We will study FORTRAN with reference to standard of FORTRAN 77

Character set
Uppercase: A-Z
Lowercase: a-z
Digits: 0-9

Special Character:
Blank = Equal + Plus - Minus * Asterisk /Slash ()parenthesis
, Comma . Decimal point $ Currency symbol ' Apostrophe : Colon
! Exclamation point _ Underscore "Quotation mark

Escape Sequence

Escape Sequence	**Character**
\n	Newline
\t	Tab
\b	Backspace
\f	Form feed
\v	Vertical tab
\0	Null
\'	Apostrophe, which does not terminate a string
\"	Quotation mark, which does not terminate a string
\\	\ or slash
\x	*x*, where *x* is any other character

Data Types Constants and Variables

INTEGER
positive and negative integral numbers and zero, size 4 bytes
REAL
positive and negative numbers with a fractional part and zero, size 4 bytes
DOUBLE PRECISION
Same as REAL but using twice the storage space.

COMPLEX
Ordered pair of REAL data: real and imaginary components
LOGICAL
Boolean data representing true or false
CHARACTER
Character strings

Operators

1. **Arithmetic operators**

Priority	**Operation**	**Symbol**	**FORTRAN 77 Expression**	**Arithmetic Expression**
right to left	Exponentiation	**	A**B	a^b
left to right	Multiplication and Division	* /	A*B A/B	$a \times b$ $a \div b$
left to right	Addition, Subtraction, and Unary Minus	+ -	A+B A-B -A	$a+b$ $a-b$ $-a$

Note: Parentheses have the highest priority and can be used to force lower priority calculations to occur before higher priority ones.

The arithmetic expression $-a^n+b \times c-d \div e$ is written in FORTRAN 77 as -AN+B*C-D/E and is evaluated in the following order:**

i. A**N
ii. B*C
iii. D/E
iv. - in front of A**N
v. + between -A**N and B*C
vi. - between B*C and D/E

Logical Operators

Three operations can be performed on LOGICAL variables. The truth table below sums this up:

A	B	.NOT. A	A .AND. B	A .OR. B
.TRUE.	.TRUE.	.FALSE.	.TRUE.	.TRUE.
.TRUE.	.FALSE.	.FALSE.	.FALSE.	.TRUE.
.FALSE.	.TRUE.	.TRUE.	.FALSE.	.TRUE.
.FALSE.	.FALSE.	.TRUE.	.FALSE.	.FALSE.

Relational Operators

Operator	Meaning	Fortran Expression	Result assuming a=5 and b=3
.EQ.	equal to	a.EQ.b	0
.NE.	not equal to	a.NE.b	1
.LT.	less than	a.LT.b	0
.LE.	less than or equal to	a.LE.b	0
.GT.	greater than	a.GT.b	1
.GE.	greater than or equal to	a.GE.b	1

Assignment operators

Variable_name= expression;

y= x+5

String concatenation

Double backward slash '//' is used for string concatenation.

FORTRAN 77 program format

It not a free format language i.e. it has very strict set of rules for writing a program.

Structure of FORTRAN 77 program

Program name
Declarations
Statements
End

Column 1 → Blank, or The character **c** or ***** means that this line contains only comment.
Column 2-5 → Blank, or Statement Label.
Column 7-72→ Statements, all information after column 72 is ignored.
Column 6 → Only one statement per line is permitted, If a statement does not fit on one line, it can be continued on the next line(s). On position 6 of the next line(s) there has to be a **'+'** symbol.

1	2-5	6	7-72	
c			Hello world program in FORTRAN	
			program Hello	
	100		write(*,*) 'hello from console'	
		+	'this is for multiple statements'	
			end program Hello	

Program	**Output**
program hello write(*,*)'Hello World' end program hello	Hello World
Sum of two Numbers	
c sum of 2 numbers in FORTRAN program sum INTEGER a,b,sum write(*,*)'enter value of a and b' read(*,*)a,b sum=a+b write(*,*)'the sum is',sum end program sum	enter value of a and b 7 8 the sum is 15
Sum of two Complex numbers	
program complex_number complex a,b,sum write(*,*)'enter first complex number' read(*,*)a write(*,*)'enter second complex number' read(*,*)b sum=a+b; write(*,*)'the sum is',sum end program complex_number	enter first complex number (4,6) enter second complex number (8,6) the sum is (12.0000000 , 12.0000000)
Concatenate two strings	
program concat_string character*20 str1,str2,str write(*,*)'enter first string' read(*,*)str1 write(*,*)'enter second string' read(*,*)str2 write(*,*)'Concatenated string	enter first string Cristiano enter second string Ronaldo Concatenated string is:Cristiano Ronaldo

is:',str1//str2 end program concat_string	

Formatted and Unformatted IO operations

- read() and write() function can be used in conjunction with format specifier.
- The general syntax of read() and write() are as follows:

read(unit#,format#) list of arguments

write(unit#,format#) list of arguments

- ✓ **unit** is a number, which has an association with a particular device.
- ✓ In General **Unit=5** is reserved for **input from the keyboard** & **Unit=6** for **output to console.**
- ✓ If ***** is used in place of **unit# & format#,** default input/output device & default format is assumed.

