# PROGRAMMING THE PALM TUNGSTEN

Edwin G. Wiggins
Webb Institute

## Introduction

Most people use a personal digital assistant (PDA) to track appointments, store phone numbers, and manage to-do lists. Many engineers have discovered that there are useful engineering programs for PDAs available for download. Some are free, while others carry modest price tags. The list of programs of interest to engineers includes calculators for humid air, steam properties, steam turbine processes, compound interest factors, and many others.

Despite the array of commercial software that is available, the author has not been able to find programs to do some of the things that interest him. In order to fill this need, it became necessary to write custom applets for his Palm Tungsten E machine. This article is the first of a series about these applets. In this article, the focus is on the programming language PocketC that is used on the Palm PDAs. A fairly simple program is presented to illustrate how this language works. Subsequent articles will describe engineering applets that the author has written for his own use. These applets will be made available for download at no charge.

## Comments on the PocketC Language

As the name suggests, PocketC is a scaled down version of the C programming language. Readers who are already familiar with C will probably find PocketC easy to master. Unfortunately, the author's programming experience began with FORTRAN in the 1960s and ended with BASIC in the 1980s. Since neither of these languages was "structured", the transition to PocketC was not particularly easy.

PocketC does not appear to be the ideal language for numerical work. The language itself supports very limited math. In order to do much beyond the four basic mathematical operations, a separate math library is required. Fortunately, the math library is available for download at no charge. Although the math library supports most of the functions and operations an engineer needs (trigonometric and hyperbolic functions, natural and base-10 logarithms, exponentiation, etc.), the author was surprised to discover that it does not appear to have an absolute value function. The workaround is to square the number and take the square root. That is clumsy, but it works.

Some other bits of numerical syntax are a bit unintuitive. For instance, to raise the number in variable "x" to the power in variable "y", the following syntax is used

pow(x,y).

That works fine, but it makes the code more difficult to read.

PocketC supports the usual sorts of looping and conditional execution structures (while, do..while, if, and if..else). There is good support for logic operations, but the syntax is a bit arcane. For example, the syntax for logical AND is "&&." Logical OR is "||."

## Auxiliary Programs

It is possible to write applets in the memo pad applet on the Palm itself and compile them on the Palm. This is very tedious if one uses Graffiti script. Perhaps with an external keyboard, it would be bearable. A much better route is to use a text editor/compiler program on a PC and download the compiled code. The author uses PocketC Desktop Edition (PDE) for applet development. It contains a capable text editor for creating the source code and a compiler to produce the object code.

Anyone who does programming knows that programs often compile just fine but do not execute properly. It can be very tedious to download programs under development to the Palm, only to find runtime errors that require further work on the PC. Fortunately, a Palm emulator is available. This emulator runs on the PC and emulates what happens on a Palm. The emulator is available for download from PalmOne at no charge. Applets can be written in PDE and transferred to the emulator by drag-and-drop for testing.

The standard Palm HotSync routine will transfer new applets from the PC to the Palm, but HotSync will also synchronize the calendar, the to-do list, the contacts list, and many other items every time it runs. Pilot Install (PI) is a freeware application that allows the user to decide exactly what will be transferred to the Palm. PI can reduce the time spent downloading new applets to the Palm.

## Documentation

Documentation for PocketC is a bit sparse. The download package includes an HTML file of documentation. When printed out, it amounts to 14 pages. There are several third-party books available. The author found Palm Programming for the Absolute Beginner by Andy Harris[1] to be quite helpful with the basics. In this book, as in several others, there is very little about working with numbers. Palm OS Programming Bible by Lonnon Foster[2] claims to be "100% comprehensive," but it appears to contain nothing about working with numbers. It may be that books about C itself would be helpful.

## A Simple Example

The simple example described below finds the roots of a quadratic equation whether they are real or complex. It is based on the familiar quadratic formula

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a} .$$

If the discriminant ($b^2 - 4ac$) is negative, there will be two complex conjugate roots. The applet presented below handles this case as well as the simpler case where both roots are real. While this applet is useful, its main purpose is to illustrate the programming process in PocketC. It demonstrates PocketC syntax and several important programming structures. Commands for input and output are included. The code is presented in sections that relate to input, calculations, and output. Throughout the code listing, extensive use of comments makes most of it self-explanatory.

The program begins with some comments (Anything following a double slash is a comment), and proceeds with the declaration of all variables that will be used. Here is the first section of the code.

```
// Quadratic Formula
// Edwin G. Wiggins
// Date: 2/8/2005

// Declare variables

main(){

    float a; // coefficient of x^2
    float b; // coefficient of x
    float c; // constant term
    float root1; // first root
    float root2; // second root
    float temp1; // temporary variable
    float temp2; // temporary variable
    float temp3; // temporary variable

    clear(); // clear the screen
    title("Quadratic Formula");
                    //  used in displays
```

All variables must be declared before execution begins. Part of the declaration is the specification of variable type. Three types are commonly used "string" for text, "int" for integers, and "float" for floating point numbers.

