Cohesion and Coupling

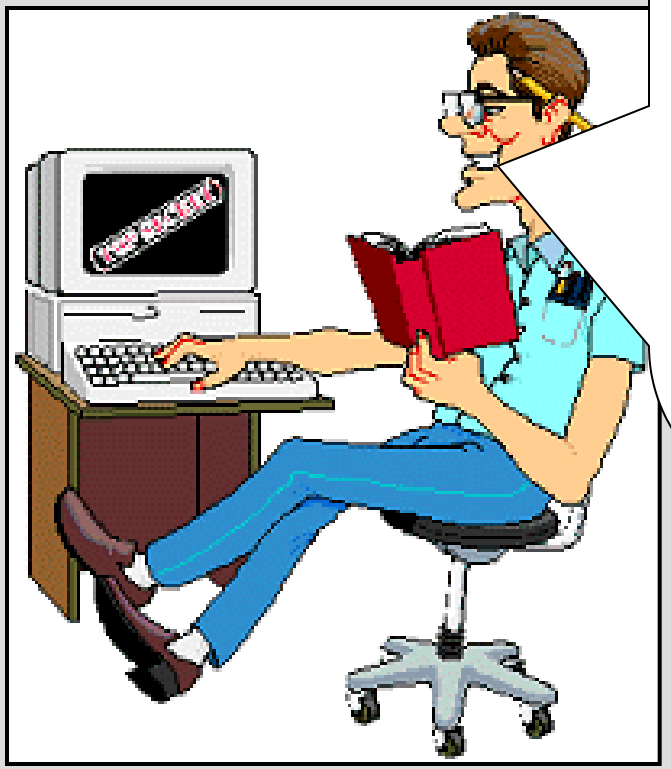
OBJECTIVES:

- To learn the levels of cohesion.

Seven levels of COHESION:

1. COINCIDENTAL
2 .LOGICAL
3. TEMPORAL
4. PROCEDURAL
5. COMMUNICATIONAL
6. SEQUENTIAL
7. FUNCTIONAL

Cohesion and Coupling

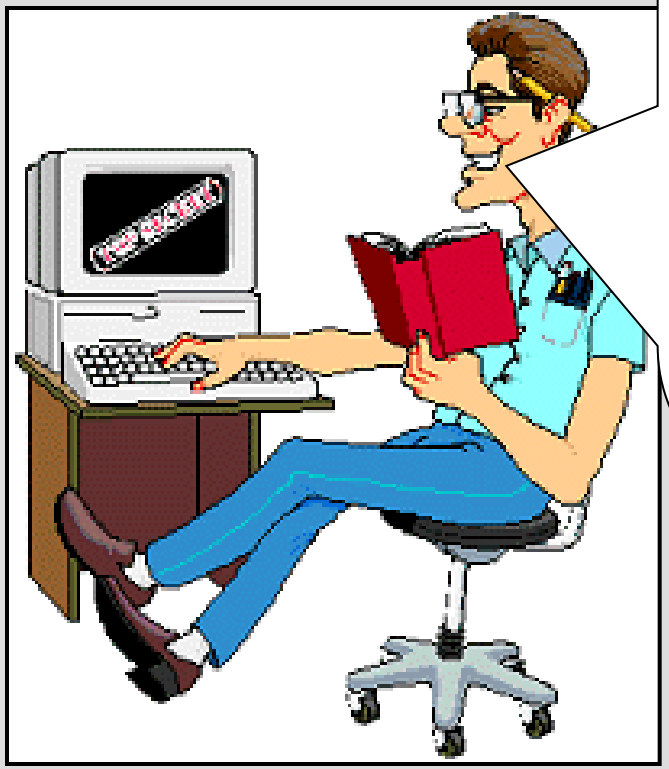


COHESION means "Glue". People "glue" statements together to form a single procedure (or module). These ways range from very bad to very good.

