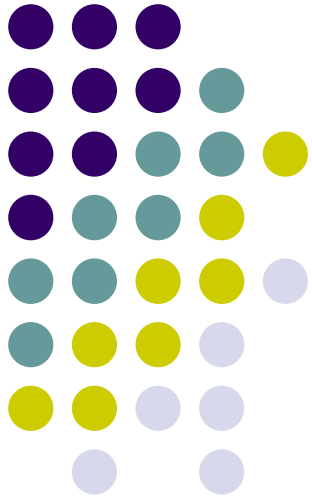
ATSC 212

FORTRAN - part 2

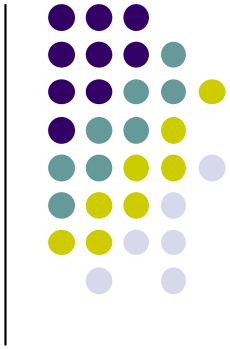
Roland Stull

roland.stull@ubc.ca



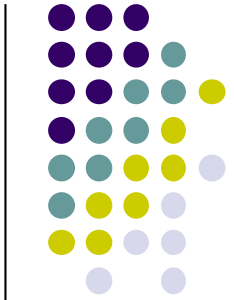
revised July 2024

Topics Today



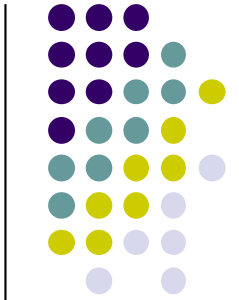
- Audiences of your code
- Functions: intrinsic and subroutine functions
- “alias”, a linux shortcut
- Modules
- More string manipulation
- More file handling
- More I/O

Write Your Code for 3 audiences



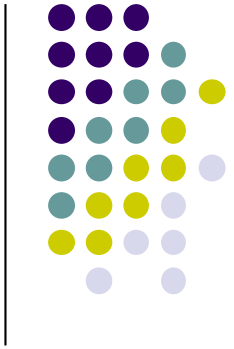
1. The programmer & colleagues
   - Give variables meaningful names
   - Always add comment lines in the code
   - Declare all variables, and indicate units
2. The user
   - As program runs, display status on screen
   - Interact with user: prompt -> response
   - Plan to handle errors in user input
3. The computer
   - Code must execute cleanly.
   - Sometimes must also be fast.

Set-up your computer



- Activate your terminal window, and your window for writing source code (else be prepared to use vi from your terminal window). If you need to compile and run on a server, finish logging into that server.
- Start with your own working copy of `wp12.f95` from last time. Assuming you already successfully compiled this code to create an executable runwp12, then run it again to remind you where we left off.
- However, if you didn't get it working at the end of the last session, I can provide you with my copy of `wp12s.f95` that you can use as a starting point for the next lessons. https://www.eoas.ubc.ca/courses/atsc212/labs/fortran-2/wp12stull.txt (then change the name from `wp12stull.txt` to `wp12.f95` .)
- If you use my version, please compile and run it, and fix it if needed.

Alias - a linux shortcut



First, from the linux command line,

define an alias, such as:

```
alias g12='gfortran wp12.95 -o runwp12'
```

To use this alias, on the linux command line type:

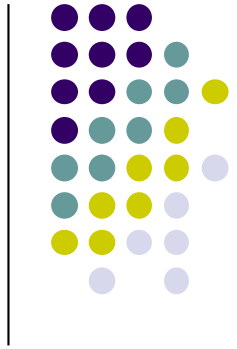
```
g12
```

Which will automatically compile the program, and produce an output file called “runwp12”.

