

FILE PROCESSING

Why Files?

- the amount of data read and / or produced is huge
- repetitive data is needed for more than one program
- data may be entered by other people or instruments

■ Opening Files

- Preparing a file for input:

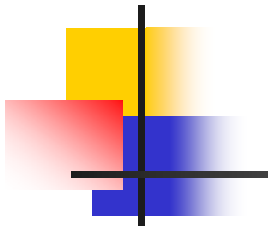
```
OPEN(UNIT = INTEGER EXPR, FILE = FILENAME, STATUS = 'OLD')
```

- Preparing a file for output:

```
OPEN(UNIT = INTEGER EXPR, FILE = FILENAME, STATUS = 'NEW')
```

Or

```
OPEN(UNIT = INTEGER EXPR, FILE = FILENAME, STATUS = 'UNKNOWN')
```



Example 1: Assume that you want to use file POINTS.DAT as an input file. The following statement will then appear before any read statement from the file:

```
OPEN(UNIT = 1, FILE = 'POINTS.DAT', STATUS = 'OLD')
```

Example 2: Assume that you want to use file RESULT.DAT as an output file. The following statement will then appear before any write statement to the file:

```
OPEN(UNIT = 1, FILE = 'RESULT.DAT', STATUS = 'UNKNOWN')
```

Reading from Files

```
READ(UNIT, *) VARIABLE LIST
```

Example 1: Find the sum of three exam grades taken from file EXAM.DAT.

Solution:

```
INTEGER  EXAM1, EXAM2, EXAM3, SUM
OPEN(UNIT = 10, FILE = 'EXAM.DAT', STATUS = 'OLD')
READ(10, *) EXAM1, EXAM2, EXAM3
SUM = EXAM1 + EXAM2 + EXAM3
PRINT*, 'SUM OF THE GRADES=', SUM
END
```

Reading from Files

READ(UNIT, *, END = NUMBER) VARIABLE LIST

Example 2: Find the average of real numbers that are stored in file NUMS.DAT. Assume that we do not know how many values are in the file and that every value is stored on a separate line.

Solution:

```
REAL NUM, SUM, AVG
INTEGER COUNT
OPEN(UNIT = 12, FILE = 'NUMS.DAT', STATUS = 'OLD')
SUM = 0.0
COUNT = 0
333 READ(12, *, END = 999) NUM
SUM = SUM + NUM
COUNT = COUNT + 1
GOTO 333
999 AVG = SUM / COUNT
PRINT*, 'AVERAGE =', AVG
END
```



Writing to Files

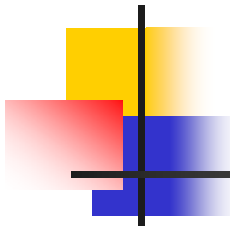
WRITE(UNIT, *) EXPRESSION LIST

Example: Create an output file CUBES.DAT that contains the table of the cubes of integers from 1 to 20 inclusive.

Solution:

```
INTEGER NUM
OPEN(UNIT = 15, FILE = 'CUBES.DAT', STATUS = 'UNKNOWN')
DO 100 NUM = 1, 20
    WRITE(15, *) NUM, NUM**3
100 CONTINUE
END
```

Working with Multiple Files



Example: Create an output file THIRD that contains the values in file FIRST followed by the values in file SECOND. Assume that every line contains one integer number and we do not know how many values are stored in files FIRST and SECOND.

Solution:

```
INTEGER NUM
OPEN(UNIT = 15, FILE = 'FIRST.DAT', STATUS = 'OLD')
OPEN(UNIT = 17, FILE = 'SECOND.DAT', STATUS = 'OLD')
OPEN(UNIT = 19, FILE = 'THIRD.DAT', STATUS = 'UNKNOWN')
33 READ(15, *, END = 66) NUM
   WRITE(19, *) NUM
   GOTO 33
66 READ(17, *, END = 30) NUM
   WRITE(19, *) NUM
   GOTO 66
30 STOP
END
```



Closing Files

CLOSE (UNIT)

Rewinding Files

REWIND (UNIT)

Example:

Given a data file 'INPUT.DAT' that contains unknown number of lines. Each line has a student ID and his grade. Write a program that reads the data from the above file and writes to a new file 'OUTPUT.DAT' in each line the ID and the grade of all the students who have grades greater than the average.

```
REAL GRADE, SUM, AVG
INTEGER ID, COUNT, K
OPEN (UNIT = 20, FILE = 'INPUT.DAT', STATUS = 'OLD')
OPEN (UNIT = 21, FILE = 'OUTPUT.DAT', STATUS = 'UNKNOWN')
SUM = 0.0
COUNT = 0
44 READ (20, *, END = 100 ) ID, GRADE
SUM = SUM + GRADE
COUNT = COUNT + 1
GOTO 44
100 AVG = SUM / COUNT
REWIND (20)
DO 50 K = 1, COUNT
    READ (20, * ) ID, GRADE
    IF (GRADE .GT. AVG) WRITE (21, *) ID, GRADE
50 CONTINUE
END
```