Seven levels of COHESION:

1. **COINCIDENTAL (LOWEST)**
2. **LOGICAL**
3. **TEMPORAL**
4. **PROCEDURAL**
5. **COMMUNICATIONAL**
6. **SEQUENTIAL**
7. **FUNCTIONAL (HIGHEST)**

Cohesion and Coupling



FUNCTIONAL cohesion is the highest level. It is not the lower six levels.

Seven levels of COHESION:

1. **COINCIDENTAL (LOWEST)**
2. **LOGICAL**
3. **TEMPORAL**
4. **PROCEDURAL**
5. **COMMUNICATIONAL**
6. **SEQUENTIAL**
7. **FUNCTIONAL (HIGHEST)**

Cohesion and Coupling

Topic: COINCIDENTAL cohesion.

COINCIDENTAL COHESION - LOWEST LEVEL

1. Coincidental cohesion implies little or no relationship among statements of code within a procedure (a module).

2. EXAMPLES:
   (1) A programmer might take a 200 line program and break it into four procedures with 50 lines of code in each. Thus, lines 1-50 would be in procedure-1 and lines 51-100 would be in procedure-2, etc.
      a. It sounds hard to believe that anyone would form procedures this way but it happened a lot in the early days of programming. Programmers might have 10 to 200 pages of code...statement after statement, page after page. When these programs became far too complex to understand, these programmers would the break up to code into procedures of a certain length and then call these procedures from a main procedure. Yes, this happened a lot.
      a. Once a professor at the Air-Force Academy stated at a national conference that his students were only allowed to put 13 statements in a procedure. (True story...I was there on a discussion panel with this misinformed professor.)
   (2) Recurring code which has <u>no complete function</u> might be spotted by the programmer and, in an effort to save memory space, placed that code in one procedure (a module). That procedure was then repeatedly called in the program.

3. If this type of procedure is changed to accommodate another calling procedure, the remaining procedures which call this coincidentally formed procedure will also need to be changed. It is also possible that the other calling modules just simply will not be able to use the changed coincidentally formed procedure.

Cohesion and Coupling

Topic: LOGICAL cohesion.

LOGICAL COHESION

1. **This type of procedure performs one or more logically related functions.**

2. **EXAMPLE:**
   (1) A "print all" procedure might consist of:
   - **a. Printing a local sales report.**
   - **b. Printing a regional sales report.**
   - **c. Printing a national sales report.**
   - **d. Printing a international sales report A "print all" procedure might consist of:**

If the user, just wanted a national sales report, and not any of the others, a flag (a switch) would need to be passed to the procedure (using arguments and parameters) to tell the procedure which report to print.

It is easy to spot this level of cohesion, because one or more flags are passed to the procedure to tell it which function to perform.

FLAGS ARE TO BE AVOIDED, BECAUSE THEY ADD COMPLEXITY TO THE CODE.

