

C++ Video Lectures

1 Lecture 1 (2019-05-07) — Course Introduction [2019-05-07]

The following is a link to the full video:

- ◇ https://youtu.be/-Jyf-U18_gI [duration: 00:48:37]

The following are links to particular offsets within the video:

- ◇ 00:00: [course_intro] SENG 475 & ECE 596C
- ◇ 00:24: [course_intro] Course Overview [multiple slides]
- ◇ 02:11: [course_intro] Prerequisites and Requirements
- ◇ 05:33: [course_intro] Course Topics
- ◇ 07:07: [course_intro] Learning Outcomes
- ◇ 09:42: [course_intro] Course Outline and Various Other Handouts
- ◇ 32:02: [course_intro] Video Lectures
- ◇ 32:37: [course_intro] Computer-Based Tutorial
- ◇ 37:10: [course_intro] Plagiarism and Other Forms of Academic Misconduct
- ◇ 41:54: [course_intro] Software Development Environment (SDE)
- ◇ 42:57: [course_intro] Prelude to SDE Demonstration
- ◇ 45:55: [course_intro] SDE Demonstration

2 Lecture 2 (2019-05-08) — Algorithms and Data Structures [2019-05-08]

The following is a link to the full video:

- ◇ <https://youtu.be/JOUZVLMJvI> [duration: 00:49:42]

The following are links to particular offsets within the video:

- ◇ 00:00: [algorithms] Algorithms [title slide]
- ◇ 01:07: [algorithms] Software Performance
- ◇ 02:16: [algorithms] Random-Access Machine (RAM) Model
- ◇ 04:17: [algorithms] Worst-Case, Average, and Amortized Complexity
- ◇ 08:21: [algorithms] Asymptotic Analysis of Algorithms
- ◇ 09:55: [algorithms] Big Theta (Θ) Notation
 - ◇ [algorithms] Big Theta (Θ) Notation (Continued)
- ◇ 12:12: [algorithms] Big Oh (O) Notation
 - ◇ [algorithms] Big Oh (O) Notation (Continued)
- ◇ 13:01: [algorithms] Big Omega (Ω) Notation
 - ◇ [algorithms] Big Omega (Ω) Notation (Continued)
- ◇ 15:32: [algorithms] Asymptotic Notation in Equations and Inequalities
- ◇ 17:06: [algorithms] Properties of Θ , O, and Ω
- ◇ 18:30: [algorithms] Additional Remarks
- ◇ 18:49: [algorithms] Remarks on Asymptotic Complexity
- ◇ 22:30: [algorithms] Some Common Complexities
- ◇ 23:32: [algorithms] Recurrence Relations
- ◇ 25:12: [algorithms] Solving Recurrence Relations
- ◇ 26:24: [algorithms] Solutions for Some Common Recurrence Relations
- ◇ 27:39: [algorithms] Iterative Fibonacci Algorithm: Time Complexity
- ◇ 30:10: [algorithms] Iterative Fibonacci Algorithm: Space Complexity
- ◇ 31:04: [algorithms] Recursive Fibonacci Algorithm: Time Complexity
- ◇ 32:47: [algorithms] Recursive Fibonacci Algorithm: Space Complexity

- ◇ 34:34: [algorithms] Amdahl's Law
- ◇ 38:02: [data_structures] Abstract Data Types (ADTs)
- ◇ 41:14: [data_structures] Container ADTs
- ◇ 43:17: [data_structures] Container ADTs (Continued)
- ◇ 45:35: [data_structures] Iterator ADTs

3 Lecture 3 (2019-05-10) — Data Structures [2019-05-10]

The following is a link to the full video:

- ◇ <https://youtu.be/1swLQCO-1Cg> [duration: 00:46:23]

The following are links to particular offsets within the video:

- ◇ 00:00: [data_structures] Container and Iterator Considerations
- ◇ 03:26: [data_structures] Container and Iterator Considerations (Continued)
- ◇ 08:23: [data_structures] List ADT
- ◇ 10:43: [data_structures] Array-Based Lists
 - ◇ [data_structures] Array-Based Lists: Diagram
- ◇ 14:38: [data_structures] Remarks on Array-Based Lists
- ◇ 19:15: [data_structures] Singly-Linked Lists
 - ◇ [data_structures] Singly-Linked Lists: Code
 - ◇ [data_structures] Singly-Linked Lists: Diagram
- ◇ 29:52: [data_structures] Remarks on Singly-Linked Lists
- ◇ 33:19: [data_structures] Singly-Linked List With Header Node
 - ◇ [data_structures] Singly-Linked List With Header Node: Code
 - ◇ [data_structures] Singly-Linked List With Header Node: Diagram
- ◇ 40:52: [data_structures] Remarks on Singly-Linked List With Header Node
- ◇ 41:49: [data_structures] Doubly-Linked Lists
 - ◇ [data_structures] Doubly-Linked Lists: Code
 - ◇ [data_structures] Doubly-Linked Lists: Diagram
- ◇ 45:55: [data_structures] Remarks on Doubly-Linked Lists [starting from end of preceding slide]

4 Lecture 4 (2019-05-14) — Data Structures, Some C++ Review (Const and Other Stuff) [2019-05-14]

The following is a link to the full video:

- ◇ <https://youtu.be/hSEUXnb0cFY> [duration: 00:49:38]

The following are links to particular offsets within the video:

- ◇ 00:00: [data_structures] Doubly-Linked List With Sentinel Node
 - ◇ [data_structures] Doubly-Linked List With Sentinel Node: Code
 - ◇ [data_structures] Doubly-Linked List With Sentinel Node: Diagram
- ◇ 05:46: [data_structures] Remarks on Doubly-Linked Lists With Sentinel Node
- ◇ 07:23: [data_structures] Stack ADT
- ◇ 08:25: [data_structures] Array Implementation of Stack
 - ◇ [data_structures] Array Implementation of Stack: Diagram
- ◇ 09:13: [data_structures] Remarks on Array Implementation of Stack
- ◇ 10:52: [data_structures] Node-Based Implementation of Stack
 - ◇ [data_structures] Node-Based Implementation of Stack: Diagram
- ◇ 11:29: [data_structures] Remarks on Node-Based Implementation of Stack
- ◇ 13:28: [data_structures] Queue ADT
- ◇ 14:43: [data_structures] Array Implementation of Queue
- ◇ 16:32: [data_structures] Remarks on Array Implementation of Queue

- ◇ 17:40: [data_structures] Array of Arrays Implementation of Queue
 - ◇ [data_structures] Array of Arrays Implementation of Queue: Diagram
- ◇ 22:03: [data_structures] Remarks on Array of Arrays Implementation of Queue
- ◇ 22:22: [data_structures] Node-Based Implementation of Queue
 - ◇ [data_structures] Node-Based Implementation of Queue: Diagram
- ◇ 22:51: [data_structures] Remarks on Node-Based Implementation of Queue
- ◇ 23:02: [data_structures] Trees
- ◇ 24:11: [data_structures] Tree Terminology (Continued 1)
- ◇ 24:42: [data_structures] Tree Terminology (Continued 2)
- ◇ 25:20: [data_structures] Binary Trees
- ◇ 25:58: [data_structures] Perfect and Complete Trees
- ◇ 26:24: [data_structures] Balanced Binary Trees
- ◇ 27:25: [data_structures] Node-Based Binary Tree
 - ◇ [data_structures] Node-Based Binary Tree: Diagram
 - ◇ [data_structures] Remarks on Node-Based Binary Tree
- ◇ 29:11: [data_structures] Array-Based Binary Tree
- ◇ 29:49: [data_structures] Array-Based Binary Tree: Diagram
 - ◇ [data_structures] Remarks on Array-Based Binary Tree
- ◇ 31:19: [data_structures] Binary Search Trees
- ◇ 33:33: [data_structures] Heaps
- ◇ 34:34: [data_structures] Set and Multiset ADTs
- ◇ 36:20: [data_structures] Map and Multimaps ADTs
 - ◇ [data_structures] Remarks on Implementation of Sets and Maps
- ◇ 38:04: [data_structures] Priority Queue ADT
- ◇ 41:01: [data_structures] Remarks on Priority Queue Implementations
- ◇ 41:40: [basics] References Versus Pointers
- ◇ 45:15: [basics] The const Qualifier
- ◇ 45:34: [basics] The const Qualifier and Non-Pointer/Non-Reference Types

5 Lecture 5 (2019-05-15) — Some C++ Review (Const and Other Stuff) [2019-05-15]

The following is a link to the full video:

