Topic 0

Class Introduction

資料結構與程式設計
Data Structure and Programming

09/16/2015

Class Information

- Class Website
  - https://ceiba.ntu.edu.tw/1041_DSnP
- Discussion board
  - FB → NTU_DSnP
  - To get in, (1) go to FB/NTU_DSnP to apply, and (2) go to https://goo.gl/Ua8k82 to leave your name
- My office:
  - EE building II - 444
  - (Tel) 3366-3644
  - (e-mail) cyhuang@ntu.edu.tw
  - Office hour: stop by or by e-mail appointment
- Class TA(s)
  - TBD

Class Information

- Required textbook: none
- Suggested reading
  - Class slides and source codes
    - Download from the Ceiba website
  - Any of your Data Structure and C++ programming textbooks
- Highly recommended (DO THEM ASAP)
  - Review C++
  - Get access to and familiar with Linux workstations

Grading (May subject to change)

- Homework 70%
- Final project 30%
- Bonus TBD

The final grades are subject to linear adjustment. Instructor will determine the average and standard deviation

授權碼

- 本課程這學期改為二類加選，並且不開放初選
- 欲修習本課程者，請在
  ，　　，
  -mail　　。
- 作業一題目載點：https://goo.gl/Do9Flp
- [Note] 我們有強大的抓抄襲程式，會在事後檢查是否有抄襲的現象，請勿以身試法，會有嚴重的後果。

修課人數限制

- 教室容量
- 教學
- 學生的
  - 去年加退選完後仍有 210人，但期中過後停修人數達 47 人
  - 22.4%, 跟以往差不多(但稍高)，不過還是希望這些人能夠及早認清自己無法負荷這門課的現實
- 解決辦法
  - 訂定修課人數上限？
  - 先做作業一，讓同學們體驗/回憶一下寫程式的樂(ㄎㄨㄥˋ)趣(ㄑㄨˇ)

What HW#1 tells you…

- C++ 很煩!! 為什麼不直接學簡單又漂亮的 Python, 或者是很潮的 Javascript 呢？
  - 身為工程學系的學生，我們除了要能夠很快的把事情做好之外，還要有把東西 optimize 10X 以上的能力
  - 不過現實是，看看你們 HW#1 的 code, 可以想像如果讓它再長大十倍、百倍、千倍… 會變成什麼樣子嗎？

關於資料結構與程式設計，這門課想要傳遞的觀念是…

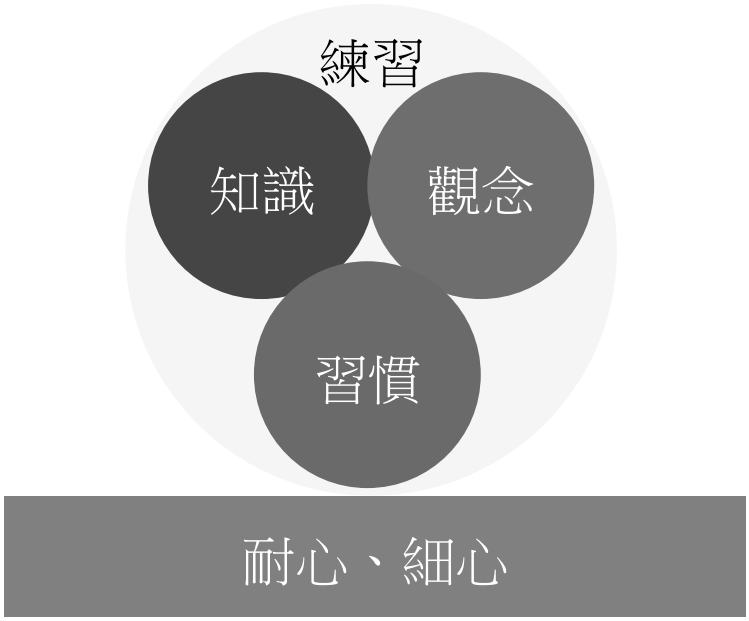
- 語言的嚴謹性
  - 電腦語言 vs. 人類語言

```
bool operator() (const Data& d1, const Data& d2) {
   return cmp(d1, d2, 0); }
bool cmp(const Data& d1, const Data& d2, size_t i) {
   if ( d1[_sortOrder[i]] < d2[_sortOrder[i]] ) { return 1; }
   else if ( d1[_sortOrder[i]] == d2[_sortOrder[i]] ) {
      if(i < _sortOrder.size()-1 ) {
         i++; cmp(d1, d2, i);
      } else {
         if(d1[_sortOrder[i]] < d2[_sortOrder[i]]) return 1;
         else return 0;
      }
   }
   else return 0;
}
-------------------------------------------------------------
bool operator() (const Data& d1, const Data& d2){
   for (int i=0; i < _sortOrder.size(); i++){
      int idx = _sortOrder[i];
      if (d1[idx] != d2[idx]) return (d1[idx] < d2[idx]);

   return false;
}
```


關於資料結構與程式設計，這門課想要傳遞的觀念是...

