Review for Midterm Exam

Introduction to Scientific Computing with Fortran 90

Material Covered: Lectures 1 - 11 (no vectors)

---

1. What is the purpose of the `implicit none` statement? Where do you put this statement in a main program? What happens if you don't have it? Where do you put this statement in a module? Why do you not have to put it in every subprogram in a module?

2. List the 5 basic data types.

3. What is the syntax to continue a statement from one line to the next? What is the syntax to put more than one statement on a single line?

4. If we define n through the parameter statement
```
integer, parameter ::  n=5
```
and later in the code we need to increase n and have the statement
```
n = 2* n
```
What will happen and why?

5. Is Fortran 90 case sensitive?

6. Write an expression to calculate

$$\frac{6.1}{2.5^4} + \sin 45^\circ + e^5 - \ln 3.1$$

7. In each of the statements list the order of (from first to last) that the operations are performed.

```
5.  + 4.**2 / 8.

exp(2.* ( 4./5.)  )  /7.
```

8. Explain the difference in the result of the following two statements
```
3.0 + 5/2     3.0 + 5.0/2.0
```

9. Explain what the following do loop does. Assume that `sum` has been declared as real and `i` as an integer. What is the final value of `sum` after the loop is finished?

```
sum= 3.0
do i = 1, 4, 2
    sum = sum * float(i)
end do
```

10. Assume `x,y` have been declared as real and set to some values; assume `okay` has been defined as a logical. Write a conditional statement which will do the following:

(i) if $0 < x < 10$ and y is any value, set `okay` to true;
(ii) if `x` is positive and $y < 20$ set `okay` to true;
(iii) set `okay` to false otherwise.

11. Give an example to illustrate the difference between a compile time error and a run time error.

12. Suppose we have set `random_seed` and have generated a random number `x` via the statement

```
call random_number (x)
```
Write statement(s) to use `x` from the above statement to produce random numbers such that $-10 \leq x \leq 10$

13. Explain in words the difference between the philosophy behind the Bisection Method and Regula Falsi. Does the error in the Bisection Method decrease at every step? Why or why not?

14. Suppose we have a program to calculate the root of a nonlinear equation $f(x) = 0$. Why do we need a stopping criteria for a numerical method to solve this problem? Discuss each of the following stopping criteria where x^k is our iterate at the kth step. What do you recommend using and why?

(i) $f(x^k) \leq$ prescribed tolerance;

(ii) $|f(x^k)| \leq$ prescribed tolerance;

(iii) $|x^{k+1} - x^k| \leq$ prescribed tolerance;

(iv) $\dfrac{|x^{k+1} - x^k|}{|x^{k+1}|} \leq$ prescribed tolerance

15. Assume we have a subroutine of the form

```
subroutine test( a, b, n)
```
where we declare `real :: a, b` and `integer:: n`. Explain why you would have an error for each of the following calls to the subroutine.

(i) `call test ( 4.2, 3)`

(ii) `call test ( w, m, v)` where `real :: w,v,` `integer :: m`

(iii) `a= test ( 4.2, 5.0, 3)` where `real :: a`

16. Suppose we have an integer function, i.e., it returns an integer value defined by

```
function row ( x )
real :: x
```
If this is the only declaration statement what happens and why? How can you fix the error?

17. Suppose we have a logical function, i.e., it returns a logical value defined by

```
function okay ( x )
logical :: okay
real :: x
```

with the calling statement

`a = okay (x)` where `real:: x, a`.

What happens and why?

18. Explain geometrically how Newton's method works. What is the (geometrical) difference between Newton and Secant Method?

19. What are the number and requirements (if any) on the starting values for Bisection, Regula-Falsi, Newton and Secant? Do the methods converge to a root for any initial guesses satisfying these criteria?

20. Assume that we have starting values for which the Newton and Secant methods both converge. Which converges faster? What about for Bisection and Regula Falsi?

21. Suppose that we have the statement

```
open ( unit = 17, file='output.txt')
```

and the real variables `a, b`, the integers `m,n` and the character `name` are declared.

(i) Give a `write` statement which writes to the file `output.txt` the real variables in exponential format using 7 digits to the right of the decimal place.

(ii) Give a `write` statement which writes to the file `output.txt` the real variables in floating point format using 4 digits to the right of the decimal place. Include the format specification in the `write` statement.

(iii) Give a `write` statement which writes to the file `output.txt` the integer variables in an integer format (where $m, n < 9999$) with the text "i=", etc. included.

(iv) Repeat (i) using a `format` statement.

(v) Give a formatted `print` statement to print to the screen which prints out the character `name` which has been declared as `character (len=20) :: name` .

22. If we have the declaration statements

```
real :: x, y, z;     integer :: m,n
```

in a separate file, can these be compiled? If we have a main program where we want to add these statements to the declarations that are currently in it, how do we do this? Where do we put this statement and what do we have to modify when we compile the main program?

23. Can a module be compiled? executed? How does another program unit get access to a module?

24. Write a `case` construct to do the following where n has been declared as an integer and x as real :

(i) if $n = 1$ increase x by one

(ii) if $2 \leq n < 5$ increase x by five

(iii) if $5 \leq n \leq 50$ increase x by ten

(iv) otherwise increase x by twenty.

25. In a `case` construct what are the allowable data types for the `selector`?

26. What do we mean by "function overloading"? by "operator overloading"?

27. Suppose we have defined an employee class (i.e., a derived data type) in the module called `class_employee` with the attributes name and pay rate by

```
type employee
    character(len=30) ::  name
    real ::  payrate
end type employee
```

Now we want to have a module `class_manager` where we define a data type manager whose components are (i) an object in the employee data structure and (ii) a logical called `is_salaried`. Write the first lines of this module up to the `contains` statement.

28. Referring to # 27, suppose we have a subroutine in each module called `print_pay_employee` and `print_pay_manager`. In a main program which uses both modules, we would like to simply call either of these routines by the generic name `print_pay`. Explain how to do this.

29. Referring to #27, give a statement which would set a member of the employee class, say `smith` using the intrinsic constructor. Set the name to "JohnSmith" and the payrate to $2150.50

30. Referring to #27, use an unformatted `write` statement to print out the payrate of a member of the employee class called "emp123". Just print to the screen.

31. If a module contains a subroutine called `get_angle` and it contains the statement

```
private ::  get_angle
```

then what happens in a calling program that uses this module when you call `get_angle`?

32. Suppose we obtained the following results while attempting to find a root of $f(x)$. What would be the next iteration for (i) the bisection method, (ii) regula falsi, (iii) Newton, (iv) Secant method?

x^k	$f(x^k)$	x^{k-1}	$f(x^{k-1})$	$f'(x^k)$
1.0	-2.1	3.0	0.4	0.5