- ◇ <https://youtu.be/lnDMJrwtA24> [duration: 00:50:13]

The following are links to particular offsets within the video:

- ◇ 00:00: [basics] The const Qualifier and Non-Pointer/Non-Reference Types
- ◇ 01:27: [basics] The const Qualifier and Pointer Types
- ◇ 05:07: [basics] The const Qualifier and Reference Types
- ◇ 09:39: [basics] The constexpr Qualifier for Variables
- ◇ 16:08: [basics] The const Qualifier and Functions
- ◇ 20:43: [basics] String Length Example: Not Const Correct
- ◇ 20:53: [basics] Square Example: Not Const Correct
 - ◇ [basics] Square Example: Const Correct
- ◇ 25:51: [basics] Square Example: Const Correct
- ◇ 27:29: [basics] Function Types and the const Qualifier
- ◇ 32:30: [exercises] [Q.1] What is Wrong With This Code?
 - ◇ [exercises] [Q.1] Solution: Use Const Qualifier Correctly

6 Lecture 6 (2019-05-17) — Some C++ Review (Const and Other Stuff), Compile-Time Computation [2019-05-17]

The following is a link to the full video:

- ◇ <https://youtu.be/KTT9boX3wyg> [duration: 00:51:14]

The following are links to particular offsets within the video:

- ◇ **00:00**: [exercises] [Q.2] What is Wrong With This Code?
 - ◇ [exercises] [Q.2] Solution: Use Const Qualifier Correctly
- ◇ **08:10**: [exercises] [Q.3] What is Wrong With This Code?
 - ◇ [exercises] [Q.3] Solution: Functions Should Be Inline
- ◇ **16:17**: [exercises] [Q.4] What is Wrong With This Code?
 - ◇ [exercises] [Q.4] Solution: Place Inline Function Definitions in Header File
- ◇ **19:22**: [exercises] [Q.5] What is Wrong With This Code?
 - ◇ [exercises] [Q.5] Solution 1: Explicit Template Instantiation
 - ◇ [exercises] [Q.5] Solution 2: Define Function Template in Header File
- ◇ **27:07**: [exercises] Remarks on Header Files and Function Declarations
- ◇ **32:33**: [exercises] [Q.6] What is Wrong With This Code?
 - ◇ [exercises] [Q.6] Solution: Place Default Arguments in Header File
- ◇ **41:02**: [basics] The constexpr Qualifier for Functions

7 Lecture 7 (2019-05-21) — Compile-Time Computation [2019-05-21]

The following is a link to the full video:

- ◇ <https://youtu.be/GZWsV7KpAw8> [duration: 00:48:50]

The following are links to particular offsets within the video:

- ◇ **00:30**: [basics] constexpr Function Example: power_int (Iterative)
- ◇ **15:55**: [basics] Compile-Time Versus Run-Time Computation
- ◇ **21:01**: [classes] constexpr Member Functions
- ◇ **23:19**: [classes] constexpr Constructors
- ◇ **24:49**: [classes] Example: constexpr Constructors and Member Functions
- ◇ **31:51**: [classes] Why constexpr Member Functions Are Not Implicitly Const
- ◇ **37:27**: [classes] Literal Types
- ◇ **44:26**: [classes] Example: Literal Types
- ◇ **46:48**: [classes] constexpr Variable Requirements

8 Lecture 8 (2019-05-22) — Compile-Time Computation, Temporary Objects [2019-05-22]

The following is a link to the full video:

- ◇ https://youtu.be/eULv_AiAFII [duration: 00:49:28]

The following are links to particular offsets within the video:

- ◇ **00:00**: [classes] Example: constexpr Variable Requirement Violations
- ◇ **02:03**: [classes] constexpr Function Requirements
- ◇ **06:22**: [classes] Example: constexpr Function Requirement Violations
- ◇ **10:50**: [classes] constexpr Constructor Requirements
- ◇ **12:42**: [classes] Example: constexpr Constructor Requirement Violations
- ◇ **15:16**: [classes] Example: constexpr and Accessing External State
- ◇ **18:15**: [classes] Example: constexpr and Immediate Initialization
- ◇ **21:55**: [classes] Debugging constexpr Functions
- ◇ **28:50**: [classes] Example: Debugging Strategies for constexpr Functions

- ◇ 30:55: [exercises] [Q.7] What is Wrong With This Code?
 - ◇ [exercises] [Q.7] Solution: Define Constexpr Function in Header
- ◇ 33:25: [exercises] [Q.8] What is Wrong With This Code?
 - ◇ [exercises] [Q.8] Answer: Invalid Constexpr Function
- ◇ 36:05: [exercises] [Q.9] What is Wrong With This Code?
 - ◇ [exercises] [Q.9] Solution: Initialize Constexpr Function Variables
- ◇ 40:48: [exercises] [Q.10] What is Wrong With This Code?
 - ◇ [exercises] [Q.10] Solution: Constexpr Requires Literal Types
- ◇ 42:16: [temporaries] Temporary Objects

9 Lecture 9 (2019-05-24) — Temporary Objects, Moving/Copying, Value Categories [2019-05-24]

The following is a link to the full video:

- ◇ <https://youtu.be/LhCHHfMh4Gg> [duration: 00:48:29]

The following are links to particular offsets within the video:

- ◇ 00:00: [temporaries] Temporary Objects
- ◇ 02:51: [temporaries] Temporary Objects (Continued)
- ◇ 06:51: [temporaries] Temporary Objects Example
- ◇ 07:54: [temporaries] Temporary Objects Example (Continued)
- ◇ 09:06: [temporaries] Prefix Versus Postfix Increment/Decrement
- ◇ 18:24: [rval_refs] Propagating Values: Copying and Moving
- ◇ 22:04: [rval_refs] Copying and Moving
- ◇ 23:50: [rval_refs] Buffer Example: Moving Versus Copying
- ◇ 25:09: [rval_refs] Buffer Example: Copying
- ◇ 27:49: [rval_refs] Buffer Example: Moving
- ◇ 30:55: [rval_refs] Moving Versus Copying
- ◇ 33:35: [lvalues] Value Categories of Expressions
- ◇ 36:39: [lvalues] Value Categories of Expressions (Continued)
- ◇ 40:36: [lvalues] Lvalues
- ◇ 43:39: [lvalues] Lvalues (Continued 1)

10 Lecture 10 (2019-05-28) — Value Categories, Moving/Copying [2019-05-28]

The following is a link to the full video:

- ◇ <https://youtu.be/C1ONBX9-vdo> [duration: 00:48:36]

The following are links to particular offsets within the video:

- ◇ 00:00: [lvalues] Lvalues (Continued 2)
- ◇ 03:14: [lvalues] Moving and Lvalues
- ◇ 07:17: [lvalues] Rvalues
- ◇ 11:33: [lvalues] Prvalues
- ◇ 14:11: [lvalues] Prvalues (Continued)
- ◇ 19:38: [lvalues] Xvalues
- ◇ 23:55: [lvalues] Moving and Rvalues
- ◇ 34:43: [lvalues] Moving and Lvalues/Rvalues
- ◇ 40:20: [lvalues] Moving/Copying and Lvalues/Rvalues

11 Lecture 11 (2019-05-29) — Copy Elision [2019-05-29]

The following is a link to the full video:

- ◇ <https://youtu.be/LCRKHycBhsQ> [duration: 00:48:31]

The following are links to particular offsets within the video:

- ◇ 00:00: [copy_elision] Copy Elision and Implicit Moving [title slide]
- ◇ 00:36: [copy_elision] Copy Elision
- ◇ 06:55: [copy_elision] Copy Elision and Returning by Value
- ◇ 31:11: [copy_elision] Return-By-Value Example 1: Summary
- ◇ 35:32: [copy_elision] Return-By-Value Example 2: Summary
- ◇ 38:54: [copy_elision] Example Where Copy Elision Allowed But Likely Impossible
- ◇ 44:09: [copy_elision] Copy Elision and Passing by Value

12 Lecture 12 (2019-05-31) — Copy Elision, Implicit Move [2019-05-31]

The following is a link to the full video:

- ◇ <https://youtu.be/QgfH-RFAFhI> [duration: 00:50:32]

The following are links to particular offsets within the video:

- ◇ 00:00: [copy_elision] Pass-By-Value Example: Summary
- ◇ 04:11: [copy_elision] Copy Elision and Initialization
- ◇ 21:27: [copy_elision] Mandatory Copy Elision Example: Factory Function
- ◇ 25:02: [copy_elision] Return Statements and Moving/Copying
- ◇ 36:36: [copy_elision] Example: Return Statements and Moving/Copying
- ◇ 40:38: [copy_elision] Use of std::move in Return Statements
- ◇ 43:03: [copy_elision] Example: Moving/Copying, Copy Elision, and Implicit Move a.k.a. [exercises] [Q.MC1] Copy, Move, or Copy Elision?

