1 **FORTRAN 77**
Chapter 1
Satish Chandra

2 **What is Fortran?**
- Fortran is a general purpose programming language, mainly intended for mathematical computations e.g. in engineering, in science.
- Fortran is an acronym for FORmula TRANslation, and was originally capitalized as FORTRAN.
- However, following the current trend to only capitalize the first letter in acronyms, we will call it Fortran.
- Fortran was the first ever high-level programming languages.

3 **What is Fortran?**
- The work on Fortran started in the 1950's at IBM and there have been many versions since. By convention, a Fortran version is denoted by the last two digits of the year the standard was proposed. Thus we have
  Fortran 66
  Fortran 77
  Fortran 90 (95)
- The most common Fortran version today is still Fortran 77

4 **Fortran 77 Basics**
- A Fortran program is just a sequence of lines of text. The text has to follow a certain syntax to be a valid Fortran program. We start by looking at a simple example:

5 **An example of Fortran program**
```
c     program circle
      real r, area
c     This program reads a real number r and prints
c     the area of a circle with radius r.
      write (*,*) 'Give radius r:'
      read (*,*) r
      area = 3.14159*r*r
      write (*,*) 'Area = ', area
      stop
      end
```

6 **Fortran 77 Basics**
- The lines that begin with a "c" are comments and has no purpose other than to make the program more readable for humans.
- Originally, all Fortran programs had to be written in all upper-case letters. Most people now write lower-case since this is more legible, and so will we.

7 **Program Organization**
- A Fortran program generally consists of a main program (or driver) and possibly several subprograms (or procedures or subroutines).
- For now we will assume all the statements are in the main program; subprograms will be treated later.

8 **Program Organization**
A FORTRAN program can be divided into three sections:
Declarations - This section consists of a group of non-executable statements at the start of the program.
Execution - This section consists of one or more statements describing the actions to be performed by the program.
Termination - This section consists of a statement (or statements) telling the computer to stop/end running the program.

9 **Program Organization**
The structure of a main program is:
- program name
- declarations

- statements
- stop
- end

10 **Program Organization**

A program consists of a series of *statements* designed to accomplish the goal to be accomplished.
There are two basic types of statements:
Executable statements describe the actions taken by the program (additions, subtractions, multiplications, divisions).
Non-executable statements provide information necessary for proper operation of the program.

11 **Column position rules**

Fortran 77 is not a free-format language. The most important rules are the column position rules:

- Col. 1 : Blank, or a "c" or "*" for comments
- Col. 2-5 : Statement label (optional)
- Col. 6 : Continuation of previous line (optional)
- Col. 7-72 : Statements
- Col. 73-80: Sequence number (optional, rarely used today)

12 **Column position rules**

- Most lines in a Fortran 77 program starts with 6 blanks and ends before column 72, i.e. only the statement field is used.
- Note that Fortran 90 allows free format.
- A line that begins with the letter "c" or an asterisk (*) in the first column is a comment.
- Comments may appear anywhere in the program.
- Well-written comments are crucial to program readability.
- A label can be any number between 1 and 9999.

13 **Types of Data**

Three common types of data are stored in a computer's memory: character, integer, real
Each type has unique characteristics and takes up a different amount of memory.

14 **Character Data**

A typical system for representing character data may include the following:

- Letters: A through Z and a through z
- Digits (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
- Miscellaneous symbols, e.g. (", ', ?, ., <, >, =, %, &,$,+, –,*,**,;, Ъ)

15 **Character Data**

- In the past, it has been common to use just one byte of memory to represent a maximum of 256 characters (ASCII).
- To represent characters found in many languages, one can use 2 bytes of memory which allows 65,536 possible characters (Unicode).

16 **Constants**

- Constants are the quantities which never change during the execution of a program.
- There are two types of constants – Integer type and Real type.

17 **Integer constants:**

- It is a whole number written as a string of digits without a decimal point or a comma.
- It is also called fixed point type. Normally a minus sign is used for a negative constant
- eg : 456, 78, –89 etc.
- where as 32.12, 12+, 12,345 etc. are not integer type

18 **Integer constants:**

- Integer data (e.g. –1, –355, 0, 1993) are represented exactly on computers. However, only a finite number can be stored.
- Most machines will use 4 bytes of memory to represent integers.
- Smallest n-bit integer: $-2^{(n-1)}$
- Largest n-bit integer: $2^{(n-1)} - 1$
- e.g. n=32 for 4-byte numbers.

19 **Real constants :**

- The fractional numbers and decimal numbers are of this type.

- They are also known as floating point type. A real constant is a string of decimal digits.

