Logical Type and Its Constant

- Logical values are *true* and *false*. In FORTRAN, they must be written as `.TRUE.` and `.FALSE.`
- Logical identifiers are declared with `LOGICAL`:

```
LOGICAL  :: Answer, Condition, Test
LOGICAL  :: Value, Yes_and_No
```

- Use `PARAMETER` to assign an identifier a logical value:

```
LOGICAL, PARAMETER :: Ans = .TRUE., Cond = .FALSE.
```

- Logical variables can also be initialized:

```
LOGICAL  :: Test = .TRUE., PreTest = .FALSE.
```

- Use assignment to store logical variables:

```
LOGICAL  :: Cond_1, Cond_2, Total

Cond_1 = .TRUE.
Cond_2 = .TRUE.
Total  = .FALSE.
```

- `WRITE(*,*)` will display a `T` for `.TRUE.` and a `F` for `.FALSE.`.
- For input, use `T` for `.TRUE.` and `F` for `.FALSE.`

Producing Logical Values – I
Relational Operators

Type	*Operators*	*Associativity*
	**	right-to-left
Arithmetic	* /	left-to-right
	+ -	left-to-right
Relational	< <= > >= == /=	none

- B**2 - 4.0*A*C .GE. 0.0
- A**2 + B**2 .EQ. C**2
- TOTAL /= SUM
- 'DOG' < 'FOX'
- 'JAN' <= 'JANUARY'
- A < B < C — **WRONG!!!!**

If `REAL` variables `x` and `y` have values 3.0 and 7.0, and `INTEGER` variables `p` and `q` have values 6 and 2, what is the result of

`x*x - y*y + 2.0*x*y /= p*q + p**3 - q**3`?

```
x*x - y*y + 2.0*x*y /= p*q + p**3 - q**3
 --> 3.0*3.0 - 7.0 * 7.0 + 2.0*3.0*7.0 /= 6*2+6**3 - 2**3
 --> 9.0 - 7.0*7.0 + 2.0*3.0*7.0 /= 6*2 + 6**3 - 2**3
 --> 9.0 - 49.0 + 2.0*3.0*7.0 /= 6*2 + 6**3 - 2**3
 --> -40.0 + 2.0*3.0*7.0 /= 6*2 + 6**3 - 2**3
 --> -40.0 + 6.0*7.0 /= 6*2 + 6**3 - 2**3
 --> -40.0 + 42.0 /= 6*2 + 6**3 - 2**3
 --> 2.0 /= 6*2 + 6**3 - 2**3
 --> 2.0 /= 12 + 6**3 - 2**3
 --> 2.0 /= 12 + 216 - 2**3
 --> 2.0 /= 228 - 2**3
 --> 2.0 /= 228 - 8
 --> 2.0 /= 220
 --> 2.0 /= 220.0
 --> .TRUE.
```

Producing Logical Values – II
Logical Operators

Type	*Operators*	*Associativity*
Arithmetic	`**`	right-to-left
	`*   /`	left-to-right
	`+   -`	left-to-right
Relational	`< <= > >= == /=`	none
Logical	`.NOT.`	right-to-left
	`.AND.`	left-to-right
	`.OR.`	left-to-right
	`.EQV.   .NEQV.`	left-to-right

- If one operand is `.TRUE.`, then `.OR.` is `.TRUE.`
- If one operand is `.FALSE.`, then `.AND.` is `.FALSE.`
- If both operands have the same value, then `.EQV.` is `.TRUE.`
- If both operands <u>**do not**</u> have the same truth values, then `.NEQV.` is `.TRUE.`

Examples

1. Let `LOGICAL` variables `Something` and `Another` have values `.TRUE.` and `.FALSE.`, respectively.

```
.NOT.  Something .AND. Another
     --> .NOT.  .TRUE.  .AND.  .FALSE.
     --> .FALSE.  .AND.  .FALSE.
     --> .FALSE.
```

2. Here is the same expression with parenthesis.

```
.NOT.  (Something .AND. Another)
     --> .NOT.  (.TRUE.  .AND.  .FALSE.)
     --> .NOT.  .FALSE.
     --> .TRUE.
```