To run your program, type in the command line:

```
./runwp12
```

…more I/O: Formatted READ

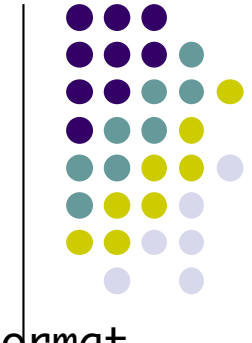


```
!Example: Reading real numbers:
 character (len=30) :: fmt          !name of string holding format
…
 fmt = "( F7.1, F7.0, 2F7.1 )"

 read(1,fmt) P, z, T, Td

! From unit 1, reads 4 numbers in a single line, each number
! occupying 7 columns (including blanks).
!=================================================
!Fn.d is for real numbers using n columns, of which decimal values are in
! last d.    Example:  F10.3  reads or prints    bbb101.325
!And if a negative sign is needed, it uses up one of the columns.
!  example: F8.2  reads or prints    b-273.15
!So when you plan the size if n, don't forget "." and "-".
!ESn.d prints real numbers in sci. notation: ES12.3 does bbb2.990E+08
!an prints a character string of length n.  Also if n is omitted,
! then a adapts to any size string.  Example: a4 prints "nice"
!In prints an integer, within n columns:  Ex: I5 prints bb365
!Errors: If number is too large to print in n columns, then "******"
```

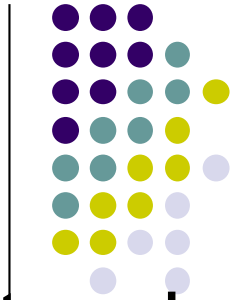
… more I/O: Formatted WRITE



```
!Example:
 character (len=30) :: fmt1, fmt2          !name of strings holding format
 character (len=1) :: tab = achar(9)       !ascii tab character
 real, dimension(5) :: T                   !an array of temperatures
 real :: x, y, z                           !real numbers
 fmt1 = "( 2(F12.4, a1) , F10.0 , I5 )"
 fmt2 = "( a , a )"
 write(2,fmt1) x, tab, y , tab, z, nlayers
 write(2,fmt2) "This file is :", filename
 write(2,"(5ES15.2)") (T(i), i=1,5)        !contains an implied do loop

!Fn.d prints real numbers using n columns, of which decimal values are in
! last d.    Example:  F10.3  prints    bbb101.325
!And if a negative sign is needed, it uses up one of the columns.
!  example: F8.2  prints    b-273.15
!So when you plan the size if n, don't forget "." and "-".
!ESn.d prints real numbers in sci. notation: ES12.3 does bbb2.990E+08
!an prints a character string of length n.  Also if n is omitted,
! then a adapts to any size string.  Example: a4 prints "nice"
!In prints an integer, within n columns:  Ex: I5 prints bb365
!Errors: If number is too large to print in n columns, then "******"
```

...more READ/WRITE formats



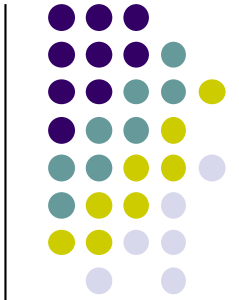
- nX in the format statement skips n characters in input, or writes n blanks in output.
- Tm in the format statement tabs to the mth character.

- Examples:

```
real :: z, speed
character (len=30) :: fmt = "(T45,F7.1,7X, F7.1)"
read(1,fmt) z, speed
```

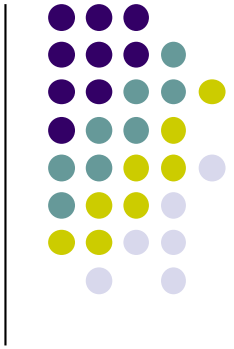
which tabs to the 45th character, then reads a real number with F7.1 format, then skips 7 characters, and reads another real number. Can be use for writes as well a reads.

Do Exercise 13 in HW-fortran2



- This exercise and most others are given at https://www.eoas.ubc.ca/courses/atsc212/labs/fortran-2/fortran2.html
- Start by doing a SaveAs (or cp in linux) to save the old version 12 as `wp13.f95` .
- Then, follow along with instructor, learning more about reading and writing formatted arrays.

