Computers in Engineering COMP 208

Formatting Input and Output

Michael A. Hawker

Verifying ISBN Numbers

```
program isbn
   implicit none
   integer :: digits(10)
   integer :: pos, sum
   logical :: valid
   read (*,"(10I1)") (digits(pos), pos = 1,10)
   sum = 0
   do pos = 1,10
        sum = sum + (11-pos)*digits(pos)
   end do
   valid = mod(sum,11) == 0
   if (valid) then
        write(*,*) "ISBN is valid"
   else
        write(*,*) "ISBN is invalid"
   end if
end program isbn
```


The second *

- The second * in the read was replaced by a format string
- A format string can be used to specify where to look for the input
- The format (`10I1`) means we expect
  - 10 values
  - Each value is an integer (`I`)
  - Each value is one digit in length (1)

FORTRAN Formats

- The READ and WRITE statements we have seen so far are called ***free-format*** statements.
- They are easy to use but we have no control over the placement of the input or appearance of the output.
- To control the appearance of the input and output, Fortran allows us to use **format specifications**

How much was that?

```
PROGRAM cost
  IMPLICIT NONE
  REAL :: price, gst, pst
  READ(*,*) price
  gst = 0.07*price
  pst = 0.075*(price + gst)
  WRITE(*,*) "Price: ",price
  WRITE(*,*) "GST: ", gst
  WRITE(*,*) "PST: ", pst
  WRITE(*,*) "Total Cost: ",price+gst+pst
END PROGRAM cost
```


The results aren't very pretty.

```
136.95
 Price:     136.9500
 GST:     9.586500
 PST:     10.99024
 Total Cost:     157.5267
```

Wouldn't this be nicer?

```
136.95
        Price:  136.95
          GST:    9.59
          PST:   10.99
   Total Cost:  157.53
```


Formats

- FORTRAN formats allow us to specify the placement of values both in output and input
- Using format descriptors we can control the appearance of output values
- Format descriptors specify
  - The appearance of output values
  - Repetition
  - Vertical positioning
  - Horizontal positioning

Fortran Formats – Method 1

There are two possible ways to specify a format.

In the first, we write the format as a character string and use it to replace the second asterisk in `WRITE(*,*)`.

```
WRITE(*,"(A15,F7.2)") "Total Cost: ", &
  price+gst+pst
```

Fortran Formats – Method 2

The most common method uses a FORMAT statement

A FORMAT statement has the syntax:

```
label FORMAT format-code
```

To use the format, we specify its label in the WRITE statement

```
    WRITE(*,100) "Total Cost: ", price+gst+pst
100 Format (A15,F7.2)
```


Cost With Formatting

```
PROGRAM cost
    IMPLICIT NONE
    REAL :: price, gst, pst
    READ(*,*) price
    gst = 0.07*price
    pst = 0.075*(price + gst)
    WRITE(*,100) "Price: ",price
    WRITE(*,100) "GST: ", gst
    WRITE(*,100) "PST: ", pst
    WRITE(*,100) "Total Cost: ",price+gst+pst
100 FORMAT (A15,F7.2)
END PROGRAM cost
```


Fortran Formats

The FORMAT **statement** in the previous example specifies a format

```
100 FORMAT (A15,F7.2)
```

A **format** is list of descriptors inside parentheses

```
( desc1, desc2, desc3, . . . )
```

Format Codes

- We will look at some of the many format codes available in FORTRAN for specifying:
  1. Real values
  2. Integer values
  3. Character values
  4. Horizontal spacing
  5. Vertical spacing

Real Values
Fixed Point Notation

```
100 Format (A15,F7.2)
```

- The second format code in the list specifies that we are to print a real number using two decimal points
- The in the code tells the computer to allow seven spaces to fit the number into

Real Numbers – Fixed Point

- The general format code for has the form
  Fw.d
- The d specifies the number of decimal places
- The w specifies the field width and includes space for
  1. d decimal digits
  2. The decimal point
  3. The whole number
  4. The sign, if the number is negative