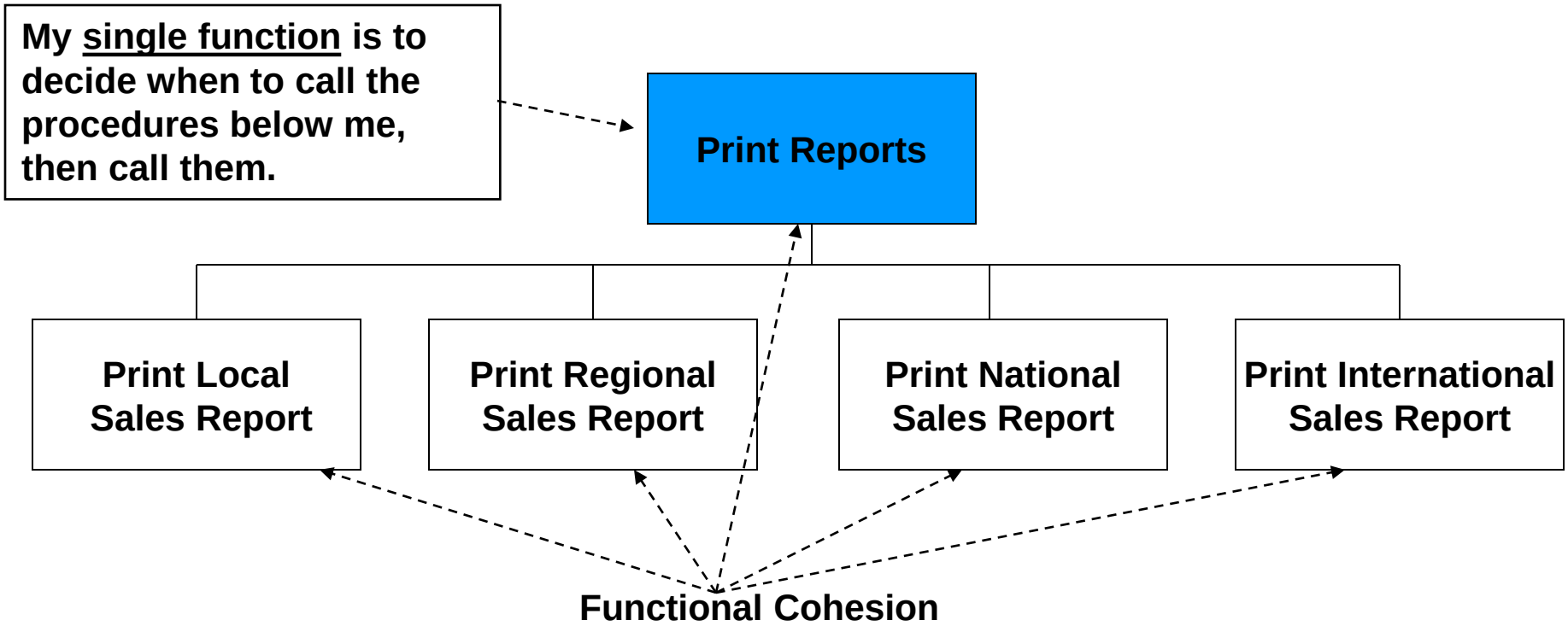
Cohesion and Coupling

Topic: LOGICAL cohesion.

LOGICAL COHESION

3. You can eliminate the need for logical cohesion, by making each report above a separate procedure. Thus, there would be four separate procedures which could be called individually when needed by using buttons, check boxes, radio button, other procedures, etc...

USING A PROCEDURE TO CALL THE OTHERS:



Topic: LOGICAL cohesion.

LOGICAL COHESION

FLAGS ARE TO BE AVOIDED, BECAUSE THEY ADD COMPLEXITY TO THE CODE.

RULE: MAKE PROCEDURES, NOT FLAGS.

Do not pass flags as arguments to other procedures to control them.

EXCEPTION: Sometimes there is no other way but to use a flag, such as the flag we used in our validation procedure. Flags within a single procedure are ok, such as in the bubble sort. But in general try to avoid them if possible.

Cohesion and Coupling

Topic: TEMPORAL (CLASSICAL) cohesion.

TEMPORAL (CLASSICAL) COHESION

1. Temporal cohesion means that the statements are grouped into a procedure and executed together during the same time-frame: for example, at the very beginning or the very end of a program.

2. EXAMPLES:
   (1) An "Initialization" procedure which does the following:
      a. Opens all files.
      b. Initializes arrays and all variables.
      c. Sets all constants
      d. Sets all flags
      e. Reads the first transaction
   (2) A "Termination" procedure which does the following:
      a. Writes array contents to disk files.
      b. Closes all files.

3. You can eliminate temporally cohesive procedures by:
   (1) Open and close files within the procedures that use those files.
   (2) Initialize all variables IMMEDIATELY before you use them.
   (3) Set flags IMMEDIATELY before you use them such as in a bubble sort.
   (4) Set constants IMMEDIATELY before you use them.
   (5) Procedures should handle their own INITIALIZATION and TERMINATION needs.

Topic: TEMPORAL (CLASSICAL) COHESION.