Modules

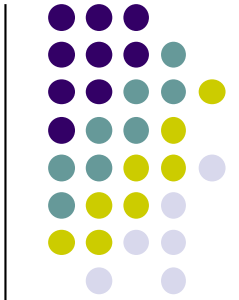


```
!A way of passing variables between different subroutines.
!Modules are global to the program.
!Example.  First, before your main program, define the module:
 module rainmod
   character(len=100) :: title!a character variable
   real, dimension(120) :: precip     !an array of real numbers
 end module rainmod


! Then, use it in any main program or subroutines where you need it:
 program precipitation
   use rainmod
   implicit none
   title = “My favorite rainy day”
   …
   call showtitle
 end program

 subroutine showtitle
   use rainmod
   implicit none
   integer :: i
   write(*,*) title
   do i = 1,120
     precip(i) = 0.05*real(i)
   enddo
 end subroutine showtitle
```

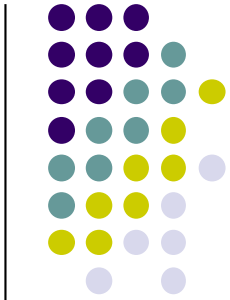
Do Exercise 14 in HW-fortran2



- Again, This exercise and most others are given at https://www.eoas.ubc.ca/courses/atsc212/labs/fortran-2/fortran2.html

- Follow along with instructor, learning about modules. A module is like a clipboard in a Mac or Windows gui. But in a module you can have many items copied to the clipboard and you can access them by name. The data stored in modules can be accessed from any subroutine that uses the module.

Intrinsic Functions
(already built in to fortran libraries)



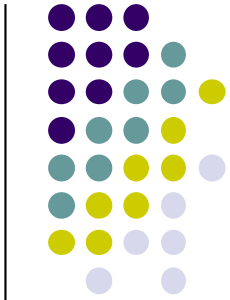
Here are some intrinsic functions. See on-line Tutorial #3 for more.

```
sin(x)        !expects x in RADIANS
sind(x)       !expects x in DEGREES
atan2(x,y)    !arctan of angle with (x,y) coord.Returns RADIANS
log(x)        !this is really the natural log (ln) (base e)
log10(x)      !here is the common log (base 10)
exp(x)        !e to power x
sqrt(x)       !square root of x
abs(x)        !absolute value of x
real(I)       !converts I to a real-number type
int(x)        !converts x to an integer type
max(a, b, c, …)  !finds the max value from a list
```

Examples of use:

```
y = cos(alpha)
z = sqrt(hippopotamus)
xmax = max(x1, x2)
```

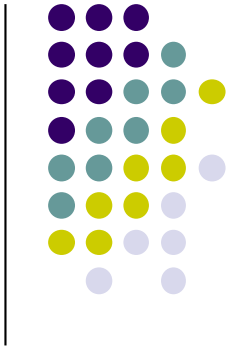
Intrinsic Functions
(already built in to fortran libraries)



gfortran users manual has full list of intrinsic functions and other info:

https://gcc.gnu.org/onlinedocs/gcc-14.1.0/gfortran/Intrinsic-Procedures.html

Function subroutines
(create your own functions)



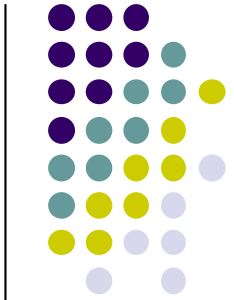
Example of definition of function:

```
real function yst(P)          !gives name and type of function
  uses soundmod              !you can access modules if needed
 implicit none               !always use strong typing
 real, intent(in) :: P       !tells compiler that P is input
 real, parameter :: Po = 100.0    !set  a constant
 yst = log(Po/P)             !finds the ln ordinate on skew-T
end function yst
```

Example of function use:

```
real :: pressure = 85       !pressure is 85 kPa
real :: yst                 !always declare type for function
real :: y                   !ordinate of skew-T diagram
…
y = yst(pressure)           !here is where function is called
```

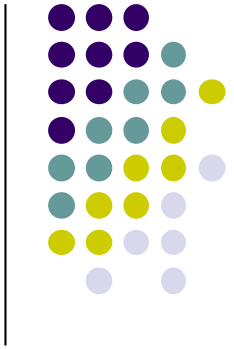
Do Exercises 15 - 17 in HW-fortran2



- Again, This exercise and most others are given at https://www.eoas.ubc.ca/courses/atsc212/labs/fortran-2/fortran2.html

- For exercise 15, follow along with instructor, learning about user-defined functions.
  What is the difference from subroutines?
- Try exercises 16 & 17 on your own. (Stull will follow along after you.)

