CSCI851 Spring-2021 Advanced Programming Lecture 01

Subject/Course Admin Introduction

Ting ZHANG

CCNU-UOW

Contact details

Office: Room 501, Science hall, NERCEL CCNU

Email: ting.zhang@mail.ccnu.edu.cn

Consultation Time

- TIME for Course CSCI851.

  Monday 10:00 am – 12:00 am (lecture, N323)

  Friday 16:10 pm – 18:10 pm (lab, ….)

  2-hour lecture and a 2-hour lab each week
- A 6-credit-point course.
- According to University policy, 1 credit point is equivalent to 2 hours of work including class attendance, per week.
- So for this course it’s about 8 hours of work outside of the classes.

- CSCI851 students, if you have little or no programming experience you can pass this subject!
  - But, you will need to work hard and take the opportunities that are given to help you learn.
  - As more experienced students we expect you to be organised, and capable of learning quickly.
- It’s likely to be daunting initially.
  - If there are terms that we are using as if everybody should know it, and you don’t know them or don’t understand them, please let us know.
    - Some of that feedback can help shape the lecture/or labs for this subject.

The lecture

- The addition of the lecture doesn’t mean we will be including a lot of additional topics.
- It will just give us time to go through some more examples and tie some of the material together more usefully.
- These deliberately run after the labs, meaning we have a chance to go over some of the particularly important lab exercises.

- The delivery method for classes will be power point slides.
- You will be able to find material for the subject on the subjects eLearning site reachable through the Moodle website:
- The slides will be provided in pdf.
  - Not necessarily much before the lecture takes place, and possibly even afterwards, because I intend to make a fair few changes.

Textbook and references

- There is an (optional) textbook for this subject:

  Lippman, Stanley B.; Lajoie, Josée; Moo, Barbara E.; C++ Primer (5th Edition), 2012.

- This book is a good C++ reference guide, however, it assumes you know nothing about programming, and therefore it may contain a lot of material you already know.

- A book that covers some good new material is:

  O'Dwyer, Arthur; Mastering the C++17:STL, 2017.

- There are also many websites which may be helpful.
- The following are a few…
  - http://www.icce.rug.nl/documents/cplusplus/
  - http://www.cplusplus.com/
  - https://stackoverflow.com/
  - http://www.cppreference.com/
  - http://en.cppreference.com/w/
  - http://www.sgi.com/tech/stl/index.html
  - https://www.bogotobogo.com/cplusplus/cpptut.php
- These sites serve different purposes, looking at formal definitions is sometimes helpful but a simple example is sometimes better.
- It’s helpful to become familiar with some C++ guidelines …

  https://github.com/isocpp/CppCoreGuidelines/blob/master/CppCoreGuidelines.md

The eLearning site

- Check the eLearning (Moodle) site for this subject regularly!
  - Any change to the subject will be announced on the eLearning site.
  - Any information posted to the eLearning site is deemed to have been notified to all students.
- Posts to the announcements section on the Moodle site, by me, are sent to the email account of everyone enrolled anyway.

Assessment: Passing this subject.

- 3 assignments: 10%+10%+10%=30%.
  - Due roughly Weeks 5, 10, and 13.
- Laboratories: 10%.
  - Basically each week.
- Final examination: 60%.

- One TF requirement:
  - At least **40%** of the exam marks: **24/60**.
  - Not meeting this requirement, and obtaining 50 or more overall, *may* result in a TF grade.
  - I'm not going to fail you for not attending labs, if you don't attend and you fail.

Laboratory exercises

- In the laboratories I expect you to work through the provided exercises.
- They are not being marked on correctness so much as that you are working through the material and making progress.
- Typically for this subject I'll aim to provide more exercises than you would expect you to complete in the time.
  - But they will range in difficulty, so some beginner exercises and some extension tasks, and you can start where you like.

Assessment environment

- Banshee: A Unix server.（MinGW/Cygwin）

- You can work at home in any C++ environment, but you need the code to compile on Banshee … (possibly lab Ubuntu)
  - And it (probably) won't be 100% compatible.
    - You can use some smart IDE's.

- You can use SSH to interact with Banshee remotely.
  - Lab clients: SSH Secure Shell Client, Client BitVise and PuTTY.
    - We will be doing this in the lab so don't stress if you don't know how. Probably demo in first L/T.

- There are a few different C++ compilers on Banshee:
  - g++, CC, gcc.
- They don’t have the same functionality so be careful.
- Generally I’ll expect assignments to come with a compilation instruction.
- More on compilation later.
- For some of the labs we may connect to a different server to be able to use C++14 and C++17 functionality.

Some notes on assessment

- Any C++ programs submitted which do not produce the desired results are likely to receive a significant deduction.
- If your program doesn't compile on Banshee (or possible lab Ubuntu), in accordance with instructions, it will likely receive zero.
  - You can always comment out problem sections!
- We will aim to return work within about two weeks of the deadline.
- Students may query about the marking to the lecturer within two weeks of receiving the marks.