- 語言的嚴謹性
  - 電腦語言 vs. 人類語言
- 程式的架構需要設計
  - 你寫的程式除了要讓電腦看得懂之外，讓人類(尤其是自己)看得懂更是重要
- 資料結構的重要性
  - 試想你有一堆等待被運算或是分析的資料，如何確定某筆資料存在？如何確認所有資料被運算過一次？如何有效率的增加或是刪除資料？
- 資料
  - 希望大家可以將「資料結構」與「程式設計」融會貫通



為什麼是 C++? 為什麼不直接學 Web/APP 就好?

- 寫程式需要的三種能力
  - 熟習文法：不同的語言，其實文法都差不多
  - 建立觀念、培養直覺：Write the code right!
  - 數理邏輯觀念：非單從寫程式中去培養
- 大部分的網路程式　　等等　入門的技術門檻

  其實都不高　但要寫得快又好　而且能夠，需要的就是良好的寫程式的 “sense”、“concept”，以及“習慣”，還有正確的 “optimization” 的概念，這些都必須從比較低階、複雜的語言　即來學，才會學得透徹

為什麼是 C++? 為什麼不直接學 Web/APP 就好?

- 我們有許多優秀的　　人才，但卻沒有一個像樣的　　產業，
- PC 時代，“軟體” 為主要獲利之 “商品”
  - 我們靠生產 Intel CPU 周邊的 IC 與零組件成就了 ‘95 ~ ‘05 的高科技奇蹟 (硬體出口產業)
- 後 PC 時代，”廣告“、”服務” 為主要的獲利方式
  - Yahoo, Google, Amazon, e-Bay, FaceBook 的崛起，我們的定位在哪裡?
  - 台灣市場太小，其實是很嚴重的問題
  - 「廣告」、「服務」可以外銷嗎？
  - APP 是個產業嗎?

「把 C++ 學好之後，再去學其它語言相對的會很容易」...

- Yes, 大部份電腦語言的語法、架構，都很相近
- 即使是不同的地方，如果你從語言設計的觀念上去了解為什麼要這樣，你就可以很容易的融會貫通
- 希望你從這門課學到的對於「程式語言」的觀念與態度，可以延續到你以後對於其他程式語言的學習

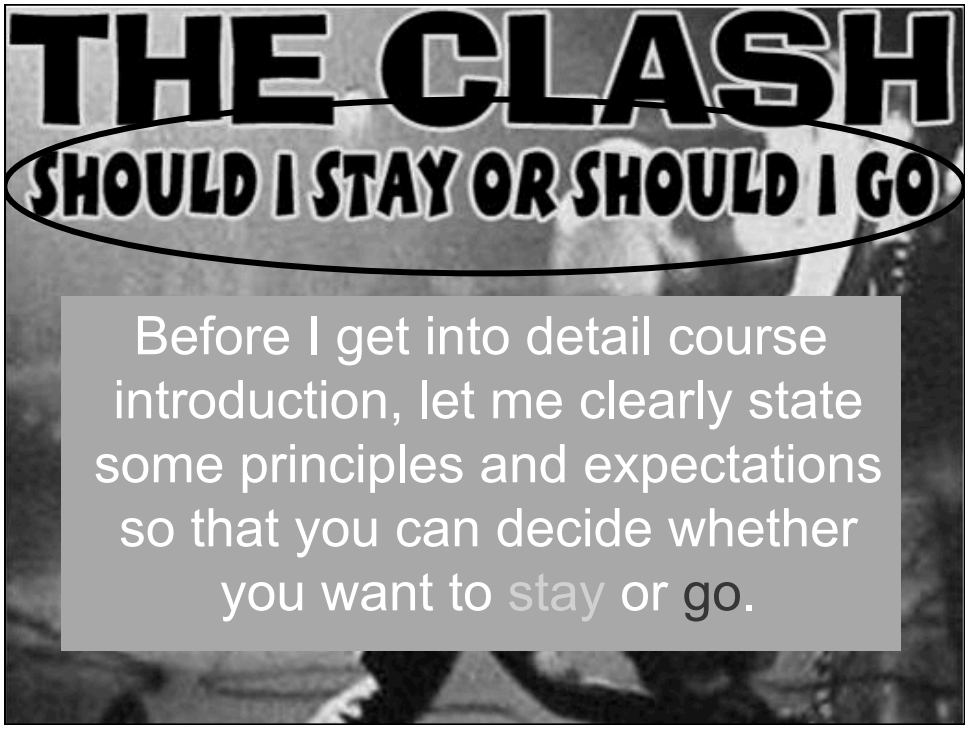
聽說這門課很操，是真的嗎?