String Manipulation: tab , trim, concatenation

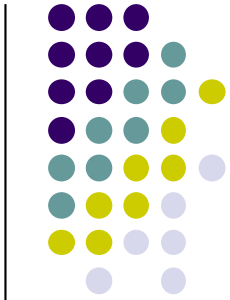


```
!To print an ascii tab character to the output file
 character (len=1) :: tab = achar(9)   !ascii tab character
…
 write(2,*) x, tab, y, tab, z     !easy to read into Excel

!To trim off any trailing blank characters
 character(len=20) :: filein
 character(len=30) :: fileout1, fileout2
 write(*,"(a)",advance="no") "The input file name is: "
 read(*,*) filein            !suppose the user typed burnaby
 fileout1= filein // "out.txt"
 write(*,*) fileout   !gives: "burnaby              out.txt"
 fileout2= trim(filein) // "out.txt"  !gives:"burnabyout.txt"

!where  //  is the concatenation operator (combines two
!strings into a longer string)
```

…more string manipulation: index, substrings



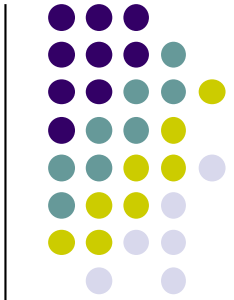
```
!Suppose a character string is “fred.txt”, and you want
!the program to create a new string “fredout.xls”.
 character(len=50) :: inname = “fred.txt”
 character(len=60) :: outname
 integer :: j

!First, use intrinsic function index to find the location
!of the “.” in the name.
  j = index(inname,”.”)
!for example, in “fred.txt”, the “.” is the 5th character.

!Use substrings to pick out a portion of a whole string;
!e.g., inname(2:4) corresponds to the letters “red”.
  outname = inname(1:(j-1)) // “out.xls”

!Thus:
  write(*,*) outname
!would print:
  fredout.xls
```

…more File Handling: output



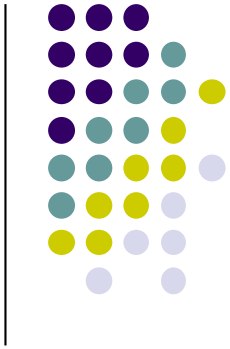
```
!Creating a new file to hold your output:
 character (len=30) :: filename   !name of file to create
 integer :: ios                   !will hold error status
 …
 filename = "myfile.txt"   !set file name or ask user
 open(2, file=filename, status='replace', iostat=ios)
 …  !don't forget to take action if ios .ne. 0

!Writing to that file:
…
 write(2,*) x, y ,z, I
 write(2,*) "This file is :", filename

!Don't forget to close the file when you are done
 close(2)

!Note: reserved unit #'s: 5=keyboard input, 6=screen output
!Note: units have global scope within any program. Can open
!in one subroutine, and use in many others, & close in other.
```

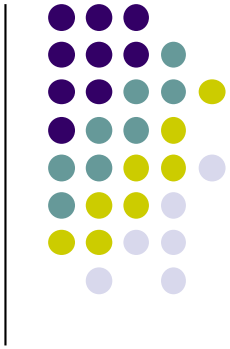
Do Exercise 18 in HW-fortran2



- Again, This exercise and most others are given at https://www.eoas.ubc.ca/courses/atsc212/labs/fortran-2/fortran2.html

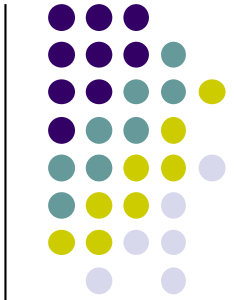
- Follow along with the instructor to create a new output file.

Other Useful Stuff-not covered



- Linking
- "Make" files
- Dynamic allocation of array sizes
- Pointers
- Designing code for multi-processors

Lab & HW - Fortran 2



In Lab:

- Continue now with exercise 18 in lab. Talk with your neighbor on strategies on how to do this. Test your code and experiment.
- If you have compiler errors, use the link from the Lab web page to see how to understand compiler error messages.
  `https://www.eoas.ubc.ca/courses/atsc212/labs/fortran-1/error-msg.html`

- On your own, do exercise 19 as homework.

Any Questions?