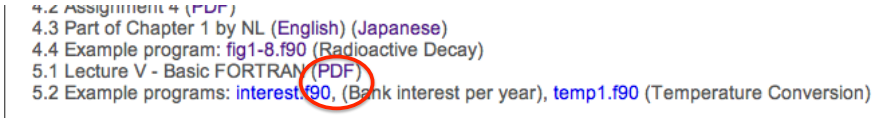
Computers in Chemistry – Lecture V

Prof. Dr. Stephan Irle
Quantum Chemistry Group
Nagoya University

Get this lecture online

- Please go to: http://qc.chem.nagoya-u.ac.jp
- Click on "Teaching"
- Click on "PPT" link of "5.1 Lecture V – Basic FORTRAN"

  userid: qcguest, password: qcigf!



Today's Lecture

- Basic elements in the FORTRAN programming language:
- Data Types, Constants, and Variables
- Operations and Functions
- The Assignment Statement
- Input/Output
- Program Composition and Format

2.1 Data Types, Constants, and Variables I

- FORTRAN provides five basic data types:

INTEGER	Numeric types
REAL	
COMPLEX	
CHARACTER	Strings of characters
LOGICAL	Two values: .FALSE. and .TRUE.

2.1 Data Types, Constants, and Variables II

- INTEGER: whole number (negative, zero, positive)

  Valid integers: 0; 137; -2516; 17745

  Invalid integers: 9,999; 16.0; --5; 7-
- REAL: ordinary decimal numbers or in exponential notation.

  Valid real constants: 1.234; -0.01536; +3.37E2

  Invalid real constants: 9,999; 63

2.1 Data Types, Constants, and Variables III

- CHARACTER: strings of characters; ANSI standard character set for FORTRAN: see Table 2-1
- CHARACTER variables have a predefined length.

  "PDQ123-A" has a length 8

  "John Q. Doe" has length 11, because blanks are counted as characters

  Apostrophe has to be included in double quotes (""), for example: "Don't"

2.1 Data Types, Constants, and Variables IV

- CHARACTER: strings of characters; ANSI standard character set for FORTRAN

TABLE 2-1 Fortran Character Set

CHARACTER	MEANING	CHARACTER	MEANING
0,...,9	digits	:	colon
A,...,Z	uppercase letters	=	equal sign
a,...,z	lowercase letters	!	exclamation mark
'	apostrophe (single quote)	&	ampersand
"	double quote	$	dollar sign
(	left parenthesis	;	semicolon
)	right parenthesis	<	less than
*	asterisk	>	greater than
+	plus sign	%	percent symbol
-	minus sign	?	question mark
/	slash	,	comman
blank	blank or space	.	period

2.1 Data Types, Constants, and Variables V

- Identifiers: names used to identify programs, constants, variables, and so on. *Must begin with a letter, which may be followed by up to 30 letters, digits, or underscores ("_").*

  Valid identifiers: Mass; Rate; Velocity; Speed_of_Light

  Invalid identifiers: R2-D2; 6Feet
- FORTRAN90 does not distinguish between upper case and lower case: "Velocity" is the same as "velocity" is the same as "vEloCiTy"

2.1 Data Types, Constants, and Variables VI

- Variables: Defined by “Type Statement”.
  INTEGER :: list of variables
  REAL :: list of variables
  CHARACTER(LEN=n) :: list of variables
  CHARACTER(n) :: list of variables
  default character length n=1
- IMPLICIT NONE should be used in every program and subroutine to avoid implicit naming convention: variables starting with the letters i,j,k,l,m,n are automatically declared of type INTEGER if not explicitly declared

2.1 Data Types, Constants, and Variables VII

- Variables initialization:
  REAL :: W = 1.0, X = 2.5, Y = 7.37, Z = -2.956
- PARAMETER attribute:
  INTEGER, PARAMETER :: Limit = 50
  REAL, PARAMETER :: Pi = 3.141593, TwoPi = 2.0 * Pi
  CHARACTER(2), PARAMETER :: Units = “cm”

2.2 Operations and Functions I

- Numeric Operations
- Addition: +
- Subtraction: –
- Multiplication: *
- Division: /
- Exponentiation: **
- Root extraction: **, e.g.: $7^{1/2}$ = 7.0 ** 0.5 = √7

Example: $B^2 - 4AC$ is written as:

B**2 – 4 * A * C

2.2 Operations and Functions II

- Important: when two constants or variables of the same type are combined using +-*/ the result has the same type as the “operands”.
- Example: 9/4 = 2, because 9 and 4 are integers and 2.25 is truncated to its integer value “2”
  Correct division in this case: 9.0 / 4.0 = 2.25
- Example: if N=2, X=2.0:
  1.0 / X = 0.5
  1 / N = 0

2.2 Operations and Functions III

- Mixed-mode expressions: combination of integer and real quantities in mathematical equations.
- BAD programming practice
- Examples:

  a) 1.0 / 4 → 1.0 / 4.0 = 0.25

  b) 3.0 + 8 /5 → 3.0 + 1 → 3.0 + 1.0 = 4.0

  c) 3 + 8.0 / 5 → 3 + 8.0 / 5.0 → 3 + 1.6 → 3.0 + 1.6 → 4.6

2.2 Operations and Functions IV

- Priority rules:

  1) all exponentiations are performed first; multiple exponentiations are performed from right to left; e.g. $5^{2^3} \neq$ 5.0 ** 2.0 ** 3.0 = 5^8

  2) all multiplication and divisions are performed next from the left to the right

  3) additions and substractions are performed last, from left to the right

2.2 Operations and Functions V

- Numeric functions:
- SQRT(argument): extracts the square root of the number in the *real-valued* argument.
  – Example: SQRT(7.0) = 7.0 ** (0.5) = 2.64575124

  If the argument is negative, an ERROR will occur.

  To calculate the square root of an integer, it is necessary to convert integer to real type:

  - Example: SQRT(7) = SQRT(REAL(7))

2.2 Operations and Functions VI

TABLE 2-2 Some Fortran Functions

FUNCTION	DESCRIPTION	TYPE OF ARGUMENT(S)*	TYPE OF VALUE
ABS(x)	Absolute value of x	Integer or real	Same as argument
COS(x)	Cosine of x radians	Real	Real
EXP(x)	Exponential function	Real	Real
INT(x)	Integer part of x	Real	Integer
FLOOR(x)	Greatest integer ≤ x	Real	Integer
FRACTION(x)	Fractional part (mantissa) of x	Real	Real
LOG(x)	Natural logarithm of x	Real	Real
MAX($x_1, \ldots, x_n$)	Maximum of $x_1, \ldots, x_n$	Integer or real	Same as arguments
MIN($x_1, \ldots, x_n$)	Minimum of $x_1, \ldots, x_n$	Integer or real	Same as arguments
MOD(x, y)	x (mod y): x - INT (x/y) * y	Integer or real	Same as arguments
INT(x)	x rounded to nearest integer	Real	Integer
REAL(x)	Conversion of x to real type	Integer	Real
SIN(x)	Sine of x radians	Real	Real
SQRT(x)	Square root of x	Real	Real
TAN(x)	Tangent of x radians	Real	Real

*In several cases, the arguments (and values) may be of extended-precision or complex types. See Chapter 9.

2.2 Operations and Functions VII

- Character Operations:
- Concatenation (to put two characters together): Operator "//"

  Example:

  a) "centi" // "meters" produces the string: "centimeters"

  b) if SquareUnits is a character variable with value "square ", then: SquareUnits // "centi" // "meters" = "square centimeters"

2.2 Operations and Functions VII

- Character Operations:
- Create substring of a string, can be extracted from a character variable or constant followed by the position of the first and last characters of the substring.
  - Example: if UNITS = "centimeters", then:
  - UNITS(4:7) = "time"

  Note: If initial position is not specified, it is assumed to be 1.

  If final position is not specified, it is assumed to be the last position in the value of the character variable

2.3 Assignment Statement

- Assignment statement:

  variable = expression
- Be careful with type conversion. Examples:
  - If an integer-valued expression is assigned to a real variable, the integer is converted to a real number
  - If a real-valued expression is assigned to an integer variable, the fractional part is truncated, and the integer part is assigned to the variable. Example: I = 2.718281828 → I = 2

  Mixed mode assignment is dangerous and should not be used!!

2.4 Input/Output I

- Two types of input/output statements: "list-directed" and "formatted"
- In this chapter we only use list-directed input/ output.
- The simplest list-directed output statement:
  - PRINT *, output-list

  Or

  - WRITE (*, *) output-list

  Example: PRINT *, "At time", Time, "seconds"

  where Time is a variable

2.4 Input/Output II

- To produce a blank line, PRINT and WRITE may be used without the output-list:

  Example: PRINT *

  or

  WRITE (*, *)

Note: the first "*" means the output to the screen, the second "*" means we do not use a fixed format (=free format)

2.4 Input/Output III

- The simplest list-directed input statement:
  - READ *, input-list

  Or

  - READ (*, *) input-list

  Example: READ *, Fahrenheit

  Care must be taken that the input data corresponds to the variable types. Otherwise, automatic type conversion will take place.

