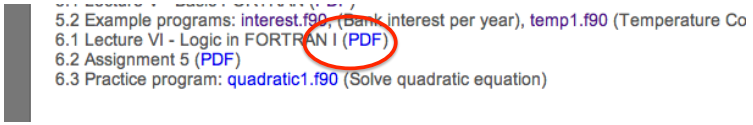
Computers in Chemistry – Lecture VI

Prof. Dr. Stephan Irle
Quantum Chemistry Group
Nagoya University

Get this lecture online

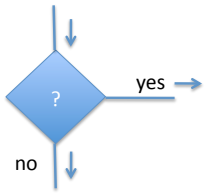
- Please go to: http://qc.chem.nagoya-u.ac.jp
- Click on "Teaching"
- Click on "PPT" link of "6.1 Lecture VI – Logic in FORTRAN I"

  userid: qcguest, password: qcigf!



Today's Lecture

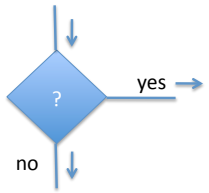
- Logical expressions: Making decisions



- Several FORTRAN statements used to repeat or select parts of the program involve logical expressions.
- Typical logical expressions involve <, >, or == (equal to) or /= (not equal to) operators

Computer logic vs. biological logic

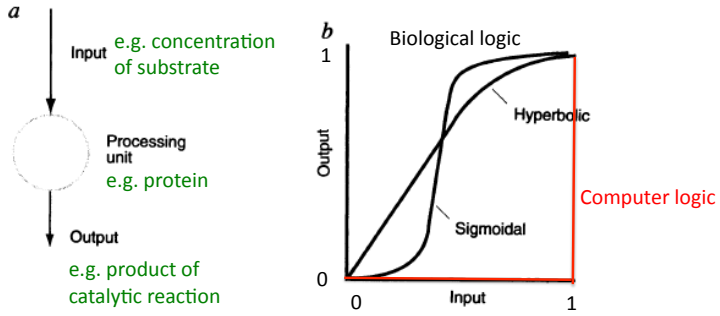
- Computer logic is binary: no (0) or yes (1)



- Logic operations in biomolecules are "fuzzy", i.e. can have continuous values in the interval [0,1], where "0" means "no action", "1" means "full scale action", and a value in between regulates the amount of "intermediate" action.

Cf: Dennis Bray, "Wetware – a computer in every living cell", Yale University Press, 2009

Computer logic vs. biological logic



Cf: Dennis Bray, "Wetware – a computer in every living cell", Yale University Press, 2009

3.1 Logical Expressions I

- Logical expressions may be simple or compound (contain at least two expressions).
- Simple logical expressions are:
  - Logical constants (.TRUE. or .FALSE.)
  - Logical variables (.TRUE. or .FALSE.)
  - Relational expressions of the form:

    expression relational-operator expression

    numeric or character (or logical) expressions

    relational-operator may be any of the following:

3.1 Logical Expressions II

SYMBOL	MEANING
< or .LT.	Is less than
> or .GT.	Is greater than
== or .EQ.	Is equal to
<= or .LE.	Is less than or equal to
>= or .GE.	Is greater than or equal to
/= or .NE.	Is not equal to

FORTRAN90 or FORTRAN77: both OK

- FORTRAN90: symbolic form; FORTRAN77: abbreviated form
- Only use one of them consistently in your code
- Note that "==" is different from "="

3.1 Logical Expressions III

- Examples for logical expressions:

  .TRUE.

  X < 5.2

  Number == -999

If X has the value 4.5, the logical expression "X < 5.2" has the value .TRUE.

If Number has the value 400, the expression "Number == -999" has the value .FALSE.

3.1 Logical Expressions IV

- If logical expression contains both arithmetic and relational operators, such as in:

  B ** 2 >= 4.0 * A * C

  The arithmetic operations are performed first; the above expression is equivalent to:

  (B ** 2) >= (4.0 * A * C)

  Example: A = 2.0, B = 1.0, C = 3.0, this is 1.0 >= 24.0, which is clearly .FALSE.

3.1 Logical Expressions V

- For character data, numeric codes are used to establish an ordering of the character set. Most common scheme is the ASCII code, which uses codes in the range from 0 to 255; for example:

A = 65	a = 97
B = 66	b = 98
…	…
Z = 90	z = 122

3.1 Logical Expressions VI

- Thus, the following are .TRUE. expressions:

  “A” < “F”

  “6” > “4”

- Two strings are compared character by character. Example:

  “cat” < “dog” is .TRUE. because “c” < “d”

  But

  “cat” > “cow” is .FALSE. because “a” < “o”.

3.1 Logical Expressions VI

- Compound logical expressions

  Performed by combining simple logical expressions by **logical operators:**

  .NOT. (negation)

  .AND. (conjunction)

  .OR. (disjunction)

  .EQV. (equivalence)

  .NEQV. (nonequivalence)

- Operators are defined by “truth tables”

3.1 Logical Expressions VII

- Truth tables: p and q are logical expressions, then:

p	.NOT. p
.TRUE.	.FALSE.
.FALSE.	.TRUE.

p	q	p .AND. q	p .OR. q	p .EQV. q	p .NEQV. q
.TRUE.	.TRUE.	.TRUE.	.TRUE.	.FALSE.	.TRUE.
.TRUE.	.FALSE.	.FALSE.	.TRUE.	.FALSE.	.TRUE.
.FALSE.	.TRUE.	.FALSE.	.TRUE.	.FALSE.	.TRUE.
.FALSE.	.FALSE.	.FALSE.	.FALSE.	.TRUE.	.FALSE.

