

# Math 248 Computers and Numerical Algorithms–Pruett

## LABORATORY ASSIGNMENT Function Subprograms in Fortran 90

C\_\_\_\_\_ In first-semester calculus, you learned that

$$\lim_{x \rightarrow +\infty} \left(1 + \frac{1}{x}\right)^x = e$$

Although you cannot *prove* this using a computer, you can nevertheless look at trends of functions that suggest certain limits. On Blackboard under DEMONSTRATIONS are the files **limit.f90** and **lim\_lib.f90** that contain the **PROGRAM limit** and the **MODULE limit\_library** on the reverse side of this paper. Copy these into your working directory, compile the MODULE, and then compile the PROGRAM. Run the program and briefly describe the results.

C\_\_\_\_\_ Modify the MODULE and the PROGRAM to approximate the following limit:

$$\lim_{x \rightarrow 0^+} (1+x)^{\frac{1}{x}}$$

Recompile, re-run, and briefly describe your results.

B\_\_\_\_\_ On paper, write a quadratic function  $f(x)$  (with integer coefficients) one of whose roots is  $\sqrt{7}$ .

B\_\_\_\_\_ Write and compile a Fortran 90 MODULE, containing a single function subprogram, which returns the value of the function  $f(x)$  above, given the value of  $x$ .

B\_\_\_\_\_ Add to your Fortran 90 MODULE above a second function subprogram that returns the value of  $f'(x)$ , given the value of  $x$ . Recompile.

B\_\_\_\_\_ Write, compile, and run a Fortran 90 PROGRAM that computes the  $\sqrt{7}$  by Newton's method and USES the two function programs in the MODULE above.

A\_\_\_\_\_ Modify your MODULE and PROGRAM above to compute  $\sqrt{a}$ , where  $a$  is *any* non-negative real number (to be read by your program).

A+\_\_\_\_\_ Write the quadratic formula on paper.

A+\_\_\_\_\_ Rationalize the numerator of the quadratic formula, and simplify the result as much as possible.

A+\_\_\_\_\_ From Blackboard DEMONSTRATIONS also copy files **quad2.f90** and **quadlib.f90** to your working directory. Files **quad2.f90** and **quadlib.f90** contain the main PROGRAM and (half of the) MODULE, respectively, of the “smart” quadratic solver discussed in class. Add another function subprogram, which exploits the alternative quadratic formula above, to the MODULE and recompile. Feel free to cut and paste.

A+\_\_\_\_\_ Run your “smart solver” and verify that the results are correct for several different sets of values  $A$ ,  $B$ , and  $C$ .

```

PROGRAM limit
!
!%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
! Program to illustrate a use of FUNCTION subprograms
!
! AUTHOR:  Dave Pruett/JMU Math
! CREDITS: None
!%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
!
  USE limit_library

  IMPLICIT NONE
  REAL (KIND=8) :: x, y
  INTEGER :: i

  DO i = 1,30

    x = 2.0**i
    y = my_function(x)
    WRITE (*,*) ' i = ', i, ' x = ', x, ' y = ', y

  END DO

  STOP
END PROGRAM limit

MODULE limit_library

CONTAINS
!
! Enter user-defined function in this subprogram
!
  FUNCTION my_function(x)

    IMPLICIT NONE
    REAL (KIND=8), INTENT(IN) :: x ! x may NOT be redefined
    REAL (KIND=8) :: my_function

    my_function = (1. + 1./x)**x ! user-defined function

    RETURN ! return value to main program

  END FUNCTION my_function

END MODULE limit_library

```