- 不要懷疑 根據多次的問卷統計 同學們覺得
  這門課的 大約 學分 每兩個星期
  要花 以上 在作業上

因為我覺得台大的學生根本修太多主科了!!

你可以去修很多其他領域的課 跨領域學習
增廣見聞

但你如果想要把一些專業科目學好 我覺得一學期應該修兩三門就好 然後每門課九學分 誤



雖然這門課很操...

- 但好處是沒有期中　期末考　不用去　教科書或是消習題
  - 不過有期末 project
  - 而且要學會自己找參考資料
- 所以如果你還要忙社團或是要參加什麼隊的，或是其他的課很重，請搞清楚你的　　，切莫**始亂終棄**!!
- 我的目標是：同學們在修了這門課之後除了對於資料結構能有正確的觀念之外，起碼要有自行 handle 1000 行程式碼的信心!

寫 1000 行程式，很難嗎？

- 當然，要看是寫什麼
- 如果是有功能性，可以解決一些問題的程式，1000 行的程式的確已經有相當的複雜度

  重點是 --- structured design and thinking!
- 人腦的思考複雜度有一定的限制，如果有超過一定數量的元素要一起考量，就會無法掌握
- 但階層式的、歸納式、模組化的思考，有助於化繁為簡，讓程式在可以 handle 的範圍內被最佳化

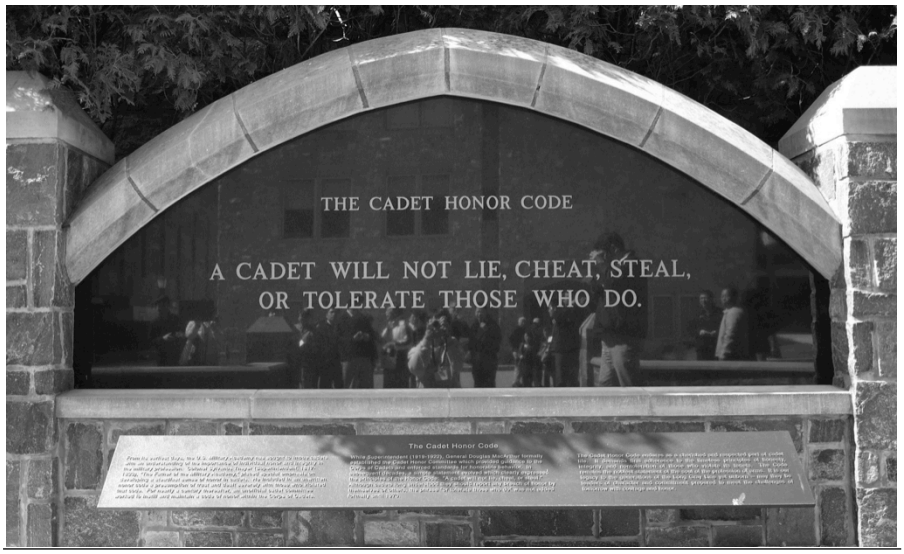
寫 1000 行程式，很難嗎？

- 10+ 行的程式
  → ， ，
- 100+ 行的程式
  → ，
  用力寫下去，大家都做得到
- 1000+ 行的程式
  → 如果有能力將 100+ 行的程式模組化，那 1000+ 行的程式要 handle 的只是最上層的 control flow，何難之有？
- 10000+ 行的程式
  → s/1000/10000, s/100/1000, repeat this!