3.1 Logical Expressions VIII

- If compound logical expressions contain arithmetic operators, relational operators, and logical operators, the operations are performed in the following order:
  1. Arithmetic operations (and functions)
  2. Relational operations
  3. Logical operators in the order .NOT., .AND., .OR., .EQV. (or .NEQV.)

3.1 Logical Expressions IX

- Example: Assume N=4
  - N**2 + 1 > 10 .AND. .NOT. N < 3

    Is equivalent to:

    (N**2 + 1 > 10) .AND. (.NOT. (N < 3))

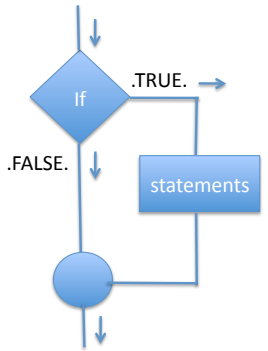
    The above is true: (.TRUE.).AND.(.TRUE.)
  - N == 3 .OR. N == 4

    The above is .TRUE. since N==4 is .TRUE.
  - N == 1 .OR. 2

    The above is .FALSE. since "2" is not a logical expression to which .OR. can be applied.

3.2 IF Constructs I

- Simple IF construct



- "If" can contain a simple or compound logical expression

3.2 IF Constructs II

- Simple IF Construct (also called “block IF construct) of the form:

```
IF (logical-expression) THEN
   statements
END IF
```

- If the logical expression is .TRUE., the statements are going to be executed, otherwise they will not be executed.

3.2 IF Constructs III

- Example:

```
IF (X >= 0) THEN
      Y = X * X
      Z = SQRT(X)
END IF
```

- More simple IF statement:

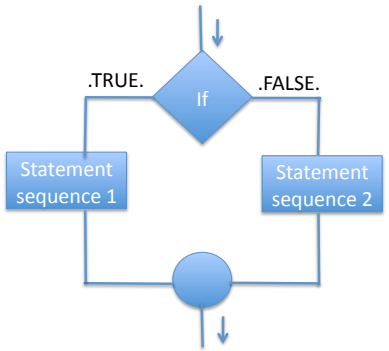
  IF (logical-expression) statement

  A single statement will be be executed if logical-expression is .TRUE.

  Example: IF (1.5 <= X .AND. X <= 2.5) PRINT *, X

3.2 IF Constructs IV

- General form of IF constructs



- Is realized by using the “ELSE” part of the IF construct

3.2 IF Constructs V

- Syntax:

```
IF (logical-expression) THEN
      statement sequence 1
ELSE
      statement sequence 2
END IF
```

3.2 IF Constructs VI

- Example: Quadratic expression
  $Ax^2 + Bx + C = 0$
  has two solutions:

Discriminant

$$x_1 = \frac{-B + \sqrt{B^2 - 4AC}}{2A} \qquad x_2 = \frac{-B - \sqrt{B^2 - 4AC}}{2A}$$

In FORTRAN notation:

X1 = (-B + SQRT(B**2 – 4.0*A*C))/(2.0*A)

X2 = (-B - SQRT(B**2 – 4.0*A*C))/(2.0*A)

Note that SQRT() gives an error if argument (=discriminant) is negative!

3.2 IF Constructs VII

- Algorithm:
- Task: Solve the quadratic equation $Ax^2 + Bx + C = 0$
- Input: A, B, C
- Output: The two real roots of the quadratic expression; if the solutions are complex, print a statement that there are no real roots
- Algorithm:
  1. Enter A, B, C
  2. Calculate discriminant = B**2 – 4.0*A*C
  3. Decide if discriminant is negative or positive: If positive, compute solutions, else display the discriminant and print a message that there are no real roots.

3.2 IF Constructs VII

- Algorithm:
- Task: Solve the quadratic equation $Ax^2 + Bx + C = 0$
- Input: A, B, C
- Output: The two real roots of the quadratic expression; if the solutions are complex, print a statement that there are no real roots
- Algorithm:
  1. Enter A, B, C
  2. Calculate discriminant = B**2 – 4.0*A*C
  3. Decide if discriminant is negative or positive: If positive, compute solutions, else display the discriminant and print a message that there are no real roots.

3.2 IF Constructs VIII

- Task: Write your own FORTRAN code without looking first at the solution, quadratic1.f90 on the webpage

3.3 IF-ELSE IF Constructs

- IF-ELSE constructs only considered selecting one of two alternatives.
- IF-ELSE IF constructs allow more than two alternatives:

```
IF (logical-expression1) THEN
      statement sequence 1
ELSE IF (local expression2) THEN
      statement sequence 2
ELSE IF (local expression3) THEN
      statement sequence 3
ELSE
     statement sequence4
END IF
```


3.4 CASE Constructs I

- Not as general as IF-ELSE IF constructs, but useful to implementing selection structures if the selection is based on the value of a single selector expression.
- Selector is an integer, character, or logical expression.

```
SELECT CASE (selector)
      CASE (label-list1)
            statement sequence 1
      CASE (label-list2)
            statement sequence 1
     CASE DEFAULT
            statement sequence 3
END SELECT
```


3.4 CASE Constructs II

- Example: ClassCode: integer variable, serves as selector.

```
SELECT CASE (ClassCode)
      CASE (1)
            PRINT *, "Freshman"     ! B1
      CASE (2)
            PRINT *, "Sophomore"    ! B2
     CASE (3)
            PRINT *, "Junior"       ! B3
      CASE (4)
            PRINT *, "Senior"       ! B4
      CASE DEFAULT
            PRINT *, "Illegal Class Selection"
END SELECT
```