Formatted Output

- Two output statements in FORTRAN
  - PRINT and WRITE

- PRINT format-descriptor, output-list

- What is a format descriptor?
  - *
    - A character constant or a character variable whose value specifies
      the format of the output.
    - The label of a FORMAT statement

- Each execution of a PRINT statement displays the values in the output list on a new line.
  - Blank output list → new line
Format Descriptors

• Specifies the format at which the values in the output list are to be displayed.

• * → List directed output

• ‘(list of format descriptors)’ or “(list of format descriptors)”

• Label FORMAT (list of descriptors)
  – Label is an integer from 1 to 99999.
What do format descriptors look like?

- Table 5-1.

- Example:
  - PRINT ‘(I3)’, N
  - Integer printed in first 3 positions of line (right justified).
  - Let N=320, what does it output?
  - Let N=32, what does it output?

- Let’s look more carefully at the format descriptors for integers, reals, and character variables.
Integer output (I descriptor)

• Values output are right justified.

• Book examples (Page 96)

• If an integer value (including a minus sign if the number is negative) requires more spaces than specified by a descriptor, the field is filled with *’s!!!
Real output (F, E, ES, and EN)

- Like integer values, right-justified
- Fw.d
  - \( W = \) width of field, \( d = \) number of digits to right of decimal place.
- If the value has more than \( d \) digits following the decimal point, it is rounded to \( d \) digits.
- If the value has fewer than \( d \) digits, the remaining are filled with zeros.
- Book examples (Page 97)
- Like integers, if the real number being output requires more spaces than specified by the descriptor, the entire field is filled with *’s.
- \( w \geq d + 3 \) Why?
E, ES, and EN

- Let $A = .12345E8$

- E (exponential) descriptor
  - Output displayed in “normalized” format
  - A minus sign, followed by a leading zero, decimal point, $d$ significant digits, then E followed by the exponent in the next four spaces.
  - $0.12345E+08$

- ES (scientific notation) descriptor
  - Same as E, but mantissa (left of decimal point) is at least one, but less than 10.
  - $1.2345E+07$

- EN (engineering) descriptor
  - Same as E, except that exponent is constrained to be a multiple of 3.
  - A non-zero mantissa is greater than or equal to 1 and less than 1000.
  - $12.345E+06$
Character Output

• Character constants can be displayed by including them in the list of descriptors of a format statement.

• Book examples (Page 98)

• Character data may also be displayed by using an A format descriptor, rA or rAw
  – r is the repeatability descriptor and w is the field width.
  – Right justified
  – If the character value exceeds the specified field width, the output consists of the **leftmost** w characters.
Positional descriptors (X and T), Repetition, Slash

- nX – inserts n blanks in an output line.
- Tc – causes the output field to begin at the specified position c on the current line (tab).
- PRINT 75, “John Q. Doe”, “CPSC”, Number 75 FORMAT (1X, A11, 3x, A4, 2X, I3)
  OR
  75 FORMAT(1X, A11, T16, A4, 2X, I3)
- Repetition
  - ‘(1X, A, F6.2, A, F6.2)’ can be written as ‘(1X, 2(A, F6.2))’
- Slash descriptor (/)
  - Causes output to begin on a new line.
  - 88 FORMAT (1x, A, 3/ 1x, 2(I10,F10.2) // 1x, 2E15.7)
Formatted Input

• Formatted input is rarely used in the physical sciences.

• Typically, data is read in from files or obtained from the user as unformatted data.

• As a result, this is where we will place our focus.
File Processing: OPEN (open-list)

- Unit specifier indicating a unit number connected to the file being opened.
- FILE = character-expression
  - Character-expression is the name of the file to be connected to the specified unit number.
- STATUS = character-expression
  - Character-expression is one of:
    - “OLD” – file already exits in the system.
    - “NEW” – file does not yet exist and is being created by the program.
    - “REPLACE” creates a new file, replacing the old one if it exits, and changes its status to “OLD”
OPEN (open-list) cont.

- **ACTION = i-o action**
  - i-o action is a character expression whose value is one of:
    - "READ" – File opened for reading only.
    - "WRITE" – File opened for writing only.
    - "READWRITE" – File opened for both reading and writing.