13 Lecture 13 (2019-06-04) — Copy Elision, Implicit Move, Exceptions [2019-06-04]

The following is a link to the full video:

- ◇ <https://youtu.be/yoA7fFfBRlI> [duration: 00:52:24]

The following are links to particular offsets within the video:

- ◇ 00:00: [exercises] [Q.MC1] Answer
- ◇ 09:44: [rval_refs] Allowing Move Semantics in Other Contexts via std::move
- ◇ 10:49: [rval_refs] Old-Style Swap
- ◇ 12:20: [rval_refs] Improved Swap
- ◇ 14:27: [rval_refs] Implication of Rvalue-Reference Type Function Parameters
- ◇ 17:34: [exceptions] Exceptions
- ◇ 18:52: [exceptions] The Problem
- ◇ 20:35: [exceptions] Traditional Error Handling
- ◇ 23:24: [exceptions] Example: Traditional Error Handling
- ◇ 25:09: [exceptions] Error Handling With Exceptions
- ◇ 27:55: [exceptions] Example: Exceptions
- ◇ 29:55: [exceptions] safe_divide Example: Traditional Error Handling
- ◇ 30:37: [exceptions] safe_divide Example: Exceptions
- ◇ 31:29: [exceptions] Exceptions Versus Traditional Error Handling
- ◇ 34:28: [exceptions] Exceptions
- ◇ 36:58: [exceptions] Standard Exception Classes
 - ◇ [exceptions] Standard Exception Classes (Continued 1)

- ◇ [exceptions] Standard Exception Classes (Continued 2)
- ◇ 37:42: [exceptions] Throwing Exceptions
- ◇ 38:39: [exceptions] Throwing Exceptions (Continued)
- ◇ 40:45: [exceptions] Catching Exceptions
- ◇ 41:41: [exceptions] Catching Exceptions (Continued)
- ◇ 43:29: [exceptions] Rethrowing Exceptions
- ◇ 44:23: [exceptions] Transfer of Control from Throw Site to Handler
- ◇ 50:22: [exceptions] Stack Unwinding Example

14 Lecture 14 (2019-06-05) — Exceptions [2019-06-05]

The following is a link to the full video:

- ◇ https://youtu.be/_jyR6ue12k4 [duration: 00:47:00]

The following are links to particular offsets within the video:

- ◇ 00:00: [exceptions] Stack Unwinding Example
- ◇ 08:38: [exceptions] Function Try Blocks
- ◇ 09:49: [exceptions] Exceptions and Construction/Destruction
- ◇ 14:06: [exceptions] Construction/Destruction Example
- ◇ 18:09: [exceptions] Function Try Block Example
- ◇ 24:53: [exceptions] The noexcept Specifier
- ◇ 29:13: [exceptions] The noexcept Specifier (Continued 1)
 - ◇ [exceptions] The noexcept Specifier (Continued 2)
- ◇ 30:34: [exceptions] The noexcept Specifier (Continued 3)
- ◇ 37:33: [exceptions] Exceptions and Function Calls
- ◇ 42:06: [exceptions] Avoiding Exceptions Due to Function Calls

15 Lecture 15 (2019-06-07) — Exceptions, Interval Arithmetic [2019-06-07]

The following is a link to the full video:

- ◇ <https://youtu.be/xMZ12vghJF4> [duration: 00:48:56]

The following are links to particular offsets within the video:

- ◇ 00:00: [exceptions] noexcept Operator
- ◇ 08:34: [exceptions] noexcept Operator (Continued)
- ◇ 17:00: [arithmetic] Interval Arithmetic
- ◇ 21:21: [arithmetic] Applications of Interval Arithmetic
- ◇ 24:11: [arithmetic] Real Interval Arithmetic
- ◇ 26:22: [arithmetic] Addition and Subtraction
- ◇ 27:54: [arithmetic] Multiplication and Division
- ◇ 28:46: [arithmetic] Floating-Point Interval Arithmetic
- ◇ 31:52: [arithmetic] Floating-Point Interval Arithmetic (Continued)
- ◇ 34:12: [arithmetic] Floating-Point Interval Arithmetic Operations
- ◇ 35:35: [arithmetic] Comparisons
- ◇ 44:18: [arithmetic] Setting and Querying Rounding Mode

16 Lecture 16 (2019-06-11) — Interval Arithmetic, Geometric Predicates and Applications [2019-06-11]

The following is a link to the full video:

- ◇ <https://youtu.be/Ec00zgwRPw4> [duration: 00:46:42]

The following are links to particular offsets within the video:

- ◇ 00:00: [arithmetic] Impact of Current Rounding Mode
- ◇ 03:55: [arithmetic] Rounding Mode Example
- ◇ 04:53: [arithmetic] Geometric Predicates
- ◇ 07:18: [arithmetic] Filtered Geometric Predicates
- ◇ 11:44: [arithmetic] Two-Dimensional Orientation Test
- ◇ 13:50: [arithmetic] Example: Two-Dimensional Orientation Test
- ◇ 14:16: [arithmetic] Convex Polygons
- ◇ 17:08: [arithmetic] Polygon Convexity Test
- ◇ 20:42: [arithmetic] Three-Dimensional Orientation Test
- ◇ 25:58: [arithmetic] Side-of-Oriented-Circle Test
- ◇ 28:37: [arithmetic] Preferred-Direction Test
- ◇ 30:32: [arithmetic] Triangulations
- ◇ 33:40: [arithmetic] Delaunay Triangulations
- ◇ 35:37: [arithmetic] Nonuniqueness of Delaunay Triangulations
 - ◇ [arithmetic] Comments on Delaunay Triangulations
- ◇ 39:37: [arithmetic] Edge Flips
- ◇ 42:21: [arithmetic] Locally-Delaunay Test
- ◇ 45:49: [arithmetic] Locally Preferred-Directions Delaunay Test

17 Lecture 17 (2019-06-12) — Geometric Predicates and Applications, Memory Management [2019-06-12]

The following is a link to the full video:

- ◇ <https://youtu.be/x3Z7Kxb32ew> [duration: 00:41:34]

The following are links to particular offsets within the video:

- ◇ 00:00: [arithmetic] Locally Preferred-Directions Delaunay Test [plus related slides]
- ◇ 08:08: [arithmetic] Lawson Local Optimization Procedure
- ◇ 11:32: [arithmetic] Finding Delaunay Triangulations with Lawson LOP
- ◇ 13:43: [data_structures] Naive Triangle-Mesh Data Structure
- ◇ 16:04: [data_structures] Naive Triangle-Mesh Data Structure Example
- ◇ 20:11: [data_structures] Half-Edge Data Structure
- ◇ 20:46: [data_structures] Half-Edge Data Structure (Continued)
- ◇ 30:05: [data_structures] Object File Format (OFF)
- ◇ 30:40: [data_structures] OFF Example (Triangle Mesh)
- ◇ 34:01: [memory_management] Memory Management
- ◇ 36:18: [memory_management] Potential Problems Arising in Memory Management
- ◇ 38:42: [memory_management] Alignment
- ◇ 39:06: [memory_management] The alignof Operator

18 Lecture 18 (2019-06-14) — Memory Management [2019-06-14]

The following is a link to the full video:

- ◇ <https://youtu.be/E31oR6H-Lv8> [duration: 00:41:56]

The following are links to particular offsets within the video:

- ◇ 00:09: [memory_management] The alignas Specifier
- ◇ 02:04: [memory_management] New Expressions
- ◇ 03:07: [memory_management] New Expressions (Continued)
- ◇ 05:49: [memory_management] Delete Expressions
- ◇ 07:22: [memory_management] Delete Expressions (Continued 1)

- ◇ 10:13: [memory_management] Delete Expressions (Continued 2)
- ◇ 11:58: [memory_management] Typical Strategy for Determining Array Size in Array Delete
- ◇ 19:21: [memory_management] New Expressions and Allocation
- ◇ 22:54: [memory_management] Allocation Function Overload Resolution
- ◇ 26:11: [memory_management] Allocation Function Overload Resolution (Continued)
- ◇ 29:03: [memory_management] New Expressions and Deallocation
- ◇ 30:37: [memory_management] Delete Expressions and Deallocation
- ◇ 31:04: [memory_management] Single-Object Operator New (i.e., operator new)
- ◇ 34:03: [memory_management] Single-Object Operator New Overloads
- ◇ 36:34: [memory_management] Single-Object Operator New Overloads (Continued)
- ◇ 37:28: [memory_management] Single-Object Operator New Examples

