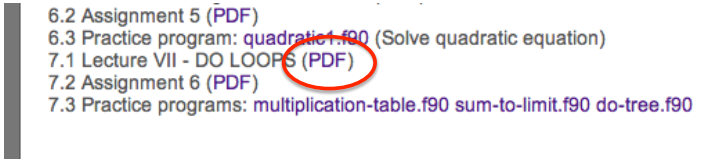
Computers in Chemistry – Lecture VII

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 "7.1 Lecture VII – DO LOOPS"

  userid: qcguest, password: qcigf!



Today's Lecture

- Repetitive execution (Do the same thing many times)
- Loop types
- Counter-Controlled DO Loops
- General DO Loops

4.1 Loop Types

- Previously, we learned a) sequence (regular program statements) and b) selection (IF statements)
- Now we learn the third basic control structure, namely c) repetition (repeat the same thing many times).
- A repetition structure or loop makes possible the repeated execution of one or more statements, called the "body of the loop"
- There are **two basic types of repetition:**
  - Repetition controlled **by a counter** (numerical variable)
  - Repetition controlled **by a logical expression**

4.2 Counter-Controlled DO Loops I

- FORTRAN 90 provides one basic loop construct, the DO construct.
- DO constructs for counter-controlled loops have the following form:

```
DO control-variable = initial-value, limit, step-size
   statement-sequence
END DO
```

- intial-value, limit, step-size are INTEGER expressions with step-size ≠ 0; step-size may be omitted in which case its default value is 1

4.2 Counter-Controlled DO Loops II

- **When the loop is executed, the following happens:**
  1. control-variable is assigned initial-value
  2. control-variable is compared to the limit to see if it is
     - less than or equal to the limit, for a positive step-size
     - greater than or equal to the limit, for a negative step-size
  3. If so, the statement sequence, called the "body of the loop", is executed
  4. The step-size is added to the control-variable, and step 2 is repeated. Otherwise, repetition is terminated.

4.2 Counter-Controlled DO Loops III

- If the termination test in step 2 is satisfied initially, the loop is never executed.
- Example:

```
DO Number = 1, 9
   PRINT *, Number, Number**2
END DO
```

The output of the loop is on the next page:

4.2 Counter-Controlled DO Loops IV

```
1  1
2  4
3  9
4  16
5  25
6  36
7  49
8  64
9  81
```

4.2 Counter-Controlled DO Loops V

- If step-size is negative, the control-variable is decremented (step-size is subtracted).
- Example:

```
DO Number = 9, 1, -1
   PRINT *, Number, Number**2
END DO
```

The output of the loop is on the next page:

4.2 Counter-Controlled DO Loops VI

```
9  81
8  64
7  49
6  36
5  25
4  16
3   9
2   4
1   1
```


4.2 Counter-Controlled DO Loops VII

- NOTE: initial-value, limit, step-size are determined BEFORE the DO loop and THE NUMBER OF REPETITIONS CANNOT BE CHANGED DURING THE DO LOOP, since it is calculated before the DO loop as:

  Number of repetitions =

  INT((limit – initial-value + step-size)/step-size)
- Statements in the body of the loop may use the control-variable, but CANNOT MODIFY ITS VALUE.

4.2 Counter-Controlled DO Loops VIII

- initial-value, limit, step-size are often variables. For example:

```
READ *, Number
DO I = 1, Number
    Sum = Sum + I
END DO
```

- Reads a value for variable Number, and the computes sum = 1 + 2 + … + Number

4.2 Counter-Controlled DO Loops IX

- The body of a DO loop may contain another DO loop. We call this construct a "**nested**" DO loop.
- Example:

```
DO M = 1, Last_M
    DO N = 1, Last_N
        Product = M * N
        Print *, M, " ", N, " ", Product
    END DO
END DO
```


4.2 Counter-Controlled DO Loops X

- Download the program multiplication-table.f90, compile, and execute
- Open X-terminal, cd "Downloads", and compile:

