

Computers in Engineering  
COMP 208

DO WHILE  
Michael A. Hawker

Sept. 27th, 2007 Do While 1

---

---

---

---

---

---

---

---

DO-WHILE

- DO ... WHILE loops are a special case used when a condition is to be tested at the top of a loop
- This is a looping structure provided in many different programming languages
- Syntax:  
`DO WHILE (logical expression)  
statement block, s  
END DO`

Sept. 27th, 2007 Do While 2

---

---

---

---

---

---

---

---

DO-WHILE

- Semantics:
  - Test the logical expression
  - If it evaluates to `.TRUE.`, execute the statement block and go back to step 1.
  - If it evaluates to `.FALSE.`, go to the statement after the `END DO`

Sept. 27th, 2007 Do While 3

---

---

---

---

---

---

---

---

## DO-WHILE

DO-WHILE loops are equivalent to

```

DO
  IF .NOT.(logical expression) EXIT
  statement block s
END DO

```

Sept. 27th, 2007 Do While 4

---

---

---

---

---

---

---

---

## Example

The DO loop of the program to compute exp(x) can be rewritten using a DO-WHILE

```

DO
  IF (ABS(Term) < Tolerance) EXIT
  Sum = Sum + Term
  Count = Count + 1
  Term = Term * (X / Count)
END DO

DO WHILE (ABS(Term) >= Tolerance)
  Sum = Sum + Term
  Count = Count + 1
  Term = Term * (X / Count)
END DO

```

Sept. 27th, 2007 Do While 5

---

---

---

---

---

---

---

---

## Warning!

- The loop only executes if the logical expression evaluates to .TRUE.
- If the value of this expression doesn't change, we will get an infinite loop
- The values of variables that the logical expression depends on must be modified within the loop
- (It still might not terminate, but at least we have a chance)

Sept. 27th, 2007 Do While 6

---

---

---

---

---

---

---

---

## Nested DO-Loops

- A DO-loop can contain other DO-loops in its body.
- This nested DO-loop, must be completely inside the containing DO-loop.
- Note that an EXIT statement transfers control out of the inner-most DO-loop that contains the EXIT statement.

Sept. 27th, 2007 Do While 7

---

---

---

---

---

---

---

---

---

---

## Nested DO-Loop Example

The outer loop has i going from 1 to 7 with step size 1. For each of the seven values of i, the inner loop iterates 9 times with j going from 1 to 9.

```

INTEGER :: i, j
DO i = 1, 7
  DO j = 1, 9
    WRITE(*,*) i*j
  END DO
END DO
    
```

There are 63 values printed in total

Sept. 27th, 2007 Do While 8

---

---

---

---

---

---

---

---

---

---

## Table of Exp(x) (preamble)

```

! -----
! This program computes exp(x) for a range of values of x using the
! Infinite Series expansion of exp(x)
! The range has a beginning value, final value and step size.
! -----
PROGRAM Exponential
IMPLICIT NONE
INTEGER      :: Count
REAL         :: Term
REAL         :: Sum
REAL         :: X
REAL         :: ExpX
REAL         :: Begin, End, Step
REAL         :: Tolerance = 0.00001

WRITE(*,*) "Initial, Final and Step please --> "
READ(*,*)  Begin, End, Step
    
```

Sept. 27th, 2007 Do While 9

---

---

---

---

---

---

---

---

---

---

### Table of Exp(x) (body)

```

X = Begin           ! X starts with the beginning value
DO
  IF (X > End) EXIT ! if X is > the final value, EXIT
  Count = 1
  Sum = 1.0
  Term = X
  ExpX = EXP(X)     ! the exp(x) from Fortran's EXP()
DO
  IF (ABS(Term) < Tolerance) EXIT
  Sum = Sum + Term
  Count = Count + 1
  Term = Term * (X / Count)
END DO

WRITE(*,*) X, Sum, ExpX, ABS(Sum-ExpX), ABS((Sum-ExpX)/ExpX)

X = X + Step
END DO
END PROGRAM Exponential

```

Sept. 27th, 2007 Do While 10

---

---

---

---

---

---

---

---

---

---

### GCD Revisited

- A more efficient way of computing the GCD of two integers is possible
- It doesn't even use division!!

Sept. 27th, 2007 Do While 11

---

---

---

---

---

---

---

---

---

---

### Some GCD Facts

- The trivial cases:
  - $\text{gcd}(k, k) = k$ , for nonzero  $k$
  - $\text{gcd}(0, k) = \text{gcd}(k, 0) = k$ , for nonzero  $k$
- The general case:
  - For  $i \geq j$ ,  $\text{gcd}(i, j) = \text{gcd}(i-j, j)$
- Using this, we can work backwards from the general case by reducing the larger of the two arguments until we reach one of the trivial cases

Sept. 27th, 2007 Do While 12

---

---

---

---

---

---

---

---

---

---

### A GCD Program

```
INTEGER :: I, J, G
DO WHILE (I /= 0 .and. J /= 0 .and. I /= J)
  IF (I>J) THEN
    I = I - J
  ELSE
    J = J - I
  END IF
END DO
IF (I == 0) THEN
  G = J
ELSE
  G = I
END IF
```

Sept. 27th, 2007 Do While 13

---

---

---

---

---

---

---

---

### Verifying ISBN Numbers

```
program isbn
  implicit none
  integer :: digits(10)
  integer :: pos, sum
  logical :: valid
  READ (*,"(10I1)") (digits(pos), pos = 1,10)
  sum = 0
  do pos = 1,10
    sum = sum + (11-pos)*digits(pos)
  end do
  valid = mod(sum,11) == 0
  if (valid) then
    write(*,*) "ISBN is valid"
  else
    write(*,*) "ISBN is invalid"
  end if
end program isbn
```

Sept. 27th, 2007 Do While 14

---

---

---

---

---

---

---

---