20 **Real constants :**
- Fortran has two types of precision namely
- Single precision (It has maximum 8 decimal places), and
- Double precision which has 16 or even more decimal places.
- When a number is too large or too small, the numbers are put in the exponent form, i.e. for .00000032 it is written as 3.2E–7 or 0.32e–6

21 **Real constants :**
- A format similar to scientific notation will be used to represent real numbers in FORTRAN.
- Real numbers occupy 4 bytes of memory. These 4 bytes will be divided as follows:
- 3 byte for the fraction (or mantissa)
- 1 byte for the exponent
- The number of bits in the exponent allows a range of approximately 10^{-38} to 10^{38}

22 **Real constants :**
- A real constant in Fortran is one of the form +w.d, -w.d , + x.d E +n, -x.d E+n etc.
- Ex: 12.3, + 0.43, 2.3454E-3 etc.
- Where as 12, 12,345.0 , E5, 12.3E+.3 etc are not valid real constants.

23 **Double precision**
- For constants that are larger than the largest real allowed, or that requires high precision, double precision should be used. The notation is the same as for real constants except the "E" is replaced by a "D". Examples:

  2.0D-1
  1D99

24 **Logical Constant**
- These can only have one of two values:

  .TRUE.
  .FALSE.
- Note that the dots enclosing the letters are required.

25 **Characters Constant**
- A character constant is a string of characters enclosed in a single or double quotes.
- Any characters represened on a computer (not just the Fortran characters) are legal in a character context, if enclosed in quotes.
- e.g. 'this is', ' The area is =', '[^]', "3.1345"

26 **Variables**
- A variable is one whose value changes during the execution of a program.
- A variable is represented by alphanumeric characters and it is a string of maximum 6 characters which should begin with an alphabet.
- Ex: pay, money, index , text, beta ,pi , lab , alpha3 etc.

27 **Variables**
- The variables are also of two types- Real & Integer.
- By default a name begins with i,j,k,l,m and n represent an integer variable and others are real variable.
- Ex: pay, beta, pi etc are real variable where as money, index, joy , lab etc are integer variables.
- epsilon , pay. , &pi are not valid variables.

28 **Problems A:**
1. An integer constant : 7.59, 62.7, 10,-36, 64.539, 25.0, 3 x 10^3
2. A real constant : 4, 345.45 , E7, 6.93, 12, 2.3E+5
3. An integer variable: x, bell, joy, month
4. A real variable : wrong, boys, index, star, long

29 **Type declaration**
- Though certain variables are real or integer by default, they can be declared otherwise by using type declaration statements like REAL , INTEGER, LOGICAL , COMPLEX etc.
- For example 'money' is an integer variable by default , we can make it real by the statement REAL money.

- Similarly 'pay' is real and to make it integer use the statement INTEGER pay

30 **Type declaration**

Every variable should be defined in a declaration. This establishes the type of the variable. The most common declarations are:

integer *list of variables*
real *list of variables*
double precision *list of variables*
complex *list of variables*
logical *list of variables*
character *list of variables*

31 **Type conversion**

- When different data types occur in the same expression, type conversion has to take place, either explicitly or implicitly. Fortran will do some type conversion implicitly. For example,

```
real x
x = x + 1
```

- will convert the integer one to the real number one, and has the desired effect of incrementing x by one.

32 **Type conversion**

- However, in complicated expressions, it is good to force the type conversions explicitly. For numbers, the following functions are available:

```
int
real
dble
ichar
char
```

- The first three have the obvious meaning.
- ichar takes a character and converts it to an integer, while char does exactly the opposite.

33 **The parameter statement**

- Some constants appear many times in a program. It is then often desirable to define them only once, in the beginning of the program.
- This is what the parameter statement is for. It also makes programs more readable.
- For example, the circle area program should rather have been written like this:

34 **The parameter statement**

```
real r, area, pi
parameter (pi = 3.14159)
c This program reads a real number r and prints
c the area of a circle with radius r.
write (*,*) 'Give radius r:'
read (*,*) r
area = pi*r*r
write (*,*) 'Area = ', area
stop
end
```

35 **The parameter statement**

- The syntax of the parameter statement is, parameter (name = constant, ... , name = constant)
- The rules for the parameter statement are:
  1. The "variable" defined in the parameter statement is not a variable but rather a constant whose value can never change
  2. A "variable" can appear in at most one parameter statement
  3. The parameter statement(s) must come before the first executable statement

36 **Assignment Statement**

- The symbol '=' is used for assignment. X = 3 means the value 3 is assigned to the variable x. Care should be taken when a constant is assigned to a variable that both should be of the same type.

- For example, i=2.54 will assign only 2 to i, similarly j=1/0.3 assign only 0 to j

37 **Arithmetic Operators**

The operators are

Addition	+
Subtraction	-
Multiplication	*
Division	/
Exponentiation	**

38 **Arithmetic Operators**

- As there are two types of constants, arithmetic is also of two types – real and integer arithmetic.
- 4/3 yields 1 where as 4/3.0 yields 1.3333333. i.e. integer division truncates.