我是個寫程式的小嫩咖，我有辦法修這門課嗎？

- 原則上絕大部分的人在你們這個年紀都是寫程式的小嫩咖，所以我想沒有問題。
- 重點還是要能有每兩個星期交一個作業，連續14周，然後再加上一個期末專題的 "**commitment**"
  - 再強調一次，要考量現實，不要輕易相信自己的意志力可以戰勝一切!
- Commitment 從何而來?
  - 首先，請確定 "把程式學好" 對你的重要性
  - 再來，請確定自己可以接受 "學習比成績重要"
  - 還有，請發誓自己 "寧願被當，也不會抄襲"

DSnP Honor Code



DSnP Honor Code

- 上課要專心，寧願翹課也不要來課堂做別的事
- 作業不抄襲，寧願被當也要從頭到尾自己寫

DSnP Honor Code

- **上課要專心，寧願翹課也不要來課堂做別的事**
  - 不點名，學生有自行決定如何學習的自由
  - 但是如果你是來教室上臉書、打電動、睡覺補眠，那對我是種不尊重，對同學也有不好的影響。
  - 如果你覺得上課的內容你都已經會了，就請不要貪圖這個學分，把座位讓給別人，或者，你也可以不用來上課。
  - 不過，上課用電腦寫寫小程式，驗證上課所學，或者是上網查詢相關資料，是 OK 的

DSnP Honor Code --- 關於抄襲

- Definition:
  code, copy/paste, code
  變成自己的作業的一部份
  - 歡迎互相討論，甚至拿別人的 code 來 study 也不會/無法禁止 (雖然這樣並不好)，但最後一定要自己獨立的寫。
- **我們有強大的抓抄襲的程式 會對所有的作業以及之前學長姊的作業去做比對 如果沒有抄襲 相似度都會很低，但如果有抄襲，不管你是改變數名稱，還是換 statements 順序... 等等，我們都可以很容易抓出來，所以請勿抱著苟且的想法。**
  - 以我們的作業複雜度而言，只要是自己寫的，一定一眼就可以看出跟抄襲的不同。
- **凡抄襲者不論多寡、理由，除該次作業 0 分之外，學期成績一律再扣 20 分 (調分後)**

一些前車之鑑...

老師您好

對不起老師...我對之前的作業有些抄襲or參考的疑慮 睡覺都睡不好,

所以還是先寄信詢問(自首)了...雖然我自認程度很輕微拉(爬過ptt對於抄襲的定義, 覺得還好??)

自從在寫HW4快到尾聲時在網路上搜到疑似老師2012年DSnP的解答...相信老師都知道, 因為實在太好搜了= =
之後我作業不懂的就會去看老師的code...

…

一些前車之鑑...

我覺得網路上那2012版的解答, 雖然部分不夠完美(EX:HW6 gate定義覺得可以再刪減),
但他對於我就像潘朵拉盒子, 一載下來看, 就是罪惡...可是當作業不懂時, 他卻是最好的來源。

……
如果真的被處罰也很甘願, 因為是我自己程式能力不足。

儘管如此...還是拜託老師開恩...即使有2012年的code, 我每次作業也是會花20小時up, 覺得努力沒有比別人少...
也常常跟同學討論code, 當然都是based on對老師code的理解, 再加上自己的詮釋。

最後, 謝謝老師看完我很長的解釋文...感謝老師開DSnP, 我學到的真的很多!!

一些前車之鑑...

(From Ric)
很遺憾的, 你沒有在學期中我一再強調抄襲的定義的時候就主動承認, 而前幾天問你的時候你也還是無法就直接承認你就是有抄襲。

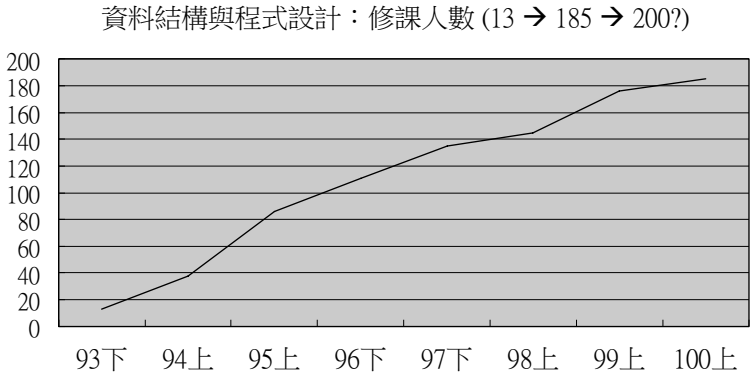
因此, 我只好按照學期初所說的規定, 將你該次的作業算成零分, 然後學期成績在調分後在扣 20 分, 因此, 你的成績將會變成 52 分(F). 希望你可以接受這樣的處置。

(Reply)
這次的經驗已經讓我飽嚐煎熬與苦頭
以後不只不敢再犯大概還會便成陰影警惕很久
這幾天也想了很多, 那些文過飾非的話大概不只是想要粉飾太平, 一部分也是因為內心本來就有所愧疚想要說服自己吧

正如教授所說:不只沒有遵守規定, 我還欠缺更多勇於認錯承擔的態度
謝謝教授, 還願意耗費時間跟我說這麼多。

有教無類? 教學品質?

