

Computers in Engineering  
COMP 208

Selection  
Michael A. Hawker

---

---

---

---

---

---

---

---

Selection

- As we have seen:
  - Every programming language must provide a selection mechanism that allows us to control whether or not a statement should be executed
  - This will depend on whether or not some condition is satisfied (such as the discriminant being positive)

Sept. 18th, 2007 Selection 2/41

---

---

---

---

---

---

---

---

FORTRAN Selection

- Used to select an alternative sequence of statements
- The keywords separate the block statements
- Has Additional Forms to Provide More Control

Sept. 18th, 2007 Selection 3/41

---

---

---

---

---

---

---

---

## IF-THEN-END IF

### • Syntax:

```
IF (logical expression) THEN
  block of statements, s1
END IF
```

### • Semantics:

- Evaluate the logical expression
- If it evaluates to .TRUE. execute s<sub>1</sub> and then continue with the statement following the END IF
- If the result is .FALSE. skip s<sub>1</sub> and continue with the statement following the END IF

Sept. 18th, 2007

Selection

4/41

---

---

---

---

---

---

---

---

## Examples of IF-THEN-END IF

```
absolute_x = x
```

```
IF (x < 0.0) THEN
  absolute_x = -x
END IF
```

```
WRITE(*,*) "The absolute value of ", x, &
  " is ", absolute_x
```

Sept. 18th, 2007

Selection

5/41

---

---

---

---

---

---

---

---

## Examples of IF-THEN-END IF

```
INTEGER :: a, b, min
READ(*,*) a, b
min = a
```

```
IF (a > b) THEN
  min = b
END IF
```

```
WRITE(*,*) "The smaller of ", &
  a, " and ", b, " is ", min
```

Sept. 18th, 2007

Selection

6/41

---

---

---

---

---

---

---

---

## Logical IF

- An even simpler form is sometimes useful.

- **Syntax:**

```
IF (logical expression) single-statement
```

- **Semantics:**

- This statement is equivalent to

```
IF (logical expression) THEN  
  single-statement  
END IF
```

- The single-statement cannot be an IF or we might end up with an ambiguous statement

Sept. 18th, 2007

Selection

7/41

---

---

---

---

---

---

---

---

## Examples of Logical IF

```
absolute_x = x
```

```
IF (x < 0.0) absolute_x = -x
```

```
WRITE(*,*) "The absolute value of ", x, &  
" is" ,"absolute_x
```

Sept. 18th, 2007

Selection

8/41

---

---

---

---

---

---

---

---

## Examples of Logical IF

```
INTEGER :: a, b, min
```

```
READ(*,*) a, b
```

```
min = a
```

```
IF (a > b) min = b
```

```
WRITE(*,*) "The smaller of ", &  
a, " and ", b, " is ", min
```

Sept. 18th, 2007

Selection

9/41

---

---

---

---

---

---

---

---

## What's Going On?

- What is a “logical expression” ?
- Where do the values `.TRUE.` and `.FALSE.` come from?
- What are those periods around the words true and false?

Sept. 18th, 2007

Selection

10/41

---

---

---

---

---

---

---

---

## Logical Data Type

- FORTRAN has a LOGICAL data type, just like it has INTEGER and REAL types
- Each type has its associated values
- There are only two values in the type LOGICAL, `.TRUE.` and `.FALSE.`
- To enable the compiler to distinguish these values from variables, we represent them with periods around the words

Sept. 18th, 2007

Selection

11/41

---

---

---

---

---

---

---

---

## Logical Data Type

- We can declare variables of type LOGICAL

```
LOGICAL :: positive_x, condition
```
- We can assign values to them

```
condition = .TRUE.
positive_x = x > 0
```
- These variables can only take on one of the two values of type logical

Sept. 18th, 2007

Selection

12/41

---

---

---

---

---

---

---

---

## Logical Expressions

- Logical expressions, such as those that appear in IF statements, return a logical value
- That is, they are expressions which evaluate to `.TRUE.` or `.FALSE.`
- We have operators that return logical values.

