HIGH-LEVEL PROGRAMMING 2

Introduction

by Prasanna Ghali

What is HLP2?

- Explain fundamental C++ language constructs and semantics:
  - Types, variables, initialization
  - Namespaces
  - Static polymorphism using functions
  - Data abstraction with `enum`s, `struct`s, `class`es
  - Generic programming using templates
  - Object-orientation including inheritance, derivation, and run-time polymorphism

What is HLP2?

- Explain use of and overall design principle of C++ standard library
  - Input/output streams
  - Types such as `std::pair`, `std::string`, `std::initializer_list`, and `std::tuple`
  - Standard template library:
    - Containers
    - Iterators
    - Algorithms
    - Function objects

What is HLP2?

- Use C++ data abstraction and encapsulation techniques to design and implement user-defined types that are:
  - usable,
  - correct,
  - well-behaved, and
  - non-trivial

Design and implement your types so that they're as easy to use correctly as built-in types and hard to use incorrectly

What is HLP2?

- Use user-defined classes in combination with different components from C++ standard library to solve computation problems

Module Learning Outcomes

- MLOs indicate *what you can do* after completing the module
- This module has six learning outcomes

Module Learning Outcomes

1. *Explain* data abstraction, generic, and object-oriented programming paradigms and styles
2. *Apply* C++'s class abstraction mechanism to define user-defined data types
3. *Apply* C++'s template mechanism to define general algorithms and classes that accept wide variety of types
4. *Apply* C++'s inheritance and dynamic polymorphism mechanisms to implement class hierarchies

Module Learning Outcomes

5. *Apply* C++ standard library data structures and algorithms to effectively solve programming problems
6. *Develop* C++ programs for small-scale computing problems by combining procedural, data abstraction, object-oriented, and generic programming styles

Teaching Strategies

- Lectures
- Quizzes and Exercises [15%]
- Programming Labs [20%]
- Programming Assignments [15%]
- Midterm and Final Assessments [50%]

Teaching Strategies: Lectures

- Introduce theoretical concepts
- Almost always accompanied by live coding demonstrations and examples of theory
- After lecture, you must read associated material in textbook and other references and program before attending next lecture!!!
- Attendance is mandatory

Teaching Strategies: Quizzes

- Provide a venue to better understand theory covered in lectures
- Involves exercises that test your comprehension
- Could be in-class or take-home
- Submission is mandatory – no submission equivalent to zero grade
- Respect submission deadlines!!!

Teaching Strategies: Programming Assignments

- Provide venue to improve problem solving skills and knowledge of C++
- Consist of programming exercises with little hand-holding
- Submission is mandatory
- Respect submission deadlines!!!

Teaching Strategies: Midterm & Final Tests

- Aim is for you and us to know how much you know
- Involves all material covered in lectures, quizzes, and assignments
- Involves reading code, analyzing code, writing code, debugging code, …
- Attendance is mandatory

Online Only

- All assessments are online only!!!

Learning Strategies

- Be an active and motivated learner
- Come prepared to every lecture and lab
- Take pride in your submissions!!!
- Get your hands dirty by programming!!!
- Expand your horizons by reading the text book
- Get help – we're here to help you succeed

Getting Help

- If you've specific questions about HLP2 material:
  - Post questions to Teams channel
  - Use instructors consultation hours on Teams
  - Use Academic Support Center
- Questions involving your grades and other private matters should be directed to your instructor
  - Emails must always have CSD1170 or CS1171 in Subject field

Are You Helping Yourself?

- We're here to help, but what have you done to help yourself?
  - Your problem solving skills will determine your future career's trajectory
  - You can learn this skill by analyzing and debugging your problem extensively before asking for help
  - Asking for help at first sign of something not working is similar to spoon feeding!!!

Academic Integrity

- You've to submit original work
  - Discussing solutions is encouraged
  - Having study groups is encouraged
- Don't take solutions
- Don't provide solutions
- Read academic integrity policy on course web page

Assessments

- Quizzes [15% of final grade]
  - Periodic in class quizzes during lectures; no specific schedule; no shows will get zero
  - Periodic take-home quizzes [some with SafeBrowser and some without]
- Labs [20% of final grade]
  - Weekly
  - Attendance required; no submissions will get zero

Assessments

- Assignments [15% of final grade]
  - Weekly
  - No submissions will get zero
- Midterm Test [20% of final grade]
  - Week 6 [provisionally set for Friday February 10]
- Final Test [30% of final grade]
  - Week 14 [provisionally set for Tuesday April 4]

Assessments

- Check your online submissions are evaluated!!!
  - Your responsibility not mine!!!
- Late submissions policy: Zero grade
- Except for medical and family emergencies that are communicated in advance to Module Coordinator [that would be me] and Registrar's Office

Grades

Description	Letter Grade
Excellent attainment of learning outcomes	A-, A, A+
Very good attainment of learning outcomes	B-, B, B+
Good attainment of learning outcomes	C-, C, C+
Adequate attainment of learning outcomes	D, D+
Failed attainment of learning outcomes	F

What is C++?