3. Let `LOGICAL` variables `a`, `b` and `c` have values `.TRUE.`, `.TRUE.` and `.FALSE.`, respectively.

```
.NOT.  a .OR. .NOT. b .AND. c
     --> .NOT.  .TRUE. .OR. .NOT. .TRUE. .AND. .FALSE.
     --> .FALSE.  .OR.  .NOT.  .TRUE.  .AND.  .FALSE.
     --> .FALSE.  .OR.  .FALSE.  .AND.   .FALSE.
     --> .FALSE.  .OR.  .FALSE.
     --> .FALSE.
```

4. Let `INTEGER` variable `n` have a value of 4:

```
n**2 + 1 > 10 .AND. .NOT. n < 3
     --> 4**2 + 1 > 10 .AND. .NOT. 4 < 3
     --> 16 + 1 > 10 .AND. .NOT. 4 < 3
     --> 17 > 10 .AND. .NOT. 4 < 3
     --> .TRUE. .AND. .NOT. 4 < 3
     --> .TRUE. .AND. .NOT. .FALSE
     --> .TRUE. .AND. .TRUE.
     --> .TRUE.
```

5. Let `INTEGER` variables `m`, `n`, `x` and `y` have values 3, 5, 4 and 2, respectively:

```
.NOT. (m > n .AND. x < y) .NEQV.  (m <= n .AND. x >= y)
  -> .NOT. (3 > 5 .AND. 4 < 2) .NEQV. (3 <= 5 .AND. 4 >= 2)
  -> .NOT. (.FALSE. .AND. 4 < 2) .NEQV. (3 <= 5 .AND. 4 >= 2)
  -> .NOT. (.FALSE. .AND. .FALSE.) .NEQV. (3 <= 5 .AND. 4 >= 2)
  -> .NOT. (.FALSE.) .NEQV. (3 <= 5 .AND. 4 >= 2)
  -> .TRUE. .NEQV. (3 <= 5 .AND. 4 >= 2)
  -> .TRUE. .NEQV. (.TRUE. .AND. 4 >= 2)
  -> .TRUE. .NEQV. (.TRUE. .AND. .TRUE.)
  -> .TRUE. .NEQV. (.TRUE.)
  -> .TRUE. .NEQV. .TRUE.
  -> .FALSE.
```

`IF-THEN-ELSE-END IF` Statement

Syntax

1. The complete form:

```
IF (logical-expression) THEN
   statements-1
ELSE
   statements-2
END IF
```

2. The complete form without `ELSE`

```
IF (logical-expression) THEN
   statements
END IF
```

3. A old form

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

Examples: `IF-THEN-ELSE-END IF`

1. Determine if an integer is even or odd.

```
INTEGER :: Number

READ(*,*)  Number
IF (MOD(Number, 2) == 0) THEN
   WRITE(*,*)  Number, ' is even'
ELSE
   WRITE(*,*)  Number, ' is odd'
END IF
```

2. Compute the absolute value of `x`:

```
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
```

3. Determine the smaller number of two

```
INTEGER  :: a, b, Smaller

READ(*,*)  a, b
IF (a <= b) THEN
   Smaller = a
ELSE
   Smaller = b
END IF
Write(*,*)  'The smaller of ', a, ' and ', &
            b, ' is ', Smaller
```

Examples: `IF-THEN-END IF`

1. Compute the absolute value of `x`

```
REAL  :: X, Absolute_X

X = .....
Absolute_X = X
IF (X < 0.0) THEN
   Absolute_X = -X
END IF
WRITE(*,*)  'The absolute value of ', x, &
            ' is ', Absolute_X
```

2. Determine the smaller of two

```
INTEGER  :: a, b, Smaller

READ(*,*)  a, b
Smaller = a
IF (a > b) THEN
   Smaller = b
END IF
Write(*,*)  'The smaller of ', a, ' and ', &
            b, ' is ', Smaller
```

3. Whenever `Counter` is a multiple of 10, display a blank line.

```
INTEGER :: Counter

IF (MOD(Counter, 10) == 0) THEN
   WRITE(*,*)
END IF
```

