Programming Techniques 2025-2026

Lecture 7: Input/Output in Fortran: Read, Print, Write, and Formatting

Hannu Parviainen

Universidad de la Laguna

October 6, 2025

Introduction to I/O in Fortran

Fortran provides three main IO statements

- `read`: Input
- `print`: Output to the console
- `write`: Output to files or console

IO formatting allows for better control over how input and output data is handled.

```
read *, v1, v2 ,v3, ...
read fmt, v1, v2 ,v3, ...
read (unit, fmt) v1, v2, v3, ...

print *, v1, v2, v3, ...
print fmt, v1, v2, v3, ...

write (unit, *) v1, v2, v3, ...
write (unit, fmt, advance, ...) v1, v2, v3,
    ...
```

Using the `read` Statement

The `read` statement is used for reading input from the user or files.

Syntax:

```
read fmt, variable
read(unit, fmt) variable
```

`*` stands for default formatting and input and output (usually the terminal).

Example:

```
program example_read
  implicit none
  integer :: a
  print *, "Enter an integer: "
  read(*, *) a
  print *, "You entered: ", a
end program example_read
```

Using the `print` Statement

The `print` statement is used for output to the console.

Syntax:

```
print fmt, "Message"
```

Example:

```
program example_print
  implicit none
  integer :: num = 10
  print *, "The value of num is: ", num
end program example_print
```

Using the `write` Statement

The `write` statement is more flexible than `print` and allows output to different IO units, like files or other devices than the console.

Syntax:

```
write(unit, format) variable
```

Example:

```
program example_write
  implicit none
  integer :: i
  open(unit=10, file='output.txt')  ! Open a file
  do i = 1, 5
    write(10, *) "Line number: ", i
  end do
  close(10)  ! Close the file
end program example_write
```

IO Formatting in Fortran

Formatting is controlled with format descriptors.

- Format descriptors:
  - `rIw`: integer.
  - `rFw.d` floating point.
  - `rEw.d` real in exponential notation.
  - `rESw.d` real in scientific notation.
  - `rAw` character string.
  - `X` space.
  - `/` blank line.
- Modifiers:
  - `r`: repeat count.
  - `w`: field width.
  - `d`: num. digits after the decimal point.

```
program format_example
  implicit none
  integer :: a = 123
  real :: b = 456.789
  real, dimension(3) :: c = [1.2, 2.2, 1.2]
  print '(I5, 10X, F8.2, /, 3F8.1)', a, b, c
end program format_example
```

```
  123              456.79
  1.2      2.2      1.2
```

Reading and Writing to Files

The `read` and `write` statements can be used to interact with files.

Use the `open` statement to open a file and associate it with a unit number.

- `open(unit, file, status, action, ...)`
- `close(unit, ...)`
- `inquire(file, exists)`

`status` can be `new`, `old`, or `replace`.

`action` can be `read`, `write`, or `readwrite`.

```
program file_io
  implicit none
  integer :: i
  open(unit=20, file='data.txt', status=
       'replace')
  do i = 1, 5
    write(20, '(I5)') i
  end do
  close(20)
end program file_io
```