Topic 1

Data Structure in Programming

Why is data structure (implementation) so important?

資料結構與程式設計

Data Structure and Programming

Sep, 2011

What is Data Structure?

- In programming, the term *data structure* refers to a scheme for organizing related pieces of information.
- The basic types of data structures include:
  - files
  - lists
  - arrays
  - records
  - trees
  - tables
- Each of these basic structures has many variations and allows different operations to be performed on the data.

http://www.webopedia.com

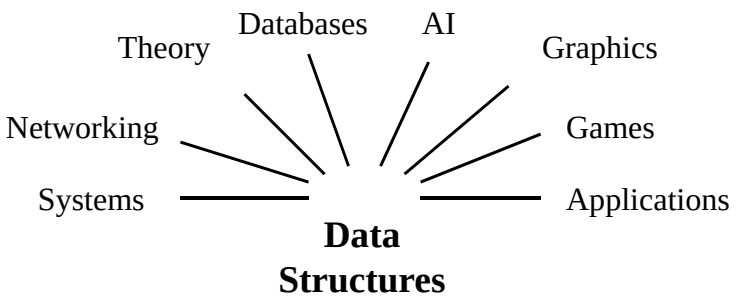
What is Data Structure?

- A data structure is a specialized format for organizing and storing data
  - General data structure types include the array, the file, the record, the table, the tree, and so on.
- Any data structure is designed to organize data to suit a specific purpose so that it can be accessed and worked with in appropriate ways.
- In computer programming, a data structure may be selected or designed to store data for the purpose of working on it with various algorithms.

http://www.whatis.com

Why study data structures?

Clever ways to organize information in order to enable efficient computation



Prof. Yen, "Data Structure", class slide

Why do YOU study Data Structure?

- Any application not related to programming?
- When you designed a program, did you keep the concept of data structure in mind?
  - Or just thinking about “if...else...”, “for...”, “while...”, “switch...”, etc
  - Or the program you wrote was just another “write-it-once-and-never-see-it-again” piece of #@$^...? (you think: why should I care?)

Structure-less Programs

[A C++/C example code]

- What are the problems you see?
  1. Deep pointer’s access
     - “a->b->c->d[i]->e = kkk;”
  2. Function with many arguments
     - “void f(int a, char* b, int *c, bool d, int& f, vector<int>& g, …);”
  3. Complicated “if()” condition
     - “if (a && b || c && d || e…)”
     - ➔
     - Difficult to write and understand
     - Prone to err

Structure-less Programs

- What are the problems you see?
  4. Global variables
     - Data contamination (unexpected bug)
     - Memory bugs/leaks
  5. Fixed size array
     - Not flexible and extensible
     - May have unexpected exception (very difficult to debug)
  6. Duplicated similar codes
     - Tend to have unexpected bugs
     - Difficult to maintain
  7. Very long “switch”
     - Slow in compilation; Performance degradation
     - Difficult to read

Do these sound familiar?

(Do you have the similar problems?)

Everybody is talking about Object Oriented Programming (OOP)

but

What is an object in a program?

Why do we need "class" (encapsulation)?

The Object in an Object Oriented Program

◆ An Object is:

➔ a structure (or container) to

- hold the value fields, and
- define the operations

of the data

➔ Data Structure

◆ A "class" is to:

➔ define the same type of objects in order to share the same:

- Data organization ➔ data members
- Data operations ➔ member functions

Program Design without Structured Objects

1. Idea, spec, flowchart
2. Focus on the control flow
   - If this, else that…
   - What if…
3. Create variables to implement the control flow
4. Oops, some variables must be seen across different functions
   - Too much work to pass as function arguments
   - Create global variables
5. Revise the control flow ➔ need to add more branches
   - Change “else” to “else if…else”
   - More and more complex conditions
   - Similar codes get duplicated
6. Ah, bugs… fix one piece of code
   - But forget to fix some similar parts of the program
   - Bugs get transformed into different bugs
7. Create workaround, hack, whatsoever…

An Object Oriented Program Should be…

1. Idea, spec, flowchart
2. Architect the program as a connection of objects
   - Server - client
   - Manager - manager
   - Node - node
   - Stage - stage, etc

   ➔ You are the BOSS, who are your employees/managers?
3. What are the data structures of these objects
   - Define class and class data fields (share / reuse)
4. What are the possible operations of the class objects to their neighboring objects
   - Define class member functions (clear interface)
5. Implement the main functions to connect the program flow
   - Declare class objects and utilize their member functions
   - Create more member functions if necessary

