ICT Programming Curriculum Meeting Notes

October 8 2015

Revised Agenda

0) Meeting Notes – September 8 2015

1) Subject Outlines
    - OOP345
    - BTP305
    - DSA555
    - BTP500
    - Revising Outlines:
        - Subject Descriptions and Learning Outcomes
        - Topics, Evaluations, Reference Material
        - Timelines, Workshop Learning Outcomes

2) Core Programming Subjects Update
    - IPC144
    - BTP100
    - OOP244
    - OOP345
    - BTP305
    - Material Allowed on Tests and the Final Exam
    - Continuing Education Conflicts

3) Working Groups
    - C++ Idioms
    - C++ Best Practices
    - Mapping Topics to Core Literacies

4) Other Business

Meeting Notes – September 8 2015

Attendees: Peter McIntyre, Ian Tipson, Catherine Leung, Tim McKenna, Mark Fernandes, Elnaz Delipsheh, Fardad Soleimanloo, Patrick Crawford, Greg Blair, Sunny Shi, Mary Lynn Manton, Chris Szalwinski

Agenda Approved

http://zenit.senecac.on.ca/wiki/imgs/Start-Up_Meeting_September_8_2015.pdf

Meeting Notes for Sep 8 will be approved unless objections submitted before October 15 2015.

Subject Outlines

OOP345 – Object Oriented Software Development Using C++

Subject Description (no proposed changes)

"This subject expands the student's skill-set in object-oriented programming and introduces the student to threaded programming. The student learns to model relationships between classes using containers, inheritance hierarchies and polymorphism in the C++ programming language and to write C++ programs that execute on multiple threads."

Learning Outcomes

– make LOs more specific
– replace raw pointers and pointer arithmetic with smart pointers

- Design collections of model objects using sequential containers and multi-dimensional arrays to ~~solve~~ organize the elements of a programming solution to a systems or business problem
- Create function objects and closures to customize ~~a programming solution~~ an algorithm for a particular ~~application~~ systems or business task
- Model generalization and specialization using inheritance hierarchies to minimize the duplication of code
- Model polymorphic behavior using interfaces, virtual functions and templates (generics) to ~~amplify the reusability of code~~ enable adaption of a programming solution to an evolving class of systems or business problems
- Implement ~~design components~~ algorithms using ~~algorithms of~~ the standard template library to ~~utilize existing technologies~~ simulate the solution of a systems or business problem
- Create ~~program components~~ algorithms of quadratic complexity to solve non-linear problems
- ~~Design~~ Implement access to model objects ~~program components~~ using ~~raw~~ smart pointers ~~and pointer arithmetic~~ to ~~access data in program memory~~ manage their lifetimes
- Design multi-tasked solutions using threading libraries to improve the performance of a program
- Design file stream objects to backup text and binary data for future restoration

BTP305 – Object Oriented Software Development Using C++

Subject Description (no proposed changes)

“This subject expands the student's skill-set in object-oriented programming and introduces the student to threaded programming. The student learns to model relationships between classes using containers, inheritance hierarchies and polymorphism in the C++ programming language and to write C++ programs that execute on multiple threads.”

Learning Outcomes (proposed changes)

- to make LOs more specific
- to place stress smart pointers over raw pointers and pointer arithmetic?

- Design collections of model objects using sequential containers and multi-dimensional arrays to ~~solve~~ organize the elements of a programming solution to a complex systems or business problem
- Create function objects and closures to customize ~~a programming solution~~ an algorithm for a particular ~~application~~ systems or business task
- Model generalization and specialization using inheritance hierarchies to minimize the duplication of code in complex hierarchies
- Model polymorphic behavior using interfaces, virtual functions and templates (generics) to ~~amplify the reusability of code~~ enable adaptation of a programming solution to an evolving class of systems or business problems
- Implement ~~design components~~ algorithms using ~~algorithms of~~ the standard template library to ~~utilize existing technologies~~ simulate the solution of a systems or business problem
- Create ~~program components~~ algorithms of quadratic complexity to solve non-linear problems
- ~~Design~~ Implement access to model objects ~~program components~~ using ~~raw~~ smart pointers ~~and pointer arithmetic~~ to ~~access data in program memory~~ manage their lifetimes
- Design multi-tasked solutions using threading libraries to improve the performance of a program
- Design file stream objects to backup text and binary data for future restoration
- Trace the execution of program code that includes a linked list to debug an application

⇨ **OOP345 and BTP305 Discussion**

⇨ Should Multi-Dimensional Arrays be included in LOs or just the topic list
  - Dynamic allocation in the topic list as optional, not in Los

⇨ Customize an algorithm
  - Decided this scope includes STL algorithms

⇨ Proposed revisions to the OOP345 and BTP305 LOs approved

DSA555 – Data Structures and Algorithms

Topic list has been updated to align better with OOP345 (Sept 10 2015)

Still to be done

- Formulate a comprehensive set of learning outcomes
- Consider 2 Lecture +2 Lab Mode of instruction

Subject Description – should we remove C++ STL | should we simplify the description?