TEMPORAL (CLASSICAL) COHESION (Continued)

4. **Temporal cohesion is easy to spot, because the code is put into one procedure and is executed because it is convenient to do so at a certain time in the program.**

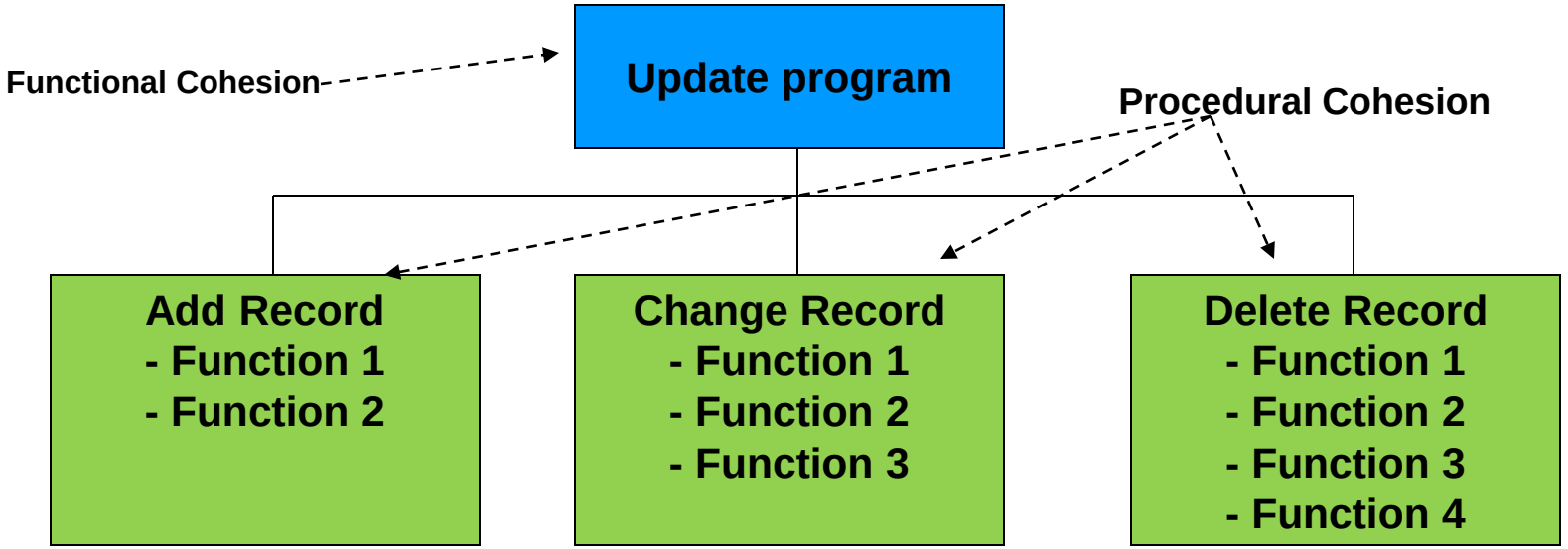
5. **Remember: The goal is to produce procedures which can do their single function independently of other procedures.**

Cohesion and Coupling

Topic: PROCEDURAL COHESION.

PROCEDURAL COHESION

1. This type of cohesion results from packaging <u>two or more single functions</u> together into a single procedure. Each procedure does ALL of the code necessary for a record addition, change, or deletion. In other words, it does the ENTIRE "Procedure".
   NOTE: The word "procedural" in procedural cohesion is being used in a generic sense. It does not mean a programming procedure such as a sub-procedure, a function procedure, or an event procedure. For example it means the "procedure" for running a household, building a jet engine, building a house, running a business, etc...
2. EXAMPLE:
   (1) Consider a file update program in which additions, changes and deletions of records are made to the file. Each procedure consists of two or more single functions. A hierarchy chart might look as below. The blue procedure's single function is CONTROL. The red procedures are doing several functions...the entire "procedure".



Cohesion and Coupling

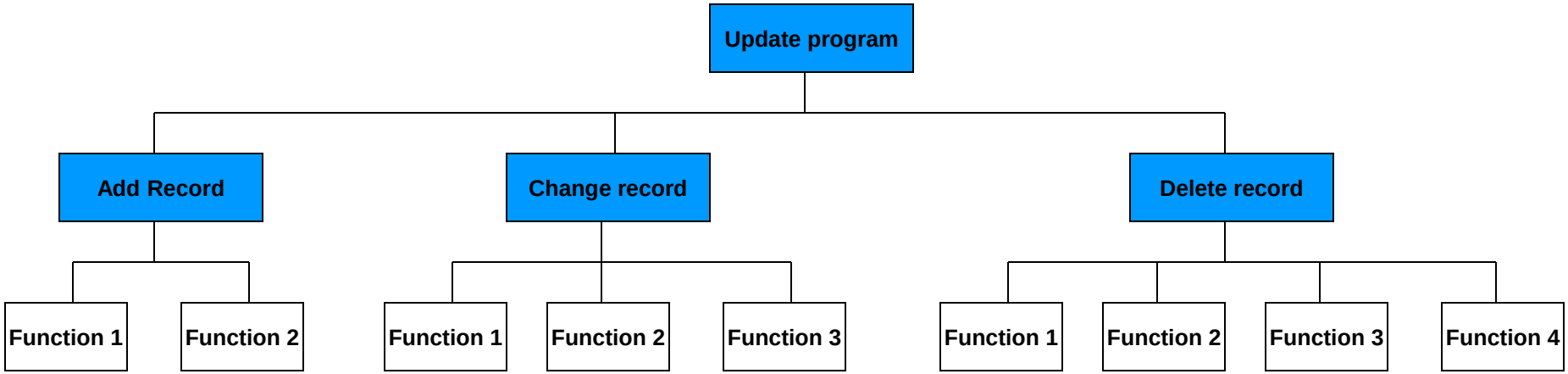
Topic: PROCEDURAL COHESION.

PROCEDURAL COHESION (Continued)

3. **To eliminate the "procedural" cohesion in this example, the program could be re-designed as indicated in the hierarchy chart below.**

4. **The blue procedures' single function is to CONTROL and the white procedures' single functions do the NUTS-AND-BOLTS work of the program. By "control" we mean that these procedures only call other procedures that have single functions and they do not do the "nuts-and-bolts" work of the program.**

5. **All procedures (modules of code) below are at the functional level of cohesion.**



Legend:

CONTROL

NUTS-AND-BOLTS

Cohesion and Coupling

Topic: PROCEDURAL COHESIONAL.

PROCEDURAL vs. TEMPORAL COHESION

TEMPORAL COHESION

(1) Separate functions or statements are grouped together simply because it is convenient to execute then at the same TIME.

PROCEDURAL COHESION

(1) Separate functions or statements are grouped together simply because they are part of a large procedure within the program.

NOTE: The word procedural in procedural cohesion is being used in a generic sense. It does not mean a programming procedure such as a sub-procedure, function procedure, or event procedure. For example it means the "procedure" for baking a cake, building a jet engine, building a house, running a business, etc...

Cohesion and Coupling

Topic: COMMUNICATIONAL COHESION.

COMMUNICATIONAL COHESION

1. **This type of cohesion results from considering all of the things that can be done with certain data and assigning all these activities a single procedure.**

2. **In other words, several functions are combined into one procedure. Each function in the process does something different to the data.**

3. **EXAMPLE:**

   DATA STRUCTURE: TRANSACTION FILE

   PROCEDURE FUNCTIONS:
   1. **Read transactions into array.**
   2. **Sort transactions**
   3. **Find the mean of the transactions.**
   4. **Print the transactions onto paper.**
   5. **Write the transaction to a CD-Writer**

Cohesion and Coupling

Topic: SEQUENTIAL COHESION.

SEQUENTIAL COHESION

1. When a procedure performs multiple processing functions on the same data and those functions represent successive transformations of the data, sequential cohesion results.

2. These procedures are easy to spot because the output from one function becomes the input to another function, etc. Since these separate functions are executed automatically in a sequence, control flags are not needed to direct the sequence of processing.

