complete a specific function. For example, an object could be created for a square. The square object would do everything that is needed in regard to a square. By encapsulating the square, you create an object that does all the work. All you or any other programmer need to know about the square is how to interact with it.

By encapsulating, you provide a shield to the inner workings of the object. This enables data abstraction. You don't need to worry about how a square works, all you need to concentrate on is how to use it. That is, you only need to worry about what the square can do, not how it does it.

Note: Encapsulation enables you to ignore how an object works and concentrate on what it does.

Polymorphism

Polymorphism means having the capability to assume many forms. Polymorphism can be applied to several different areas of object-oriented programming. It is mainly used in calling a function. A function can be called in many different ways and still provide the same results.

In the example of the square, you may want to call the square object using four points, three points, a single point and a line length, or any of a multitude of other ways. Using any of these sets of parameters, you would expect to get the same results. In a procedural language such as C, you are required to write a different function for each different set of parameters. In addition, each function needs a unique name. In C++, using polymorphism, you may still need to write more than one function; however, you give each of them the same name. Based on the information passed, the correct function is automatically called. Unlike C, in C++ the person using the square function doesn't need to worry about which square function to call—they will all have the same name.

Inheritance

Inheritance is the third identifier of an object-oriented language. Having a square is a beginning, but what if you want a cube? A cube is a special kind of square. It has all the characteristics of a square with a third dimension added. By using a square to create the cube, you can use all of the characteristics of the square. The cube inherits all the characteristics of the square.

Synopsis of C++

As you can see, an object-oriented language has features that are foreign to a procedural language such as C. Because of this, C++ programs cannot be ported to C. The inverse of this, porting C to C++, can occur; however this would not take advantage of C++. In reality, this would be using the C++ compiler to compile your C program. This wouldn't make your C program a C++ program.

Warning: Many C programmers have started using C++ compilers; however, this does not make them C++ programmers. It is when you quit programming procedurally and start using the object-oriented constructs that you become a C++ programmer.

By using encapsulation, polymorphism, and inheritance in your program, you can create many benefits to yourself and your users. Because functions are encapsulated,

they are easy to upgrade and replace. As long as the interfaces remain constant, you can change the inner workings. With polymorphism, you can add different ways of calling functions without the need to change any pre-existing code. Because of inheritance, once you create an object, you can use it to build on bigger and better objects. While the learning curve of C++ is a little higher than C, once it's learned, there are benefits to be gained.

Note: Most books that teach C++ assume that you already know C.

Other Platforms

In addition to specializing your C development or moving to other languages to supplement your C programming, you can also move to other platforms. This book has concentrated on developing DOS-based applications. With your understanding of application development, you can either continue to develop DOS applications, or you may choose to move to a different platform. A *different platform* means using a different operating environment or operating system.

The C language can be used on almost every platform. You would need to obtain a C compiler for the given platform. Several of the most commonly used platforms are DOS, UNIX, Windows, OS/2, and System 7 (Macintosh). While the bulk of the C language is the same, some system-level commands are going to differ. In addition, in multitasking operating systems/environments such as Windows and OS/2, you must write your programs to interact with the operating system.

C Utilities

What you've learned in this book should be quite useful. You created a useful library along with several neat applications. In fact, you could market your applications. If you know other programmers, you could give them your TYAC library and header files. Include some documentation on how to use each function, and they will be able to link the TYAC library with their own code. You don't even have to give them the source code if they are using the same compiler as you.

You may now be wondering about other libraries. If you can give your library away, does this mean that you could get other people's libraries? The answer is yes! In fact,

many advanced programmers wouldn't write functions like the ones presented in this book. They would buy a prepackaged library instead.

Note: While you can buy libraries to do everything that was presented in this book, there are a couple of reasons why you wouldn't want to. The main reason is for the sake of learning. By creating these functions as you have done in this book, you learn the background of application development. Additionally, by creating them yourself, you don't have to pay for them! The final reason for creating functions yourself is control. You control exactly what the functions do.