“What makes a piece of code "good"? What is the best way to represent data? How do we determine the efficiency of our programs? Students taking this subject will learn to write and apply standard algorithms and data structures to real programming problems. They will learn how an algorithm's efficiency is measured. The features and limitations of a number of data structures and situations when they should be employed will be discussed. Aside from traditional structures and algorithms, students will also learn about related topics including: encryption algorithms, heuristic searches, and information retrieval. Additionally students will be introduced to the C++ Standard Template Library.”

⇨ Cathy agrees to removing C++ STL

Learning Outcomes – will need to update to meet Seneca QA Standards

- ~~En~~Incorporate data structures into the applications they write
- Implement searching and sorting algorithms
- Create their own data structures
- Determine which algorithm or data structure to use in different situations

BTP500 – Data Structures and Algorithms

Still to be done

- Formulate a comprehensive set of learning outcomes
- Consider 2 Lecture + 2 Lab Mode of instruction

Compare and contrast with DSA555 above

Topic list needs slight realignment to BTP305

Subject Description – is this description satisfactory?

“In this course, the student learns to write and apply, to real programming problems, the classical data structures and algorithms from computer science, such as searching and sorting algorithms, stacks, queues, linked lists and trees. Algorithm efficiency, encryption algorithms, randomized algorithms and heuristic search are among other topics investigated. The C++ Standard Template Library and the Java libraries are also looked at, as examples of data structure implementation.”

Learning Outcomes – will need to update to meet Seneca QA Standards

- methodically test and debug complex Java and C++ programs
- analyze a programming problem in order to determine the suitability of implementing a variation on a well-known data structure or algorithm as part of the solution
- compare the relative efficiencies of competing algorithms
- determine the shortcomings or limitations of a particular data structure or algorithm

- use the data structures provided in the standard Java and C++ libraries
- describe the features and limitations of a particular data structure or algorithm
- implement variations of well-known data structures and algorithms using Java and C++
- identify, through the application of research techniques, and use in a programming solution, well known data structures or algorithms not specifically covered in class
- produce technical and usage documentation for data structures/algorithms developed

⇨ DSA555 and BTP500 Discussion

- ⇨ Ian – success rate in both courses is pretty good, so "if it ain't broke, don't fix it"
- ⇨ Cathy will do the LOs for DSA555 and will meet with Lew Baxter to discuss BTP500 LOs
- ⇨ Cathy does not mind 2 Lectures + 2 Labs, but does not want to be activity-restricted
  - Peter suggested use of active learning classrooms as an option
  - Initial conclusion is 4 hours in the classroom is OK
- ⇨ Associative Containers: Map and Hash discussion – where to put it
  - Add to STL in 345/305 or add STL Associative Containers to 555/500
  - We really don't have room in 345/305
  - Initial conclusion leave STL Associative Containers out of the curriculum
- ⇨ Peter and Ian – promote and encourage all CPA students to take DSA as one of their pro options
- ⇨ BTP500 – Complexity Theory
  - BSD Consent renewal requested an appropriate balance of theory and practice: "To improve coverage of algorithmic complexity, the program will increase the accountability, emphasis and awareness of this important topic. This enhancement to the curriculum will begin in the summer 2014 term, with BTP500 – Data Structures and Algorithms course"
  - Chris suggested that a bit more than one week be spent on this topic

Revising Outlines (FYI)

Individual Subjects

Faculty initiative -> QA approval -> Chair Approval -> Posting

- Chris – this is separate from the topic list, etc. – voted in Curriculum meetings

Topic List, Evaluations, Reference Material

Participating faculty initiative (week 7) -> Coordinator approval -> Posting

- Peter – deadline for faculty recommendations is end of Week 10 (Nov 18 2015)