Examples: IF

1. Compute the absolute value of x

```
REAL  :: X, Absolute_X

X = .....
Absolute_X = X
IF (X < 0.0)  Absolute_X = X
WRITE(*,*)  'The absolute value of ', x, &
            ' is ', Absolute_X
```

2. Determine the smaller of two

```
INTEGER  :: a, b, Smaller

READ(*,*)  a, b
Smaller = a
IF (a > b)  Smaller = b
Write(*,*)  'The smaller of ', a, ' and ', &
            b, ' is ', Smaller
```

3. Whenever Counter is a multiple of 10, display a blank line.

```
INTEGER :: Counter

IF (MOD(Counter, 10) == 0)  WRITE(*,*)
```

Programming Example 1

The roots of $ax^2 + bx + c = 0$ can be computed as follows:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

where $b^2 - 4ac$ must be non-negative to have 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
   WRITE(*,*) 'a = ', a
   WRITE(*,*) 'b = ', b
   WRITE(*,*) 'c = ', c
   WRITE(*,*)

!  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)       ! first root
      root2 = (-b - d)/(2.0*a)       ! second root
      WRITE(*,*)  'Roots are ', root1, ' and ', root2
   ELSE                              ! complex roots
      WRITE(*,*)  'There is no real roots!'
      WRITE(*,*)  'Discriminant = ', d
   END IF

END PROGRAM  QuadraticEquation
```

Programming Example 2

Two examination papers are written at the end of the course. The final mark is either the average of the two papers, or the average of the two papers and the class record mark (all weighted equally), whichever is the higher. The program should reads in the class record mark and the marks of the papers, computes the average, and shows PASS ($\geq 50\%$) or FAIL ($< 50\%$).

```
PROGRAM  FinalMark
   IMPLICIT  NONE

   REAL    :: Mark1, Mark2         ! the marks of the papers
   REAL    :: Final                ! the final marks
   REAL    :: ClassRecordMark      ! the class record mark

   REAL, PARAMETER :: PassLevel = 50.0  ! the pass level

   READ(*,*)  ClassRecordMark, Mark1, Mark2

   Final = (Mark1 + Mark2) / 2.0
   IF (Final <= ClassRecordMark) THEN
      Final = (Mark1 + Mark2 + ClassRecordMark) / 3.0
   END IF

   WRITE(*,*)  'Class Record Mark : ', ClassRecordMark
   WRITE(*,*)  'Mark 1            : ', Mark1
   WRITE(*,*)  'Mark 2            : ', Mark2
   WRITE(*,*)  'Final Mark        : ', Final

   IF (Final >= PassLevel) THEN
      WRITE(*,*)  'Pass Status       : PASS'
   ELSE
      WRITE(*,*)  'Pass Status       : FAIL'
   END IF

END PROGRAM  FinalMark
```

Programming Example 3

The area of the triangle formed by three sides with lengths a, b and c can be computed with Heron's formula area $= \sqrt{s(s-a)*(s-b)*(s-c)}$, where $s = (a+b+c)/2$.

```
PROGRAM  HeronFormula
   IMPLICIT  NONE

   REAL     :: a, b, c             ! three sides
   REAL     :: s                   ! half of perimeter
   REAL     :: Area                ! triangle area
   LOGICAL  :: Cond_1, Cond_2      ! two logical conditions

   READ(*,*)  a, b, c

   WRITE(*,*)  "a = ", a
   WRITE(*,*)  "b = ", b
   WRITE(*,*)  "c = ", c
   WRITE(*,*)

   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
```

Nested IF-THEN-ELSE-END IF

You can use `IF-THEN-ELSE-END IF` within another. For example:

```
IF (logical-expression) THEN
   ......
   IF (logical-expression) THEN
      ......
   ELSE
      ......
   END IF
   ......
ELSE
   ......
END IF
```

```
IF (logical-expression) THEN
   ......
ELSE
   ......
   IF (logical-expression) THEN
      ......
   ELSE
      ......
   END IF
   ......
END IF
```

Examples

1. If `x` is greater than, equal to or less than zero, display `+`, `0` or `-`

