Computer Science 1510

Lecture 9

January 25, 2016

Lecture Outline

- Examples involving selection and repetition

Terminating input

- When we wish to input a list of numbers, or characters, we need to have some way to identify the end of the list.

  There are several ways to do this:

  – Allow a fixed number of inputs; say `N`.
    Input can then be done with a counted loop.
    The number `N` can itself be an input.
  – Designate a special *terminal value* which can be used to determine that the input list is complete.
    Input can then be done with a `DO WHILE` loop, or and `EXIT` statement when that terminal value is found.
  – Terminate input on a special condition, such as reading to the end of a file.

Program 1: Mean time to failure

```
PROGRAM Mean_Time_to_Failure
!------------------------------------------------------------------------
! Program to read a list of failure times, count them, and find the
! mean time to failure.  Values are read until an end-of-data flag
! is read.  Identifiers used are:
!  INPUT:
!    A list of failure times (FailureTime = current failure time read)
!  OUTPUT:
!    NumTimes        : the number of failure time readings
!    MeanFailureTime : the mean time to failure
!------------------------------------------------------------------------
  IMPLICIT NONE
  INTEGER :: NumTimes
  REAL :: FailureTime, Sum, MeanFailureTime
  REAL, PARAMETER :: EndDataFlag = -1.0
  Sum=0.0
  NumTimes=0
  WRITE(*,*) "Enter failure time of", EndDataFlag, "to stop."

  DO
     WRITE(*,*) "Enter failure time:"
     READ(*,*) FailureTime
     ! If end-of-data, terminate repetition
     IF (FailureTime == EndDataFlag) EXIT
     NumTimes = NumTimes + 1
     Sum = Sum + FailureTime
  END DO
  IF (NumTimes /= 0) THEN
     MeanFailureTime = Sum / NumTimes
     WRITE(*,*)
     WRITE(*,*) "Number of failure time readings:", NumTimes
     WRITE(*,*) "Mean time to failure:", MeanFailureTime
  ELSE
     WRITE(*,*) "No failure times were entered."
  END IF
END PROGRAM Mean_Time_to_Failure
```

Example 2: Approximating sin(x) and cos(x)

- Both the sine and cosine functions are represented by infinite series:

$$\sin(x) = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$$

$$\cos(x) = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n}}{(2n)!} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots$$

- Thus, we can approximate $\sin(x)$ or $\cos(x)$ for some x by computing some finite number of terms in the corresponding sum, call this number of terms n.

- The following program approximates the value of $\sin(x)$ using the above sum.

Program 2: Approximating sine

```
PROGRAM Approx_sine
   IMPLICIT NONE
   INTEGER::i,n
   REAL::x,term,approx
   WRITE(*,*) 'Enter a value for x'
   READ(*,*) x
   WRITE(*,*) 'How many terms would you like to include?'
   READ(*,*) n
   term=x
   approx=x
   DO i=1,n-1
       term=(-1)*term*(x**2/(2*i*(2*i+1)))
       approx=approx+term
   END DO
   WRITE(*,*) 'sin(',x,') is approximately',approx
END PROGRAM Approx_sine
```

Program 3: Approximating sine/cosine

```
PROGRAM Sin_cos
   IMPLICIT NONE
   INTEGER::i,n
   REAL::x,term,approx
   CHARACTER::func
   WRITE(*,*) 'Enter s for sin, and c for cos or q to quit'
   READ(*,*) func
   DO WHILE(func/='q')
      WRITE(*,*) 'Enter a value for x'
      READ(*,*) x
      WRITE(*,*) 'How many terms would you like to include?'
      READ(*,*) n
      SELECT CASE (func)
         CASE('s')
            term=x
            approx=x
            DO i=1,n-1
               term=(-1)*term*(x**2/(2*i*(2*i+1)))
               approx=approx+term
            END DO
            WRITE(*,*) 'sin(',x,') is approximately',approx
         CASE('c')
            term=1
            approx=1
            DO i=1,n-1
               term=(-1)*term*(x**2/(2*i*(2*i-1)))
               approx=approx+term
            END DO
            WRITE(*,*) 'cos(',x,') is approximately',approx
         CASE DEFAULT
            WRITE(*,*) 'Invalid entry, try again'
      END SELECT
      WRITE(*,*) 'Enter s for sin, and c for cos or q to quit'
      READ(*,*) func
   END DO
END PROGRAM Sin_cos
```

Example 3: Euclid's algorithm

- Suppose that you have two integers m and n, and you want to find their greatest common divisor (GCD).

- According to Euclid's algorithm we do the following. Taking $m = 1976$ and $n = 1032$,

$$\begin{aligned}
1976 &= 1032 \times 1 + 944 \\
1032 &= 944 \times 1 + 88 \\
944 &= 88 \times 10 + 64 \\
88 &= 64 \times 1 + 24 \\
64 &= 24 \times 2 + 16 \\
24 &= 16 \times 1 + 8 \\
16 &= 8 \times 2 + 0
\end{aligned}$$

Then the GCD is 8 (the final divisor).

Euclid's algorithm - Why it works

- Suppose that we want to find the GCD of a and b.
- We know that $a = sd$ and $b = td$ for some integers s and t, where d is the GCD.
- If we divide a by b we obtain a quotient q and a remainder r such that $a = qb + r$.

  $$\Longrightarrow r = a - qb = sd - qtd = (s - qt)d$$

  Thus d is also a divisor of r.
- Therefore, the GCD of a and b is also the GCD of b and r.
- Since $r < b$ we will eventually obtain $r = 0$.

Program 4: Euclid’s algorithm

```
PROGRAM Euclid
   IMPLICIT NONE
   INTEGER::m,n,q,r
   WRITE(*,*) 'Enter two integers m and n, where m>n'
   READ(*,*) m,n
   r=1
   DO WHILE (r/=0)
      q=m/n
      r=m-q*n
      WRITE(*,*) m,' = ',n,' * ',q,' + ',r
      m=n
      n=r
   END DO
   WRITE(*,*) 'The greatest common divisor is ',m
END PROGRAM Euclid
```