19 Lecture 19 (2019-06-18) — Memory Management [2019-06-18]

The following is a link to the full video:

- ◇ https://youtu.be/W_GazLV6qcg [duration: 00:48:04]

The following are links to particular offsets within the video:

- ◇ 00:00: [memory_management] Array Operator New (i.e., operator new[])
- ◇ 01:50: [memory_management] Array Operator New Overloads
- ◇ 02:57: [memory_management] Array Operator New Overloads (Continued)
- ◇ 03:31: [memory_management] Array Operator New Examples
- ◇ 11:54: [memory_management] Single-Object Operator Delete (i.e., operator delete)
- ◇ 13:44: [memory_management] Single-Object Operator Delete Overloads
- ◇ 14:16: [memory_management] Single-Object Operator Delete Examples
- ◇ 20:57: [memory_management] Array Operator Delete (i.e., operator delete[])
- ◇ 21:36: [memory_management] Array Operator Delete Overloads
- ◇ 21:42: [memory_management] Array Operator Delete Examples
- ◇ 22:14: [memory_management] Motivation for Placement New
 - ◇ [memory_management] Motivation for Placement New: Diagram
- ◇ 31:00: [memory_management] Placement New
- ◇ 36:59: [memory_management] Placement New Examples
- ◇ 43:24: [memory_management] Direct Destructor Invocation
- ◇ 46:15: [memory_management] Pseudodestructors

20 Lecture 20 (2019-06-19) — Memory Management [2019-06-19]

The following is a link to the full video:

- ◇ <https://youtu.be/xK0bs70kzC8> [duration: 00:49:07]

The following are links to particular offsets within the video:

- ◇ 00:00: [memory_management] std::addressof Function Template
- ◇ 02:29: [memory_management] std::addressof Example
- ◇ 04:25: [memory_management] The std::aligned_storage Class Template
- ◇ 05:48: [memory_management] Optional Value Example
- ◇ 07:17: [memory_management] Optional Value Example: Diagram
- ◇ 08:12: [memory_management] Optional Value Example: optval.hpp
- ◇ 19:57: [memory_management] Optional Value Example: User Code
- ◇ 22:10: [memory_management] Handling Uninitialized Storage
- ◇ 22:55: [memory_management] Functions for Uninitialized Storage
- ◇ 26:37: [memory_management] Functions for Uninitialized Storage (Continued)
- ◇ 27:47: [memory_management] Some Example Implementations
- ◇ 31:04: [memory_management] Bounded Array Example

- ◇ 31:19: [memory_management] Bounded Array Example: Diagram
- ◇ 32:46: [memory_management] Bounded Array Example: aligned_buffer.hpp
- ◇ 34:44: [memory_management] Bounded Array Example: array.hpp (1)
- ◇ 39:00: [memory_management] Bounded Array Example: array.hpp (2)
- ◇ 44:22: [memory_management] Bounded Array Example: array.hpp (3)
- ◇ 48:40: [memory_management] Bounded Array Example: array.hpp (4)

21 Lecture 21 (2019-06-21) — Memory Management, Intrusive Containers, Pointers to Members [2019-06-21]

The following is a link to the full video:

- ◇ <https://youtu.be/Tlo0KliV-xY> [duration: 00:49:10]

The following are links to particular offsets within the video:

- ◇ 00:00: [memory_management] Vector Example
- ◇ 01:48: [memory_management] Vector Example: Diagram
- ◇ 02:43: [memory_management] Vector Example: vec.hpp (1)
- ◇ 06:55: [memory_management] Vector Example: vec.hpp (2)
- ◇ 12:48: [memory_management] Vector Example: vec.hpp (3)
- ◇ 17:01: [memory_management] Vector Example: vec.hpp (4)
- ◇ 20:49: [memory_management] Vector Example: vec.hpp (5)
- ◇ 24:02: [memory_management] Vector Example: vec.hpp (6)
- ◇ 27:38: [data_structures] Intrusive Containers
- ◇ 33:25: [data_structures] Shortcomings of Non-Intrusive Containers
- ◇ 35:28: [data_structures] Advantages of Intrusive Containers
- ◇ 38:27: [data_structures] Disadvantages of Intrusive Containers
- ◇ 42:40: [data_structures] Disadvantages of Intrusive Containers (Continued)
- ◇ 45:21: [classes] Pointers to Members
- ◇ 47:58: [classes] Pointers to Members (Continued)

22 Lecture 22 (2019-06-25) — Pointers to Members, Intrusive Containers, Caches [2019-06-25]

The following is a link to the full video:

- ◇ <https://youtu.be/3rCHYD5VE2U> [duration: 00:52:44]

The following are links to particular offsets within the video:

- ◇ 00:00: [classes] Pointers to Members for Data Members
- ◇ 06:05: [classes] Pointers to Members Example: Accumulate
- ◇ 14:53: [data_structures] Intrusive Doubly-Linked List With Sentinel Node
 - ◇ [data_structures] Intrusive Doubly-Linked List With Sentinel Node: Code (Continued)
 - ◇ [data_structures] Intrusive Doubly-Linked List With Sentinel Node: Code
 - ◇ [data_structures] Intrusive Doubly-Linked List With Sentinel Node: Diagram
- ◇ 25:39: [data_structures] Remarks on Intrusive Doubly-Linked List With Sentinel Node
- ◇ 25:52: [data_structures] Examples of Intrusive Containers
- ◇ 27:03: [cache] The Memory Latency Problem
- ◇ 28:32: [cache] Principle of Locality
- ◇ 31:05: [cache] Memory Hierarchy
- ◇ 32:48: [cache] Caches
- ◇ 35:57: [cache] Memory and Cache
- ◇ 37:38: [cache] Block Placement
- ◇ 40:04: [cache] Block Placement (Continued)

- ◇ 42:35: [cache] Direct-Mapped Cache Example
- ◇ 43:31: [cache] K-Way Set-Associative Cache Example
- ◇ 44:28: [cache] Fully Associative Cache
- ◇ 45:03: [cache] Block Identification
- ◇ 46:43: [cache] Decomposition of Memory Address
- ◇ 48:53: [cache] Block Replacement
- ◇ 50:26: [cache] Write Policy

23 Lecture 23 (2019-06-26) — Caches, Cache-Efficient Algorithms [2019-06-26]

The following is a link to the full video:

- ◇ <https://youtu.be/ZV3L0rsHuV0> [duration: 00:50:24]

The following are links to particular offsets within the video:

- ◇ 00:00: [cache] Cache Misses
- ◇ 02:14: [cache] Virtual Memory
- ◇ 03:20: [cache] Virtual Address Space
- ◇ 05:38: [cache] Address Translation
- ◇ 07:21: [supplemental] [Q.C2] Virtual Memory Exercise
- ◇ 08:39: [supplemental] [Q.C2] Virtual Memory Exercise (Continued)
- ◇ 14:03: [cache] Translation Lookaside Buffer (TLB)
- ◇ 15:59: [cache] Virtual and Physical Caches
- ◇ 17:28: [cache] Virtual Versus Physical Caches
- ◇ 19:37: [cache] Virtually-Indexed Physically-Tagged (VIPT) Caches
- ◇ 20:15: [cache] VIPT Cache Example
- ◇ 23:06: [cache] Cache Performance
- ◇ 23:50: [cache] Intel Core i7
- ◇ 24:42: [cache] ARM Cortex A8
- ◇ 25:43: [cache] Cache-Efficient Algorithms
- ◇ 26:56: [cache] Code Transformations to Improve Cache Efficiency
- ◇ 28:30: [data_structures] Row-Major Versus Column-Major Order
- ◇ 29:42: [cache] Array Merging Example
- ◇ 31:50: [cache] Loop Interchange Example
- ◇ 33:17: [cache] Loop Fusion Example
- ◇ 35:25: [cache] Blocking Example
- ◇ 37:20: [cache] Blocking Example (Continued 0.5)
- ◇ 40:54: [cache] Blocking Example (Continued 1)
- ◇ 42:11: [cache] Blocking Example (Continued 2)
- ◇ 44:48: [cache] Cache-Aware Versus Cache-Oblivious Algorithms
- ◇ 47:24: [cache] Tall Caches