◆ Well, as you can see, the class was overbooked.



Should I stay or should I go?

- After taking the class, somebody liked it, but somebody hated it.
- 去年 210 個選課的同學，最後 47 個停修，7 個被當掉

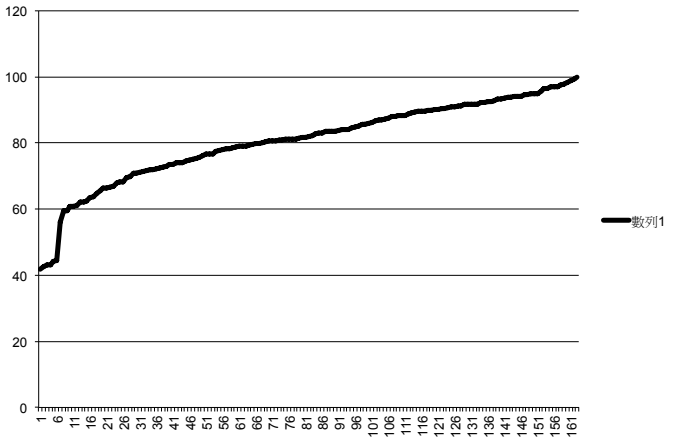
  // 當掉/停修/修課人數
  - 電機大二: 1/3/37
  - 電機大三: 2/29/117
  - 電機大四: 1/4/20
  - 外系 (物理、數學、心理、機械、化工、工科海洋、生機、資工): 1/7/28
  - 研究所 (電機、電子、生醫、物理、生機、資管、資工): 2/4/8

Some statistics about the grades

	103-1	102-1	101-1	100-1	99-1	98-1
外系	79 (21/27)	76 (10/13)	70 (3/4)	80 (6/8)	82 (5/7)	61 (5/5)
電二	82 (33/37)	86 (21/26)	78 (36/38)	83 (29/32)	83 (27/27)	--- (0/1)
電三	82(86/117)	81 (59/65)	82 (80/89)	79 (98/110)	84 (101/106)	88 (74/83)
電四	77 (15/20)	78 (8/11)	73 (7/8)	82 (31/31)	86 (17/22)	82 (27/38)
資工	80 (1/1)	NA	NA	92 (2/2)	84 (7/11)	87 (11/15)
研究所	62 (2/8)	57 (4/7)	86 (5/6)	63 (1/2)	52 (2/3)	58 (3/3)

Some statistics about the grades

- Average: 80.7
- A+: 48
  A: 21
  A-: 30
  B+: 14
  B: 13
  B-: 11
  C+: 5
  C: 6
  C-: 8
  F: 7



"Should I stay or should I go?"

- Please check on your own:

1. Do I have the eager to improve my programming skill?
   - 光有"希望"是不夠的，要有"渴望"才行。
2. Am I willing to spend more than 10 hours per week on the homework?
   - 獨力完成，不抄襲，也不要當寄生蟲。
3. Do I agree that "learning" is the most important thing in class?
   - 心態上要能接受"學習"比"分數"重要。

FAQs & Suggestions

- Can I take this class as I am not an NTUEE student?
  - You are also welcome, but you are advised to find someone to study and discuss together.
- Can I sit in this class?
  - Well, technically there is no restriction on sitting-in.
  - However, since the number of students is way too high, please leave the seats to the students who take this class.
- Is this the last time I offer this class?
  - Nobody knows. But I will try to sign in this class as long as it is possible.
  - Please note that other professors also offer this class in different semesters.
- My only request to you: 做人要甘願!!
  - If you decide to stay in this class, you need to know that this is a heavy class, and this is YOUR decision.
  - Don't blame on me if you find it too heavy-loaded!

歡喜修課，甘願承受

- 說實在的，DSnP 是 NTU(EE) 的奇蹟!
  - 需要大家共同的珍惜
- 非誠勿試，please!!