```
IF (x > 0) THEN
   WRITE(*,*)  '+'
ELSE
   IF (x < 0) THEN
      WRITE(*,*)  '-'
   ELSE
      WRITE(*,*)  '0'
   END IF
END IF
```

2. Given `x`, display `-x`, `x*x` or `2*x` if `x` is less than zero, in the range of 0 and 1 inclusive, or greater than 1:

```
IF (x < 0) THEN
   WRITE(*,*)  -x
ELSE
   IF (x < 1) THEN
      WRITE(*,*)  x*x
   ELSE
      WRITE(*,*)  2*x
   END IF
END IF
```

3. Determine the smallest of three:

```
IF (a < b) THEN
   IF (a < c) THEN
      Result = a
   ELSE
      Result = c
   END IF
ELSE
   IF (b < c) THEN
      Result = b
   ELSE
      Result = c
   END IF
END IF
```

IF-THEN-ELSE IF-END IF Statement

When you have a series tests against a common variable, `IF-THEN-ELSE IF-END IF` can simplify a lot.

Syntax

```
IF (logical-expression-1) THEN
   statements-1
ELSE IF (logical-expression-2) THEN
   statements-2
ELSE IF (logical-expression-3) THEN
   statements-3
   ........
ELSE
   statements-ELSE
END IF
```

Examples

1. If `x` is greater than, equal to or less than `0`, display `+`, `0` or `-`:

```
IF (x > 0) THEN
   WRITE(*,*)  '+'
ELSE IF (x == 0) THEN
   WRITE(*,*)  '0'
ELSE
   WRITE(*,*)  '-'
END IF
```

2. If `x` is less than 0, in the range of 0 and 1 inclusive, or greater than 1, display `-x`, `x*x`, or `2*x`:

```
IF (x < 0) THEN
   WRITE(*,*)  -x
ELSE IF (x <= 1) THEN
   WRITE(*,*)  x*x
ELSE
   WRITE(*,*)  2*x
END IF
```

3. Determine the smallest of three:

```
IF (a < b .AND. a < c) THEN
   Result = a
ELSE IF (b < a .AND. b < c) THEN
   Result = b
ELSE
   Result = c
END IF
```

4. If `x` is less than 50, the letter grade is `F`; if `x` is less than 60 and greater than or equal to 50, the letter grade is `D`, etc.

```
INTEGER           :: x
CHARACTER(LEN=1)  :: Grade
IF (x < 50) THEN
   Grade = 'F'
ELSE IF (x < 60) THEN
   Grade = 'D'
ELSE IF (x < 70) THEN
   Grade = 'C'
ELSE IF (x < 80) THEN
   Grade = 'B'
ELSE
   Grade = 'A'
END IF
```

Programming Example 1

Solve $ax^2 + bx + c = 0$; but distinguish repeated root ($b^2 - 4ac = 0$).

```
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
   WRITE(*,*) 'a = ', a
   WRITE(*,*) 'b = ', b
   WRITE(*,*) 'c = ', c
   WRITE(*,*)

!  compute the discriminant d

   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 roots!'
         WRITE(*,*)  'Discriminant = ', d
      END IF
   END IF
END PROGRAM  QuadraticEquation
```

Programming Example 2

Solve $ax^2 + bx + c = 0$ and deal with **all** possible cases.

```
PROGRAM  QuadraticEquation
   IMPLICIT  NONE
   REAL  :: a, b, c
   REAL  :: d
   REAL  :: root1, root2
          ........................................
   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 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 roots!'
         WRITE(*,*)  'Discriminant = ', d
      END IF
   END IF
END PROGRAM  QuadraticEquation
```

Programming Example Example 3

Display a, b and c in increasing order:

<table>
<tr><td colspan="6" align="center">$a < b$</td></tr>
<tr><td colspan="3" align="center">$a < c$</td><td colspan="3" align="center">$b < c$</td></tr>
<tr><td colspan="2" align="center">$b < c$</td><td rowspan="2" align="center">$c \leq a \leq b$</td><td colspan="2" align="center">$a < c$</td><td rowspan="2" align="center">$c \leq b \leq a$</td></tr>
<tr><td align="center">$a \leq b \leq c$</td><td align="center">$a \leq c \leq b$</td><td align="center">$b \leq a \leq c$</td><td align="center">$b \leq c \leq a$</td></tr>
</table>