Exercises

What will be printed by the following program?

```
INTEGER M, K
OPEN ( UNIT = 30, FILE = 'INPUT.DAT', STATUS = 'OLD')
READ ( 30, *, END = 20) ( M, K = 1, 100)
20 PRINT*, M, K - 1
END
```

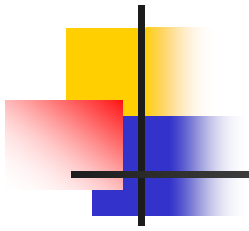
Assume that the file 'INPUT.DAT' contains the following:

```
1    2    3
4    5
6    7    8    9
6
```

The output

6 10

What will be printed by the following program?



```
INTEGER J, K
OPEN ( UNIT = 4, FILE = 'FF1.DAT', STATUS = 'OLD')
DO 50 J = 1, 100
    READ ( 4, *, END = 60) K
50 CONTINUE
60 PRINT*, ' THE VALUES ARE:'
   PRINT*, K, J
END
```

The contents of the file 'FF1.DAT' are:

```
20  50  67  45  18  -2  -20
88  66  77 105  55 300
```

The output	
THE VALUES ARE:	
88	3

What will be printed by the following program?

```
INTEGER M
OPEN ( UNIT = 10, FILE = 'INPUT.DAT',STATUS = 'OLD')
READ (10,*) M
20 IF ( M .NE. -1) THEN
    PRINT*, M
    READ(10, *, END = 30) M
    GOTO 20
ENDIF
PRINT*, 'DONE'
30 PRINT*, 'FINISHED'
END
```

The output

```
7
3
9
4
DONE
FINISHED
```

Assume that the file 'INPUT.DAT' contains the following :

```
7
3
9
4
-1
```

What will be printed by the following program?

```
INTEGER A, B
```

```
OPEN ( UNIT = 20, FILE = 'INPUT.DAT', STATUS = 'OLD')
```

```
OPEN ( UNIT = 30, FILE = 'OUTPUT.DAT', STATUS = 'NEW')
```

```
READ*,A, B
```

```
READ(20, *) A, B, A
```

```
WRITE(30,*) A, B
```

```
READ(20, *, END = 40) A, B
```

```
40 WRITE(30, *) A, B
```

```
END
```

The output

6	5
8	5

Assume the input for the program is:

10 11

Assume that the file 'INPUT.DAT' contains the following data

4 5
6 7
8



OUTPUT DESIGN

PRINT K, expression list

K FORMAT (specification list)

The following are some of the carriage control characters used to control the vertical spacing:

- ' ' : single spacing (start printing at the next line)
- ' 0 ' : double spacing (skip one line then start printing)
- ' - ' : triple spacing (skip 2 lines then start printing)
- ' 1 ' : new page (move to the top of the next page before printing)
- ' + ' : no vertical spacing (start printing at the beginning of the current line irrespective of what was printed before)

I Specification (Iw)

for integer numbers

Example : Assume $K = -244$ and $M = 12$. What will be printed after the execution of the following print statements ?

10 PRINT 10, K
FORMAT(' ', 14)

-	2	4	4
---	---	---	---

20 PRINT 20, K, M
FORMAT(' ', 15, 16)

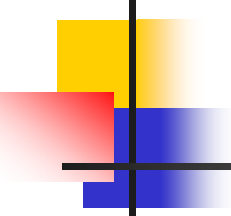
	-	2	4	4					1	2
--	---	---	---	---	--	--	--	--	---	---

30 PRINT 30, K
PRINT 40, M
FORMAT(' ', 13)
40 FORMAT('0', 12)

*	*	*
1	2	

50 PRINT 50, K + M
FORMAT(' ', 15)

	-	2	3	2
--	---	---	---	---



```
PRINT 60, M + 1.0
60  FORMAT(' ', I3)
```

Error Message
Type Mismatch

```
PRINT 70, K
PRINT 70, M
70  FORMAT(' ', I4)
```

-	2	4	4
		1	2

```
PRINT 80, K
PRINT 90, M
80  FORMAT(' ', I4)
90  FORMAT('+', I8)
```

-	2	4	4			1	2
---	---	---	---	--	--	---	---

```
PRINT 99, N
99  FORMAT(' ', I3)
```

?	?	?
---	---	---

F Specification (Fw.d)

for real numbers

Example : Assume $X = -366.126$, $Y = 6.0$ and $Z = 20.97$. What will be printed after the execution of the following print statements ?