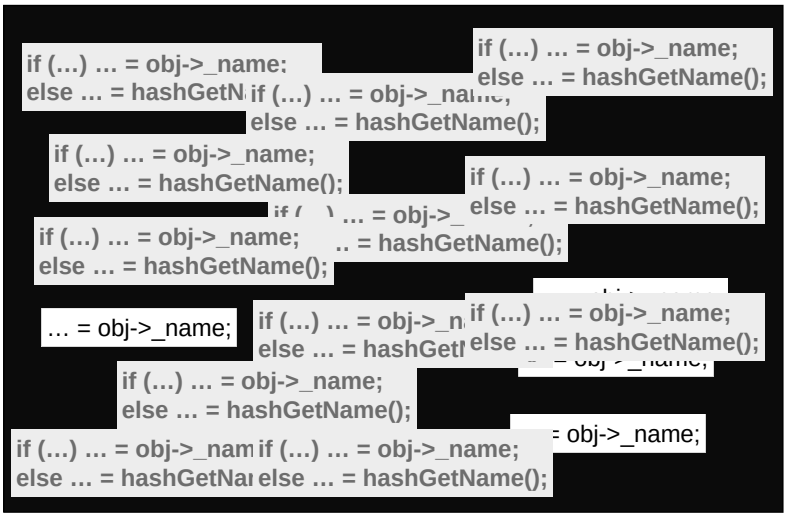
When you first learned C++, did you ever think that method like ---

```
class MyClass
{
        const string& getName() const
        { return _name; }
};
```

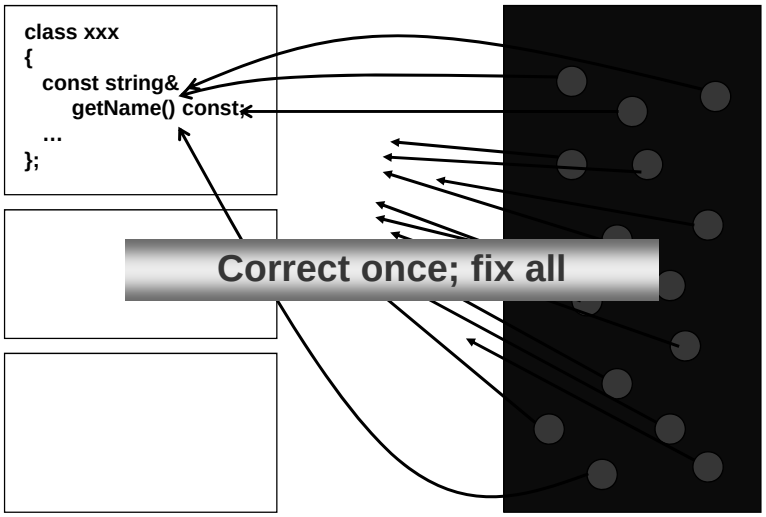
is too trivial and even sounds stupid?

(Why not use “obj->_name” directly?)

If you use “obj->_name” directly…



But if you use “obj->getName()”…



Member Functions in a Class

- Therefore, create member functions for data manipulation
  - Maximize the sharing between different objects
- Use “private/protected” to ensure the encapsulation
- Use “const” to make sure some methods are “read only”

(More to cover later…)

Container Classes

- A class that hold objects of the same data type (class) in some particular way

1. Linear (sequence)
   - List, array
2. Look-up (associative)
   - Set, map, hash
3. Topological
   - Tree, diagram, graph, BDD

Standard Template Library (STL)

- First drafted by Alexander Stepanov and Meng Lee of HP in 1992
  - Became IEEE standard in 1994
- A C++ library of container classes, algorithms, and iterators
  - Provides many of the basic algorithms and data structures of computer science
- The STL is a *generic* library
  - Platform independent
  - Its components are heavily parameterized
  - Almost every component in the STL is a template.
- An useful reference: http://www.sgi.com/tech/stl/
- More to cover in lecture #4

Since STL is pretty good,
why do we implement
our own DS classes here?

Using STL or implement on your own?

- For program prototyping, using STL is a good choice.
- For production tools, you may consider implementing container classes on your own.
  - Customized classes
  - Special functions for better efficiency
  - (Sometimes) Portability issue

Conclusions

- A structured (object oriented) program must have a well architected data structure
- The knowledge of data structure is practically useful when applied in programming
- Standard Template Library (STL) provides an exemplar implementation of various container, algorithm, and functional object classes

Coming up next…

- Programming on Linux workstations
  - Please try to install Linux on your computer or find a machine that runs linux (e.g. PC Room)
- Download: (http://www.linux.com/directory/Distributions/Desktop)
  1. Fedora project sponsored by RedHat http://fedora.redhat.com/
  2. Ubuntu Linux: http://www.ubuntu.com/
  3. Debian Linux: http://www.debian.org/
  4. Suse Linux: http://www.novell.com/linux/, http://www.opensuse.org/
  5. Mandriva (Mandrake) Linux : http://www.mandriva.com/
  6. Gentoo Linux: http://www.gentoo.org/
- References
  - Official Linux home: http://www.linux.org/
  - Linux Taiwan: http://www.linux.org.tw/
    - Not very active, but lots of useful links
  - A nice Linux portal site (Chinese): 鳥哥的 Linux 私房菜館 (http://linux.vbird.org/)

Coming up next…

- C++ advanced features review
  - Please pick up the C++ basics before the meeting
- References
  - "C++ How to Program 6th edition", by Deitel and Deitel
  - MSDN C++ library
  - http://www.cplusplus.com/doc/tutorial/
  - http://www.cppreference.com/
  - SGI STL guide: http://www.sgi.com/tech/stl/

One Last Thing...

- Please pay attention to the class website and BBS for class announcements...
- Your comments and questions are very welcome!!