From *A Brief Review of FORTRAN 77* by John H. Mathews
(to accompany "NUMERICAL METHODS: FORTRAN Programs", Prentice Hall, 1995).

General Remark:
Fortran does not distinguish between upper and lower case.

Specified Columns:
Each line of FORTRAN code must be written using the following column conventions.

Column	Contents
1	"C" or "*" for a comment line
1-5	Statement number or label (non-signed integer).
6	Blank or character for continuation line.
7-72	FORTRAN statement
73-80	Sequence number (usually omitted).

Form of a FORTRAN Subroutine

```
SUBROUTINE <subroutine-name>[<actual-argument-list>]
    {<specification-statements>}
    {<executable-statements>}
RETURN
END
```

Form of a Subroutine call

```
CALL <subroutine-name>[<actual-argument-list>]
```

Form of a FORTRAN Function

```
[<type>] FUNCTION <function-name> [(<dummy-assignment list>)]
    {<specification-statements>}
    {<executable-statements>}
RETURN
END
```

Specification Statements (or Type Declaration Statements)

```
INTEGER  I,J,Row
REAL  A0,B0,Max,X,Y
DOUBLE PRECISION
COMMON  A0,B0
COMMON  /BlockA/ A,B,X,Y
```

Variables
Variable names can be from one to six characters long. Variables beginning with I,J,K,L,M,N are presumed to be integers unless they are declared otherwise.

Assignment Statements

```
Y = A*X**2 + B*X + C
```

Arrays

```
DIMENSION A(1:50), M(1:10,1:10)
```

Arithmetic Operations

+	Addition		
-	Subtraction	/	Division
*	Multiplication	**	Exponentiation

Relational Operators

.EQ.	Equal to		
.NE.	Not equal to	.LE.	Less than or equal to
.LT.	Less than	.GE.	Greater than or equal to
.GT.	Greater than		

Logical Operators

.NOT. Complement
.AND. True if both operands are true
.OR. True if either (or both) operands are true

IF (Block) Control Statement

```
IF (<logical-expression-#1>) THEN
   {<executable-statements>}
ELSEIF (<logical-expression-#2>) THEN
   {<executable-statements>}
         #
ELSE
   {<executable-statements>}
ENDIF
```

DO (Block) Control Statement

```
DO  K = M1, M2, Mstep
   {<executable-statements>}
ENDDO
```

WHILE (Block) Control Statement

```
WHILE (<logical-expression>)
   {<executable-statements>}
REPEAT
```

Input and Output

```
READ  *,<input-variable-name-list>
PRINT  *
PRINT  *,<output-expression-list>
WRITE  (*,*)
WRITE  (*,*) <output-expression-list>
```

Pause Statement

```
PAUSE
```

Stop Statement

```
STOP
```

Mathematical Functions

COS(X)	cosine (radians)	ALOG(X)	natural logarithm base e
SIN(X)	sine (radians)	LOG10(X)	common logarithm base 10
TAN(X)	tangent (radians)	SQRT(X)	square root
EXP(X)	exponential exp(x)	ABS(X)	absolute value
COSH(X)	hyperbolic sine	INT(X)	conversion to integer
SINH(X)	hyperbolic cosine	FLOAT(I)	conversion to real number type
TANH(X)	hyperbolic tangent	REAL(I)	conversion to real number type
ACOS(X)	inverse cosine (radians)	DBLE(X)	conversion to double precision type
ASIN(X)	inverse sine (radians)	CMPLX(X)	conversion to double precision type
ATAN(X)	inverse tangent (radians)		

Put a “D” in front of the name for functions with input and output of DOUBLE type, i.e. use DCOS instead of COS