Sept. 18th, 2007

Selection

13/41

---

---

---

---

---

---

---

---

## Relational Operators

- Relational operators compare two values and return the result `.TRUE.` or `.FALSE.`  
`<, <=, >, >=, ==, /=`
- Relational operators are of lower precedence than all arithmetic operators  
`2 + 7 >= 3 * 3 → .TRUE.`
- There is no associativity  
`a < b < c → illegal`

Sept. 18th, 2007

Selection

14/41

---

---

---

---

---

---

---

---

## == or = ?

- Note that `==` is the FORTRAN (and C) syntax for a relational operator meaning “is equal to”
- The expression `x == y` has the value `.TRUE.` if x and y are equal and `.FALSE.` if x and y are not equal
- A single equal sign (`=`) is the FORTRAN (and C) syntax for assignment
- The statement `x = y` means assign the value of y to the variable x

Sept. 18th, 2007

Selection

15/41

---

---

---

---

---

---

---

---

## == or = ?

- ✱ In FORTRAN you will get an error message if you use either operator incorrectly
- ✱ When we study C, we will see a program can still work but give incorrect results if you confuse these operators

Sept. 18th, 2007

Selection

16/41

---

---

---

---

---

---

---

---

## The Missing ELSE

- ✱ There is another more complex selection mechanism we can use
- ✱ The IF-THEN-ELSE-END IF form allows us to choose between two alternatives
- ✱ It allows us to choose whether or not to perform a one set of actions or another
- ✱ We either perform one action or another before we continue

Sept. 18th, 2007

Selection

17/41

---

---

---

---

---

---

---

---

```
! -----  
! Solve Ax^2 + Bx + C = 0  
! -----  
PROGRAM QuadraticEquation  
  IMPLICIT NONE  
  ! **** Same old declarations and set up ****  
  ! compute the square root of discriminant d  
  
  d = b*b - 4.0*a*c  
  IF (d >= 0.0) THEN  
    ! is it solvable?  
    d = SQRT(d)  
    root1 = (-b + d)/(2.0*a)  
    root2 = (-b - d)/(2.0*a)  
    WRITE(*,*) "Roots are ", root1, " and ", root2  
  ELSE  
    ! complex roots  
    WRITE(*,*) "There is no real root!"  
    WRITE(*,*) "Discriminant = ", d  
  END IF  
END PROGRAM QuadraticEquation
```

Sept. 18th, 2007

Selection

18/41

---

---

---

---

---

---

---

---

## IF-THEN-ELSE-END IF

### • Syntax:

```
IF (logical expression) THEN
  block of statements, s1
ELSE
  block of statements, s2
END IF
```

### • Semantics:

- Evaluate the logical expression
- If it evaluates to .TRUE. execute s<sub>1</sub> and then continue with the statement following the END IF
- If it evaluates to .FALSE. execute s<sub>2</sub> and continue with the statement following the END IF

Sept. 18th, 2007

Selection

19/41

---

---

---

---

---

---

---

---

---

---

## Is a Number Even or Odd?

```
IF (MOD(number, 2) == 0) THEN
  WRITE(*,*) number, " is even"
ELSE
  WRITE(*,*) number, " is odd"
END IF
```

Sept. 18th, 2007

Selection

20/41

---

---

---

---

---

---

---

---

---

---

## Is A Number Even or Odd? (alternate)

```
IF (number/2*2 == number) THEN
  WRITE(*,*) number, " is even"
ELSE
  WRITE(*,*) number, " is odd"
END IF
```

Sept. 18th, 2007

Selection

21/41

---

---

---

---

---

---

---

---

---

---

## Find Absolute Value

```
REAL :: x, absolute_x
x = ...
IF (x >= 0.0) THEN
  absolute_x = x
ELSE
  absolute_x = -x
END IF
WRITE(*,*) "The absolute value of ", &
  x, " is ", absolute_x
```

Sept. 18th, 2007

Selection

22/41

---

---

---

---

---

---

---

---

## Which value is smaller?

```
INTEGER :: a, b, min
READ(*,*) a, b
IF (a <= b) THEN
  min = a
ELSE
  min = b
END IF
WRITE(*,*) "The smaller of ", a, &
  " and ", b, " is ", min
```

Sept. 18th, 2007

Selection

23/41

---

---

---

---

---

---

---

---

## Quadratic Roots Revisited

- The problem of finding the roots of a quadratic is a bit more complicated than we have been assuming
- If the discriminant is zero there is only a single root

Sept. 18th, 2007

Selection

24/41

---

---

---

---

---

---

---

---



```

! -----
! Solve Ax^2 + Bx + C = 0
! Detect complex roots and repeated roots.
! -----
PROGRAM QuadraticEquation
IMPLICIT NONE
! **** same old declarations and setup statements omitted ****
d = b*b - 4.0*a*c

IF (d > 0.0) THEN           ! distinct roots?
  d = SQRT(d)
  root1 = (-b + d)/(2.0*a)  ! first root
  root2 = (-b - d)/(2.0*a)  ! second root
  WRITE(*,*) 'Roots are ', root1, ' and ', root2
ELSE
  IF (d == 0.0) THEN        ! repeated roots?
    WRITE(*,*) 'The repeated root is ', -b/(2.0*a)
  ELSE                       ! complex roots
    WRITE(*,*) 'There is no real root!'
    WRITE(*,*) 'Discriminant = ', d
  END IF
END IF
END PROGRAM QuadraticEquation

