ADVANCED INPUT AND OUTPUT FORTRAN PROGRAMMING

Zerihun Alemayehu
AAiT.CED  Rm E119B
SIMPLE INPUT AND OUTPUT

- READ, WRITE and PRINT statements are called *list-directed*
  - READ*, a, b, c
  - READ(*,*) a, b, c
  - PRINT*, a, b, c
  - WRITE (*, *) a, b, c
SIMPLE INPUT AND OUTPUT

READ (*, *)  
Read from keyboard  
Use default format

WRITE (*, *)  
Write to screen  
Use default format
FORMAT STATEMENT

- used to read or write data in a form other than the default format.

- label FORMAT ( flist ... )

- where flist is a list of format descriptors which include → I, F, E, ES, EN, D, L, A, H, T, TL, TR, X, P, SP, S

READ (*,100) i, j
READ (*,FMT=200) x, y
WRITE (*,200) x, y

.....
100 FORMAT ( 2I )
200 FORMAT ( 2F10.6 )
We can also specify the FORMAT descriptor list as a *character string* directly in the WRITE or READ statement, as follows:

```plaintext
READ ( *, '(2I)' ) I, J
WRITE ( *, '(2F12.6)' ) X, Y
```

This has the advantage of improved clarity.
FORMS OF FORMAT

WRITE (100) i, x ! Format in FORMAT statement
100 FORMAT (1X, I6, F10.2)

CHARACTER (len = 20) :: string ! Format in character variable
string = '(1X, I6, F10.2)'
WRITE (*,string) i, x

WRITE (*, '(1X, I6, F10.2)') i, x ! Format in character constant
Edit Descriptors

- Edit descriptors specify exactly how values should be converted into a character string on an output device or internal file, or converted from a character string on an input device or internal file.
- The edit descriptors are defined in terms of the following key letters:
  - a - repeat count
  - w - width of field - total number of characters
  - m - number of digits
  - d - digits to right of decimal point
  - e - number of digits in exponent
INTEGR E Edit D escriptors

- I, Iw, Iw.m - integer data
- May be repeated i.e. aI, aIw, aIw.m
- If w is too small to represent the number then on output w asterisks are printed and on input the leftmost w digits are read
- For example: I, I6, I10.3, 5I, 4I6.4
- I6.4 specifies a total of 6 characters including a sign with a minimum of 4 digits thus:
- WRITE (6, `(I6.4)`) 56
- would output two spaces followed by 0056
REAL (FIXED POINT FORM)

- Possible forms: F, Fw, Fw.d, aF, aFw, aFw.d
- If no decimal point is supplied d digits are read as the fractional part.
- For example: F10.5, F12.6, 5F14.7
- F12.6 specifies a total of 12 characters including decimal point and where required a minus sign with 6 digits following the decimal point, thus:
- WRITE(*, `(2F12.6)`) 12.6, -131.4567891 would output bbb12.600000 and -131.456789 where b represents a blank space
REAL - EXPONENTIAL FORM

- Possible forms: E, Ew, Ew.d, aE, aEw, aEw.d
- The E descriptor specifies a number in the following form S0.XXXESXX
  where S signifies a sign, X digits and the character E separates the
  mantissa from the exponent
- On output the exponent is adjusted to place the most significant digit
  to the right of the decimal point eg. 0.123E-2
- Two alternative forms are available
  - EN (Engineering) - the exponent is always divisible by 3 and the value
    before the decimal point lies in the range 1..1000
  - ES (Scientific) - the value before the decimal point always lies in the
    range 1..10
CHARACTER - EDIT DESCRIPTORS

- Possible forms: A, aA, Aw, aAw
- Use to read or write single characters or character strings
- If a field width w is greater than the number of characters then the characters are right justified and space filled
- On input the character string does not need to be enclosed in quotes
LOGICAL - EDIT DESCRIPTORS

- **L** - logical data
- On input T, F, .TRUE., FALSE. are acceptable
- On output T or F will be written
**Horizontal Positioning**

- **Skip Character Positions**
  - `aX` - skip specified number of characters
  - On input characters are ignored
  - On output the required number of spaces is written

- **Column descriptor**
  - `Tc` – go to column number `c`
**Horizontal Positioning**

CHARACTER(len=10) :: first_name = 'James'
CHARACTER :: initial = 'R'
CHARACTER(len=16) :: last_name = 'Johnson'
CHARACTER(len=9) :: class = 'COSC 2301'
INTEGER :: grade = 92
WRITE (*,100) first_name, initial, last_name, grade, class
100 FORMAT (1X, A10, 1X, A1, 1X, A10, 4X, I3, T51, A9)

<table>
<thead>
<tr>
<th>James</th>
<th>R Johnson</th>
<th>92</th>
<th>COSC 2301</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 10 15 20 25 30 35 40</td>
<td>45 50 55 60</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Other Special Characters

- `/` specifies take a new line
- `:` terminate I/O if list exhausted
- `()` group descriptors, normally for repetition
e.g. 4(I5.5,F12.6)
- `\` Output the character string specified
FORMATTED READ