2.5 Program Composition and Format I

- **General form of a FORTRAN program** is
  - Heading, beginning with "PROGRAM name" statement
  - Specification part
  - Execution part
  - Subprogram part
  - "END PROGRAM" statement

2.5 Program Composition and Format II

- **PROGRAM name**
- "name" is a program identifier, should be a single string, can be 31 characters long
- Following the PROGRAM statement, it is best to write an **opening documentation**: what the program does, what are the variables used, what is input and what is output
- Next: **specification part**, beginning with:
- IMPLICIT NONE

2.5 Program Composition and Format III

- Then: declare the variables and their types
- COMPLEX :: list-of-complex-variables
- REAL :: list-of-real-variables
- INTEGER :: list-of-integer variables
- CHARACTER :: list-of-character-variables, including their lengths
- LOGICAL :: list-of-logical-variables

2.5 Program Composition and Format IV

- PROGRAM and declaration statements are *non-executable statements,* assignments, input/output etc. are *executable statements*
- Execution can be stopped by using the STOP statement
- The **program has to be terminated** by the END PROGRAM statement

2.5 Program Composition and Format V

- General composition in FORTRAN 90:
- Program lines may have up to 132 characters
- A line may contain more than one statement, provided that the statements are separated by ";" (it is good practice in FORTRAN NOT to do that)
- An ampersand "&" must be placed at the end of a line that will be continued in the next line, and another "&" must be placed at the beginning of the next line.
- Example: PRINT *, "Enter the initial" &
  &Velocity (m/sec):"

2.5 Program Composition and Format VI

- Any character following an exclamation mark "!" will be treated as a comment and ignored by the compiler
- If a statement requires a **statement label**, it must precede that statement (the statement must begin with this label).

  Example: GO TO 9999

  9999 STOP

2.6 Practice I

- Get interest.f90 from qc.chem.nagoya-u.ac.jp

```
PROGRAM Interest
!-------------------------------------------------------------------------
! This program calculates the interest on a bank deposit for one year,
! given the amount of the deposit and the interest rate.
! Variables used are:
!    Deposit          : bank deposit (Yen)
!    Interest_Rate    : bank interest rate (percent)
!    Earned_Interest  : how much interest the deposit can earn (Yen)
!
! Input : Deposit, Interest_Rate
! Output : Earned_Interest
!-------------------------------------------------------------------------
IMPLICIT NONE
REAL :: Deposit, Interest_Rate, Earned_Interest

! Get values for Deposit, Interest_Rate

PRINT *, "Enter the deposited amount (Yen) and the bank &
         &interest rate per year (percent):"
READ *, Deposit, Interest_Rate

! Compute the earned interest in one year
Earned_Interest = Deposit * (Interest_Rate / 100.0)

! Display earned interest
Print *, "Earned interest =", Earned_Interest, " Yen"

END PROGRAM Interest
```


2.6 Practice II

- Get temp1.f90 from qc.chem.nagoya-u.ac.jp

```
PROGRAM Temperature_Conversion_1
!-------------------------------------------------------------------------
! Program to convert a temperature on the Celsius scale to the
! corresponding temperature on the Fahrenheit scale.
! Variables used are:
!   Celsius:      : temperature on the Celsius scale
!   Fahrenheit:   : temperature on the Fahrenheit scale
!
! Input: Celsius
! Output: Fahrenheit
!-------------------------------------------------------------------------

  IMPLICIT NONE
  REAL :: Celsius, Fahrenheit

  ! Obtain Celsius temperature
  PRINT *, "Enter temperature in degree Celsius:"
  READ *, Celsius

  ! Calculate corresponding Fahrenheit temperature
  Fahrenheit = (9/5) * Celsius + 32.0

  ! Display temperature
  PRINT *, Celsius, "degrees Celsius = ", &
           Fahrenheit, "degrees Fahrenheit"

END PROGRAM Temperature_Conversion_1
```

What is wrong?

2.6 Practice III

- Write your own programs to practice the example expressions given today.
- Try to understand unit conversion in FORTRAN.

- This concludes today's lecture.