```
IF (a < b) THEN                    ! a < b here
   IF (a < c) THEN                 !   a < c     : a the smallest
      IF (b < c) THEN              !      b < c  : a < b < c
         WRITE(*,*)  a, b, c
      ELSE                         !      c <= b : a < c <= b
         WRITE(*,*)  a, c, b
      END IF
   ELSE                            !   a >= c    : c <= a < b
      WRITE(*,*) c, a, b
   END IF
ELSE                               ! b <= a here
   IF (b < c) THEN                 !   b < c     : b the smallest
      IF (a < c) THEN              !     a < c   : b <= a < c
         WRITE(*,*)  b, a, c
      ELSE                         !     a >= c  : b < c <= a
         WRITE(*,*)  b, c, a
      END IF
   ELSE                            !   c <= b    : c <= b <= a
      WRITE(*,*)  c, b, a
   END IF
END IF
```

SELECT CASE Statement

Syntax

```
SELECT CASE (selector)
   CASE (label-list-1)
      statements-1
   CASE (label-list-2)
      statements-2
   CASE (label-list-3)
      statements-3
     .............
   CASE (label-list-n)
      statements-n
   CASE DEFAULT
      statements-DEFAULT
END SELECT
```

- `selector` is an expression whose result must be of type `INTEGER`, `CHARACTER` or `LOGICAL`. No `REAL` is allowed.
- A `label` has one of the following four forms:

Label	*Menaing*
$: x$	all values $\leq x$
$x :$	all values $\geq x$
$x : y$	all values in the range of x and y $(x \leq y)$
x	the value of x itself

- A `label-list` is a number of `label`s separated by commas

Examples

1. Count the number of Doctor MD (`DrMD`), Ph.D. (`PhD`), master (`MS`) and bachelor (`BS`) using `CHARACTER` variable `Title`. Otherwise, add 1 to `Others`.

```
CHARACTER(LEN=4) :: Title
INTEGER          :: DrMD = 0, PhD = 0, MS = 0
INTEGER          :: BS = 0, Others = 0

SELECT CASE (Title)
   CASE ("DrMD")
      DrMD = DrMD + 1
   CASE ("PhD")
      PhD = PhD + 1
   CASE ("MS")
      MS = MS + 1
   CASE ("BS")
      BS = BS + 1
   CASE DEFAULT
      Others = Others + 1
END SELECT
```

2. Old problem again. Display the sign of `x`

```
INTEGER :: Number, Sign

SELECT CASE (Number)
   CASE ( : -1)
      WRITE(*,*)  Sign = -1
   CASE (0)
      WRITE(*,*)  Sign = 0
   CASE (1 : )
      WRITE(*,*)  Sign = 1
END SELECT
```

3. Display `Freshman`, `Sophomore`, `Junior`, or `Senior` if the value of `Class` is 1, 2, 3 or 4; otherwise, display `Hmmmm, I don't know`.

```
INTEGER :: Class

SELECT CASE (Class)
   CASE (1)
      WRITE(*,*)  'Freshman'
   CASE (2)
      WRITE(*,*)  'Sophomore'
   CASE (3)
      WRITE(*,*)  'Junior'
   CASE (4)
      WRITE(*,*)  'Senior'
   CASE DEFAULT
      WRITE(*,*)  "Hmmmm, I don't know"
END SELECT
WRITE(*,*)  'Done'
```

4. Determine the content of `c`

```
CHARACTER(LEN=1) :: c

SELECT CASE (c)
   CASE ('a' : 'j')
      WRITE(*,*)  'One of the first ten letters'
   CASE ('l' : 'p', ''u' : 'y')
      WRITE(*,*)  'One of l, m, n, o, p, q, u, v, w, x, y'
   CASE ('z', 'q' : 't')
      WRITE(*,*)  'One of z, q, r, s, t'
   CASE DEFAULT
      WRITE(*,*)  'Other characters, which may not be letters'
END SELECT
```

5. Try different values of `Number` please.