READ ([UNIT=]u, [FMT=]fmt [,IOSTAT=ios]
[,ERR=errorlabel] [,END=endlabel]) [list]

- UNIT - is an I/O device
  - an integer expression,
  - an asterisk (which implies the standard input unit),
  or
  - a character variable in the case of an internal file.
  - READ (UNIT=10,FMT=100,ERR=10,IOSTAT=ios)
FORMATTED WRITE

WRITE (UNIT=] u, [FMT=] fmt [, IOSTAT = ios] [, ERR=errorlabel]) [list]

WRITE (UNIT=file1, FMT=100, REC=recordnumber, ERR=10) newline
Files and file processing
INTERNAL FILES

*Internal file:* is a character variable (or array) which may be written to or read from.

```fortran
CHARACTER (LEN=50) :: CAPTION
NYEAR = 2000
WRITE (CAPTION, 10) NYEAR
10 FORMAT ('SALES FIGURES FOR THE FINANCIAL YEAR', I4)

CHARACTER (30) STRING
STRING = "1984"
READ( STRING, 10) NYEAR
10 FORMAT(I4)
```
EXTERNAL FILES

- In FORTRAN, a file is said to exist if a program is able to access it.
- A file is a stream of data, arranged into records.
  - The records are all either formatted, or unformatted.
  - Files may be accessed sequentially, or directly;
- Sequential files: A sequential file may be thought of as a continuous tape, where records are located sequentially along the tape.
  - only from the beginning
OPEN STATEMENT

- The OPEN statement is used to connect a unit number to a file specifying certain properties for that file which differ from the defaults.
- It can be used to create or connect to an existing file.
- OPEN( [UNIT = ]u, speclist )
  - u is file unit number, and speclist is a list of specifiers
- FILE=filename
  - where filename is a valid filename for the particular system.
  - e.g. FILE='output.test'
OPEN STATEMENT

- STATUS=st
  - where st may be 'OLD', 'NEW', 'REPLACE', 'SCRATCH' or 'UNKNOWN'.
  - If 'OLD' → the file must exist;
  - if 'NEW' → the file must not exist;
  - if 'REPLACE' and the file exists it will be deleted before a new file is created;
  - and if 'SCRATCH' the file will be deleted when closed.
  - In general use 'OLD' for input and 'NEW' for output.

- ERR=label
  - GOTO label if an error occurs opening the file.
OPEN STATEMENT

- **IOSTAT=ios**
  - *ios* is an integer variable which is set to zero if the statement is executed successfully or to an implementation dependent constant otherwise.

- **FORM=fm**
  - *fm* may be 'FORMATTED' or 'UNFORMATTED', the default is 'FORMATTED' for sequential files and 'UNFORMATTED' for direct access files.

- **ACCESS=acc**
  - *acc* may be 'SEQUENTIAL' or 'DIRECT'
CLOSE STATEMENT

This statement permits the orderly disconnection of a file from a unit either at the completion of the program, or to alter a property of the file.

CLOSE ([UNIT=]u [,IOSTAT=ios] [,ERR=label] [,STATUS=st])

- **st** can be 'KEEP' or 'DELETE'. The value 'KEEP' cannot be applied to a file opened as 'SCRATCH'.
- CLOSE (10)
- CLOSE (UNIT=10, ERR=10)
- CLOSE (UNIT=NUNIT, STATUS='DELETE',ERR=10)
INQUIRE STATEMENT

This statement may be used to check the status of a file or the connection to a file.

INQUIRE (inquiry-list)

- where inquiry-list may be either
- FILE=fname or UNIT=unum
- EXIST=lex ! true or false
- OPENED=lod ! true or false
- NUMBER=unum ! unit number
- NAME=fnm ! filename
- ACCESS=acc ! 'DIRECT' or 'SEQUENTIAL'
- FORMATTED=fmt ! 'YES' or 'NO'
- FORM=frm ! 'FORMATTED' or 'UNFORMATTED'
A file's records may be *unformatted*. They take up much less storage than formatted records.

```fortran
INTEGER, DIMENSION (10) :: A = (/ (I, I = 1,10) /)
OPEN( 1, FILE = 'TEST', FORM = 'UNFORMATTED' )
WRITE (1) A
REWIND (1)
A = 0             ! Just to be sure
READ (1) A
PRINT*, A
CLOSE (1)
END
```
**DIRECT ACCESS FILES**

- A direct access file is a random access table-like structure which may be inspected or amended.
- May be created and accessed using the RECL and ACCESS='DIRECT'

```fortran
CHARACTER (LEN=200) :: STAFFRECORD

OPEN (UNIT=10, FILE='STAFF.RECORDS', &
RECL=200, ACCESS='DIRECT', ......)

READ(UNIT=10,REC=20) STAFFRECORD

WRITE(UNIT=10,REC=20) STAFFRECORD
```
NON ADVANCING I/O

This permits records to be read in sections (for example a long record of unknown length) or to create a neat user-interface where a prompt for input and the user's response appear on the same line.

WRITE(*,*,ADVANCE='NO') 'Enter new value: '
READ(*,*) I

If the user enters the value 10 this would appear on the screen as

Enter new value: 10