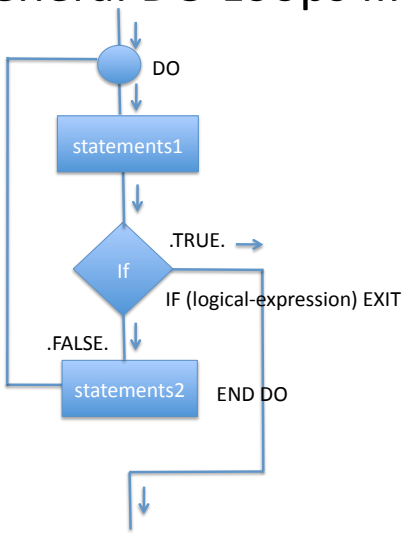
```
cd Downloads
pico multiplication-table.f90
gfortran –o multiplication-table.x multiplication-table.f90
./multiplication-table.x
```


4.3 General DO Loops I

- Counter-controlled DO loops execute the statements in the body of the loop for a pre-defined number of times.
- Sometimes we wish to determine the number of repetitions DURING the DO loop.
- For this, we need GENERAL DO loops (DO-EXIT constructs)

4.3 General DO Loops III

- Flow-chart:



4.3 General DO Loops IV

- FORTRAN 90 form of DO-EXIT construct:

```
DO
   statement-sequence-1
   IF (logical-expression) EXIT
   statement-sequence-2
END DO
```

- Example: consider previous example

  Sum = 1 + 2 + … + Number

- Question: For which smallest Number is Sum greater than Limit?

4.3 General DO Loops V

- To solve this problem, we must perform the following steps:
  1. Enter the value for Limit
  2. Initialize Number and Sum to 0
  3. Repeat the following:
     a. If Sum > Limit, terminate repetition; otherwise continue with the following
     b. Increment Number by 1
     c. Add Number to Sum
  4. Display Number and Sum

4.3 General DO Loops VI

- The following is an implementation of the solution to this problem:

```
DO
   IF (Sum > Limit) EXIT   ! Terminate repetition
   ! Otherwise continue with the following
   Number = Number + 1
   Sum = Sum + Number
END DO
```


4.3 General DO Loops VII

- Task: Write your own FORTRAN code without looking first at the solution, sum-to-limit.f90

Good luck.

DO-loop solution for assignment III

```
PROGRAM Do_Tree
!-------------------------------------------------------------------------
! Program to print the Christmas tree from Assignment III using a
! Counter-controlled DO loop.
! Variables:
!    Maxlength: maximum number of columns in a line
!    Line:       The line to be printed
!    IStar:      Number of Stars (counter)
!    IPos:       Position in line variable
!    MaxStar:    How many Stars is maximum (limit)
!    StarStep:   How many Stars more in each line
!      (width of the tree, step-size)
! Input: MaxStar
! Output: Christmas tree
!-------------------------------------------------------------------------

   IMPLICIT NONE
   INTEGER, PARAMETER :: Maxlength = 72
   INTEGER :: IStar, MaxStar, StarStep, IPos
   CHARACTER*1 :: Star="*"
   CHARACTER*(Maxlength) :: Line

   PRINT *, "How many stars more in each line:"
   READ *, StarStep

   PRINT *, "Maximum number of stars in a line:"
   READ *, MaxStar

   ! Check if MaxStar fits in the Line of length Maxlength
   IF (MaxStar > Maxlength) THEN
      PRINT *, "Maxstar", MaxStar, "is greater than", Maxlength
      STOP
   END IF
```


DO-loop solution for assignment III

```
   ! Initialize the line with blanks"
     DO IPos = 1, Maxlength
        Line(IPos:IPos) = " "
     END DO

     PRINT *,"*"
     DO IStar = 1, MaxStar-2, StarStep
        DO IPos = IStar, IStar+StarStep
           Line(IPos:IPos) = "*"
        END DO
        PRINT *, Line
     END DO

END PROGRAM DO_Tree
```

RUNNING THE PROGRAM:

```
[stephan@hawk trash]$ ./do-tree.x
 How many stars more in each line:
2
 Maximum number of stars in a line:
12
 *
 ***
 *****
 *******
 *********
 ***********
```