According to page 1 of every ISO C++ standard:

C++ is a general-purpose programming language based on the C programming language […]. C++ provides many facilities beyond those provided by C, including additional data types, classes, templates, exceptions, namespaces, operator overloading, function name overloading, references, free store management operators, and additional library facilities.

Why Learn C++?

- C++ is bigger, more complex, more nuanced, and more expressive
- Provides you more control in how program runs
- You've more things to understand, more things to control, more ways to go wrong, more difficulty in debugging
- You've to take more care in designing program to take into account additional complexity
- To use C++ effectively means to know more about what's going on "under the hood"

Why Learn C++?

- Because of complexity, payoff in learning C++ is at least threefold for you

Why Learn C++?

1) C++ is still best and most commonly used language where fine-grained control over performance is paramount
   - You'll have a job for next few decades

Why Learn C++?

2) C++ is both low-level and high-level
   - C++ makes visible low-level details that you might not otherwise experience
   - This allows you to get insights into how to make your programs use memory more effectively and make your programs run faster in any other programming language you might use

Why Learn C++?

3) C++ introduces collection of language features that you'll almost certainly see in other languages you've to learn in future
   - Makes learning new languages easier

Why Learn C++?

Even if you never use C++ again, learning C++ will make you a better problem-solver and a better programmer *in any language*

Design Goals of C++

- When learning any new language, important to understand its main design goals
  - What was language intended to be used for?
  - How was language intended to be used for?

Design Goals of C++

- General-purpose programming language
- Mid-level language
- Compiled language
- Statically-typed language
- Multi-paradigm language
- Provide transition path
- Provide zero-cost overhead

Prerequisites: Things You Should Know From HLP1

- Built-in data types, literals, `sizeof` operator
- Variables: declarations, definitions, initialization
- Statements: Operands, operators, expressions, statements, precedence & associativity, order of operand evaluation
- Selection statements: `if`, `else` clause, `switch`
- Iteration statements: `while`, `for`, `do while`
- Meaning of keywords `break`, `continue`, `return`

Prerequisites: Things You Should Know From HLP1

- Functions: declarations, definitions, function parameter vs. function argument, pass-by-value semantics
- Idea of separate compilation and linking: header files, source files, object files, executable, compilation steps [preprocessing, assembling, compiling]

Prerequisites: Things You Should Know From HLP1

- C standard library: must have good knowledge of common standard library functions declared in `<math.h>`, `<string.h>`, `<stdlib.h>`, `<ctype.h>`, …
- `const` keyword and its uses [e.g., on variables, on parameters]

Prerequisites: Things You Should Know From HLP1

- Pointers: purpose, problems solved, usage
- Regions of program memory: text area, data area, BSS, stack, heap
- Meaning of keywords `static`, `extern`, `auto`, `register`, `volatile`
- Pointers
- Allocation/deallocation of dynamic memory and potential problems involved in use of dynamically allocated memory

Prerequisites: Things You Should Know From HLP1

- Built-in data structures, like arrays and structures (memory storage, access)
- Relationship between arrays and pointers, pointer arithmetic, compact pointer expressions
- Two-dimensional arrays: definition, memory representation, passing to and returning from functions
- Writing algorithms & problem-solving process [this is the Achilles heel for many of you]

History and Evolution of C++

Year	Description
1979	First implementation of “C with classes”
1983	Renamed to C++
1985	The C++ Programming Language 1st Edition
1990	The Annotated C++ Reference Manual
1991	The C++ Programming Language 2nd Edition
1998	First ISO Standard [C++98]
2003	Small amendments [C++03]
2011	Major release of C++11 [Modern C++]
2014	Minor updates for C++14
2017	Minor updates for C++17
2020	Major release of C++20 [New Modern C++]
2023	Minor updates for C++23