```

Sept. 18th, 2007                      Selection                      25/41

---

---

---

---

---

---

---

---

---

---

### IF-THEN-ELSE IF-END IF

- The nested IF statements in the last example are a bit complicated
- When we use IF to select between several (not just two) alternatives, we end up with more than a single END IF, one for each of the branches
- Let's simplify this

Sept. 18th, 2007                      Selection                      26/41

---

---

---

---

---

---

---

---

---

---

### Syntax of IF-THEN-ELSE IF-END IF

```

IF (logical-exp, e1) THEN
  statement block, s1
ELSE IF (logical-exp, e2) THEN
  statement block, s2
ELSE IF (logical-exp, e3) THEN
  statement block, s3
. . . . .
ELSE
  statement block, se
END IF

```

Sept. 18th, 2007                      Selection                      27/41

---

---

---

---

---

---

---

---

---

---

## Semantics of IF-THEN-ELSE IF-END IF

- Evaluate  $e_1$
- If the result is `.TRUE.`, execute  $s_1$  and go on to the statement that follows the `END IF`
- If the result is `.FALSE.`, evaluate  $e_2$ . If it is `.TRUE.`, execute  $s_2$  and go on to the statement that follows the `END IF`
- If the result of  $e_2$  is false, repeat this process.
- If none of the expressions  $e_i$  evaluate to `.TRUE.`, execute  $s_e$  and then go on to the statement that follows the `END IF`

Sept. 18th, 2007

Selection

28/41

---

---

---

---

---

---

---

---

```
! -----  
! Solve Ax^2 + Bx + C = 0  
! Detect complex roots and repeated roots.  
! -----  
PROGRAM QuadraticEquation  
  IMPLICIT NONE  
  ! **** same old declarations and setup statements omitted ****  
  
  d = b*b - 4.0*a*c  
  
  IF (d > 0.0) THEN                ! distinct roots?  
    d = SQRT(d)  
    root1 = (-b + d)/(2.0*a)        ! first root  
    root2 = (-b - d)/(2.0*a)        ! second root  
    WRITE(*,*) 'Roots are ', root1, ' and ', root2  
  ELSE IF (d == 0.0) THEN          ! repeated roots?  
    WRITE(*,*) 'The repeated root is ', -b/(2.0*a)  
  ELSE                              ! complex roots  
    WRITE(*,*) 'There is no real root!'  
    WRITE(*,*) 'Discriminant = ', d  
  END IF  
END PROGRAM QuadraticEquation
```

Sept. 18th, 2007

Selection

29/41

---

---

---

---

---

---

---

---

## Quadratic Roots Final Version

- The problem of finding the roots of a quadratic has some more complications
- What if  $a$  is zero. Dividing by  $2.0*a$  would cause an error.
- If  $a$  is zero, the equation is linear, not quadratic
- If  $a$  and  $b$  are zero but  $c$  isn't there is no solution

Sept. 18th, 2007

Selection

30/41

---

---

---

---

---

---

---

---

```

! -----
! Solve Ax^2 + Bx + C = 0
! Now, we are able to detect the following:
! (1) unsolvable equation
! (2) linear equation
! (3) quadratic equation
!     (a) distinct real roots
!     (b) repeated root
!     (c) no real roots
! -----

PROGRAM QuadraticEquation
IMPLICIT NONE

REAL :: a, b, c
REAL :: d
REAL :: root1, root2

! read in the coefficients a, b and c

READ(*,*) a, b, c

```

---

---

---

---

---

---

---

---

---

---

---

---

```

IF (a == 0.0) THEN ! could be a linear equation
IF (b == 0.0) THEN ! the input becomes c = 0
IF (c == 0.0) THEN ! all numbers are roots
WRITE(*,*) 'All numbers are roots'
ELSE ! Unsolvable
WRITE(*,*) 'Unsolvable equation'
END IF
ELSE ! linear equation
WRITE(*,*) 'This is linear equation, root = ', -c/b
END IF
ELSE ! ok, we have a quadratic equation
d = b*b - 4.0*a*c
IF (d > 0.0) THEN ! distinct root
d = SQRT(d)
root1 = (-b + d)/(2.0*a) ! first root
root2 = (-b - d)/(2.0*a) ! second root
WRITE(*,*) 'Roots are ', root1, ' and ', root2
ELSE IF (d == 0.0) THEN ! repeated roots?
WRITE(*,*) 'The repeated root is ', -b/(2.0*a)
ELSE ! complex roots
WRITE(*,*) 'There is no real root!'
WRITE(*,*) 'Discriminant = ', d
END IF
END IF
END PROGRAM QuadraticEquation