There are a multitude of packages that can be purchased. These packages may be libraries or utilities. When you purchase a library, it may or may not have the source code with it. Most libraries that you can purchase don't automatically come with the source code. Many offer the source code at an additional cost. If you don't plan on ever modifying the functions that are provided in the library, then you shouldn't need the source code. By having the source code, you can modify the functions if they don't meet your exact needs. Some purchased libraries may charge a royalty. This means that if you use the library's functions in your program, you must pay a fee each time you distribute your program. Most purchased libraries don't charge a royalty; however, you should always check.

Following are some of the different libraries and utilities that are available. They are listed here to provide you with an idea of some of the packages that are available. You can find information on packages such as these in several places. You can check your local computer stores, you can talk to local user groups with special interest groups specializing in C/C++, or you can check computer magazines.

Note: The following information is provided to give you an idea of the different types of libraries and utilities available. Their inclusion here is not to be considered an endorsement. You should always research the available utilities before purchasing.

Graphics Libraries

Following are several graphics libraries:

Accusoft Image Format Library	By Accusoft Corporation. This set of routines enables you to add the capability to work with TIFF, PCX, DCX, TGA, GIF, BMP, and many other graphics formats. Included are functions that enable you to use several effects including zooming and rotating.
FONT-TOOLS	By Autumn Hill Software. These tools enable you to create, edit, and use fonts within your applications. In addition to providing the functions, a large sample of fonts is also included.
GraphiC	By Scientific Endeavors Corporation. This set of functions can be used to help create scientific and engineering graphics in your applications. This includes the capability to create linear, log, contour, polar, 3D bars, and more.
Graphics-MENU	By Island Systems. This is a set of GUI functions. GUI stands for Graphical User Interface. These routines help you to create several different constructs that are all valuable to an application. These include icons, menus, checklists, entry and edit screens, pop-up messages, and more. The routines also support mouse control.
GX Effects	By Genus Microprogramming. This is a library of special effects. These can be added to your program to display graphics using wipe, split, crush, slide, sand, drip, diagonal, spiral, random, or exploding screens.
PCX ToolKit	By Genus Micro programming. This is a toolkit that enables you to incorporate graphics into your programs. With over 100 routines, you can display, save, scale, and manipulate PCX files into almost any C program you create.
Virt-Win	By TeraTech. This is a set of routines that will help you save and restore the screen. In addition, you'll be able to create screens that are larger then the displayed windows. Using routines provided, you'll be able to pan and zoom different areas.

Communications Libraries

Following are several communications libraries:

C Asynch Manager	By Blaise Computing, Inc. This is a set of library routines that enables you to create asynchronous communications into your applications. This includes several protocols including XMODEM, YMODEM, and ZMODEM. The support is for Hayes-compatible modems.
Essential Communications	By South Mountain Software, Inc. This is a set of asynchronous communications library routines that supports interrupt-driven communications for both receiving and transmitting. Speeds of up to 115,200 baud can be supported along with up to 34 ports. Support for XMODEM, YMODEM, Kermit, and ZMODEM are all provided along with several other protocols.
Greenleaf CommLib	By Greenleaf Software, Inc. This is an asynchronous communications library for both DOS and Windows programs. This library supports several protocols including XMODEM, YMODEM, ZMODEM, and ASCII.
Magna•Comm C	By SoftDesign International, Inc. This is a library that provides full modem support. These fully interrupt-driven routines are for several protocols including ASCII, XMODEM, YMODEM, and ZMODEM.

File Management Libraries

Following are several file management libraries:

AccSys for Paradox	By Copia International. These routines provide an interface between C and Paradox. This includes multi-user capabilities along with routines for working with table files and primary and secondary indexes.

AccSys for dBASE	By Copia International. These routines enable you to work with dBASE DBF, MDX, and NDX files. In addition, DBT memo files are also supported. Different levels of support are provided for dBASE II, dBASE III, dBASE III Plus, and dBASE IV.
Btrieve	By Novell, Inc. This is a comprehensive set of routines that will help you incorporate Btrieve file access into your applications. These routines will help in writing applications that will use NetWare Btrieveä .
CodeBase	By Sequiter Software, Inc. This is a set of utilities for accessing dBASE and Clipper files. This includes functions that support multi-user access to the databases. Included are functions for creating pull-down menus, pop-up windows, and data entry and edit screens.
Essential B-Tree	By South Mountain Software, Inc. This set of functions are for creating single- and multiple-key files. In addition, there is support for fixed- and variable-length records.

