modules will use each other’s services.

MODULES: Each module solves a separate problem.

Fundamental idea for logical organization of software: Organize it

- Not acceptable for CS1333 C++ projects.
- For larger programs … is unworkable
- Achieves Consistency.

Single Header FILE (Str. 9.3.1)

consistent” (Str. 9.2.1)

preprocessor [to the compiler and later linked together must be

etc.,” must be consistent … the source code submitted] By the

types in all declarations of the same object, function, class,"
USES this module. (Guarantees consistency.)

Implementation files AND in every other module that

- The header is #include'd in the module's code

  (what it provides)

- Each module has its own interface that specifies its interface

- Plus make rebuilds efficient.

- Enables physical organization to reflect logical organization

  - Multiple Header Files (Section 9.3.2)
Aggregate Definitions: char hello[ ] = "Hello";

Simple Static Data Definitions: int 6topcount;

{ return n; } ... Function Definitions: int mystri( ) {

Header files MUST NOT contain:

Comments that DOCUMENT the interface.

Constant definitions: const int MAXLINLEN = 80;

Static Data declarations: extern int 6topcount;

Function declarations: extern int mystri( const char *);

Type definitions: struct Point int x, y;

CS1233 Software Construction Standards Header files may contain:
ONE module this body implements.

- Data or function DEFINITIONS that do not belong to the header file.
- Data type DEFINITIONS for interfaces (that should be in the body file) MUST NOT CONTAIN:

module: int &topcount = 0; char hello[] = "hello";

- Definitions of any static, global variables belonging to this module.
- The DEFINITIONS of functions implemented by this module.

USES.
#include<every module this body

- #include for the INTERFACE of EVERY module this body implements.

Body files MUST CONTAIN:

-
module's implementation.

• Constant or other definitions/declarations private to this

Body files MAY CONTAIN: