Introduction to FORTRAN

A Brief Summary of GNU FORTRAN

Ashik Iqubal

Department of Physics
Ramakrishna Mission Vivekananda University
Belur Math, Howrah

ashik.iqubal@gmail.com

August 31, 2012

FORTRAN: Data Types

- INTEGER
- REAL
- COMPLEX
- CHARACTER
- LOGICAL

FORTRAN: Data Type Examples

Integer INTEGER :: variable1, variable2, ...

Real REAL :: variable1, variable2, ...

Complex COMPLEX :: variable1, variable2, ...

Character CHARACTER(len=character length) :: variable1, variable2, ..

Logical LOGICAL :: variable1, variable2, ...
LOGICAL :: FLAG
FLAG = .TRUE. or .FALSE

Arrays REAL, DIMENSION(10) :: VAR

FORTRAN: Arithmetic Operators

+	Addition
–	Subtraction
*	Multiplication
/	Division
**	Exponentiation

FORTRAN: Conditional IF Statement

Code

```
IF (condition) THEN
statements
END IF
```

statements are evaluated if *condition* is true

FORTRAN: Nested Conditional Statement

Code

```
IF (condition1) THEN
statements block 1
ELSE IF (condition2) THEN
statements block 2
.....
ELSE
statements
END IF
```

FORTRAN: Named Block IF Conditional Statement

Code

```
[label:] IF (condition1) THEN
statements block 1
ELSE IF (condition2) THEN [label]
statements block 2
.....
ELSE [label]
statements
END IF
```

FORTRAN: Relational Operators

$<$	less than
$<=$	less than or equal to
$>$	greater than
$>=$	greater than or equal to
$==$	equal to
$/=$	not equal to

FORTRAN: Logical Operators

.AND.	AND
.OR.	OR
.EQV.	Logical Equivalence
.NEQV.	Logical Non-Equivalence
.NOT.	NOT

FORTRAN: Order of Evaluation

1. All arithmetic operations are evaluated first from left to right
2. All relational operators are evaluated working from left to right
3. All .NOT. operators are evaluated
4. All .AND. operators are evaluated working from left to right
5. All .OR. operators are evaluated working from left to right
6. All .EQV. and .NEQV. operators are evaluated working from left to right

Parenthesis can be used to change the default order of evaluation

FORTRAN: DO Loops

Code

```
DO
statements
IF (exit-condition) EXIT statements
END DO
```

(Repeatedly) executes *statements* between DO and END DO until *exit-condition* is true

FORTRAN: DO WHILE Loops

Code

```
DO WHILE (condition)
statements
END DO
```

If *condition* is true, repeatedly executes *statements* between DO and END DO

FORTRAN: Iterative Loops

Code

```
DO index = istart, iend, increment
statements
END DO
```

1. index = istart
2. if index*increment < iend*increment , then it executes the *statements*
3. index = index + increment
4. Repeat steps 2 - 3

FORTRAN: Named Loops

Code

```
[label:] DO index = istart, iend, increment
statements
IF (cycle-condition) CYCLE [label]
statements
IF (exit-condition) EXIT [label]
statements
END DO
```

FORTRAN: Named Loops contd.

Code

```
[label:] DO
statements
IF (cycle-condition) CYCLE [label]
statements
IF (exit-condition) EXIT [label]
statements
END DO
```

FORTRAN: CYCLE and EXIT Statements

- *EXIT* statement exits loops block, jumping immediately to the next statement outside of the loop.
- *CYCLE* statement continues the loop after skipping the remaining statements in its current iteration.
- *GOTO* statement transfers control to another part of the program

FORTRAN: Function

Code

```
FUNCTION function-name (input-variables)
IMPLICIT NONE
REAL/INTEGER, INTENT(IN) :: input-variables
REAL/INTEGER, :: function-name
statements
function-name = expression
END FUNCTION function-name
```

FORTRAN: Recursive Function

Code

```
RECURSIVE FUNCTION function(input-var) RESULT(answer)
IMPLICIT NONE
REAL/INTEGER, INTENT(IN) :: input-var
REAL/INTEGER :: answer
statements
answer = expression
END FUNCTION function
```

FORTRAN: Subroutine

Code

```
SUBROUTINE subroutine-name (input-variables, output-variables)
IMPLICIT NONE
REAL/INTEGER, INTENT(IN) :: input-variables
REAL/INTEGER, INTENT(OUT) :: output-variables
REAL/INTEGER, INTENT(INOUT) :: common
input/output-variables
statements
END SUBROUTINE subroutine-name
```

Using RETURN in the subroutine returns to the calling program
Subroutines can be called anywhere in the program by using :

Code

```
CALL subroutine-name(input-variables, output-variables)
```

FORTRAN: Recursive Subroutine

If the subroutine is used recursively, then use

Code

```
RECURSIVE SUBROUTINE subroutine-name (variables)
declarations and statements
END SUBROUTINE subroutine-name
```

FORTRAN: Subroutine contd.

Subroutines/Functions are generally placed at the end of the program after using a CONTAINS statement

Code

```
main program
.......
CONTAINS
SUBROUTINE subroutine-name (variables)
........
END SUBROUTINE subroutine-name
END
```