Week 9 Lecture 1

Modules

Moving into Team Work

- You will be building a database program
  - It will read and write a database
- Team based development will help you learn communications skills
- Today we discuss modular design and development

What is Modularity?

- *Modules* are pieces of a program that interacts with the rest of the program in a controlled fashion.
- From outside the module you can access only selected functions and variable.
- The module provides a *controlled interface*.

Functions are modules

- Functions provide an example of modularity
  - The function declaration indicates the inputs (parameters) and outputs (return value)
  - What happens inside the function cannot be affected from outside.
  - E.g.: Bubble sort and Merge sort both take and array and sort it, but they do it in different ways.

Compilation Units are Modules

- Compilation unit: .h + .c file
  - e.g. calc.c + calc.h
- The .h file defines the controlled interface
  - Only the functions in the .h file can be used outside the module.
  - Only the variables defined as external can be used outside the module.

Libraries are Modules

- Libraries provide a set of related functions
  - Libraries may consist of multiple compilation modules.
- How the functions interact is not visible from outside the library.
- We simply use their behavior.
- We use a .h file to specify the syntax.

Developing Modules

- What is the collection of functionality provided by the module?
- What are the inputs to the module?
  - Usually function calls
- What are the outputs to the module?
  - Usually function returns
- What behavior is private to the module?
  - Functions that cannot be called from outside

Modules in C

- Header files (.h) specify which functions can be used from outside the module.
  - The specify the parameters and the return values.
- Functions that are not specified in the header files cannot be used outside the module.

Calculator Example

- In the calculator, we have only one module.
  - The module is used by main to give the user access to the calculator
  - It is used by test to make sure that all of the functions pass be basic tests.
- If we were going to develop it further we might want to have multiple modules.
  - e.g. int operations, floating point operations, array operations ...

Module Development

- Design
  - Before programming, we sketch out the modules
  - This helps us focus our work on parts of the program without attending to all of it.
- Refactoring
  - After programming, we revisit the modules to make sure they make sense
  - Will people coming after us understand how the program fits together.

Design

- We think in terms of things and actions
  - E.g. Nouns and verbs
- First step in analysis is what are the things in the program; what do they do

Database: CRUD

- C: Create – Add a new record
- R: Read – Read a record
- U: Update – Update (or add) a record
- D: Delete – Delete a record

Database Interface (1)

- int add(record);
- record find(index);
- int modify(record);
- int delete(index);

Database Interface (2)

- db: sequence of student records
- record:
  - s_no: unique number
  - name: string
  - grade: int

Database Interface (3)

- int add(stu)
  - Adds student_rec stu to database; returns 1
  - If there is already one with s_no, returns 0
- stu find(s_no)
  - returns the student record with serial number, s_no
  - Returns error record if not found. (Cannot return 0)
- int modify(stu):
  - alters the record with stu.sno to match the record passed.
  - If there is no such record a new one is added to the database.
  - Returns 0 on failure
- int delete(s_no)
  - removes the record with s_no.
  - Returns 0 on failure