Course Outline

Part 1: Introduction

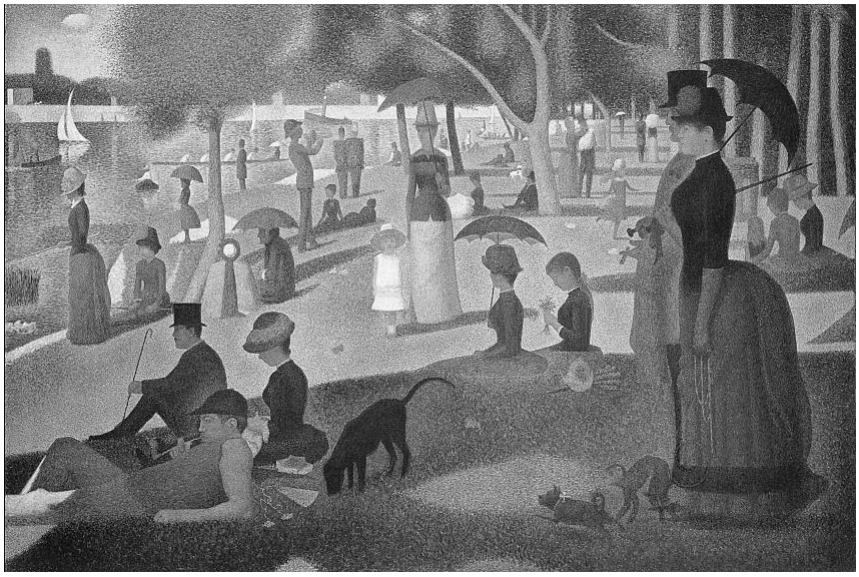
0. Class Introduction
1. Data Structure in Programming

   Why is data structure (implementation) so important?

1. Data Structure in Programming: *Why is data structure (implementation) so important?*

- Why do you learn DS?
  - When will you use it in your daily life, if you don't apply it in your programs…?
- "Programming is an art; DS is the spirit of the art."
  - If you know how to cleverly utilize DS in your codes, you will definitely produce an elegant program.
  - Masterpiece? 99% perseverance and 1% talent
- "Writing program is an ego thing,

  while writing a SW tool/framework needs cooperation"



Georges Seurat, *"A Sunday Afternoon on the Island of La Grande Jatte"*, 1884-1886

Data Structure in Programming

- As we will see, "programming" is nothing more than "storing" and "operating" data.
- "Data structure", in general, includes all types of "structured storage" in which data can be "operated" in various ways.
- "Object oriented programming (OOP)" teaches you how to use "structured data type" (e.g. *class*) to write a good program.

How to be a good programmer?

- My observation
  - Achievements in ACM or programming contests do NOT necessarily imply good programming skill.
  - It just means that you are smart, or at most, good in math and logic.
- Our objective here is not just to be a good programmer, but a good program **designer**.
  - Has the capability to plan, architect, and manage a large scaled program.

Course Outline

Part 1: Introduction

0. Class Introduction
1. Data Structure in Programming

   Why is data structure (implementation) so important?
2. Programming on Linux Workstations

   A peek in the real engineering world

2. Programming on Linux Workstations:
peek in the real engineering world

- Why Linux? Why not M$ Windows?
- History of Linux OS
- Basic survival guide on Linux
- Writing programs on Linux
  - Shell commands
  - Compiler
  - Makefile
  - Debugger

Overview of this course

Part 1: Introduction

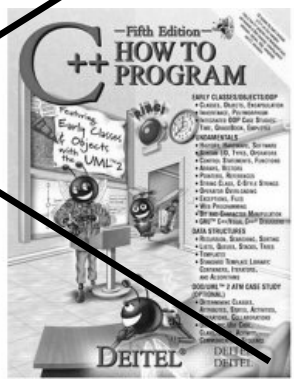
Part 2: Polishing Your Programming Skills

Part 3: Data Structure Revisited

Part 4: Putting What You Learn Together

3. C++ Advanced Features Review: *When can/should I use them?*

- Object, pointer, reference
- Const, static, extern, type cast
- Namespace
- Constructor, destructor
- #include, #define, #ifdef
- Enum, union, bit slicing
- Public, private, friend
- Inheritance, virtual, polymorphism
- Operator overload
- Template
- Functional object
- Stream classes
- String
- Exception handling



3. C++ Advanced Features Review: *When can/should I use them?*

- Understanding “variables”
  - Object, pointer, reference
  - Const, static, extern, type cast
  - #define, typedef
  - Namespace
- Understanding “classes”
  - Constructor, destructor
  - Enum, union, bit slicing
  - Public, private, friend

3. C++ Advanced Features Review: When can/should I use them?

- Understanding “overloading”
  - Function & operator overloading
  - Function & class template
- Understanding “polymorphism”
  - Class inheritance, virtual function
  - Functional object
- Understanding “libraries”
  - #include, #ifdef
  - Stream classes
  - String
- Exception handling

Homework #2

- Target due: Week #4 (10/06)
  - A command line reader
  - Thorough understanding of “pointers”
  - Basic program design
  - Ref code: 836/938 lines C++ (last year’s)
  - New feature(s) may be added...

A short version of "Computer Programming" class?

- NO!!
- If you don't have any background in C++ (or C) ...
  - You probably have chosen the wrong class.
- If you are poor in C++ programming...
  - Well, you are definitely NOT the only one, so you are very welcome!!
  - Please pay attention to the lectures in this topic, and make sure you can commit enough time on homework

You may think I cover way too many details in C++... (Why bother to understand them?)

- Remember:
  Programming is a computer science.
  - There is NO random bug!!
    Everything happens for a reason.
  - You need to be rationale, and be "precise on the details".
  - ➔ Capability to handle the complexity!!
- But...
  Programming is also an art.
  - A good program looks beautiful!!
  - A beautiful program is beautiful for a reason.
  - A good design is a MUST, and easy to maintain to make the program live long!
  - ➔ Sense to manage the complexity!!

Homework #3

- Target due: 10/20
  - Complete command interface and a simple command-line number converter.
  - Learn how to write a structured code
  - Homework description file: 16 pages
  - Ref code: 1789(2114)/2409 lines C++

Course Outline

Part 2: Polishing Your Programming Skills

3. C++ Advanced Features Review:
   When can/should I use them?
4. Memory Management:
   How to gain 30% performance improvement easily
5. STL Basics:
   The Standard Template Libraries
6. What is a Good Program?
   Software engineering point of view

4. Memory Management: *How to gain 30% performance improvement easily*

- Where's your bug?
  - Segmentation fault, bus error, etc
- Constructor and destructor
- Fragmentation
- System memory allocation/deletion
- Implement your own memory manager
- Garbage collection
- Cache effect

Homework #4

- Target due: 11/03
  - Memory management
  - Pointers (again), basic data structure
  - Ref code: 1387(2643)/2865 lines C++

5. STL Basics:
The Standard Template Libraries

- Why template libraries?
- Why standard?
- The standard template libraries
  1. Container classes
     - List, array, map, hash, stack, string, bitvector, etc…
  2. Iterators
     - Forward, bidirectional, random, etc
  3. Algorithms
     - For_each, sort, partial_sum, sort, etc.
  4. Functional object
     - Unary, binary, arithmetic, etc
  5. Utility
  6. Memory allocation

6. What is a Good Program?
software engineering point of view

- What do you suffer most in programming?
  - Coding? Compiling? Debugging?
- Which one is more important?
  - Best or complete algorithm?
  - Least instructions/sub-routines called?
  - Least memory used?
  - Smaller size of code?
  - More (or less) advanced language features?
  - Easier to debug and maintain?
  - Nicely documented?
  - Easily reusable?
- Coding style guideline

7. Computational Complexity: *Time and space tradeoffs*

- Review of complexity analysis
- Why should I care?
- What's the most frequently encountered problem?
- What's your best bet?

Overview of this course

Part 1: Introduction

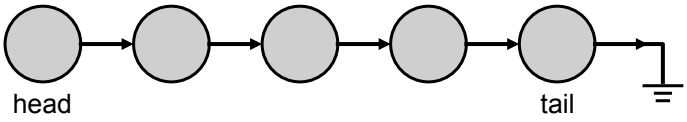
Part 2: Polishing Your Programming Skills

Part 3: Data Structure Revisited

Part 4: Putting What You Learn Together

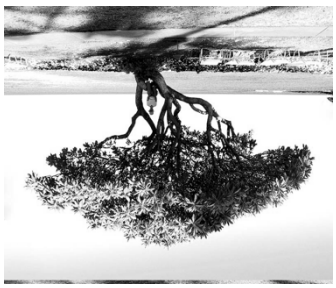
8. Dynamic Array vs. Linked List: *Which one is better?*

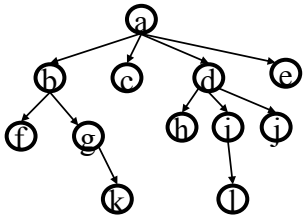
- Linear data types
- Static vs. dynamic array
- Why dynamic array? Why not linked list?
- How to evaluate their performance?
  - Runtime vs. memory usage



9. Tree: *How to search data faster than linear time?*

- Non-linear data types
- Decision trees
- Tree traversal
- Balanced trees
- Implementation issues



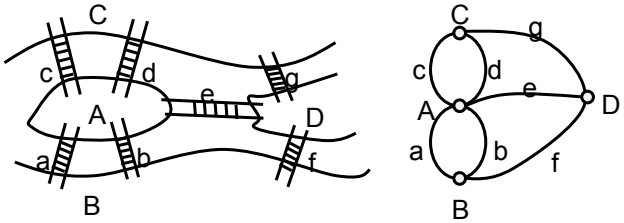


Homework #5

- Target due: 11/17
- Implementation and comparison of various data structures
  - Linked list
  - Dynamic array
  - Binary search tree
- Ref code: 1418(2674)/3104 lines in C++

10. Graph and Circuit: *From CS to EE applications*

- Tree vs. graph
- Basic graph theories
- Graph traversal problems
- Loop handling
- How to design data structure for a circuit netlist?



