Software Design

File Management

Multi-File Programs

- We need a way to split our code into many separate files so that we can partition our code
  - We often are given code libraries from other developers or companies
  - It can also help to put groups of related functions into a file
- bmplib.h has prototypes for functions to read, write, and show .BMP files as well as constant declarations
- bmplib.cpp has the implementation of each function
- gradient.cpp has the main application code
  - It #include's the .h file so as to have prototypes and constants available

Key Idea: The *.h* file tells you *what* library functions are available; The *.cpp* file tells you *how* it does it

Multi-File Compilation

- Three techniques to compile multiple files into a single application
  - Use 'make' with a 'Makefile' script
    - We will provide you a 'Makefile' whenever possible and it contains directions for how to compile all the files into a single program
    - To use it just type 'make' at the command prompt
  - Compile all the .cpp files together like:

```
$ g++ -g -o gradient  gradient.cpp bmplib.cpp
```

    - Note: NEVER compile .h files

Multi-File Compilation (cont.)

- Three techniques to compile multiple files into a single application
  - Compile each .cpp files separately into an "object file" (w/ the –c option) and then link them altogether into one program:

```
$ g++ -c bmplib.cpp -o bmplib.o
$ g++ -c demo.cpp -o demo.o
$ g++ -g demo.o bmplib.o -o demo
```

  - The first two commands produce .o (object) files which are non-executable files of 1s and 0s representing the code
  - The last command produces an executable program by putting all the .o files together
  - It is approach 'Makefiles' use and the way most real programs are compiled