Ex. 1: The arithmetic expression for a+(b-c)/dxe is a+(b-c)/(d*e)

Ex. 2: The arithmetic statement for $\alpha = 2x^2/(x^2+y^2)^3$ is alpha =2.0*x**2/(x**2 +y**2)**3

39 **Arithmetic Operators**

Hierarchy of operations

1. Contents of "( )" are evaluated first, starting from the innermost and going outward.
2. Exponentials are evaluated, from right to left.
3. Multiplies/divides are evaluated, from left to right.
4. Adds/subtracts are evaluated, from left to right.

e.g. a=3.,b=2.,c=5.,d=4.,e=10.,f=2.,g=3.

output =a*b+c*d+e/(f**g**1)

output = … = 27.25

40 **Relational Operators**

Relational operator	Syntax	Meaning
.LT.	x .LT. y	x < y
.GT.	x .GT. y	x > y
.LE.	x .LE. y	x ≤ y
.GE.	x .GE. y	x ≥ y
.EQ.	x .EQ. y	x = y
.NE.	x .NE. y	x ≠ y

41 **Relational Operators**

- If two expressions are connected by a relational operator then it is called logical expression
- It is either true or false. i.e. the value of the expression is either .TRUE. or .FALSE.
- For example , suppose x=12 y=8 then (x .LT. y) is .FALSE.

42 **Logical Operators**

- Logic operators are operators with one or two logical operands that yield a logical result.
- Logical expressions can be combined by the logical operators..AND., .OR., .NOT., .EQV., .NEQV.
- The hierarchy of logical operators is .NOT., .AND., .OR., .EQV., .NEQV.
- The rule for order of precedence is that arithmetic expressions are evaluated first, then relational operators, and finally logical operators.

43 **Logical Operators**

- Example: Let a,b,c,d be real/integer variables.

(a<b) .OR. (c<d)

'(' , ')' can be omitted

a<b .OR. c<d

!But may confuse us; so use '(' , ')'

(a<b) .OR. (c<d)

For a = 2 , b = 3 , c = 5 , d = 3 the value of the logical expressions above is .TRUE.

44 **Logical Operators**

Let log1 = (a < b) and log2 = (c < d) , then

log1	log2	log1.OR.log2	log1.AND.log2	.NOT.log1
.TRUE.	*.TRUE.*	*.TRUE.*	*.TRUE.*	*.FALSE.*
.TRUE.	*.FALSE.*	*.TRUE.*	*.FALSE.*	*.FALSE.*
.FALSE.	*.TRUE.*	*.TRUE.*	*.FALSE.*	*.TRUE.*

.FALSE.	*.FALSE.*	*.FALSE.*	*.FALSE.*	*.TRUE.*

log1	log2	log1.EQV.log2	log1.NEQV.log2
.TRUE.	*.TRUE.*	*.TRUE.*	*.FALSE.*
.TRUE.	*.FALSE.*	*.FALSE.*	*.TRUE.*
.FALSE.	*.TRUE.*	*.FALSE.*	*.TRUE.*
.FALSE.	*.FALSE.*	*.TRUE.*	*.FALSE.*

45 **Problems B:**

1. If x=10.0 and y=15.0 then what is the value of
   i. x.GT.y .OR. x.LT.y
   ii. x.GT.y .AND. .NOT. x.LT.y
2. if a=4.0, b=12.5, ix=2, jx=5 then what is the value of
   i. x=a/b
   ii. k=jx/ix
   iii. n=ix+2/jx+a+b

46 **Functions**

- Fortran functions are quite similar to mathematical functions: They both take a set of input arguments (parameters) and return a value of some type.
- Fortran 77 also has some built-in functions. A simple example illustrates how to use a function:

  x = cos(pi/3.0)

  Here cos is the cosine function, angle is in radian, so x will be assigned the value 0.5

47 **Library Functions**

There are many built-in functions in Fortran 77.

abs	absolute value
min	minimum value
max	maximum value
mod	remainder
sqrt	square root
exp	exponential (natural)
log	logarithm (natural)
log10	logarithm base 10

48 **Library Functions**

sin	sine
cos	cosine
tan	tangent
asin	arcsine
acos	arccosine
atan	arctangent
sinh	hyperbolic sine
cos h	hyperbolic cosine
tanh	hyperbolic tangent

49 **Problems C:**

Find the Fortran equivalent of

1. $\cos(\log_{10}(3x+y))$
2. $\log_{10}(x^2+y^2)$
3. e^x+e^y
4. $\tan^{-1}\sqrt{(\sin^2|a|)}$

50 **ENDS**

Chapter 1