Introduction to FORTRAN

1) Real and integer variables

Examples: 5 is an integer

5,132 is a real number (and it's also rational, but we don't care about that)

FORTRAN defaults:

A-H, O-Z Real variables

I-N Integer variables

Storage (on a 32-bit machine)

Binary code (0's & 1's)

Integer:

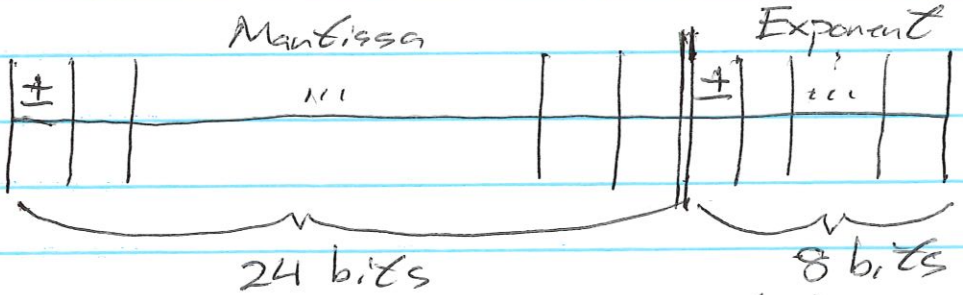
±	2^{30}			...	2^6	2^5	8	4	2	1
					1	0	1	0	0	1

Highest number: $\sim 2^{31} = 10^{31 \log 2}$

$\cong 10^{31(0.3)}$

$= 10^9$

Real



Highest exponent: $2^7 = 10^{7(0.3)} = 10^{2.1}$

$\cong 100$

Accuracy: 1 part in $2^{23} = 10^{23(0.3)}$

$\cong 10^7$

14

- Some computers have 64-bit chips $\Rightarrow$ 1 part in 10^{14}

Integer arithmetic

You have to be careful doing arithmetic with integer variables!

Examples:

Real variables

A = 1.

B = A/2.

Answer: B = 0,5

Integer variables

I = 1

J = I/2

Answer! J = 0

⇒ Integer arithmetic rounds down to the nearest integer, or whole number.

2) Character variables

Can also have variables that store characters, e.g.

N = 'NAME'

or N = 4HNAME (Hollerith format)

- Need 4 bits per character, so you can store names up to 8 characters in our single precision variable

3) The "IMPLICIT NONE" statement

If you include this statement at the beginning of your program, as your textbook advises you to do, then the FORTRAN defaults do not apply. Instead, each variable must be explicitly defined!

```
INTEGER :: I, J, K
REAL :: A, B, C
CHARACTER :: N
```

- I personally think this is a bad idea, as I see many more errors coming from undefined variables than I do from misused defaults. But, it's a matter of taste...

4) Format statements

The hardest thing about FORTRAN for many students is mastering format statements and doing input/output (IO).

Unformatted I/O:

```
READ *, X      (reads from keyboard)
PRINT *, X     (goes to screen)
```

Formatted I/O:

```
PRINT 100, X
100 FORMAT (E12.5)
```

Format types:

Real: E format Ex: E10.3 ±0.732E+04 (10 chars; 3 chars)

F format Ex: F5.2 27.15 (5 chars)

Integer: I format Ex: I3 137

Character: A format Ex: A4 NAME

5) Vectors and arrays

You can create vectors & matrices using the dimension statement (or the REAL & INTEGER statements)

```
DIMENSION(3) :: X, Y
```

or

```
REAL(3) :: X, Y
INTEGER(3) :: I, J
```

6) Data initialization

There are several ways to initialize a variable. The simplest is with an executable statement:

```
X = 3.
```

- Can also use a data statement

```
REAL :: X=3.
```

or, in F77:

```
DATA X/3./
```

- Finally, if you want to run the program with multiple inputs (as for Homework #1), you can read from the keyboard:

```
READ *, X
```

7) Reading from and writing to files

One can define input or output files as following:

```
OPEN (UNIT=1, file='filename')
```

Then, within the program:

```
READ (1, 100) X, Y, Z
```

or

```
WRITE (1, 100) X, Y, Z
100 FORMAT (3E10.3)        (for example)
```

- One can also protect input files from being overwritten by using:

```
OPEN (UNIT=1, file='filename', STATUS
    = 'OLD')
```

8) Now, skip to the numerical notes on Newton's method in one dimension

9) Outline of first program:

Step 1: Initialize X
Step 2: DO loop for Newton step

```
Ex:  DO I=1, 10
     [illegible]
     F = ...
     FP = ...
     X = ...
     PRINT *, X, F, FP
     END DO
```

Step 3: Add test for convergence. Want

$$\left|\frac{\Delta X}{X}\right| < 1.E-5$$

Can test this using an IF statement:

```
REL = ABS (DX/X)
IF(REL .< 1.E-5) EXIT
```

Step 4: Required end statements

```
STOP  (stops execution)
END   (end of program unit, for
       the compiler)
```