Homework #6

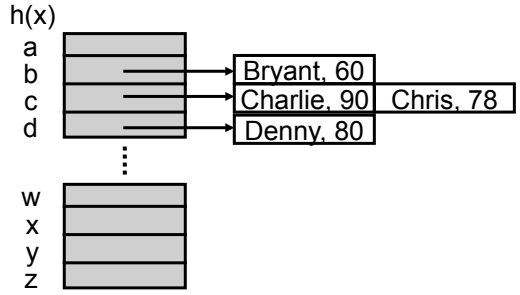
- Target due: 12/01
- A circuit parser
  - I/O and file streams
  - Graph/Circuit data structure
  - Hash/Map usage
  - Boolean logic
- Ref code: 1480(2736)/3803 lines in C++
- A special lecture note on “Lex and Yacc” may be offered

11. Heap, Set and Map: *How to store sorted data?*

- Review of sorting algorithms
- Review of binary (balanced) trees
- Complexity analysis
- Alternative ways of implementation
- Standard Template Library (STL) revisit

12. Cache vs. hash:
Virtual memory in your program

- Review on hash
- Alternative to hash
- What's the difference?
- Computational cache/hash



Homework #7

- Target due: 12/15
- Implementation and practical applications of various data structures
  - Heap
  - Hash
  - Cache
- Ref code: 1381(2637)/2808 lines in C++
- New requirements will be added this year…

13. Bit Vector and Matrix:
All about numerical operations

- Bitwise operations
- Beyond 32/64 bits
- Multi-valued system
- Dense vs. sparse matrix
- Matrix operations
- Linear algebra…

Overview of this course

Part 1: Introduction

Part 2: Polishing Your Programming Skills

Part 3: Data Structure Revisited

Part 4: Putting What You Learn Together

Final Project

- Functionally Reduced And-Inverter Graph (FRAIG)
  - Read in a circuit netlist (HW6)
  - Perform circuit optimization (graph operations)
  - Simulate the circuit (graph traversal, Boolean operations)
  - Collect functionally equivalent candidate pairs (efficient hash implementation)
  - Define the “magic number” to control the program flow (engineering sense)
- Ref code: 4572(5828)/8063 lines in C++
- 30% of the final grade!! Please start earlier!!

Overview of this course

Part 1: Introduction

Part 2: Polishing Your Programming Skills

Part 3: Data Structure Revisited

Part 4: Putting What You Learn Together

Class Schedule

09/16	Intro, DS in Prog., Linux		
09/23	Linux Prog., C++ Review	HW2 out	
09/30	C++ Review (variables)		
10/07	C++ Review (class, polymorphism)	HW3 out	HW2 due
10/14	C++ Review (polymorphism, I/O)		
10/21	Mem Mgr	HW4 out	HW3 due
10/28	STL,Good Prog., Complexity		
11/04	Array and List, Tree	HW5 out	HW4 due
11/11	Tree		

Class Schedule

11/18	Graph	HW6 out	HW5 due
11/25	Heap, set, Map		
12/02	Cache and Hash	HW7 out	HW6 due
12/09	Final Project Discussion		
12/16	Final Project Discussion	Proj. out	HW7 due
12/23	Final Project Discussion		
12/30	Special Topic		
01/06	Special Topic		
01/13	Final exam week		
01/20	Final project week		Proj. due

Homework Assignments and Final Project

- Once again, get yourself familiar with the C++ programming on Linux ASAP!!
  - You MUST compile your code on Linux or OS X environment.
- Homework turn-in
  - Through NTU Ceiba class website
  - Please pay attention to the rules on the class website
  - Filenames, compression rules, etc.
- No copying/pirating
  - If happens, -20 for your term grade!!
- Don't miss any homework!!
  - ~10% of your term grade...
- Do not delay
  - 1 day → - 1/3
  - 2 days → - 2/3
  - 3 days and up → 0