```
INTEGER :: Number, Range

SELECT CASE (Number)
   CASE ( : -10, 10 : )
      Range = 1
   CASE (-5:-3, 6:9)
      Range = 2
   CASE (-2:2)
      Range = 3
   CASE (3, 5)
      Range = 4
   CASE (4)
      Range = 5
   CASE DEFAULT
      Range = 6
END SELECT
```

Value of Number	*Value of Range*	**Why?**
≤ -10	1	`CASE (:-10, :10)`
-9, -8, -7, -6	6	`CASE DEFAULT`
-5, -4, -3	2	`CASE (-5:-5, 6:9)`
-2, -1, 0, 1, 2	3	`CASE (-2:2)`
3	4	`CASE (3, 5)`
4	5	`CASE (4)`
5	4	`CASE (3, 5)`
5, 6, 7, 8	2	`CASE (-5:-3, 6:9)`
≥ 10	1	`CASE (:-10, :10)`

Programming Example 1

Read in three marks, compute their average, round the average, and determine its letter grade using the following table:

Range	≥ 90	≥ 85	≥ 80	≥ 75	≥ 70	≥ 65	≥ 60	< 60
Grade	A	AB	B	BC	C	CD	D	F

```
PROGRAM  LetterGrade
   IMPLICIT   NONE
   REAL               :: Mark1, Mark2, Mark3
   REAL               :: Average
   CHARACTER(LEN=2)   :: Grade

   READ(*,*)  Mark1, Mark2, Mark3
   Average = (Mark1 + Mark2 + Mark3) / 3.0
   SELECT CASE (NINT(Average))      ! round Average before use
      CASE (:59)                    ! <= 59 -------------> F
         Grade = 'F '
      CASE (60:64)                  ! >= 60 and <= 64 ---> D
         Grade = 'D '
      CASE (65:69)                  ! >= 65 and <= 69 ---> CD
         Grade = 'CD'
      CASE (70:74)                  ! >= 70 and <= 74 ---> C
         Grade = 'C '
      CASE (75:79)                  ! >= 75 and <= 79 ---> BC
         Grade = 'BC'
      CASE (80:84)                  ! >= 80 and <= 84 ---> B
         Grade = 'B '
      CASE (85:89)                  ! >= 84 and <= 89 ---> AB
         Grade = 'AB'
      CASE DEFAULT                  ! >= 90 -------------> A
         Grade = 'A '
   END SELECT
     ................................
END PROGRAM  LetterGrade
```

Programming Example 2

Read a character, determine if it is a vowel, a consonant, one of the four operators ($+, -, *, /$), a space, or something else.

```
PROGRAM  CharacterTesting
   IMPLICIT  NONE

   CHARACTER(LEN=1) :: Input

   READ(*,*)  Input

   SELECT CASE (Input)
      CASE ('A' : 'Z', 'a' : 'z')       ! rule out letters
         WRITE(*,*)  'A letter is found : "', Input, '"'
         SELECT CASE (Input)            ! a vowel ?
            CASE ('A', 'E', 'I', 'O', 'U', 'a', 'e', 'i', 'o','u')
               WRITE(*,*)  'It is a vowel'
            CASE DEFAULT                ! it must be a consonant
               WRITE(*,*)  'It is a consonant'
         END SELECT
      CASE ('0' : '9')                  ! a digit
         WRITE(*,*)  'A digit is found : "', Input, '"'
      CASE ('+', '-', '*', '/')         ! an operator
         WRITE(*,*)  'An operator is found : "', Input, '"'
      CASE (' ')                        ! space
         WRITE(*,*)  'A space is found : "', Input, '"'
      CASE DEFAULT                      ! something else
         WRITE(*,*)  'Something else found : "', Input, '"'
   END SELECT

END PROGRAM  CharacterTesting
```