Memory Management Libraries

“C” EMM Library	By SilverWare, Inc. This is a set of functions that allow for low-level connections between your application and Expanded Memory Manager. Also included are several high-level functions. These high-level functions help you to access and map expanded memory.
E-MEM	By TeraTech. This set of routines will help you to work with both extended and expanded memory.

General Libraries

Following are several general libraries:

C Utility Library	By South Mountain Software, Inc. This is a set of routines for a multitude of functions. This

	includes creating pull-down menus, pop-up windows, list boxes, and more. Included with the routines are a menuing system, a screen painter, and a code generator. Using these functions and utilities, you will be able to create entry and edit screens with mouse support.
Greenleaf Functionsä	By Greenleaf Software, Inc. This is a large library of varying routines. These include sound functions, graphics functions, directory operation functions, file search functions, and more.
GX Printer	By Genus Microprogramming. This set of functions provides comprehensive graphics printer support.
Hold Everything	By South Mountain Software, Inc. This is a set of routines that enables you to create programs that call other programs. When the called program is complete, you can return to where the original program left off.
/*resident_C*/	By South Mountain Software, Inc. This is a C library that provides functions to help create TSR programs. (TSR stands for Terminate and Stay Resident.)
Vermont Views Plus	By Vermont Creative Software. This is a set of routines that helps create text-based applications such as those presented in this book. This includes the capability to add menus, entry and edit screens, scrolling, help, and more. Mouse support is also included.

Summary

Today, you were presented with material to help set your direction. Now that you have experience in developing full-fledged applications, you are ready to journey into other areas of development. The area you choose can be one of many. You may choose to continue developing entry and edit applications, or you may choose to change your focus. Other areas to concentrate on include graphics programming, using other languages such as C++ or assembler, and using other platforms such as Windows,

UNIX, or OS/2. In today's material, you were presented with some information on each of these different areas that you can choose to focus on.

It isn't necessary to create every routine that you are interested in using. The TYAC.LIB library that you created could be given away. If you create your own libraries, you could sell them, with or without their source code, to others. Several companies offer libraries of routines for sale. Generally, the routines offered center around specific topics. Such topics could include communications routines, graphics routines, file management routines, memory management routines, and more. Several products are listed in today's materials to help you get a feel for what is available.

Q&A

Q Is C going away only to be replaced by C++?

A Probably not. While more and more people are moving to C++, not many of them understand the constructs involved in C++. Most programmers using C++ are using it as a C compiler. There are, however, advantages to using C++ over C.

Q Is it better to create your own libraries or buy them?

A Many of the libraries that can be purchased are royalty free. This means you can include the functions in your application and not worry about owing any money to anyone. Whether a library is worth the cost can be determined in many ways. The easiest way is to figure out how long it would take you to create the functions yourself. Multiply the time by the value you place on your time. The resulting figure compared to the cost of the library should tell you if it is worth the cost.

Workshop

The Workshop provides quiz questions to help you solidify your understanding of the material covered and exercises to provide you with experience in using what you've learned.

Quiz

1. Has this book provided you with a complete knowledge of everything you can do with C?
2. What are three areas where you could increase your C knowledge?
3. What other languages do most C programmers consider learning?
4. What is a benefit of using assembler?
5. What are three characteristics of an object-oriented language?
6. Can a C compiler compile a C++ program?
7. Can a C++ compiler compile a C program?
8. What are some of the platforms that C can be used on?

Exercises

1. **ON YOUR OWN:** Look through a few computer magazines. If you have C programming magazines, they would be even better. Look for adds showing what utilities and libraries are available for C.
2. **ON YOUR OWN:** Take what you have learned in *Teach Yourself Advanced C in 21 Days* and apply it to your own applications.
3. **ON YOUR OWN:** Continue to work with C. Work to improve the functions that were presented in this book. In addition, create your own functions and add them to your library.