24 Lecture 24 (2019-06-28) — Cache-Efficient Algorithms [2019-06-28]

The following is a link to the full video:

- ◇ <https://youtu.be/BC-e0hw6kAQ> [duration: 00:44:45]

The following are links to particular offsets within the video:

- ◇ 00:00: [cache] Idealized Cache Model
- ◇ 02:20: [cache] Remarks on Assumption of Optimal-Replacement Policy
- ◇ 03:45: [cache] Cache-Oblivious Algorithms
- ◇ 04:32: [cache] Scanning

- ◇ 09:44: [cache] Array Reversal
- ◇ 14:48: [cache] Naive Matrix Transposition
- ◇ 16:29: [cache] Naive Matrix Transposition: Performance
- ◇ 21:31: [cache] Cache-Oblivious Matrix Transposition
- ◇ 22:50: [cache] Cache-Oblivious Matrix Transposition (Continued)
- ◇ 24:47: [cache] Cache-Oblivious Matrix Transposition Example 1A [Part 1]
- ◇ 26:52: [handout] Transpose Algorithm Pseudocode
- ◇ 29:38: [handout] Matrix Subblock Characterization
- ◇ 30:57: [cache] Cache-Oblivious Matrix Transposition Example 1A [Part 2]
- ◇ 32:48: [cache] Cache-Oblivious Matrix Transposition Example 2
- ◇ 34:47: [cache] Cache-Oblivious Matrix Transposition: Performance
- ◇ 36:40: [cache] Naive Matrix Multiplication
- ◇ 39:07: [cache] Naive Matrix Multiplication: Performance

25 Lecture 25 (2019-07-03) — Cache-Efficient Algorithms, Concurrency [2019-07-03]

The following is a link to the full video:

- ◇ <https://youtu.be/NTUnun-YjyQ> [duration: 00:46:39]

The following are links to particular offsets within the video:

- ◇ 00:00: [cache] Cache-Oblivious Matrix Multiplication
- ◇ 02:16: [cache] Cache-Oblivious Matrix Multiplication (Continued 1)
- ◇ 05:55: [cache] Cache-Oblivious Matrix Multiplication (Continued 2)
- ◇ 06:44: [cache] Cache-Oblivious Matrix Multiplication Example 1
- ◇ 13:02: [cache] Cache-Oblivious Matrix Multiplication: Performance
- ◇ 15:14: [cache] Cache-Oblivious Matrix Multiplication Revisited
- ◇ 17:52: [cache] Cache-Oblivious Matrix Multiplication Revisited Example 2
- ◇ 20:48: [cache] Discrete Fourier Transform (DFT)
- ◇ 24:03: [cache] Cache-Oblivious Fast Fourier Transform (FFT)
- ◇ 29:41: [cache] Example: Four-Point DFT
- ◇ 32:15: [cache] Example: Four-Point DFT (Continued 1)
- ◇ 33:41: [cache] Example: Four-Point DFT (Continued 2)
- ◇ 34:01: [cache] Cache-Oblivious FFT: Performance
- ◇ 37:40: [concurrency] Processors
- ◇ 39:38: [concurrency] Processors (Continued)
- ◇ 41:29: [concurrency] Why Multicore Processors?
- ◇ 44:35: [concurrency] Concurrency

26 Lecture 26 (2019-07-05) — Concurrency [2019-07-05]

The following is a link to the full video:

- ◇ https://youtu.be/U__YDW14DA0 [duration: 00:47:06]

The following are links to particular offsets within the video:

- ◇ 00:00: [concurrency] Why Multithreading?
- ◇ 03:51: [concurrency] Memory Model
- ◇ 06:47: [concurrency] Sequential Consistency (SC)
- ◇ 09:36: [concurrency] Sequential-Consistency (SC) Memory Model
- ◇ 12:34: [concurrency] Load/Store Reordering Example: Single Thread
- ◇ 15:20: [concurrency] Load/Store Reordering Example: Multiple Threads
- ◇ 20:00: [concurrency] Atomicity of Memory Operations

- ◇ 21:46: [concurrency] Data Races
- ◇ 25:34: [concurrency] Torn Reads
- ◇ 28:57: [concurrency] Torn Writes
- ◇ 31:11: [concurrency] SC Data-Race Free (SC-DRF) Memory Model
- ◇ 34:36: [concurrency] C++ Memory Model
- ◇ 39:53: [concurrency] The std::thread Class
- ◇ 43:03: [concurrency] The std::thread Class (Continued)

27 Lecture 27 (2019-07-09) — Concurrency [2019-07-09]

The following is a link to the full video:

- ◇ <https://youtu.be/1CkqUsDFPnE> [duration: 00:45:55]

The following are links to particular offsets within the video:

- ◇ 00:00: [concurrency] std::thread Members
- ◇ 01:49: [concurrency] std::thread Members (Continued)
- ◇ 03:06: [concurrency] Example: Hello World With Threads [First Half]
- ◇ 05:15: [lambdas] Hello World Program Revisited
- ◇ 09:22: [lambdas] Linear-Function Functor Example
- ◇ 21:27: [concurrency] Example: Hello World With Threads [Second Hal]
- ◇ 23:00: [concurrency] Example: Thread-Function Argument Passing (Copy/Move Semantics)
- ◇ 25:23: [concurrency] Example: Thread-Function Argument Passing (Reference Semantics)
- ◇ 30:32: [concurrency] Example: Moving Threads
- ◇ 33:16: [concurrency] Example: Lifetime Bug
- ◇ 36:38: [concurrency] The std::thread Class and Exception Safety
- ◇ 38:21: [concurrency] The std::thread Class and Exception Safety (Continued)

28 Lecture 28 (2019-07-10) — Concurrency [2019-07-10]

The following is a link to the full video:

- ◇ https://youtu.be/U_hiEvfgf0Q [duration: 00:43:18]

The following are links to particular offsets within the video:

- ◇ 00:00: [concurrency] Happens-Before Relationships
- ◇ 03:12: [concurrency] “Earlier in Time” Versus Happens Before
- ◇ 09:02: [concurrency] Sequenced-Before Relationships
- ◇ 10:21: [concurrency] Sequenced-Before Relationships (Continued)
- ◇ 11:14: [concurrency] Inter-Thread Happens-Before Relationships
- ◇ 12:37: [concurrency] Summary of Happens-Before Relationships
- ◇ 13:15: [concurrency] Synchronizes-With Relationships
- ◇ 17:01: [concurrency] Examples of Synchronizes-With Relationships
- ◇ 17:50: [concurrency] Synchronizes-With Relationship: Thread Create and Join
- ◇ 23:19: [concurrency] Shared Data
- ◇ 24:50: [concurrency] Race Conditions
- ◇ 28:42: [concurrency] Critical Sections
- ◇ 30:43: [concurrency] Data-Race Example
- ◇ 32:33: [concurrency] Example: Data Race (Counter)
- ◇ 34:46: [concurrency] Example: Data Race and/or Race Condition (IntSet)

29 Lecture 29 (2019-07-12) — Concurrency [2019-07-12]

The following is a link to the full video:

- ◇ https://youtu.be/nH11640_vh0 [duration: 00:47:21]

The following are links to particular offsets within the video:

- ◇ 00:00: [concurrency] Mutexes
- ◇ 03:10: [concurrency] The std::mutex Class
- ◇ 05:44: [concurrency] std::mutex Members
- ◇ 08:02: [concurrency] Example: Avoiding Data Race Using Mutex (Counter) (mutex)
- ◇ 11:00: [concurrency] Synchronizes-With Relationships: Mutex Lock/Unlock
- ◇ 18:57: [concurrency] The std::scoped_lock Template Class
- ◇ 21:22: [concurrency] std::scoped_lock Members
- ◇ 22:14: [concurrency] Example: Avoiding Data Race Using Mutex (Counter) (scoped_lock)
- ◇ 24:22: [concurrency] Example: Avoiding Data Race Using Mutex (IntSet) (scoped_lock)
- ◇ 32:44: [concurrency] Acquisition of Multiple Locks
- ◇ 35:26: [concurrency] Example: Acquiring Two Locks for Swap (Incorrect)
- ◇ 38:56: [concurrency] Example: Acquiring Two Locks for Swap [scoped_lock]
- ◇ 39:20: [concurrency] The std::unique_lock Template Class
- ◇ 41:55: [concurrency] std::unique_lock Members
- ◇ 43:28: [concurrency] std::unique_lock Members (Continued)
- ◇ 43:55: [concurrency] Example: Avoiding Data Race Using Mutex (Counter) (unique_lock)

