# Syllabus for the 1 Day Fortran 90 Course

AC Marshall

September 2, 1997



**Abstract**

*The course is spread over four 50 minute lectures which should be scheduled for two sessions either side of lunch, each of these sessions should be further partitioned by a tea / coffee break which can also be used for a discussion of points raised in the lectures. The timetable does not include a practical session.*

*The following topics will be covered to varying degrees: data objects and declarations, expressions, control flow, arrays, intrinsics, procedures, modules, pointers, derived types, object-oriented programming, parameterised data types.*

*The package contains lecture slides and teachers notes annotating the corresponding overhead slide.*

## 1 Introduction

FORTRAN 77 was always the first choice for scientific and engineering software due, in the main, to two reasons:

- compiler availability;
- efficiency

Fortran 90 was designed with the latter point very much in mind (which explains some of the restrictions imposed in the standard). Since its definition and standardisation, it became clear that the functionality of FORTRAN 77 needed to be extended in order to keep in line with current software engineering practises and it was out of this need that Fortran 90 emerged.

## 2 Course Format

The course is comprised of the following:

- instructions for use,
- this syllabus,
- OHP originals (for photocopying onto slides),
- lecturers guide (notes for guidance),
- student course notes,
- resource list.

## 3 Syllabus

The syllabus and lecture content is given below.

## Session 1: Overview of Fortran 90

Topics covered:

- language evolution
  - standardisation process.
- design goals
- drawbacks of FORTRAN 77
  - fixed source form, absence of parallelism, lack of dynamic storage, non-portability, lack of user-defined data type, lack of explicit recursion, reliance on unsafe storage association features.
- Fortran 90 new features
  - free source form, array syntax, array intrinsics, dynamic storage and pointers, portable data types, derived data types and operators, recursion, object based programming (`MODULE`s), procedure interfaces, new control structures, user defined generic procedures, enhanced I/O.
- Fortran 90 source form
- attributed declarations
- overview of new control constructs
  - `IF` construct names, `SELECT CASE`, `DO` names, `END DO` terminators, `EXIT` and `CYCLE` statements.
- overview of procedures
  - Internal procedures, `INTENT` attributes, `OPTIONAL` and keyword arguments, recursion and `INTERFACE`s.
- short overview of array facilities
  - arrays as objects, elemental operations, sectioning, array valued intrinsics, masked assignment.
- overview of modules
  - object based programming: definitions of user types, declarations of constants, declaration of variables, accessibility statements, definition of procedures, declarations of generic procedure names and operator symbols, the `USE` statement, global storage without `COMMON`. Examples of object based programs.
- overview of user-defined entities
  - defined types: declarations, selecting components, constructors, examples. Overloading and definition of operators and assignment, generic interfaces (intrinsic and user defined), examples.
- overview of pointers
  - pointer attribute, target attribute, pointer association, dereferencing, aliasing.
- pointers and recursive data structures
  - linked lists, assignment between structures containing pointer components.
- overview of parameterised data types
  - `SELECTED_INT_KIND`, `SELECTED_REAL_KIND` intrinsics, range specification, portability, `KIND` function, constants of a selected integer kind, kinds and procedure arguments, kinds and generic interfaces.
- overview of new I/O features

- advantages of new Fortran 90 features
  - how unsafe FORTRAN 77 features map to new Fortran 90 syntax.

- obsolescent features of Fortran 90
  - philosophy and Fortran 90 equivalents, list of obsolescent features.

- undesirable features
  - things which are not yet obsolete but which should be avoided: fixed source form layout, implicit declaration of variables, `COMMON` blocks, assumed size arrays, storage and sequence association, `ENTRY` statements and the computed `GOTO` statement.

## Session 2: Arrays

Topics covered:

- ☐ basic array terminology
  - ◇ visualisation, rank, bounds, extent, size, shape and conformance, declarations, `DIMENSION` attribute, default lower bounds, zero-sized arrays, indexing.
- ☐ array conformance
  - ◇ scalar conformance, conformance between arrays, difference between size and shape.
- ☐ array element ordering
  - ◇ FORTRAN 77 storage association, conceptual column major form.
- ☐ array syntax
  - ◇ whole arrays, elements, sections.
- ☐ whole array expressions
  - ◇ operations between conformable arrays, the concept of parallel execution, elemental intrinsic function invocation, examples.
- ☐ array sections
  - ◇ specification and visualisation.
- ☐ array sections
  - ◇ subscript-triplets, default subscript-triplet values, examples of triplets: strided sections, zero-sized sections, reversed sections.
- ☐ array inquiry intrinsics
  - ◇ inquiry intrinsics in procedures, `LBOUND` and `UBOUND`, `SHAPE`, `SIZE` and `ALLOCATED`.
- ☐ vector-valued subscripts
  - ◇ restrictions, examples.
- ☐ array constructors
  - ◇ `(/ .. /)` constructors, implied `DO` statements, `RESHAPE`.
- ☐ `RESHAPE` intrinsic function
- ☐ deferred-shape arrays
  - ◇ heap storage, `ALLOCATABLE` attribute, `ALLOCATE` statement, `STAT=` specifier, reclaiming storage, `DEALLOCATE` statement, `ALLOCATED` function, memory leakage
- ☐ masked assignment
  - ◇ `WHERE` statement, masks, `WHERE` construct, execution process, example.
- ☐ dummy array arguments
  - ◇ explicit-shape arrays, assumed-shape arrays.
- ☐ assumed-shape arrays
  - ◇ recommended method, example, restrictions.
- ☐ automatic arrays
  - ◇ temporary local objects, declaration, restrictions.

