4. Loops
Control Constructs: Loops

- while loops
- iterative loops (counting loops)
While loop

```
do
    .......
    if (logical_expression) exit
    .......
end do
```

```
do while (logical_expression)
    .......
end do
```

WHILE LOOP could result in infinite looping.
Example:
Read a set of positive real numbers $x_i$ and calculate the mean $\bar{x}$ and standard deviation $s$ of the data.

Note that:
• Input of the data is ended by entering a negative number.
• At least three numbers are needed to calculate the statistics.

$$\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$$

$$s = \sqrt{\frac{N \sum_{i=1}^{N} x_i^2 - (\sum_{i=1}^{N} x_i)^2}{N(N-1)}}$$
program stats_2
! Calculate mean and standard deviation of input data set
! containing at least 3 input positive real values.

implicit none
integer :: n = 0 ! The number of input samples
real :: x = 0. ! An input data value
real :: sum_x = 0. ! The sum of the input values
real :: sum_x2 = 0. ! The sum of the squares of the input values
real :: x_bar ! The average of the input samples
real :: std_dev ! The standard deviation of the input samples

! While loop to read input values
do
  ! Read in next value
  write (*,*) 'Enter number:'
  read (*,*) x
  write (*,*) 'The number is ', x

  ! Test of loop exit
  if (x < 0.) exit
  ! Otherwise, accumulate sums
  n = n + 1
  sum_x = sum_x + x
  sum_x2 = sum_x2 + x**2
end do

>Continued on next page...
! Check to see if we have enough input data
if (n < 2) then
    write (*,*) 'At least 2 values must be entered!'
else
    x_bar = sum_x / real(n)
    std_dev = sqrt((real(n)*sum_x2-sum_x**2)/(real(n)*real(n-1)))

    write (*,*) 'The mean of this data set is: ', x_bar
    write (*,*) 'The standard deviation is: ', std_dev
    write (*,*) 'The number of data points is: ', n
end if

end program
Iterative loop (counting loop)

```plaintext
do index = istart, iend[, incr]
   ...
   statement block
   ...
end do
```

• `istart`, `iend`, `incr` can be constants, variables, or expression of integer.
• If `incr` is not specified, `incr = 1`.
• `incr` can be positive or negative.
do index = istart, iend [, incr]
  statement block
end do
Examples:

\begin{align*}
do \ i &= 1, 10 \\
\quad \ldots \\
\quad \text{end do} \\
\quad \quad i &= 1, 2, 3, \ldots, 10 \\

\begin{align*}
do \ i &= 1, 10, 2 \\
\quad \ldots \\
\quad \text{end do} \\
\quad \quad i &= 1, 3, 5, 7, 9 \\

\begin{align*}
do \ i &= 1, 10, -1 \\
\quad \ldots \\
\quad \text{end do} \\
\quad \quad \text{Initially, } 1 < 10, \text{ so exits}

\begin{align*}
do \ i &= 3, -3, -2 \\
\quad \ldots \\
\quad \text{end do} \\
\quad \quad i &= 3, 1, -1, -3
\end{align*}
\end{align*}
do index = istart, iend[, incr]
    statement block
end do

• **Number of iteration to be performed by the DO loop:**

\[
\text{iter} = \frac{\text{iend} - \text{istart} + \text{incr}}{\text{incr}}
\]

• **Logical condition to continue the loop:**

\[
\text{index} \times \text{incr} \leq \text{iend} \times \text{incr}
\]
Example:
Read $n$ real numbers $x_i$ and calculate the mean $\bar{x}$ and standard deviation $s$ of the data.

Note that:
• Input the number of data, $n$, then read the data.
• At least three numbers are needed to calculate the statistics.
• The numbers can be positive or negative.
program stats_3
! Calculate mean and standard deviation of input data set, 
! where each input value can be positive, negative or zero.

implicit none
integer :: i           ! Loop index 
integer :: n = 0       ! The number of input samples
real :: x = 0.         ! An input data value
real :: sum_x = 0.     ! The sum of the input values
real :: sum_x2 = 0.    ! The sum of the squares of the input values
real :: x_bar          ! The average of the input samples
real :: std_dev        ! The standard deviation of the input samples

! Get the number of data to input
write(*,*) 'Enter number of data: ' 
read(*,*) n

! Check to see if we have enough input data
if (n < 2) then
    write(*,*) 'At least 2 values must be entered.'
else

>Continued on next page...
! Loop to read input values
do i = 1, n
    write(*,*) 'Enter number: '
    read(*,*) x
    write(*,*) 'The number is ', x
    sum_x = sum_x + x
    sum_x2 = sum_x2 + x**2
end do