Timeline, Workshop Learning Outcomes

Participating faculty initiative (week 0) -> Posting

    - Chris – touchy point – workshop LOs to be decided by multi-section faculty by week 0 – all of us need to recognize the flow of content through each course and with respect to the set of core courses – workshop LOs are not open to revision by individual faculty who are expected to abide by the initial multi-section decision
    - Timeline Posted Start of Week 10 (Nov 18 2015)
        - Proposed vs Mandatory – part of contract with the students – hence: Mandatory
        - Individual faculty to communicate variations to students caused by scheduling and holiday conflicts

Core Programming Subjects Update

OOP244 – Introduction to Object Oriented Programming

Web Site Dynamic Component (Workshops, Final Project, Instructors)

- Redirection to the current Semester's Content
- https://scs.senecac.on.ca/~oop244/dynamic/workshops/index.html

Experience with Interactive Scripts for Workshops and the Final Project

⇨ Workshop and Final Project Submission Scripts

- Fardad has designed an interactive script for submitting workshops and assignments
    - Script compiles source code, reports warnings and errors, compares execution output to expected output and submits the source code to the instructor only if every check passes
    - Students continue to work on source code until script allows submission
    - Script displays the output lines that are unexpected to facilitate debugging
    - Instructor prepares the configuration file for each workshop and the final project
- Chris reviewed his experience after teaching one of Fardad's classes – most students stayed until the end and kept working to meet the script's requirements – recommends using this script in all core programming courses
- Fardad outlined usage of the script
- Cathy requested a copy of the script

Evaluation Weights for the Winter Semester

⇨ Faculty decided to retain the evaluation weights for the Winter semester

	Fall - OOP244	Winter - OOP244
Final Project	20%	20%
Workshops/Lab Work	30%	30%
Quizzes	15%	15%
Midterm Test	25%	25%
Final Exam	10%	10%

Windows and Linux Environments

- Do we make Visual Studio IDE a standard tool in this course?
- Is matrix Compilation and Execution sufficient?
- See below

BTP200 – The Object-Oriented Paradigm Using C++

Web Site Dynamic Component (Workshops, Final Project, Instructors)

- Redirection to the current Semester's Content
- https://scs.senecac.on.ca/~btp200/dynamic/workshops/index.html

Evaluation Weights for the Winter Semester

⇨ Recommendations

- Chris – should we align BTP200 with OOP244 (esp. low exam weight)
- Olivier – teaching in the Winter – prefers 20% for final - keep the summer weights or move 5% from the mid-term to the final project for 50/50 distribution
- Peter – prefers 20% final for this BSD course
- Ian – no problem with 20% for the Final Project
- No further discussion

	Summer – BTP200	Winter – BTP200
Final Project	15%	20%
Workshops/Lab Work	30%	30%
Quizzes	15%	15%
Midterm Test	20%	15%
Final Exam	20%	20%

Experience with Interactive Scripts for the Final Project

⇨ Recommendation

- Chris – found Fardad's scripts extremely useful to the students for the final project

Windows and Linux Environments

- Do we make Visual Studio IDE a standard tool in this course?
- Is matrix Compilation and Execution sufficient?

⇨ OOP244 and BTP200 Discussion

⇨ Use of IDE's in the Core Courses

- Chris
  - Need to expose students to Windows and Linux platforms
  - Getting more difficult to access the Visual Studio command line
  - IDEs such as VS facilitate debugging – we have been discussing introducing debugging for many semesters – this is one way to incorporate it
- Cathy – gdb is enough – objectionable mutterings for some – no second
- Ian – Visual Studio has won the battle of the IDE's
- Olivier – personally would not use Visual Studio to teach BTP200 – doesn't like VS and thinks it is a little too complex for what students have to do – uses CentOS with GNOME editor and clang++ compiler – likes CodeBlocks
- Fardad – has been using VS for many semesters
- No overall strong objection to IDEs – no recommendation agreed

IPC144 – Programming Fundamentals Using C

Web Site Dynamic Component (Workshops Assignments, Instructors)

- Redirection to the current Semester's Content
- https://scs.senecac.on.ca/~ipc144/dynamic/workshops/index.html
- Not discussed

Weekly Coordination Meetings

- …
- Not discussed

Evaluation Weights for the Winter Semester

- Ian – weak students learn the material over several semesters: no trouble passing weak ones
- Cathy wants definite values (not ranges) – values must be exactly divisible by no of deliverables
- Agreed to a Final Project with milestones instead of distinct assignments (copying OOP244)
- Cathy wants to go back to IPC144 Winter instructors for approval (only Pat and Cathy present)

	Fall - IPC144	Winter – IPC144	Fall - OOP244
Assignments	30%	20%	20%
Workshops/Lab Work	15% to 20%	30%	30%
Quizzes	5% to 10%	15%	15%
Midterm Test	15%	25%	25%
Final Exam	30%	10%	10%

Windows and Linux Environments

- Do we make Visual Studio IDE a standard tool in this course?
- Is matrix Compilation and Execution sufficient?