F Format Example

Example

```
      REAL :: x=1.0, y=1100.1003
      WRITE(*, 900) x, y
900   FORMAT (F3.1, F10.4)
```

F3.1 is format code for x and F10.4 is for y

```
 1.0#1100.1003
```


Variations on a Theme

```
real :: x=1.0, y=1100.1003
write (*,"(F3.1,F11.4)") x, y
write (*,"(F3.1,F10.4)") x, y
write (*,"(F3.1,F9.4)") x, y
write (*,"(F3.1,F8.4)") x, y
```

- Results:

```
1.0  1100.1003
1.0 1100.1003
1.01100.1003
1.0********
```


Oops!

- What happened in the last example?
- Whenever a value to be output does not fit into the allocated field width, w, the computer just outputs w *'s\
- This is true of any type of value, not just real numbers

Real Numbers Exponential Notation

- The E format descriptor has the form
  `Ew.d`
- They are displayed as a normalized number between 0.1 and 1.0, multiplied by a power of 10
- The output is in the form
  `±0.ddddE±ee`
- The number of significant digits is specified by d, the exponent uses two places
- We must have w≥d+7

E Code Variants

Example:

```
real :: y=1100.1003
write (*,"(E15.8)") y
write (*,"(E15.4)") y
write (*,"(E15.2)") y
write (*,"(E12.8)") y
```

Results:

```
 0.11001003E+04
     0.1100E+04
       0.11+E04
************
```


Integer Numbers "I" Format Codes

- The general format code for has the form

  `Iw`
- The w specifies the field width
- Numbers are right justified
- If a number doesn't fit, *'s are output

Character Values "A" Format Code

- The general format code for has the form

  Aw
- The w specifies the field width
- Strings are right justified
- If a number doesn't fit, the first w characters are output
- If w is left out, the entire character string is printed

Cost With Formatting

```
PROGRAM cost
    IMPLICIT NONE
    REAL :: price, gst, pst
    READ(*,*) price
    gst = 0.07*price
    pst = 0.075*(price + gst)
    WRITE(*,100) "Price: ",price
    WRITE(*,100) "GST: ", gst
    WRITE(*,100) "PST: ", pst
    WRITE(*,100) "Total Cost: ",price+gst+pst
100 FORMAT (A15,F7.2)
END PROGRAM cost
```


Repetition Factors

- A format code or group of codes can be repeated by putting a value in front
- For example:

  `10I1` means output (or input) 10 digits

  `5(A3, I5)` is equivalent to

  `A3, I5, A3, I5, A3, I5, A3, I5, A3, I5`

Horizontal Spacing

- To skip a space horizontally, we have the format code X
- Using a repetition factor, nX, indicates "skip n spaces"

```
      INTEGER :: a=1000
      WRITE (*,100) "a=", a
 100  FORMAT(A, 4X, I4)
```

- Output

```
a=    1000
! ####
```


Vertical Spacing

- To skip a space vertically, we have the format code /
- Using a repetition factor, n/, indicates "skip n lines"

```
      INTEGER::a=1000
      WRITE(*,100) "a=", a
 100  FORMAT(A, 2/, I4)
```

- Output

```
a=
#
#
1000
```


Format on Input

- When using format with a READ statement, the input values must be positioned according to the format specifications

Format on Input

- Example

```
      INTEGER :: a,b
      READ(*,100) a,b
  100 FORMAT(2I5)
```

- This reads the first 5 characters on the input line, converts them to an integer and stores the result in a.
- It then reads the next five characters, converts them and stores the result in b

Formatted Read

Correct inputs for Format code `2I5`:

```
1234567890     → a=12345, b=67890
123456         → a=12345, b=6
####12345#     → a=1, b=2345
###1234567890  → a=12, b=34567
```

Incorrect inputs:

```
1234,5678
123456789a
12, 14
```


Reading Fixed Point Reals

Example

```
READ(*, "(F5.1)") x
```

Results

```
##3.4       → x=3.4
123.456     → x=123.4
12345       → x=1234.5
```