! Calculate statistics
x_bar = sum_x / real(n)
std_dev = sqrt((real(n)*sum_x2-sum_x**2)/(real(n)*real(n-1)))

write(*,*) 'The mean of this data set is: ', x_bar
write(*,*) 'The standard deviation is: ', std_dev
write(*,*) 'The number of data points is: ', n
end if
end program
Notes:

```plaintext
do i = 1, 4
    ...
    i = 2
end do

will produce infinite loops

do i = 1, 5
    ...
    if(i >= 3) exit
    ...
end do

branch out of a do loop is ok

do i = 1, 10, -1
    ...
    100 ...
    end do
    goto 100

but, branch into the loop is illegal
```
• CYCLE statement

```plaintext
do i = 1, 5
   if(i == 3) cycle
   write(*,*) i
end do
```

the output will be 1, 2, 4, 5

• EXIT statement

```plaintext
do i = 1, 5
   if(i == 3) exit
   write(*,*) i
end do
```

the output will be 1, 2
• Nesting loops (loop within loop)

```fortran
do i = 1, 5
  do j = 1, 4
    write(*,*) i, j
  end do
end do
```

Output:

```
1 1
1 2
1 3
1 4
2 1
2 2
...
```

• Named loops

```fortran
outer: do i = 1, 3
  inner: do j = 1, 4
    write(*,*) i, j
  end do inner
end do outer
```
The following loops all result in compilation errors:

```fortran
do i = 1, 3
   do j = 1, 4
      write(*,*) i, j
   end do
end do

outer: do i = 1, 3
   inner: do j = 1, 4
      write(*,*) i, j
   end do
end do outer

outer: do i = 1, 3
   inner: do j = 1, 4
      write(*,*) i, j
   end do outer
end do inner
```
Using CYCLE and EXIT in nested loops:

```fortran
program test_cycle_1
integer :: i, j, product
do i = 1, 2
  do j = 1, 3
    if (j==2) cycle
    product = i*j
    write(*,*)(i, ' * ', j, ' = ', product)
  end do
end do
end program test_cycle_1
```

Output:

```
1 * 1 = 1
2 * 1 = 2
```

```fortran
program test_cycle_2
integer :: i, j, product
do i = 1, 2
  do j = 1, 3
    if (j==2) exit
    product = i*j
    write(*,*)(i, ' * ', j, ' = ', product)
  end do
end do
end program test_cycle_2
```

Output:

```
1 * 1 = 1
2 * 1 = 2
```
program test_cycle_3
integer :: i, j, product

OUTER: do i = 1, 2
  INNER: do j = 1, 3
    if (j==2) cycle OUTER
    product = i*j
    write(*,*) i, ' * ', j, ' = ', product
  end do INNER
end do OUTER
end program test_cycle_2

Output:

1 * 1 = 1
2 * 1 = 2
Example: Flight of a ball

- Given a throwing velocity $v_0$ and angle $\theta$, find the flying range.
- What is the throwing angle which results in maximum flying range?
- Assume you have not learned calculus yet.
Coordinate of the flying trajectory:

\[ x(t) = v_{0x} t \]
\[ y(t) = v_{0y} t + \frac{1}{2} gt^2 \]

The ball will hit the ground when \( y(t) = 0 \):

\[ 0 = v_{0y} t + \frac{1}{2} gt^2 \]

The time interval of the flight:

\[ t_f = -\frac{2v_{0y}}{g} \]

The total range of flight:

\[ x(t_f) = x \left( -\frac{2v_{0y}}{g} \right) = v_{0x} \times \left( -\frac{2v_{0y}}{g} \right) = -\frac{2v_0^2}{g} \cos \theta \sin \theta \]
The total range of flight:

\[ x(t_f) = x\left(-\frac{2v_{oy}}{g}\right) = v_{ox} \times \left(-\frac{2v_{oy}}{g}\right) = -\frac{2v_0^2}{g} \cos \theta \sin \theta \]

The throwing angle which results in maximum range of flight occurs when:

\[
\frac{d}{d\theta} \left[x(t_f)\right] = 0
\]

\[
\Rightarrow \frac{d}{d\theta} \left[-\frac{2v_0^2}{g} \cos \theta \sin \theta \right] = -2 \frac{v_0^2}{g} \cos 2\theta = 0
\]

\[ \therefore \theta = \frac{\pi}{4} = 45^\circ \]

But since you have not learned calculus yet, you then decide to write a Fortran program for a given throwing velocity:

• to compute the flying range by varying the throwing angle, and then
• to find the throwing angle which results in the maximum range.
PROGRAM ball
! Calculate distance traveled by a ball thrown at a specified
! angle THETA and at a specified velocity VO from a point on the
! surface of the earth, ignoring the effects of air friction and
! the earth's curvature.