30 Lecture 30 (2019-07-16) — Concurrency [2019-07-16]

The following is a link to the full video:

- ◇ <https://youtu.be/0LT1FMvkIoA> [duration: 00:44:37]

The following are links to particular offsets within the video:

- ◇ 00:00: [concurrency] The std::lock Template Function
- ◇ 01:01: [concurrency] Example: Acquiring Two Locks for Swap [unique_lock and lock]
- ◇ 01:51: [concurrency] Static Local Variable Initialization and Thread Safety
- ◇ 03:16: [concurrency] Condition Variables
- ◇ 07:40: [concurrency] The std::condition_variable Class
- ◇ 13:26: [concurrency] std::condition_variable Members
- ◇ 14:30: [concurrency] std::condition_variable Members (Continued)
- ◇ 15:32: [concurrency] Example: Condition Variable (IntStack)
- ◇ 27:50: [concurrency] Latches
- ◇ 29:56: [concurrency] Latch Example: User Code
- ◇ 32:03: [concurrency] Latch Example: latch_1.hpp
- ◇ 37:15: [concurrency] The std::condition_variable_any Class
- ◇ 38:44: [concurrency] Thread Pools
- ◇ 42:07: [concurrency] Simple Thread Pool Interface Example

31 Lecture 31 (2019-07-17) — Concurrency, More Exceptions [2019-07-17]

The following is a link to the full video:

- ◇ https://youtu.be/DeLP03S_cVM [duration: 00:45:53]

The following are links to particular offsets within the video:

- ◇ 00:00: [concurrency] Simple Thread Pool Interface Example
- ◇ 03:44: [exceptions] Resource Management
- ◇ 05:31: [exceptions] Resource Leak Example
- ◇ 07:17: [exceptions] Cleanup
- ◇ 08:43: [exceptions] Exception Safety and Exception Guarantees
- ◇ 13:13: [exceptions] Exception Guarantees
- ◇ 20:24: [exceptions] Resource Acquisition Is Initialization (RAII)

- ◇ 21:43: [exceptions] Resource Leak Example Revisited
- ◇ 30:25: [exceptions] RAII Example: Stream Formatting Flags
- ◇ 35:15: [exceptions] Other RAII Examples
- ◇ 37:55: [exceptions] Appropriateness of Using Exceptions
- ◇ 41:40: [exceptions] Enforcing Invariants: Exceptions Versus Assertions

32 Lecture 32 (2019-07-19) — Smart Pointers [2019-07-19]

The following is a link to the full video:

- ◇ https://youtu.be/_VV1B1J97ug [duration: 00:42:43]

The following are links to particular offsets within the video:

- ◇ 00:00: [smart_ptrs] Memory Management, Ownership, and Raw Pointers
- ◇ 02:36: [smart_ptrs] Smart Pointers
- ◇ 05:15: [smart_ptrs] The std::unique_ptr Template Class
- ◇ 08:27: [smart_ptrs] The std::unique_ptr Template Class (Continued)
- ◇ 10:37: [handout] Move Operation for unique_ptr
- ◇ 13:17: [handout] Why unique_ptr Is Not Copyable
- ◇ 16:16: [smart_ptrs] std::unique_ptr Member Functions
- ◇ 17:41: [smart_ptrs] std::unique_ptr Member Functions (Continued)
- ◇ 18:13: [smart_ptrs] std::unique_ptr Example 1
- ◇ 21:48: [smart_ptrs] Temporary Heap-Allocated Objects
- ◇ 24:07: [smart_ptrs] Decoupled Has-A Relationship
- ◇ 28:19: [smart_ptrs] The std::shared_ptr Template Class
- ◇ 31:25: [smart_ptrs] The std::shared_ptr Template Class (Continued)
- ◇ 39:09: [smart_ptrs] std::shared_ptr Reference Counting Example
 - ◇ [smart_ptrs] std::shared_ptr Reference Counting Example (Continued 1)
 - ◇ [smart_ptrs] std::shared_ptr Reference Counting Example (Continued 2)

33 Lecture 33 (2019-07-23) — Smart Pointers, Vectorization [2019-07-23]

The following is a link to the full video:

- ◇ https://youtu.be/D_8Hfchp09A [duration: 00:48:07]

The following are links to particular offsets within the video:

- ◇ 00:00: [smart_ptrs] std::shared_ptr Member Functions
- ◇ 00:48: [smart_ptrs] std::shared_ptr Member Functions (Continued)
- ◇ 02:23: [smart_ptrs] Prefer Use of std::make_shared
- ◇ 04:08: [smart_ptrs] std::shared_ptr Example
- ◇ 12:31: [smart_ptrs] std::shared_ptr and const
- ◇ 13:46: [smart_ptrs] Factory Function
- ◇ 15:17: [smart_ptrs] Example: Shared Pointer to Subobject of Managed Object
- ◇ 18:04: [smart_ptrs] Example: Shared Pointer to Subobject of Managed Object (Continued 1)
- ◇ 20:51: [smart_ptrs] Example: Shared Pointer to Subobject of Managed Object (Continued 2)
- ◇ 24:35: [smart_ptrs] Example: Shared Pointer to Subobject of Managed Object (Continued 3)
- ◇ 25:17: [smart_ptrs] Example: Shared Pointer to Subobject of Managed Object (Continued 4)
- ◇ 27:36: [smart_ptrs] Example: std::shared_ptr
- ◇ 30:00: [smart_ptrs] Example: std::shared_ptr (Continued)
- ◇ 32:58: [vectorization] Vector Processing
- ◇ 34:33: [vectorization] Scalar Versus Vector Instructions
- ◇ 36:10: [vectorization] Vector-Memory and Vector-Register Architectures
- ◇ 38:13: [vectorization] Vector-Register Architectures
- ◇ 40:56: [vectorization] Vector Extensions

- ◇ 42:53: [vectorization] Intel x86/x86-64 Streaming SIMD Extensions (SSE)
- ◇ 44:18: [vectorization] Intel x86/x86-64 Advanced Vector Extensions (AVX)
- ◇ 46:09: [vectorization] ARM NEON

34 Lecture 34 (2019-07-24) — Vectorization [2019-07-24]

The following is a link to the full video:

- ◇ <https://youtu.be/Thv9FA60XH8> [duration: 00:47:52]

The following are links to particular offsets within the video:

- ◇ 00:00: [vectorization] Checking for Processor Vector Support on Linux
- ◇ 01:06: [vectorization] Vectorization
- ◇ 03:14: [vectorization] Conceptualizing Loop Vectorization
- ◇ 06:56: [vectorization] Approaches to Vectorization
- ◇ 14:17: [vectorization] Auto-Vectorization
- ◇ 16:34: [vectorization] GCC Compiler and Vectorization
- ◇ 17:36: [vectorization] GCC Compiler Options Related to Vectorization
- ◇ 18:58: [vectorization] GCC Compiler Options Related to Vectorization (Continued)
- ◇ 21:09: [vectorization] Clang Compiler and Vectorization
- ◇ 21:39: [vectorization] Clang Compiler Options Related to Vectorization
- ◇ 22:58: [vectorization] Assessing Quality of Vectorized Code
- ◇ 24:48: [vectorization] Assessing Quality of Vectorized Code (Continued)
- ◇ 27:57: [vectorization] Auto-Vectorization with Hints
- ◇ 29:43: [vectorization] Obstacles to Vectorization
- ◇ 34:04: [vectorization] Data Dependencies and Vectorization
- ◇ 35:05: [vectorization] Flow Dependencies
- ◇ 37:38: [vectorization] Flow Dependence Example 1
- ◇ 40:34: [vectorization] Flow Dependence Example 1: Sequential Loop
- ◇ 41:54: [vectorization] Flow Dependence Example 1: Vectorized Loop
- ◇ 44:38: [vectorization] Flow Dependence Example 2
- ◇ 46:55: [vectorization] Output Dependencies

35 Lecture 35 (2019-07-26) — Vectorization [2019-07-26]

The following is a link to the full video:

- ◇ <https://youtu.be/dIpS5ME6SKs> [duration: 00:49:29]

The following are links to particular offsets within the video:

- ◇ 00:00: [vectorization] Control-Flow Dependencies and Vectorization
- ◇ 02:07: [vectorization] Aliasing
- ◇ 04:15: [vectorization] Aliasing and Optimization: An Example
- ◇ 06:18: [vectorization] Aliasing and Vectorization: An Example
- ◇ 12:29: [vectorization] The `__restrict__` Keyword
- ◇ 19:13: [vectorization] Noncontiguous Memory Accesses
- ◇ 20:54: [vectorization] Data Alignment
- ◇ 24:57: [vectorization] Handling Misaligned Data
- ◇ 26:54: [handout] Example: Handling Misaligned Data
- ◇ 29:44: [vectorization] Controlling Alignment of Data
- ◇ 32:07: [vectorization] Informing Compiler of Data Alignment
- ◇ 35:56: [vectorization] Profitability of Vectorization
- ◇ 38:00: [vectorization] Vectorization Example: Version 1
- ◇ 40:12: [vectorization] Vectorization Example: Version 2
- ◇ 41:31: [vectorization] Vectorization Example: Version 3

- ◇ 45:33: [vectorization] Vectorization Example: Invoking add Function
- ◇ 47:02: [vectorization] Basic Requirements for Vectorizable Loops

36 Lecture 36 (2019-07-30) — Vectorization [2019-07-30]

The following is a link to the full video:

- ◇ <https://youtu.be/gjnI4khPj5k> [duration: 00:14:39]

The following are links to particular offsets within the video:

- ◇ 00:00: [vectorization] OpenMP SIMD Constructs
- ◇ 02:09: [vectorization] OpenMP simd Pragma
- ◇ 05:28: [vectorization] OpenMP declare simd Pragma
- ◇ 07:05: [vectorization] OpenMP SIMD-Related Pragma Clauses
- ◇ 08:29: [vectorization] OpenMP SIMD-Related Pragma Clauses (Continued)
- ◇ 08:50: [vectorization] Example: Vectorized Loop
- ◇ 12:34: [vectorization] Example: Vectorized Loop and Function

37 Lecture 37 (2019-07-31) — Final Course Wrap-Up [2019-07-31]

The following is a link to the full video:

- ◇ <https://youtu.be/li216eCidB0> [duration: 00:30:16]

The following are links to particular offsets within the video:

- ◇ 00:00: [wrapup] Any Questions About the Final Exam?
- ◇ 14:31: [wrapup] Open Discussion on Ways to Improve Course
- ◇ 15:56: [wrapup] Lecture Slides and Videos
- ◇ 20:45: [wrapup] Course Experience Survey (CES)

38 Extra (2019-07-25) — Preliminary Information for Final Exam [2019-07-25]

The following is a link to the full video:

- ◇ <https://youtu.be/HQx3F--UzYA> [duration: 00:13:48]

The following are links to particular offsets within the video:

- ◇ 00:00: Final Exam Information

39 Getting Started — Compiling and Linking [2017-04-13]

The following is a link to the full video:

- ◇ <https://youtu.be/w5s7XgnLHoo> [duration: 00:13:19]

The following are links to particular offsets within the video:

- ◇ 00:00: [start] Title
- ◇ 00:16: [start] Section: Getting Started
- ◇ 00:42: [start] Section: Building Programs: Compiling and Linking
- ◇ 00:45: [start] hello Program: hello.cpp
- ◇ 02:52: [start] Software Build Process
- ◇ 03:26: [start] Software Build Process
- ◇ 05:10: [start] GNU Compiler Collection (GCC) C++ Compiler
- ◇ 05:54: [start] GNU Compiler Collection (GCC) C++ Compiler
- ◇ 11:28: [start] Manually Building hello Program

40 Version Control — Introduction [2017-04-06]

The following is a link to the full video:

- ◇ https://youtu.be/9s9_DLH1jaY [duration: 00:06:50]

The following are links to particular offsets within the video:

- ◇ 00:00: [vcs] Title
- ◇ 00:16: [vcs] Section: Version Control Systems
- ◇ 00:21: [vcs] Version Control Systems
- ◇ 01:46: [vcs] Centralized Version Control
- ◇ 03:38: [vcs] Distributed Version Control
- ◇ 04:34: [vcs] Pros and Cons of Distributed Version Control

41 Git — Introduction [2017-04-08]

The following is a link to the full video:

- ◇ https://youtu.be/scm2kxsX_Rk [duration: 00:17:25]

The following are links to particular offsets within the video:

- ◇ 00:00: [git] Title
- ◇ 00:16: [git] Section: Git
- ◇ 00:22: [git] Git
- ◇ 01:30: [git] Users of Git
- ◇ 01:55: [git] Repositories
- ◇ 02:42: [git] Revision History and Directed Acyclic Graphs
- ◇ 04:31: [git] Branching Workflows
- ◇ 05:48: [git] Local Picture
- ◇ 07:32: [git] Local and Remote Picture
- ◇ 09:21: [git] HEAD
- ◇ 10:46: [git] Remote-Tracking Branches
- ◇ 11:57: [git] Remote-Tracking Branches (Continued)
- ◇ 14:27: [git] Git Configuration
- ◇ 15:16: [git] Git on One Slide

42 Git — Demonstration [2017-04-05]

The following is a link to the full video:

- ◇ <https://youtu.be/8VHC7vzWihw> [duration: 00:13:10]

The following are links to particular offsets within the video:

- ◇ 00:00: [git] Title
- ◇ 00:16: [git] Section: Git Demonstration
- ◇ 00:21: [git] Demonstration

43 Build Systems — Introduction [2017-04-12]

The following is a link to the full video:

- ◇ https://youtu.be/FPcK_swg-f8 [duration: 00:03:25]

The following are links to particular offsets within the video:

- ◇ 00:00: [build] Title
- ◇ 00:16: [build] Section: Build Tools
- ◇ 00:25: [build] Build Tools
- ◇ 02:23: [build] Examples of Build Tools

44 Make — Introduction [2017-04-12]

The following is a link to the full video:

- ◇ https://youtu.be/FsGAM2pXP_Y [duration: 00:27:56]

The following are links to particular offsets within the video:

- ◇ 00:00: [make] Title
- ◇ 00:16: [make] Section: Make
- ◇ 00:20: [make] Make
- ◇ 02:34: [make] make Command
- ◇ 05:47: [make] Makefiles
- ◇ 07:18: [make] Makefiles (Continued 1)
- ◇ 09:46: [make] Makefiles (Continued 2)
- ◇ 11:43: [make] Makefile for hello Program
- ◇ 17:00: [make] Makefile for hello Program
- ◇ 17:49: [make] Makefile for hello Program
- ◇ 18:48: [make] Makefile for hello Program
- ◇ 26:47: [make] Commentary on Makefile for hello Program

45 CMake — Introduction [2017-04-16]

The following is a link to the full video:

- ◇ <https://youtu.be/Ak6cGZshduY> [duration: 00:22:13]

The following are links to particular offsets within the video:

- ◇ 00:00: [cmake] Title
- ◇ 00:16: [cmake] Section: CMake
- ◇ 00:21: [cmake] CMake
- ◇ 01:30: [cmake] Users of CMake
- ◇ 01:47: [cmake] Build Process
- ◇ 03:16: [cmake] Build Process (Diagram)
- ◇ 03:43: [cmake] CMake Basics
- ◇ 06:41: [cmake] In-Source Versus Out-of-Source Builds
- ◇ 08:37: [cmake] The cmake Command
- ◇ 09:41: [cmake] The cmake Command (Options)
- ◇ 10:37: [cmake] The cmake Command (Continued 1)
- ◇ 12:18: [cmake] The cmake Command for Building
- ◇ 13:28: [cmake] The cmake Command for Building (Command Usage)
- ◇ 14:59: [cmake] Hello World Example [Part 1]
- ◇ 17:19: [cmake] Hello World Example [Part 2]
- ◇ 17:56: [cmake] Hello World Demonstration [Part 1]
- ◇ 20:03: [cmake] Hello World Demonstration [Part 2]

46 CMake — Examples [2017-04-18]

The following is a link to the full video:

- ◇ <https://youtu.be/cDWOECgupDg> [duration: 00:27:43]

The following are links to particular offsets within the video:

- ◇ 00:00: [cmake] Title
- ◇ 00:16: [cmake] Section: Examples
- ◇ 00:21: [cmake] OpenGL/GLUT Example
- ◇ 00:44: [cmake] OpenGL/GLUT Example: Source Code
- ◇ 01:05: [cmake] OpenGL/GLUT Example: CMakeLists File