10 PRINT 10, X
FORMAT(' ', F11.5)

	-	3	6	6	.	1	2	6	0	0
--	---	---	---	---	---	---	---	---	---	---

20 PRINT 20, X
FORMAT(' ', F8.2)

	-	3	6	6	.	1	3
--	---	---	---	---	---	---	---

30 PRINT 30, X
FORMAT(' ', F7.3)

*	*	*	*	*	*	*
---	---	---	---	---	---	---

40 PRINT 40, Z
FORMAT('+', I5)

Error Message
Type Mismatch

X Specification (nX)

to insert blanks

Example : The following program:

```
REAL A, B
A = -3.62
B = 12.5
PRINT 5, A, B
5  FORMAT(' ', F5.2, F4.1)
END
```

Prints the following output

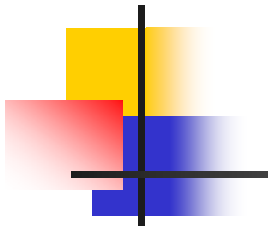
-	3	.	6	2	1	2	.	5
---	---	---	---	---	---	---	---	---

If we modify the FORMAT statement using X Specification as follows:

```
5  FORMAT(' ', F5.2, 3X, F4.1)
```

The output

-	3	.	6	2				1	2	.	5
---	---	---	---	---	--	--	--	---	---	---	---



The X specification can be used as a carriage control character.

The following pairs of format statements print the same output

```
10  FORMAT(' ', I2)
```

is equivalent to

```
10  FORMAT(1X, I2)
```

- - - - -

```
20  FORMAT(' ', 2X, F4.1)
```

is equivalent to

```
20  FORMAT(3X, F4.1)
```

Exercises

What will be printed by the following program?

```
REAL X
X = 123.8367
PRINT 10, X, X, X
10 FORMAT(' ', F7.1, 2X, F6.2, F9.5)
END
```

		1	2	3	.	8			1	2	3	.	8	4	1	2	3	.	8	3	6	7	0
--	--	---	---	---	---	---	--	--	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

What will be printed by the following program?

```
INTEGER J, K, N
K = 123
J = 456
N = 789
PRINT 10, N
PRINT 11, J
PRINT 12, K
10 FORMAT(' ', 6X, I3)
11 FORMAT('+', 3X, I3)
12 FORMAT('+', I3)
END
```

1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---

Literal Specification

string in a format statement

Example : What will be printed by the following program?

```
REAL AVG
AVG = 65.2
PRINT 5, AVG
5  FORMAT(' ', 'THE AVERAGE IS ', F4.1)
END
```

The output: THE AVERAGE IS

6	5	.	2
---	---	---	---

A Specification (Aw) for character expressions

Example1 : What will be printed by the following program?

```
CHARACTER TEXT*5  
TEXT = 'KFUPM'  
PRINT 55, TEXT, TEXT, TEXT  
55  FORMAT(' ', A, 3X, A3, 3X, A9)  
END
```

- w may be omitted. If w is omitted, the number of characters will be determined by the length of the character string
- if the string has more characters than the w positions: print the left-most w characters
- if the string has fewer characters than the w positions: print the string right-justified

K	F	U	P	M				K	F	U								K	F	U	P	M
---	---	---	---	---	--	--	--	---	---	---	--	--	--	--	--	--	--	---	---	---	---	---

Example2 : What will be printed by the following program?

```
CHARACTER TEXT*7  
TEXT = 'KFUPM'  
PRINT 66, TEXT, TEXT, TEXT  
66 FORMAT(' ', A, 3X, A3, 3X, A9)  
END
```

K	F	U	P	M						K	F	U						K	F	U	P	M		
---	---	---	---	---	--	--	--	--	--	---	---	---	--	--	--	--	--	---	---	---	---	---	--	--

Example3 : What will be printed by the following program?

```
CHARACTER TEXT*4  
TEXT = 'KFUPM'  
PRINT 77, TEXT, TEXT, TEXT  
77 FORMAT(' ', A, 3X, A3, 3X, A9)  
END
```

K	F	U	P				K	F	U									K	F	U	P
---	---	---	---	--	--	--	---	---	---	--	--	--	--	--	--	--	--	---	---	---	---

L Specification (Lw)

for logical expressions

Example1 :

```
PRINT 10, .TRUE.  
10  FORMAT(' ', L4)  
END
```



Example2 :

```
PRINT 20, .FALSE. , .TRUE.  
20  FORMAT (' ', L2 , 3X , L3)  
END
```



Example:

```
INTEGER K
LOGICAL M
REAL X, Y
CHARACTER TEXT*8
K = 346
X = 56.352
Y = -24.628
M = .NOT. .TRUE.
TEXT = 'FORTRAN'
PRINT 10, X, Y
PRINT 20, Y, K
PRINT 30, TEXT, TEXT
PRINT 40, K
PRINT 50, M , .TRUE. , 'FINAL'
10  FORMAT(' ', 2X, 'X=', F7.4, 3X, 'Y=', F5.1)
20  FORMAT(' ', 3X, 'Y=', F6.2, 2X, 'K=', I3)
30  FORMAT('0', 9X, A, 1X, A2)
40  FORMAT('+', 4X, 'K=', I2)
50  FORMAT(4X, 'M=', L4 , 2X , L3 , 1X, A)
END
```

		X	=	5	6	.	3	5	2	0				Y	=	-	2	4	.	6	
				Y	=	-	2	4	.	6	3			K	=	3	4	6			
						K	=	*	*		F	O	R	T	R	A	N		F	O	
						M	=				F				T		F	I	N	A	L