Extensions etc.

- If you require additional time to complete an assignment you must submit claims for extensions electronically via SOLS, ***before the DUE date*****.**
- You may be granted an extension if your circumstances warrant it.
- Of course, if you are in hospital for the last week or similar, and cannot get in contact I will understand.

http://www.uow.edu.au/students/sols

A word on Academic misconduct ….: Plagiarism and similar concerns …

- The Academic Integrity Policy, available at:
  http://www.uow.edu.au/about/policy/UOW058648.html

- … describes academic misconduct including:

  f. Plagiarism

  i. Using another person’s ideas, designs, words or any other work without appropriate acknowledgement ;

  ii. Re-using one’s own work without appropriate acknowledgement.

- I suggest you read that document about what is considered misconduct.

- There are two primary concerns for us:
  - Students copying directly from sources, or copying without appropriate referencing.
  - Students copying each other.
    - You can discuss ideas but need to use your own words!
    - With code and mathematical solutions you need to work fairly independently.
- Don’t just copy code from websites:
  - At the very least comment it, but if the code is mostly the work of others you are going to get zero.

Subject Content Introduction

Assumed background

- There is no assumption as to prior programming experience.
  - BCompSc students should know C, Java;
  - MCompSc students may have no programming experience, it’s likely going to be a fair bit of work in that case.

Subject Description

- The subject develops a thorough understanding of programming features, which are implemented in the C++ programming language. It comprises of four main components, namely procedural-based, object-based, object-oriented and generic programming. The subject addresses topics including memory management issues and dynamic memory allocation; classes; STL sequential and associative containers; operator overloading; advanced features in object-oriented programming; C++ RTTI; templates and exception handling; the latest C++ features (e.g. C++11 and C++14 standards).

So … what is this subject about?

- This subject looks beyond the object-based/object-oriented content, and beyond the limitations of Java.

- We use C++ as a vehicle for exploring a range of programming: Procedural, object-based, object-oriented, and generic.

- We look at some differences between Java and C++, for example the memory management models.

  – Due to the undergraduate structure there will be references to Java.

Subject Learning Outcomes …

- On successful completion of this subject, students will be able to, to varying degrees:
  - Design and implement solutions to problems with the C++ programming language.
  - Design and implement procedural-based programming to solve problems.
  - Design and implement objects providing encapsulation, inheritance and polymorphism.
  - Design solutions to problems through the use of generic programming.
  - Design object-oriented solutions to problems.
  - Incorporate advanced features in C++ to achieve efficient implementations.

Looking at the SLOs …

- … the term *design and implement* occurs three times, and design a couple more.
- The programming in the subject title doesn’t just refer to coding.
  - Inevitably there will be a fair bit of syntax covered in this subject, but …
  - … I hope the balance will move more towards understanding principles as the subject develops.

Topics covered …

- Introduction
- C++ Foundations I
- C++ Foundations II: Getting started and Procedural Programming.
- C++ Foundations III: Pointers, classical arrays and ...
- C++ Foundations IV: Control structures, loops, and various other topics ...
- C++ Foundations V: Handling files.
- Getting organised I: Pre-processing, macros, and Makefiles
- Getting organised II: Structs, unions, randomness, and time.
- Getting organised III: Exceptions (Part 1), namespaces, and defensive programming
- Getting organised IV: Debugging and profiling
- Programming with Class I: Fundamental syntax and some introductory UML
- Programming with Class II: Constructors, Destructors, ...
- Programming with Class III: Class/object relations
- Programming with Class IV: Class/class relations
- Programming with Class V: Overloading operators and making friends
- Programming withClass VI: Polymorphism, multiple inheritance
- Runtime type identification and casting
- Creating libraries
- Some miscellaneous topics
- Generic Programming I: Function templates and compiile time functionality
- Generic Programming II: Class templating
- Generic Programming III: Containers and iterators
- Generic Programming IV: The Standard Template Library (STL)
- Genetic Programming V: Template Compilation models
- Some final bits and pieces

