# Function Design: Cohesion and Coupling

CS2023 Winter 2004

# Outcomes: Function Design

- “Code Complete”, Chapter 5 (on reserve in library)
- After the conclusion of this section you should know what to aim for when designing functions: they should be strongly cohesive and loosely coupled.

# Cohesion

- How closely are the operations in a function related?
  - sin()
  - sinAndTan()
- Strong cohesion
  - Function should do one thing well *and not do anything else*

# Types of Cohesion

- **Functional cohesion**
  - The best kind!
  - Sin(), GetCustomerName(), CalcLoanPayment()
  - Can be very short
- Sequential cohesion
- Temporal cohesion
- Logical cohesion
- ...

# Sequential Cohesion

- Operations performed in a specific order, share data from step to step, but don't perform a complete function

Program: Open File, Read File, Perform Calculations, Output Results, and Close File

**`DoStep1()`**
Open File
Read File
Perform Calculations

**`DoStep2()`**
Output Results
Close File

# Removing Sequential Cohesion

- Names should have verb + object

**`GetFileData()`**
Open File
Read File

**`MassageData()`**
Perform Calculations

**`OuputFileData()`**
Output Results
Close File

# Temporal Cohesion

- Operations combined in a function because all done at same time

```
Startup()
```

- Best to have this kind of function call other functions

```
ReadConfigFile()
InitializeMemory()
ShowInitialScreen()
```

# Logical Cohesion

- Function does one of several things depending on a control flag parameter

`OutputAll()`

- Better to have distinct functions for each operation

`OutputSummary(), OutputReport(), OutputDetailedReport()`

- Can still have logically cohesive function call these

# Coupling

- How strongly functions are related to each other
- Want loose coupling: independent functions
- Coupling criteria:
  - **Size** (how many variables shared?)
  - **Intimacy** (parameters, global data, files)
  - **Visibility** (coupling by global data is sneaky)
  - **Flexibility** (how easily can connections be changed?)

# Levels of Coupling

- Data coupling
  - simple data
  - data structures
- Control coupling
  - One function passes data to another function that tells it what to do
- Global data coupling
  - tolerable if global data is read-only, or if global data couples closely related functions in a module

# Why Loose Coupling?

- Reduces program complexity, allowing programmer to focus on one thing at a time.
- If functions are too closely coupled then complexity is not reduced.

# Structured Design

- Iterative process: multilevel decomposition
- Top-down decomposition
  - design top-level first
  - postpone working out details until lower level of design
- Bottom-up decomposition
  - what does this system need to do?
- Can use both!