IMPLICIT NONE

! Declare constants
REAL, PARAMETER :: DEGREES_2_RAD = acos(-1.)/180. ! Deg => rad conv.
REAL, PARAMETER :: GRAVITY = -9.81 ! Gravity accel. (m/s)

! Declare variable types, definitions, & units
REAL :: v0 ! Throwing velocity (m/s)
INTEGER :: theta ! Throwing angle in degree
REAL :: radian ! Throwing angle in radian
REAL :: range ! Flying range (m)
REAL :: max_range = 0. ! Maximum flying range (m)
INTEGER :: max_degrees = 0 ! Angle of maximum flying range (degree)

! Input initial velocity
write(*,*) 'Initial velocity v0 (m/s) = '
read(*,*) v0

>Continued on next page...
! Loop over all specified angles.
angle: DO theta = 0, 90
  ! Convert angle into radians
  radian = real(theta) * DEGREES_2_RAD

  ! Calculate range in meters.
  range = (-2. * v0**2 / GRAVITY) * SIN(radian) * COS(radian)

  ! Write out the range for this angle.
  WRITE (*,*) 'Theta = ', theta, ' degrees; Range = ', range, ' meters'

  ! Compare the range to the previous maximum range. If this
  ! range is larger, save it and the angle at which it occurred.
  IF ( range > max_range ) THEN
    max_range = range
    max_degrees = theta
  END IF
END DO angle

! Output the maximum flying range and the angle at which it occurred.
WRITE (*,*) ' '
WRITE (*,*) 'Max range = ', max_range, ' at ', max_degrees, ' degrees'

END PROGRAM ball
Exercise 1

Write a Fortran program using WHILE loop to calculate the sum of the square of even integers between and including the range assigned by the user. Output the result on the screen.

Note that:

• The integers can be positive or negative.
• The first integer can be greater or less than the second integer.
Exercise 2

Write a Fortran program that will read an arbitrary number of real values and find the maximum and minimum of the input data.

Prompt the user for the number of data to be entered and use a DO loop (iterative loop) to read the input data.
Exercise 3

A trigonometric function can be represented by an infinite series. For example, the infinite series of \( \sin x \) is

\[
\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots = \sum_{n=1}^{\infty} (-1)^{n-1} \frac{x^{2n-1}}{(2n-1)!}
\]

where \( x \) is in unit of radian and \(-\infty < x < \infty\).

Since a computer cannot sum up the infinite number of terms in the series, the infinite series is \textit{truncated} after a finite number of terms. The truncated series, therefore, can only calculate the function to the precision of the floating point on the computer. The truncated infinite series for \( \sin x \) is

\[
\sin x = \sum_{n=1}^{N} (-1)^{n-1} \frac{x^{2n-1}}{(2n-1)!}
\]

where \( N \) is the number of terms to be retained in the series.

Write a Fortran program that:

• Reads in a value of \( x \) in degrees and then calculates the sine of \( x \) using the sine intrinsic function.
• Next calculate the sine of \( x \) using above truncated infinite series to the prescribed accuracy which is an input value.
• Output the values of the sine of \( x \) calculated using both intrinsic function and the truncated series, and the number of terms of the truncated series required.
• Be careful with how you evaluate and sum up the terms of the series.
• Be careful with the limited precision in the floating-point computation.
• Be careful to avoid infinite looping.
<table>
<thead>
<tr>
<th>$n$</th>
<th>$n!$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>24</td>
</tr>
<tr>
<td>5</td>
<td>120</td>
</tr>
<tr>
<td>6</td>
<td>720</td>
</tr>
<tr>
<td>7</td>
<td>5040</td>
</tr>
<tr>
<td>8</td>
<td>40320</td>
</tr>
<tr>
<td>9</td>
<td>362880</td>
</tr>
<tr>
<td>10</td>
<td>3628800</td>
</tr>
<tr>
<td>11</td>
<td>39916800</td>
</tr>
<tr>
<td>12</td>
<td>479001600</td>
</tr>
<tr>
<td>13</td>
<td>6227020800</td>
</tr>
<tr>
<td>14</td>
<td>87178291200</td>
</tr>
<tr>
<td>15</td>
<td>1307674368000</td>
</tr>
<tr>
<td>16</td>
<td>20922789888000</td>
</tr>
<tr>
<td>17</td>
<td>355687428096000</td>
</tr>
<tr>
<td>18</td>
<td>6402373705728000</td>
</tr>
<tr>
<td>19</td>
<td>121645100408832000</td>
</tr>
<tr>
<td>20</td>
<td>24329020081766400000</td>
</tr>
<tr>
<td>25</td>
<td>$1.551121004 \times 10^{25}$</td>
</tr>
<tr>
<td>50</td>
<td>$3.041409320 \times 10^{64}$</td>
</tr>
<tr>
<td>70</td>
<td>$1.197857167 \times 10^{100}$</td>
</tr>
<tr>
<td>100</td>
<td>$9.332621544 \times 10^{157}$</td>
</tr>
</tbody>
</table>

range in exponent: $10^{-38} \sim 10^{38}$