Identifiers: 1/2

- **A Fortran 90 identifier can have no more than 31 characters.**
- **The first one must be a letter. The remaining characters, if any, may be letters, digits, or underscores.**
- ***Fortran 90 identifiers are CASE INSENSITIVE.***
- **Examples: `A`, `Name`, `toTAL123`, `System_`, `myFile_01`, `my_1st_F90_program_X_`.**
- **Identifiers `Name`, `nAmE`, `naME` and `NamE` are the same.**

Identifiers: 2/2

- **Unlike Java, C, C++, etc, *Fortran 90 does not have reserved words*. This means one may use Fortran keywords as identifiers.**
- **Therefore, `PROGRAM`, `end`, `IF`, `then`, `DO`, etc may be used as identifiers. Fortran 90 compilers are able to recognize keywords from their "positions" in a statement.**
- **Yes, `end = program + if/(goto - while)` is legal!**
- **However, avoid the use of Fortran 90 keywords as identifiers to minimize confusion.**

Declarations: 1/3

- Fortran 90 uses the following for variable declarations, where `type-specifier` is one of the following keywords: **`INTEGER`**, **`REAL`**, **`LOGICAL`**, **`COMPLEX`** and **`CHARACTER`**, and `list` is a sequence of identifiers separated by commas.

```
type-specifier :: list
```

- Examples:

```
INTEGER :: Zip, Total, counter
REAL    :: AVERAGE, x, Difference
LOGICAL :: Condition, OK
COMPLEX :: Conjugate
```

The PARAMETER Attribute: 2/4

- Since CHARACTER identifiers have a length attribute, it is a little more complex when used with PARAMETER.
- Use (LEN = *) if one does not want to count the number of characters in a PARAMETER character string, where = * means the length of this string is determined elsewhere.

```
CHARACTER(LEN=3), PARAMETER :: YES = "yes"   ! Len = 3
CHARACTER(LEN=2), PARAMETER :: NO = "no"     ! Len = 2
CHARACTER(LEN=*), PARAMETER :: &
                  PROMPT = "What do you want?" ! Len = 17
```

The PARAMETER Attribute: 4/4

- By convention, PARAMETER identifiers use all upper cases. However, this is not mandatory.
- For maximum flexibility, constants other than 0 and 1 should be PARAMETERized.
- A PARAMETER is an alias of a value and is *not* a variable. Hence, one cannot modify the content of a PARAMETER identifier.
- One can may a PARAMETER identifier anywhere in a program. It is equivalent to replacing the identifier with its value.
- The value part can use expressions.

Variable Initialization: 2/2

- **Variable initialization is very similar to what we learned with `PARAMETER`.**
- **A variable name is followed by a `=`, followed by an expression in which all identifiers must be constants or `PARAMETER`s defined *previously*.**
- **Using an un-initialized variable may cause un-expected, sometimes disastrous results.**

```
REAL :: Offset = 0.1, Length = 10.0, tolerance = 1.E-7
CHARACTER(LEN=2) :: State1 = "MI", State2 = "MD"
INTEGER, PARAMETER :: Quantity = 10, Amount = 435
INTEGER, PARAMETER :: Period = 3
INTEGER :: Pay = Quantity*Amount, Received = Period+5
```

Arithmetic Operators

- There are four types of operators in Fortran 90: arithmetic, relational, logical and character.
- The following shows the first three types:

Type	*Operator*						*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

Mixed Mode Expression: 1/2

- If operands have different types, it is *mixed mode*.
- `INTEGER` and `REAL` yields `REAL`, and the `INTEGER` operand is converted to `REAL` before evaluation. Example: `3.5*4` is converted to `3.5*4.0` becoming single mode.
- Exception: `x**INTEGER`: `x**3` is `x*x*x` and `x**(-3)` is `1.0/(x*x*x)`.
- `x**REAL` is evaluated with `log()` and `exp()`.
- Logical and character cannot be mixed with arithmetic operands.

List-Directed WRITE: 2/3

- **Here is a simple example:**


```
INTEGER :: Target
REAL :: Angle, Distance
CHARACTER(LEN=*), PARAMETER ::          &
  Time = "The time to hit target ",     &
  IS = " is ",                          &
  UNIT = " sec."

Target = 10
Angle = 20.0
Distance = 1350.0
WRITE(*,*) 'Angle = ', Angle
WRITE(*,*) 'Distance = ', Distance
WRITE(*,*)
WRITE(*,*) Time, Target, IS,    &
           Angle * Distance, UNIT
```

means length is determined by actual count

continuation lines

print a blank line

Output:

```
Angle =  20.0
Distance =  1350.0
The time to hit target  10  is  27000.0  sec.
```

List-Directed WRITE: 3/3

- **The previous example used `LEN=*` , which means the length of a `CHARACTER` constant is determined by actual count.**
- **`WRITE(*,*)` without any expression advances to the next line, producing a blank one.**
- **A Fortran 90 compiler will use the *best* way to print each value. Thus, indentation and alignment are difficult to achieve with `WRITE(*,*)`.**
- **One must use the `FORMAT` statement to produce good looking output.**