- random number intrinsic
- vector and matrix multiply intrinsics
  - DOT_PRODUCT and MATMUL. Examples of use.
- array location intrinsics
  - MAXLOC and MINLOC intrinsics, examples.
- array reduction intrinsics
  - the concept of reduction functions, ALL, ANY, COUNT, MAXVAL, MINVAL, PRODUCT and SUM, the optional DIM argument.

## Session 3: Modules

Topics covered:

- □ overview of modules
  - ◇ global object declaration, interface declaration, procedure declaration, controlled object accessibility, packages of whole sets of facilities and semantic extension.
- □ modules — visualisation
- □ global data using modules
  - ◇ unsuitability of `COMMON`, static storage and scope.
- □ global data example
- □ procedure encapsulation
  - ◇ benefits and example.
- □ modules — object-based programming
  - ◇ introduction to program packaging and encapsulation.
- □ derived type constructors in modules
- □ user-defined generic interfaces
  - ◇ generic interface syntax, encapsulation — module procedures, restrictions, overload sets, resolution of invocation, example.
- □ extending generic intrinsic interfaces
- □ operator overloading
  - ◇ syntax, monadic and dyadic overloads and restrictions.
- □ example of operator overloading
- □ defining new operators
  - ◇ operator syntax, monadic and dyadic defined operators, syntax considerations and restrictions.
- □ example of defining new operators
- □ user defined operator precedence
- □ user-defined assignment
  - ◇ explicitly programmed user-defined assignment, overloading the = operator, assignment overload sets, syntax restrictions, redefinition of previous overloads.
- □ example of user-defined assignment
- □ data hiding
  - ◇ objects with `PRIVATE` and `PUBLIC` attributes, derived types with hidden internal structure, accessibility examples.
- □ `USE` statement
  - ◇ use-association, name clashes, use only clause, examples.
- □ semantic extension

## Session 4: Miscellaneous Features

Topics covered:

- parameterised data types
  - kind selection, numeric portability.
- integer kind selection
  - SELECTED_INT_KIND intrinsic function, specification of numeric range, default kinds, portability.
- constants of a selected integer kind
  - syntax, portability.
- real kind selection
  - SELECTED_REAL_KIND intrinsic, selecting precision and exponent range, portability, COMPLEX data types, DOUBLE PRECISION data types (use REAL).
- KIND function
- mixed kind expression evaluation
  - kind promotion, pitfalls, examples.
- kinds and procedure arguments
  - arguments correspondence, generic interfaces.
- logical kind selection
  - selecting different logical kinds, implications.
- characters of different kinds
  - default character set, selection of different character sets, restrictions, mixing character sets, specification of literals, intrinsic functions.
- mathematical intrinsic functions
  - ASIN, ACOS, ATAN, ATAN2, TAN, COS, SIN, TANH, COSH, SINH, EXP, LOG, LOG10, SQRT.
- numeric intrinsic functions
  - ABS, AINT, ANINT, CEILING, FLOOR, CMPLX, DBLE, DIM, INT, MAX, MIN, MOD, MODULO, REAL and SIGN.
- character intrinsic functions
  - ACHAR, ADJUSTL, ADJUSTR, CHAR, IACHAR, ICHAR, INDEX, LEN, LEN_TRIM, LGE, .., LLT, REPEAT, TRIM and VERIFY.
- bit variables and bit manipulation intrinsic functions
  - representation of bit variables. Intrinsics: BTEST, IAND, IBCLR, IBITS, IBSET, IEOR, IOR, ISHFT, ISHFTC, NOT and MVBITS. Examples of use.
- array construction intrinsics
  - MERGE, SPREAD, PACK and UNPACK.
- TRANSFER intrinsic
- Fortran 95
  - FORALL, nested WHERE, ELEMENTAL and PURE procedures, user-defined functions in initialisation expressions, automatic deallocation, deleted features, obsolescent features, language tidy-ups and ambiguities. Rationale for changes.

- ☐ overview of High Performance Fortran
  - ◇ distributed memory processing, philosophy. Syntax extensions: `FORALL`, new intrinsics, `PURE` and `ELEMENTAL` procedures, compiler directives: `PROCESSORS`, `ALIGN`, `TEMPLATE`, `DISTRIBUTE`, distribution methods, and examples.