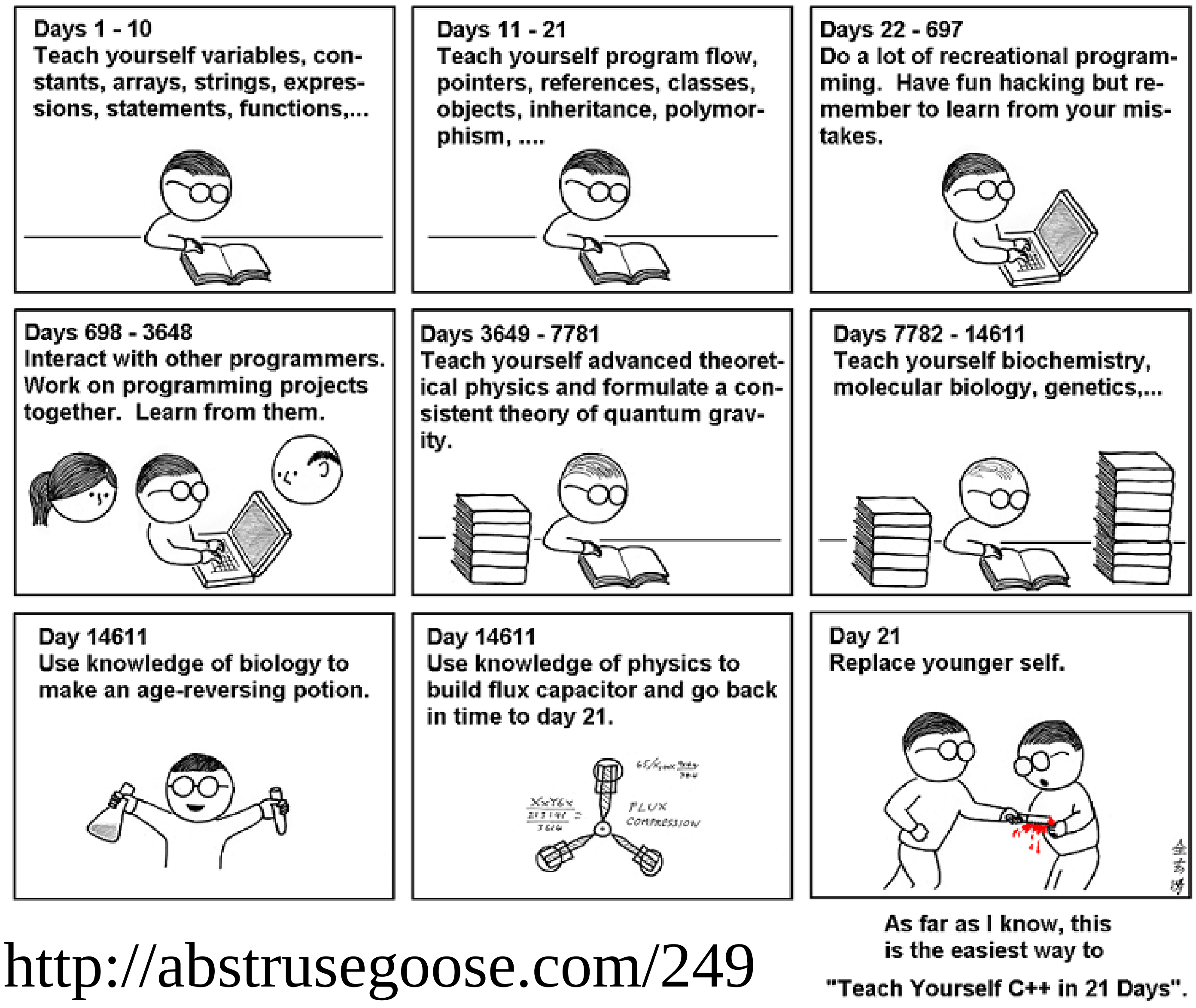
Languages evolve

- C++ continues to evolve.

- Don’t think of it as static.
  - This is true of most (programming) languages.

- We will try an integrate some of the more modern features of C++ (11 for sure, 14?, 17?) into this subject; but there isn’t time to cover close to everything.

Learning to code in 21 days …


Days 1 - 10
Teach yourself variables, constants, arrays, strings, expressions, statements, functions,...
Days 11 - 21
Teach yourself program flow, pointers, references, classes, objects, inheritance, polymorphism, ...
Days 22 - 697
Do a lot of recreational programming. Have fun hacking but remember to learn from your mistakes.
Days 698 - 3648
Interact with other programmers. Work on programming projects together. Learn from them.
Days 3649 - 7781
Teach yourself advanced theoretical physics and formulate a consistent theory of quantum gravity.
Days 7782 - 14611
Teach yourself biochemistry, molecular biology, genetics,...
Day 14611
Use knowledge of biology to make an age-reversing potion.
Day 14611
Use knowledge of physics to build flux capacitor and go back in time to day 21.
FLUX COMPRESSION
Day 21
Replace younger self.
As far as I know, this is the easiest way to "Teach Yourself C++ in 21 Days".

http://abstrusegoose.com/249

Practice makes better …

- 21 days isn’t enough.
  - For this subject or for competence.
- When there are examples in the lecture notes they are just that, examples, not points to learn.
  - You should practice variations and learn by making mistakes.
  - It’s better to make mistakes in your own practice than in the assignment.
- The examples in the labs are there to help.
  - Do them! Ask questions!
- For those of you who haven’t done (any/much) programming/coding before it’s important that you get started early, yesterday would have been a good time!