3. EXAMPLES:
   - (1) Functions in one procedure:
     a. Search array for a certain name.
     b. Place the name in another array and sort the array.
     c. Change the name from last-first to first-last.
     d. Print first-last name.

   - (2) Functions in one procedure:
     a. Read numbers from a disk file into an array.
     b. Find the mean of an array.
     c. Find deviation of each number from the mean.
     d. Sort deviations from the highest to the lowest
     e. Print the deviations in order from highest to the lowest.

Cohesion and Coupling

Topic: FUNCTIONAL COHESION.

FUNCTIONAL COHESION

1. **In functional cohesion, all the statements in the procedure are related in the performance of a single function.**

2. **All of the procedures below have functional cohesion:**
   - **(1) Sort an array**
   - **(2) Search an array**
   - **(3) Find largest value in an array.**
   - **(4) Find the mean of the numbers in an array.**
   - **(5) Merge the data in one array with another.**
   - **(6) Print data entry format on the CRT screen.**
   - **(7) Input all values from the screen.**
   - **(8) Print report headings.**
   - **(9) Find square root**
   - **(10) Compute random number.**
   - **(11) Process Payroll (BOSS MODULE)**
     Note: This procedure's single function is to call other procedures. Thus, its function is CONTROL of other procedures.

General rule: If you can say that the "procedure does this AND it does that" then the procedure may be doing more than one function.

If you can say that the "procedure does this OR it does that" then the procedure may be doing more than one function.

Cohesion and Coupling

Topic: SUMMARY

OOP AND FUNCTIONAL COHESION

1. **You should try to make methods perform a single function.**
   Methods typically consist of:
   a. **Sub-procedures**
   b. **Function procedures.**

2. **OOP programmers have gotten very sloppy concerning functional cohesion.**

3. **Classes in OOP typically consist of several methods that should perform a single function.**

4. **The heart-and-soul of Structured Design**
   a. **COHESION**
   b. **COUPLING (Covered in subsequent slides.)**

Before the concepts of structural design were created by Larry Constantine, programmers wrote code anyway they want.

<u>Larry Constantine is the FATHER of structured design</u>.

Cohesion and Coupling

Topic: SUMMARY

SUMMARY

1. **As we move up the scale towards functional cohesion, several effects should be observed:**
   a. **The procedures become more independent.**
   b. **The procedures become easier to change.**
   c. **The procedures become more useful because they can be used again in other programs, as well as in the original program.**
   d. **There is no length limit for a single function. A procedure should be as long as necessary to complete a single function. However, if a procedure becomes longer than a single page, take a second look at the procedure. It may be doing more than one function.**

2. **IF A PROCEDURE IS NOT AT ONE OF THE SIX LOWER LEVELS OF COHESION, THEN IT IS AT THE FUNCTIONAL LEVEL. THAT IS, IT IS DOING A SINGLE FUNCTION.**

Cohesion and Coupling

Topic: Coupling (Module-Level Variables)

Deadly: Module-Level Variables

Module-Level Variables

Module-Level Variables

Module-Level Variables

1. **As you know, these variables can be used in any procedure within a form or class in which they are declared, and if declared public in a module they can be used in any procedure within the program. If declared Public, we call the variable a Global-Level variable.**

2. **In functional cohesion, we want each procedure to function independently of others. In other words, we want these procedures to be loosely coupled. We don't want them to the tightly coupled by sharing variable locations (module-level variables).**

3. **Using module-level variables produces the most severe form of coupling, because the side-effects are unpredictable and the sources of the side-effect are very difficult to isolate.**

4. **Are similar to Global variable. In VB.NET, Module-Level variables are available anywhere in a form's code. Global variables, however, are worse because they are available anywhere in the program.**

Cohesion and Coupling

Topic: Coupling (Module-Level Variables)