Next, the program asks the user to enter the coefficients of the quadratic expression, and it assigns these numbers to the variables a, b, and c. The basic tool here is the "gets" statement. The words in quotes inside the "gets" function are displayed on the screen and execution pauses for the user to enter a value. The statement "a=…" assigns the entered value to the variable a, which has been declared above. Entries by the user are treated as strings unless otherwise specified. The command "(float)" converts the entry to a floating point number.

```
// Get input

    clear();
    a = (float) gets ("Coefficient of x^2? ");
    clear();
    b = (float) gets ("Coefficient of x? ");
    clear();
    c = (float) gets ("Constant term? ");
    clear();
```

At this point, calculations begin. First the value of the discriminant is calculated. This value is assigned to variable "temp1." There are three cases to be dealt with. In the first case, the discriminant is zero, and there is a double root. For instance, the expression $x^2+2x+1$ has a double root at -1. In the second case, the discriminant is positive. In this case, there are two real roots. For instance, $x^2-1$ has roots at +1 and -1. The third case is more complicated. If the discriminant is negative, there are two complex conjugate roots. In the code below, there are three "if" statements that select the case from the list above. Two aspects of the "if" statement should be carefully noted. In Case 1, the "if" statement reads "if (temp1==0)." The double equals sign is required in this structure, because a single equals sign is used in assignment statements. Also note that the code to be executed when the test is "true" is enclosed in braces {}.

```
// Perform calculations

// calculate discriminant

temp1= b*b-4*a*c;

// Case 1: The discriminant is zero.
        There is a double root.

if (temp1==0) {
        root1=-b/(2*a);
        root2=root1;
} // end if

// Case 2: The discriminant is positive.
        Calculate two real roots.

if (temp1>0) {
        root1=(-b+sqrt(temp1))/(2*a);
        root2=(-b-sqrt(temp1))/(2*a);
} //end if
```

In Case 3, the discriminant is negative, so its square root is imaginary. PocketC and its math library do not support complex arithmetic, so special handling is needed. The variable "temp2" contains the real part of the root. The variable "temp3" contains the square root of the negative of the discriminant. These two temporary variables are used in the output section to give the appearance of a complex number.

```
// Case 3: The discriminant is negative.
    Calculate complex conjugate roots.

if (temp1<0) {
        temp2=-b/(2*a);
        temp3=sqrt(-temp1)/(2*a);
} //end if
```

In the final section of the applet, the results are displayed on the screen of the Palm. The basic output statement is "puts." The argument of the "puts" statement can be text or numbers. Text must be enclosed in double quotes. A carriage return is generated by "\n." Numbers are displayed in scientific notation unless otherwise specified. The "format" statement is used to control the display of numbers. Thus the statement "format(temp1,3)" calls for display of

the value of the variable “temp1” with 3 decimal places. Trailing zeros are displayed if necessary to meet the format requirement.

```
// Display results

        // Cases 1 and 2, real roots

        if ( temp1 >= 0 ) {
        puts( "discriminant is " +
              format(temp1,3) + "\n\n" );
```

The “puts” statement immediately below displays the quadratic expression on the screen in recognizable form. Note that “x^2 + ” and “x + ” are text elements. The coefficients a, b, and c are formatted with no decimal places, since they are expected to be integers.

```
puts("The quadratic is: " + "\n\n" +
  format(a,0) + "x^2 + " + format(b,0) +
  "x + " + format(c,0) + "\n\n");
```

The next two “puts” statements display the real roots to three decimal places. The “wait()” statement pauses execution so the user can read the results. Execution resumes when the user taps a “clear” button on the screen.

```
puts ("Root 1= " + format( root1, 3 ) + "\n");
puts ("Root 2= " + format( root2, 3 ));
wait();
} //end if
```

If the roots are complex, the variables “temp2” and “temp3” above are used along with text in a “puts” statement to give the appearance of a complex number. The user must keep in mind that the “i” characters that appear are actually text elements.

```
// Case 3, complex conjugate roots

if (temp1<0) {
puts( "discriminant is " +
                format(temp1,3) + "\n" );

puts("The quadratic is:" + "\n\n" +
   format(a,0) + "x^2 + " + format(b,0) +
   "x + " + format(c,0) + "\n\n");


puts ("root1= " + format( temp2,3 ) +
  "+" + format( temp3,3 ) + " i" + "\n");

puts ("root2= " + format( temp2,3 ) + "-"
  + format( temp3,3 ) + " i");


      wait();
      } //end if
```

The “exit()” statement transfers the display back to the main Palm application screen, and “}” tells the complier that this is the end of the program. “// end main” is simply a comment.

```
                exit();


            } // end main
```

Note that each executable statement above ends with a semicolon. With his background in FORTRAN and BASIC, where a carriage return marks the end of a statement, the author had great difficulty remembering to end each statement with a semicolon. Failure to do so results in a compiler error in the <u>following</u> line of code. Care must also be taken to ensure that both parentheses and braces balance. That was not so hard to remember. Indenting in the code above is optional, but it helps keep track of the flow of execution. The compiler ignores all spaces and blank lines, so they can be used liberally to increase readability. It also ignores a double slash and anything that follows.

## Conclusion

Useful custom applets can be created by anyone with a basic understanding of programming. Each computer language has its own peculiar syntax, but they tend to share the same programming structures. The author experienced some difficulty, partly due to the lack of good documentation, but the result is a useful applet.

The next article in this series will describe an applet that finds real roots of polynomials up to fourth order by means of successive approximation. This program makes use of additional features of the PocketC language.

Subsequent articles will describe engineering applets. These applets will be made available for download at no charge.

## References


1. Harris, Andy, Palm Programming for the Absolute Beginner, Prima Publishing, 2001.

2. Foster, Lonnon R., Palm OS Programming Bible 2nd ed., Wiley Publishing Inc., 2002.


## Biographical Information

Edwin G. Wiggins holds BS, MS, and Ph.D. degrees in chemical, nuclear, and mechanical engineering respectively from Purdue University. He is the Mandell and Lester Rosenblatt Professor of Marine Engineering at Webb Institute in Glen Cove, NY. Ed is a past chairman of the New York Metropolitan Section of the Society of Naval Architects and Marine Engineers (SNAME) and a past regional vice president of SNAME. A Centennial Medallion and a Distinguished Service Award recognize his service to SNAME. As a representative of SNAME, Ed Wiggins serves on the Engineering Accreditation Commission of the Accreditation Board for Engineering and Technology.