

HIGH-LEVEL PROGRAMMING 2

Introduction

by Prasanna Ghali

What is HLP2?

2

- Explain fundamental C++ language constructs and semantics:
 - ▣ Types, variables, initialization
 - ▣ Namespaces
 - ▣ Static polymorphism using functions
 - ▣ Data abstraction with `enums`, `structs`, `classes`
 - ▣ Generic programming using templates
 - ▣ Object-orientation including inheritance, derivation, and run-time polymorphism

What is HLP2?

3

- Explain use of and overall design principle of C++ standard library
 - ▣ Input/output streams
 - ▣ Standard template library:
 - Containers
 - Iterators
 - Algorithms, and
 - Function objects

What is HLP2?

4

- 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?

5

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

Module Learning Outcomes

6

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

Module Learning Outcomes

7

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 that accept wide variety of types
4. *Apply* C++'s inheritance and dynamic polymorphism mechanisms to implement class hierarchies

Module Learning Outcomes

8

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

Assessments

9

- 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

10

- Assignments [15% of final grade]
 - ▣ Weekly
 - ▣ No submissions will get zero
- Midterm Test [20% of final grade]
 - ▣ Week 7 [10 am on Monday February 21]
- Final Test [30% of final grade]
 - ▣ Week 14 [Thursday April 14]

Assessments

11

- 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

12

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

What is C++?

13

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++?

14

- ❑ 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++?

15

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

Why Learn C++?

16

- 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++?

17

- 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 programming language you might use

Why Learn C++?

18

- 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++?

19

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++

20

- 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++

21

- 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 HLP 1

22

- 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 HLP 1

23

- 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 HLP 1

24

- 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 HLP 1

25

- 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 and dynamic memory allocation (usage of standard library and potential problems)

Prerequisites: Things You Should Know From HLP 1

26

- 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++

27

Year	Description
1979	First implementation of “C with classes”
1983	Renamed to C++
1985	The C++ Programming Language 1 st Edition
1990	The Annotated C++ Reference Manual
1991	The C++ Programming Language 2 nd 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