# Fortran 90 Summary Sheet

R F Evans 2010

## General

| | |
|---|---|
| **PROGRAM NAME** | Program statement and name, only 1 allowed |
| **END PROGRAM NAME** | End of program statement |
| **IMPLICIT NONE** | Never, never, never omit this! Equivalent to "declare all possible variables that have not been declared" |
| **STOP** | Stop execution of the program |
| **! COMMENT** | Comment control character |
| **SUBROUTINE NAME(arg1,arg2…)** | Subroutine declaration |
| **END SUBROUTINE NAME** | End of subroutine |
| **INTEGER FUNCTION NAME(arg1…)** | Integer function declaration |
| **END FUNCTION NAME** | End of function |
| **SAVE** | Saves value of variable between successive calls |
| **INTENT(IN), INTENT(OUT), INTENT(INOUT)** | Optional variable options which specifies the intended use of arguments |

## Variables

| | |
|---|---|
| **INTEGER :: NAME** {= n} | Declares an integer variable (initialised to n) |
| **DOUBLE PRECISION :: NAME** {=n_dp} | Declares a double precision real variable |
| **REAL :: NAME** | Declares a low precision real variable |
| **INTEGER, PARAMETER :: NAME** = n | Initialisation option, read only variable |

## Operators

| | | | |
|---|---|---|---|
| **=** | Assignment operator | **+** | Addition operator |
| ***** | Multiplication operator | **-** | Subtraction operator |
| ****** | Exponentiation operator | **/** | Division operator |
| **==** | **.eq.** Equality | **/=** | **.ne.** Not equal |
| **<** | **.lt.** Less than | **>** | **.gt.** Greater than |
| **<=** | **.le.** Less than or equal to | **>=** | **.ge.** Greater than or equal to |
| **.not.** | Logical NOT | **.and.** | Logical AND |
| **.or** | Logical OR | **.eqv.** | Logical Equivalence |

## Program Control

| | |
|---|---|
| **DO LOOP_VAR = START, END {, INC}** | Specifies a DO loop, with variable, start and end indexes, and increment. Loop variable must be of type integer |
| **END DO** | End of do loop |
| **LOOP_NAME : DO** i=1,10 | Loop name – useful for outer loops |
| **END DO LOOP_NAME** | Named end do |
| **EXIT** | Break out of DO loop |
| **IF (CONDITION) THEN** | Executes following statements if condition is true |
| **END IF** | End of IF statement |
| **ELSE IF(CONDITION)** THEN | Nested IF statement |

## Common Intrinsic Functions

| | |
|---|---|
| **SIN(N), ASIN(N), SINH(N)** | Returns sine/ arc sine/ hyperbolic sine of N in radians |
| **COS(N), ACOS(N), COSH(N)** | Returns cosine/ arc cosine/ hyperbolic cosine of N in radians |
| **TAN(N), ATAN(N), TANH(N)** | Returns tangent/ arc tangent/ hyperbolic tangent of N in radians |
| **ATAN2(X,Y)** | Returns arctangent in range $-\pi:\pi$ depending on sign of Y |
| **SQRT(N)** | Returns square root of N |
| **EXP(N)** | Returns exponential of N |
| **LOG(N)** | Returns natural logarithm of N |
| **LOG10(N)** | Returns logarithm of N to base 10 |
| **SIGN(N)** | Returns sign of N |
| **DBLE(N)** | Converts N to double precision for assignment |
| **INT(N)** | Converts N to integer for assignment, rounding down |
| **NINT(N)** | Returns nearest integer to N |
| **MOD(N,I)** | Returns N modulo P |

## Arrays

| | |
|---|---|
| **INTEGER, DIMENSION(10) :: ARRAY** | Integer array of 10 numbers |
| **INTEGER :: ARRAY(0:9)** | Integer array of 10 numbers starting at 0 |
| **INTEGER :: ARRAY2D (1:3,1:5)** | 2D array with specified bounds |
| **INTEGER, ALLOCATABLE :: ARRAY(:)** | 1D allocatable integer array |
| **ALLOCATE(ARRAY(1:10), ALLOCSTAT)** | Allocation statement with error checking |
| **DEALLOCATE(ARRAY), DEALLOCSTAT)** | Deallocate array to free memory |
| **ARRAY(:) = 0** | Assign zero to all numbers in array |
| **ARRAY(1:5) = 5** | Assign "5" to elements 1-5 |
| **ARRAY2D(1,:)=ARRAY(6:10)** | Copy elements from one array to another |

## File Input and Output

| | |
|---|---|
| **OPEN(UNIT=15,FILE="file.txt")** | Basic opening of file at unit 15 |
| **OPEN(15,STATUS="OLD",FILE="old.txt")** | Opening of existing file (read only) |
| **OPEN(15,"NEW","new.txt",IOSTAT=ios)** | Opening of new file with error checking |
| **READ(UNIT=15,FMT=*,IOSTAT=ios) N** | Read N from file with error checking |
| **WRITE(UNIT=15,FMT="F12.5") N** | Write N to file as formatted text |
| **CLOSE(UNIT=15)** | Close unit 15 |
| IF(ios!=0)EXIT "Error in file read!" | Check for file read error |