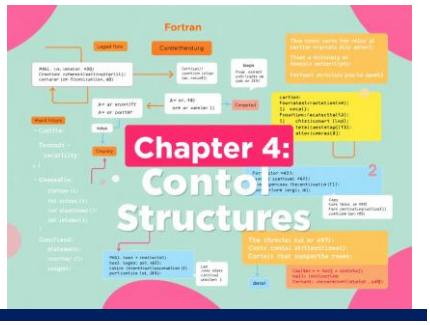
Chapter 4:
Control Structures



Chapter 3: Control Structures in Fortran

3.1 Introduction

Control structures allow the program execution to be directed based on conditions or iterations. They are essential for:

- Modifying the normal sequential flow of a program.
- Performing logical tests and making decisions.
- Repeating instructions a certain number of times or until a condition is met.

Types of Control Structures:

1. **Conditional Structures**: Allow decision-making (e.g., IF-THEN-ELSE).
2. **Iterative Structures (Loops)**: Allow repeating instructions (e.g., DO, DO WHILE, GOTO).

3.2 Conditional Statements (Decision Making)

Fortran provides several types of conditional tests to make decisions.

3.2.1 Simple **IF** Statement

Tests a condition and executes an instruction if the condition is true.

Syntax:

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

Example:

```
IF (x > 0) THEN
PRINT *, 'x is positive'
END IF
```

3.2.2 **IF-THEN-ELSE** Statement

Allows choosing between two blocks of instructions based on a condition.

Syntax:

```
IF (condition) THEN
statements_if_true
```

```
ELSE
statements_if_false
END IF
```

Example:

```
IF (x >= 10) THEN
PRINT *, 'x is greater than or equal to 10'
ELSE
PRINT *, 'x is less than 10'
END IF
```

3.2.3 **IF-THEN-ELSEIF-ELSE** Statement

Tests multiple conditions.

Syntax:

```
IF (condition1) THEN
statements1
ELSEIF (condition2) THEN
statements2
ELSE
statements3
END IF
```

Example:

```
IF (x < 0) THEN
   PRINT *, 'x is negative'
ELSEIF (x == 0) THEN
   PRINT *, 'x is zero'
ELSE
   PRINT *, 'x is positive'
END IF
```

3.2.4 **SELECT CASE** Statement

An alternative to IF when multiple options are possible.

Syntax:

```
SELECT CASE (variable)
CASE (value1)
statements1
```

```
CASE (value2)
statements2
CASE DEFAULT
default_statements
END SELECT
```

Example:

```
SELECT CASE (grade)
CASE (0:9)
PRINT *, 'Fail'
CASE (10:14)
PRINT *, 'Pass'
CASE (15:20)
PRINT *, 'Good'
CASE DEFAULT
PRINT *, 'Invalid grade'
END SELECT
```

3.3 Iterations (Loops)

Loops allow repeating a block of instructions a certain number of times or until a condition is met.

3.3.1 **DO** Loop (Fixed Counter Loop)

Used to execute a block of instructions a determined number of times.

Syntax:

```
DO variable = start, end, step
statements
END DO
```

Example:

```
DO i = 1, 5
   PRINT *, 'Iteration number', i
END DO
```

Output:

```
Iteration number 1
Iteration number 2
```

```
Iteration number 3
Iteration number 4
Iteration number 5
```

3.3.2 **DO WHILE** Loop (Conditional Loop)

Used when the number of iterations is unknown in advance.

Syntax:

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

Example:

```
i = 1
DO WHILE (i <= 5)
   PRINT *, 'Value of i:', i
   i = i + 1
END DO
```

3.3.3 Infinite **DO** Loop with **EXIT**

Allows breaking the loop once a condition is met.

Syntax:

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

Example:

```
i = 1
DO
   PRINT *, 'Value of i:', i
   IF (i >= 5) EXIT
   i = i + 1
END DO
```

3.3.4 CYCLE Statement (Skipping an Iteration)

Skips to the next iteration without executing the remaining statements in the loop.

Example:

```
DO i = 1, 5
   IF (i == 3) CYCLE
   PRINT *, 'i =', i
END DO
```

Output:

```
i = 1
i = 2
i = 4
i = 5
(Value i = 3 is skipped.)
```

3.3.5 **GOTO** Statement (Discouraged)

Allows jumping directly to a labeled line of code.

Syntax:

```
n = 1
10 PRINT *, 'n =', n
n = n + 1
IF (n <= 5) GOTO 10
```

This loop works but makes the code harder to read and maintain. Using DO or WHILE is recommended instead.

Conclusion

Control structures are essential for writing efficient programs in Fortran.

- **Conditional statements (IF, SELECT CASE)** allow decision-making.
- **Loop structures (DO, WHILE, EXIT)** enable repeating actions.
- **CYCLE and EXIT statements** help control iterations.