- **POSITION = character-expression**
  - Character expression is one of:
    - "REWIND" – positions file at its initial point.
    - "APPEND" – positions file at the end.
    - "ASIS" – leaves position unchanged. (default)

- **IOSTAT = status-variable**
  - Status-variable is an integer variable.
  - Status-variable = 0 if file is opened successfully.
  - Status-variable > 0 otherwise. Usually represents an error message found in the system manuals.
Examples

• OPEN (UNIT = 12, FILE = "RAOB.DAT", &
  STATUS = "OLD", ACTION = "READ", &
  POSITION = "REWIND", IOSTAT = OpenStat)

• Typically, it is best not to hardwire UNIT and FILE.
  – Create variables to store their values.
  – Use single statement syntax to open several files.
File Processing: CLOSE (close-list)

- UNIT specifier
  - Must include at least this.

- IOSTAT clause

- STATUS clause

- After a file is closed, it may be re-opened using an OPEN statement.
WRITE (control-list) output-list

- Unit specifier: integer expression whose value designates the output device, or an *.
  - UNIT = unit-specifier or unit-specifier
- Format specifier: may be any of the forms allowed by the PRINT statement.
  - FMT = format-specifier or format-specifier
- ADVANCE = clause statement
  - ADVANCE = character-expression
  - Character-expression is either “YES” or “NO”
  - Should I advance to a newline after output?
  - Default=“YES”
- Other useful file processing commands
Examples

- WRITE(6, *) Temperature, dewpoint
- WRITE(6, FMT= *) Temperature, dewpoint
- WRITE(UNIT = 6, FMT= *) Temperature, dewpoint
- WRITE(Output_Unit, *) Temperature, dewpoint
- WRITE(UNIT= Output_Unit, FMT = *) Temperature, dewpoint.
- WRITE(*,*) Temperature, dewpoint.
Read (control-list) input-list

- Unit specifier indicating the input device.

- Format specifier

- ADVANCE = clause (similar to write statement)

- IOSTAT = clause
  - Used to detect an input error or end-of-file condition.

- Other useful processing commands.
Examples

• READ (5,*) time, temp, dewp

• READ (UNIT = 5, FMT = *) time, temp, dewp

• READ (IN, *) time, temp, dewp
  – Where IN has a value of 5.
File Input/Output

• Once a file is open, it can be written to or read from using the WRITE and READ statements.
• IOSTAT can be used to detect the end of file condition (EOF) or an input error.
• When a READ containing an IOSTAT = status-variable statement is executed, status-variable is:
  – A positive value if an input error occurs.
  – A negative value if end of data (EOF) occurs.
  – Zero is neither EOF or an input error occurs.
Example

• DO
  
  READ (12, *, IOSTAT= InputStat) ID, Temp, Pressure

  IF (InputStat < 0) EXIT

  IF (InputSat > 0) STOP

  Count = Count + 1

  (Other processing)

END DO
Formatted Input

• READ format-specifier, input-list
  – Format specifiers and descriptors are essentially the same as those described for output.

• Integer Input
  – READ 5, I, J, K
    5 FORMAT (I6, I4, I7)
  – Blanks within numeric fields are ignored unless specifically designated otherwise (BZ descriptor)

• Real Input
  – Can be entered without decimal points, or with the decimal point as part of the input.
Formatted Input Examples

• A = 6.25, B = -1.9, C = 75.0, D = .182, E = 625.327

• READ ‘(F3.2, 2F3.1, F3.3, F6.3)’, A, B, C, D, E
  – Can be entered as: 625-19750182625327

• READ ‘(F4.2, 2F4.1, 2F8.3)’, A, B, C, D, E
  – Can be entered as:
    • _625_-19_750_____182__625327

• In the second method, the position of the decimal point in the value entered overrides the position specified by the descriptor.
  – 9423.68
  – F6.2 (without decimal point)
  – F7.2 (with decimal point)
Character Input

- All characters in the field associated with the A descriptor are read.
- Fourscore and seven years ago
- Character (6) :: Speech1, Speech2
  READ ‘(2A)’, Speech1, Speech2
  Speech1 = Foursc, Speech2 = ore an
- AB12345;an,apple a day
  Speech1 = AB1234, Speech2 = 5;an,a
- Read ‘(A2, A12)’, Speech1, Speech2
  Speech1= AB----, Speech2=an,app
REWIND/BACKSPACE

- **REWIND unit-number**
  - Repositions file at its beginning

- **BACKSPACE unit-number**
  - Repositions file at the beginning of the preceding line.

- More Examples.