Fortran Programming Cheatsheet

A quick reference guide to Fortran programming, covering syntax, data types, control structures, and more. Useful for both beginners and experienced Fortran developers.

Fundamentals

Basic Syntax

Program Structure:

```
program program_name
  ! Declarations
  implicit none
  ! Statements
  ...
end program program_name
```

Comments:

```
! This is a comment.
```

Continuation:

```
result = very_long_expression + &
         another_long_expression
```

Case Insensitivity:

Fortran is generally case-insensitive.

Data Types

`integer`	Whole numbers (e.g., `1`, `-5`, `1000`)
`real`	Floating-point numbers (e.g., `3.14`, `-2.0`, `0.0`) - single precision by default
`double precision`	Floating-point numbers with higher precision (e.g., `3.1415926535`)
`complex`	Numbers with real and imaginary parts (e.g., `(1.0, 2.0)`)
`logical`	Boolean values: `.true.` or `.false.`
`character`	Strings of characters (e.g., `'Hello'`, `"Fortran"`)

Variable Declaration

```
integer :: age
real :: temperature
double precision :: pi
character(len=20) :: name
logical :: is_valid
```

Implicit None:

Always use `implicit none` to force explicit declaration of all variables.

Operators & Control Flow

Operators

`+`	Addition
`-`	Subtraction
`*`	Multiplication
`/`	Division
`**`	Exponentiation
`==` or `.EQ.`	Equal to
`/=` or `.NE.`	Not equal to
`.GT.`	Greater than
`.GE.`	Greater than or equal to
`.LT.`	Less than
`.LE.`	Less than or equal to

Control Flow

IF Statement:

```
if (condition) then
  ! Statements
else if (condition) then
  ! Statements
else
  ! Statements
end if
```

DO Loop:

```
do i = start, end, increment
  ! Statements
end do
```

SELECT CASE Statement:

```
select case (expression)
  case (value1)
    ! Statements
  case (value2)
    ! Statements
  case default
    ! Statements
end select
```

Arrays and I/O

Arrays

Declaration:

```
integer :: matrix(3, 3)
real :: vector(10)
```

Array Slicing:

```
vector(1:5)  ! Elements 1 to 5
matrix(:, 1)  ! First column
```

Array Operations:

Fortran supports element-wise array operations.

Input/Output

Reading from standard input:

```
read *, variable1, variable2
```

Writing to standard output:

```
print *, "Result:", result
```

Formatted I/O:

```
print '(I5, F10.2)', integer_value, real_value
```

File I/O:

```
open(unit=10, file='data.txt', status='old')
read(10, *) data
close(10)
```

Subroutines & Functions

Subroutines

Definition:

```
subroutine subroutine_name(arg1, arg2)
  ! Declarations
  implicit none
  ! Statements
  ...
end subroutine subroutine_name
```

Calling a Subroutine:

```
call subroutine_name(value1, value2)
```

Functions

Definition:

```
function function_name(arg1, arg2)
result(result_variable)
  ! Declarations
  implicit none
  ! Statements
  result_variable = ...
end function function_name
```

Calling a Function:

```
result = function_name(value1, value2)
```

Modules

Definition:

```
module module_name
  ! Declarations
  implicit none
  ! Public and private variables and
subroutines
contains
  ! Subroutine and function definitions
end module module_name
```

Using a Module:

```
use module_name
```