FORTRAN Reference Card

Program Structure

`PROGRAM` *name*	Begin program *name*
`END PROGRAMM` *name*	
`SUBROUTINE` *name*	Begin subroutine *name*
`END SUBROUTINE` *name*	
`MODULE` *name*	Begin module *name*
`END MODULE` *name*	

Fortran Preprocessors

`IMPLICIT NONE`	Avoid using predefined data types

Intrinsic Data Types

`INTEGER`	Number without fractional part
`REAL`	Floating-point format
`COMPLEX`	`(a,b)` Complex number
`CHARACTER`	String of characters enclosed in ' or "
`LOGICAL`	`.true.` or `.false.`

Derived Data Types

`TYPE` *type1*	Define a new structure
`END TYPE`	
`TYPE, EXTENDS(type1) ::` *type2*	Extending an existing type
`TYPE(type1) :: name`	Create type
`type1%component`	Access component of type

Example:

```
type Books
  character(len = 50) :: title
  integer :: book_id
end type Books
type(Books) :: book1
book1%title = "Night"
book1%book_id = 1
```

Additional Attributes

`KIND` = *val*	Define presicion of a real
val = 4 (32bit), 8(64bit)	GNU Fortran compiler
`PARAMETER`	Value is set to be constant
`DIMENSION`	Assign dimension to an object
`POINTER`	Object will be pointer to content
`TARGET`	Object is target for a pointer
`ALLOCATABLE`	Object can be allocated
`PRIVATE`	Only access object in module
`PUBLIC`	Not privat

Example: `INTEGER, PARAMETER :: x, value`
`CHARACTER(len=20) :: name` Fortran is *case insensitive.*

Arrays

Miscellaneous

`!`	Comment (older versions: `C`)
`&`	Continue statement in new line

Statement labels are numbers without meaning but they can be used to refer to a statement.
Example: `100 output = x + y`

Flow Control

`DO` *name*	While loop
`IF (` *logical_expr* `) EXIT`	Exit condition
`END DO`	
`DO index = istart, iend, incr`	Iterative do loop
`Statements`	
`END DO`	

Loops and branching statements can have names.
Example: `loopname: DO [..] END DO loopname`

`STOP`*'optional string'*	Terminate program
`ERROR STOP` *'error msg'*	Informs system that program failed after terminating ✿

The `STOP` statement is more or less reduntant.

Operators

Operations beginning with highest in hierarchy.

Exponentiation	`**`		
Multiplication	`*`	Division	`/`
Addition	`+`	Subtraction	`-`

`=>` Assign target/object to pointer

Important Functions

`date_and_time`	Get the date and time
`random_seed(size=k)`	Get a random number of size k

External functions are called with `CALL`.

Math Functions

`INT(x)`	Integer part of x	`INT(2.95)` → 2
`NINT(x)`	Round x	`NINT(2.95)` → 3
`CEILING(x)`	Nearest integer above x	`CEILING(2.95)` → 3
`FLOOR(x)`	Nearest integer below x	`FLOOR(2.95)` → 2
`REAL(i)`	Convert integer to real	
`SQRT(x)`	Square root of x for $x \geq 0$	
`ABS(x)`	Absolute value of x	
`SIN(x), SIND(x)`	Sine of x (in radians, degrees)	
`COS(x), COSD(x)`	Cosine of x (radians, degree)	
`TAN(x), TAND(x)`	Tangent of x (radians, degree)	
`EXP(x)`	e to the xth Power	
`LOG(x), LOG10(x)`	Natural logarithm, Base 10-logarithm	
`MOD(a,b)`	Modulo function	
`MAX(a,b), MIN(a,b)`	Pickes larger/smaller of a and b	
`NORM2(array)`	Calculate Euclidean norm (L_2 norm)✿	
`ERF(x), ERFC(X)`	(Complementary) Error function ✿	

Input/Output

`WRITE (*,*)`	Print to standard output stream
`READ (*,*)`	Read from standard input stream
`PRINT *,`	Print to standard output stream

Formating I/O

Compilation gfortran

Using gfortran on a UNIX-like system.
`gfortran myprogram.f -o myprogram.out`
File extensions (recommendation is first one):
`file.f90` free-form source, no preprocessing
`file.F90` free-form source, preprocessing
`file.f` fixed-form source, no preprocessing
`file.F` fixed-form source, preprocessing
Several files:
First compile subfiles `gfortran -c module.f90` Then all
`gfortran main.f90 module.o -o main.o`
Other Options:

`-std=f95`	Set standard for compiler (`f2003, f2008, gnu=default, legacy`)
`-Wextra -Wall -pedantic`	Recommended warnings
`-c`	Necessary if only module or subroutine compilation

Comments

Variable types are indicated by the used character:
x = real; i= int; a,b= int/real
Functions from standard after 95 are marked with a ✿ .

2022 Tracy Kiszler; No guarantee for anything :)
https://github.com/TracyMcBean/Fortran_cheatsheet