```

---

---

---

---

---

---

---

---

---

---

---

---

## What Day is Tomorrow?

- Here is a new problem to solve.
  - Given today's date (day,month,year)
  - Compute and output tomorrow's date
- What's the problem?
- If the date is the last day of the month, we have to update the day and month
- If it is the last day of the year, we also have to update the year

---

---

---

---

---

---

---

---

---

---

---

---

## First Validate the Data

```
PROGRAM nextday
IMPLICIT NONE
INTEGER :: day, month, year
INTEGER :: lastday
WRITE (*,*) "Please enter the date, day month and year:"
READ (*,*) day, month, year

! validate month

IF (month < 1 .OR. month > 12) THEN
  WRITE (*,*) "Invalid month"
  STOP
END IF

! Validation of year and day omitted to save space
```

Sept. 18th, 2007

Selection

34/41

---

---

---

---

---

---

---

---

## Compute the last day of the month

```
IF (month == 1 .OR. month == 3 .OR. month == 5 .OR. &
    month == 7 .OR. month == 8 .OR. month == 10 .OR. &
    month == 12) THEN
  lastday = 31
ELSE IF (month == 4 .OR. month == 6 .OR. month == 9 .OR. &
    month == 12) then
  lastday = 30
ELSE IF ((mod(year,4) == 0 .AND. mod(year,100) /= 0) .OR. &
    mod(year,400) == 0) THEN
  lastday = 29
ELSE
  lastday = 28
END IF
```

Sept. 18th, 2007

Selection

35/41

---

---

---

---

---

---

---

---

## Compute Tomorrow's Date

```
! The usual case
day = day + 1

! Handling the end of the month or year

IF (day > lastday) THEN
  day = 1
  month = month + 1
  IF (month > 12) THEN
    month = 1
    year = year + 1
  END IF
END IF

WRITE (*,*) day, month, year

END PROGRAM nextday
```

Sept. 18th, 2007

Selection

36/41

---

---

---

---

---

---

---

---

## Logical Operators

More complex logical expressions can be formed using logical operators

The Logical Operators listed in order of decreasing precedence are:

.NOT.  
.AND. (or &&)  
.OR. (or ||)  
.EQV. (or ==), .NEQV. (or /=)

The precedence of all logical operators is lower than all relational operators

They all associate from left to right

Sept. 18th, 2007

Selection

37/41

---

---

---

---

---

---

---

---

## Area of a Triangle

Heron's formula gives the area of a triangle in terms of the lengths of its sides.

$$\text{area} = \sqrt{s(s-a)(s-b)(s-c)}$$

Where a, b, and c are the lengths of the sides and

$$s = \frac{a+b+c}{2}$$

Sept. 18th, 2007

Selection

38/41

---

---

---

---

---

---

---

---

## Area of a Triangle

• To use it, we must make sure that the sides form a triangle.

• There are two necessary and sufficient conditions:

- All side lengths must be positive
- The sum of any two sides must be greater than the third

Sept. 18th, 2007

Selection

39/41

---

---

---

---

---

---

---

---

## Area of a Triangle (program preamble)

```
!-----  
!  
! Compute the area of a triangle using Heron's formula  
!-----  
!  
PROGRAM HeronFormula  
  IMPLICIT NONE  
  
  REAL    :: a, b, c           ! three sides  
  REAL    :: s                 ! half of perimeter  
  REAL    :: Area  
  LOGICAL :: Cond_1, Cond_2  
  READ(*,*) a, b, c
```

Sept. 18th, 2007

Selection

40/41

---

---

---

---

---

---

---

---

## Area of a Triangle (main body of program)

```
Cond_1 = (a > 0.) .AND. (b > 0.) .AND. (c > 0.0)  
Cond_2 = (a+b > c) .AND. (a+c > b) .AND. (b+c > a)  
IF (Cond_1 .AND. Cond_2) THEN  
  s = (a + b + c) / 2.0  
  Area = SQRT(s*(s-a)*(s-b)*(s-c))  
  WRITE(*,*) "Triangle area = ", Area  
ELSE  
  WRITE(*,*) "ERROR: this is not a triangle!"  
END IF  
  
END PROGRAM HeronFormula
```

Sept. 18th, 2007

Selection

41/41

---

---

---

---

---

---

---

---