- See above – no consensus reached

Proposal to Reformat the Delivery

- …
- Not discussed

BTP100 – Programming Fundamentals Using C

Web Site Dynamic Component (Workshops Assignments, Instructors)

- Redirection to the current Semester's Content
- https://scs.senecac.on.ca/~btp100/dynamic/workshops/index.html

Weekly Coordination Meetings

- …
- Not discussed (Elnaz and Peter Liu will meet)

Evaluation Weights for the Winter Semester

- Elnaz intends to follow OOP244 experiment

	Fall – BTP100	Winter – BTP100	Summer – BTP200
Final Project	15%	20%	15%
Workshops/Lab Work	30%	30%	30%
Quizzes	15%	15%	15%
Midterm Test	20%	25%	20%
Final Exam	20%	10%	20%

Windows and Linux Environments

- Do we make Visual Studio IDE a standard tool in this course?
- Is matrix Compilation and Execution sufficient?

Proposal to Reformat the Delivery

- …
- Not discussed

OOP345 – Object-Oriented Software Development Using C++11

Revised Timeline

- Direct path to STL with detail features deferred to last third of the semester

- https://scs.senecac.on.ca/~oop345/pages/timeline.html
- Not discussed

Evaluation Weights for the Winter Semester

- Not discussed

	Fall – OOP345	Winter – OOP345	Fall – BTP305
Final Project	15%		15%
Workshops/Lab Work	30%		30%
Quizzes	15%		15%
Midterm Test	20%		20%
Final Exam	20%		20%

Windows and Linux Environments

- Do we make Visual Studio IDE a standard tool in this course?
- Is matrix Compilation and Execution sufficient?
- Not discussed

BTP305 – Object-Oriented Software Development Using C++11

Revised Timeline

- Direct path to STL detail features deferred to last third of the semester
- https://scs.senecac.on.ca/~btp305/pages/timeline.html
- Not discussed

Evaluation Weights for the Winter Semester

- Not discussed

	Fall – BTP305	Winter – BTP305
Final Project	15%	
Workshops/Lab Work	30%	
Quizzes	15%	
Midterm Test	20%	
Final Exam	20%	

Windows and Linux Environments

- Do we make Visual Studio IDE a standard tool in this course?
- Is matrix Compilation and Execution sufficient?

- Elliott uses Visual Studio
- Greg does not use Visual Studio
- Not discussed

Material Allowed on Tests and the Final Exam

- Do we restrict the material allowed to what is stated on the subject outline?
- Not discussed

Faculty of Continuing Education Conflicts

IPC144

- Subject Description and LOs match
- Topic list, reference text, mode and assessment weights DO NOT match

OOP244

- Subject Description and LOs match
- Topic list, reference text, mode and assessment weights DO NOT match

OOP345

- Subject Description and LOs match
- Topic list, reference text, mode and assessment weights DO NOT match

⇨ Ian – some discussions have been held with ConEd: ??

Proposed Winter Working Groups

- Volunteers for All Groups Needed

C++ Idioms

C++ does not protect the software developer from misuse of the language. Idioms are examples usage that has been popularized by the experts. Applying these idioms assists in producing quality code.

Understanding idioms helps an instructor explain why C++ code is written in a certain way rather than in any arbitrary way.

https://en.wikibooks.org/wiki/More_C%2B%2B_Idioms (June 18 2014)

"C++ has indeed become too "expert friendly" at a time where the degree of effective formal education of the average software developer has declined. However, the solution is not to dumb down the programming languages but to use a variety of programming languages and educate more experts. There has to be languages for those experts to use–and C++ is one of those languages." Bjarne Stroustrup (2006). "The Problem with Programming" Computer News Nov 28 2006.

⇨ Not discussed

C++ Best Practices

https://www.gitbook.com/book/lefticus/cpp-best-practices/details

https://isocpp.org/wiki/faq/Coding-standards

http://www.amazon.com/exec/obidos/ASIN/0321113586/

http://stroustrup.com/JSF-AV-rules.pdf

⇨ Not discussed

Mapping Topics to Core Literacies

⇨ Not discussed

Other Business

⇨ Not discussed