Format specifications:

label FORMAT (format_specifications)

I Format

Used for formatting integer variable or constant

Syntax: Iw, where w is the width of the integer data

F Format

Used for formatting real variables.

Syntax: Fw.d, where w is the total width & d is precision of real data.

F4.2 → read 6743 as 67.43

F4.1 → read 6743 as 674.3

F4.0 → read 6743 as 6743

X Format:

Used for formatting character or string data

Syntax: nX, skips n columns.

E Format

Used for formatting real numbers in exponential form.

Syntax: **Ew.d**, where **w** is total field width including exponent and **d** is the decimal width or precision of the exponent.

E6.0 would read 1234E2 as $1234x10^2$ → 123400

E6.1 would read 1234E2 as $123.4x10^2$ → 12340

T Format:

Used for starting output from n^{th} column.

Syntax: nT, where n is the column number from which output starts.

Illustrate concept of Format specifiers	
program formatting integer x,y real r write(6,*)'enter any 2 decimal numbers' read(5,100)x,y write(6,*)'enter an real number' read(5,300)r 100 format(I4,I2) write(6,*)'data according to format 100:' write(6,100)x,y write(6,*)'data according to format 300:' write(6,300)r write(6,*)'data according to format 200:' write(6,200)x,y 200 format(I4,4x,I2) 300 format(F4.2) c **4x** leaves 4 space before printing value of y end program formatting	enter any 2 decimal numbers 200 2 123 data according to format 100: 200 2 data according to format 300: 1.23 data according to format 200: 200 2

Control structures

GOTO:

Used to jump from instruction from one location to other.

→Unconditional goto

Syntax:

goto label

→Computed goto

Syntax:

Goto (s_1,s_2,s_3,….,s_n) , integer expression

- when value of integer expression is equal to 1, control is transferred to statement number 1, if 2 then transferred to s2 and so on.

- If the value of integer expression is negative or greater than n then the goto statement is ignored.

Logical IF

Syntax:

IF(expression) THEN

Statement(s)

END IF

IF ELSE

IF(expression) THEN

True block of Statement(s)

ELSE

False block of Statement (s)

ENDIF

ELSE IF Ladder

IF (expression 1) THEN

Statement 1

ELSE IF (expression2) THEN

Statement 2

…….

ELSE IF (expression n) THEN

Statement n

ELSE

Default statement

END IF

Arithmetic IF

Syntax:

IF(expression) s1,s2,s3

First of all the value of expression is evaluated.

➔ If **Value of expression** is **–ve**, program branches to statement s1.

➔ If value is **Zero** branches to **s2**

➔ If value is **+ve** it branches to **s3**

DO LOOP

Syntax:

```
    do label, integer_varaible = initial_value, final_value, step_size
    block of statement(s)
label continue
```

Alternative syntax:

```
do integer_variable =initial_value, Final_value, Step_size
blocke of statement(s)
enddo
```

Prime or Composite in FORTRAN	

program prime
integer n,i
write(*,*)'enter a number'
read(*,*)n
do 200, i=2,n-1,1
if (mod(n,i).eq.0) THEN
write(*,*)'composite number'
goto 201
end if
200 continue
write(*,*)'prime number'
201 stop
end program prime
	enter a number
	23
	prime number
	
	enter a number
	22
	composite number

Array in FORTRAN

Declaration and Initialization of 1 D array and 2 D array

Syntax (1D):

data_type array_name (size)

integer a(50)

Fifty integers naming a(1), a(2),……..,a(50)

Syntax (2D):

data_type array_name (row_size, column_size)

integer a(3,3)

Nine integer elements declared as:

a(1,1), a(1,2), a(1,3),

a(2,1), a(2,2), a(2,3),

a(3,1), a(3,2), a(3,3)

→Initialization by using data statement

data variable list /value list/

Integer a(4)

data a(1),a(2),a(3),a(4) / 11,12,13,14/

→Initialization by using implied do loop

Syntax:

Array_name(counter_variable), counter_variable = initial_value, final_value, step_size

integer a(50)

read(*,*) a(i), i=1,50,1

Alternatively it can be done as:

do 100 i=1,50,1

read(*,*) a(i)

100 continue

Sorting numbers in ascending order	
program sorting	enter number of terms

```
      integer a(100),n,i,j,temp
      write(*,*)'enter number of terms'
      read(*,*)n
      write(*,*)'enter numbers'
      do 100 i=1,n,1
      read(*,*)a(i)
100    continue
      do 200, i=1,n,1
      do 300, j=i+1,n,1
         if (a(i).ge.a(j))then
            temp=a(i)
            a(i)=a(j)
            a(j)=temp
         endif
300   continue
200    continue
         write(*,*)'the entered number in   ascending order:'
          write(*,*)(a(i),i=1,n)
      end program sorting
```

```
5
 enter numbers
5
10
7
15
8
the entered number in ascending order:
5        7        8        10        15
```