Deadly: Module-Level Variables

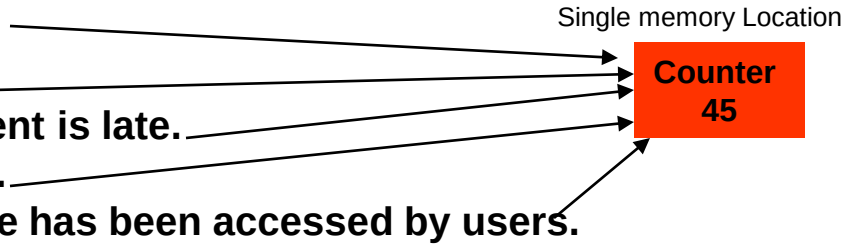
Module-Level Variables (Global Variables)

Module-Level Variables (Global Variables)

Module-Level Variables

5. EXAMPLE:

1. **Assume you have declared a variable called Counter as a module-level variable and that the program has 10,000 procedures.**
   Let's further assume that the program is being developed by 5 separate groups of programmers. Since Counter is a popular name for a variable, it is likely that it will be used by all of these groups, but for different purposes.
   <u>Purposes for Counter</u>:
   a. **Counts pages while printing a report.**
   b. **Counts the number of records in files**
   c. **Counts the number of days the payment is late.**
   d. **Counts the number of sick-days used.**
   e. **Counts the number of times a web-site has been accessed by users.**



Cohesion and Coupling

Topic: Coupling (Module-Level Variables)

Deadly: Module-Level Variables

Module-Level Variables (Global Variables)

Module-Level Variables (Global Variables)

Module-Level Variables

6. **I have allowed you to use module-level variables because the book uses them extensively, and I tried to keep confusion to a minimum in this course.**
7. **Typically, we have used module-level variables to keep a memory location active when we have repeatedly called a module. As you know, local variable locations are lost when we exit the procedure in which the local variable was declared.**

8. **HOW TO AVOID MODULE-LEVEL VARIABLES:**
   a. **<u>Static</u> Local Variables**
      (1) **These variables retain their value as long as the form is loaded, which is generally the life of the project, rather than being reinitialized for each call to the procedure in which they are declared.**
      (2) **In other words, if we declare a variable static within a procedure, the variable will retain its values during repeated calls to the procedure.**

Cohesion and Coupling

Topic: Coupling (Module-Level Variables)

Deadly: Module-Level Variables

Module-Level Variables
(Global Variables)

Module-Level Variables
(Global Variables)

Module-Level Variables

(3) How to declare a static variable:

```
Static intCounter As Integer          'Counts the number of records in a file.
```

(4) Static declarations can only be used in procedures.

TYPES OF COUPLING

1. **COMMON COUPLING**
   a. **Using module-level variables is an example of this type of coupling.**
   b. **It occurs when two or more procedures share access to a common pool of data, such as an array, or single variables. What would happen, if we changed a module-level variable from a decimal value to an integer value in our program? Assume the value is being used by 15 different procedures. (Answer: We might need to change our code in at least 5 or 6 of these procedures. In other words, these modules are tightly coupled by the use of a single variable.)**
   c. **Do you see how "initialization" procedures contribute to common coupling?**

Cohesion and Coupling

Topic: Coupling (Module-Level Variables)

Deadly: Module-Level Variables

Control coupling

Stamp Coupling

TYPES OF COUPLING

2. **CONTROL COUPLING**
   a. This occurs when "control" flags are passed as arguments to other procedures in order to control the execution of code in the receiving procedure. What level of cohesion would the called procedure have? Answer: Logical.
   b. Changes in one procedure that has the flag can propagate changes in other procedures. Thus, control coupling makes it difficult for modules to function independently.

3. **STAMP COUPLING**
   a. Happens when unnecessary data items are passed in arguments and procedures. Don't pass the "whole" record when only one field in that record is need by the called procedure.
   b. EXAMPLE: Ten procedures receive a record with 30 fields, and each module processes three fields. If just one field changes in size or data type, then all ten procedures will require code changes.

Attempt to keep your Module-Level variables and Global-Level variables to a minimum.

Use Static variables and pass values from one procedure to another using arguments and parameters.