- ◇ 03:48: [cmake] OpenGL/GLUT Example: Demonstration
- ◇ 05:25: [cmake] CGAL Example
- ◇ 05:57: [cmake] CGAL Example: Source Code
- ◇ 06:20: [cmake] CGAL Example: CMakeLists File
- ◇ 08:43: [cmake] CGAL Example: Demonstration
- ◇ 10:25: [cmake] HG2G Example: Overview
- ◇ 11:09: [cmake] HG2G Example: Library Source Code
- ◇ 12:36: [cmake] HG2G Example: Application Source Code
- ◇ 13:13: [cmake] HG2G Example: CMakeLists Files
- ◇ 16:13: [cmake] HG2G Example: CMakeLists Files
- ◇ 17:24: [cmake] HG2G Example: CMakeLists Files (Continued 1)
- ◇ 21:01: [cmake] HG2G Example: Demonstration [Part 1]
- ◇ 25:46: [cmake] HG2G Example: Demonstration [Part 2]

47 Basics — Introduction [2015-04-06]

The following is a link to the full video:

- ◇ <https://youtu.be/mPlwuVWKQmg> [duration: 00:06:07]

The following are links to particular offsets within the video:

- ◇ 00:00: [basics] Title
- ◇ 00:17: [basics] Disclaimer
- ◇ 00:40: [basics] Section: C++ Basics
- ◇ 00:45: [basics] The C++ Programming Language
- ◇ 02:59: [basics] Comments
- ◇ 04:10: [basics] Identifiers
- ◇ 05:44: [basics] Reserved Keywords

48 Basics — Objects, Types, and Values [2015-04-08]

The following is a link to the full video:

- ◇ <https://youtu.be/FlbBgg5IamY> [duration: 01:09:06]

The following are links to particular offsets within the video:

- ◇ 00:00: [basics] Title
- ◇ 00:17: [basics] Section: Objects, Types, and Values
- ◇ 00:23: [basics] Fundamental Types
- ◇ 02:08: [basics] Fundamental Types (Continued)
- ◇ 04:21: [basics] Literals
- ◇ 05:02: [basics] Character Literals
- ◇ 06:34: [basics] Character Literals (Continued)
- ◇ 07:20: [basics] String Literals
- ◇ 09:11: [basics] Integer Literals
- ◇ 11:39: [basics] Integer Literals (Continued)
- ◇ 14:02: [basics] Floating-Point Literals
- ◇ 15:38: [basics] Boolean and Pointer Literals
- ◇ 16:24: [basics] Declarations and Definitions
- ◇ 18:25: [basics] Examples of Declarations and Definitions
- ◇ 20:18: [basics] Arrays
- ◇ 22:20: [basics] Array Example
- ◇ 24:23: [basics] Pointers
- ◇ 28:28: [basics] Pointer Example
- ◇ 32:22: [basics] References

- ◇ 36:28: [basics] References Example
- ◇ 39:29: [basics] Addresses, Pointers, and References
- ◇ 49:24: [basics] Type Aliases with typedef Keyword
- ◇ 50:47: [basics] Type Aliases with using Statement
- ◇ 52:40: [basics] The extern Keyword
- ◇ 54:21: [basics] The const Qualifier
- ◇ 01:03:46: [basics] The volatile Qualifier
- ◇ 01:06:26: [basics] The auto Keyword

49 Basics — Operators and Expressions [2016-03-20]

The following is a link to the full video:

- ◇ <https://youtu.be/hwI4IHEUMZs> [duration: 00:22:11]

The following are links to particular offsets within the video:

- ◇ 00:00: [basics] Title
- ◇ 00:16: [basics] Section: Operators and Expressions
- ◇ 00:25: [basics] Operators
- ◇ 01:00: [basics] Operators (Continued 1)
- ◇ 02:09: [basics] Operators (Continued 2)
- ◇ 02:49: [basics] Operators (Continued 3)
- ◇ 03:05: [basics] Operators (Continued 4)
- ◇ 03:22: [basics] Operator Precedence
- ◇ 04:47: [basics] Operator Precedence (Continued 1)
- ◇ 04:52: [basics] Operator Precedence (Continued 2)
- ◇ 05:55: [basics] Operator Precedence (Continued 3)
- ◇ 06:00: [basics] Operator Precedence (Continued 4)
- ◇ 06:30: [basics] Alternative Tokens
- ◇ 06:55: [basics] Expressions
- ◇ 09:56: [basics] Short-Circuit Evaluation
- ◇ 13:42: [basics] The sizeof Operator
- ◇ 15:08: [basics] The constexpr Qualifier for Variables
- ◇ 20:11: [basics] The static_assert Statement

50 Basics — Control-Flow Constructs [2015-04-09]

The following is a link to the full video:

- ◇ https://youtu.be/kEKy_TwNbEE [duration: 00:23:20]

The following are links to particular offsets within the video:

- ◇ 00:00: [basics] Title
- ◇ 00:17: [basics] Section: Control-Flow Constructs: Selection and Looping
- ◇ 00:29: [basics] The if Statement
- ◇ 01:24: [basics] The if Statement (Continued)
- ◇ 02:43: [basics] The if Statement: Example
- ◇ 03:51: [basics] The switch Statement
- ◇ 05:21: [basics] The switch Statement: Example
- ◇ 06:47: [basics] The while Statement
- ◇ 07:50: [basics] The while Statement: Example
- ◇ 09:02: [basics] The for Statement
- ◇ 11:01: [basics] The for Statement (Continued)
- ◇ 11:50: [basics] The for Statement: Example
- ◇ 13:45: [basics] The Range-Based for Statement

- ◇ 15:55: [basics] The do Statement
- ◇ 17:14: [basics] The do Statement: Example
- ◇ 18:05: [basics] The break Statement
- ◇ 19:37: [basics] The continue Statement
- ◇ 21:06: [basics] The goto Statement

51 Basics — Functions [2016-03-20]

The following is a link to the full video:

- ◇ <https://youtu.be/NHS1726zvmE> [duration: 01:00:57]

The following are links to particular offsets within the video:

- ◇ 00:00: [basics] Title
- ◇ 00:16: [basics] Section: Functions
- ◇ 00:24: [basics] Parameters and Arguments
- ◇ 02:08: [basics] Function Declarations and Definitions
- ◇ 03:52: [basics] Functions
- ◇ 06:15: [basics] Functions (Continued)
- ◇ 07:15: [basics] Function: Examples
- ◇ 08:28: [basics] The main Function
- ◇ 10:15: [basics] The main Function (Continued)
- ◇ 11:43: [basics] The main Function (Continued)
- ◇ 12:58: [basics] Lifetime
- ◇ 14:15: [basics] Pass-By-Value Versus Pass-By-Reference
- ◇ 16:00: [basics] Pass-By-Value Versus Pass-By-Reference
- ◇ 18:26: [basics] Pass By Value
- ◇ 20:43: [basics] Pass By Reference
- ◇ 22:51: [basics] Pass-By-Reference Example
- ◇ 26:19: [basics] Pass-By-Reference Example (Continued)
- ◇ 30:02: [basics] Inline Functions
- ◇ 32:13: [basics] Inlining of a Function
- ◇ 33:44: [basics] The constexpr Qualifier for Functions
- ◇ 37:01: [basics] Constexpr Function Example: power_int (Iterative)
- ◇ 39:53: [basics] Compile-Time Versus Run-Time Computation
- ◇ 42:59: [basics] Function Overloading
- ◇ 46:19: [basics] Default Arguments
- ◇ 48:13: [basics] Argument Matching
- ◇ 51:17: [basics] Argument Matching: Example
- ◇ 58:01: [basics] The assert Macro

52 Basics — Input/Output [2016-03-21]

The following is a link to the full video:

- ◇ <https://youtu.be/MFSA1-1d2Bc> [duration: 00:12:42]

The following are links to particular offsets within the video:

- ◇ 00:00: [basics] Title
- ◇ 00:16: [basics] Section: Input/Output (I/O)
- ◇ 00:22: [basics] Basic I/O
- ◇ 02:33: [basics] Basic I/O Example
- ◇ 04:55: [basics] I/O Manipulators
- ◇ 06:26: [basics] I/O Manipulators (Continued)
- ◇ 08:23: [basics] I/O Manipulators Example

53 Basics — Miscellany [2016-03-21]

The following is a link to the full video:

- ◇ <https://youtu.be/IcPgHnmWy-8> [duration: 00:08:13]

The following are links to particular offsets within the video:

- ◇ [00:00](#): [basics] Title
- ◇ [00:16](#): [basics] Section: Miscellany
- ◇ [00:23](#): [basics] Namespaces
- ◇ [02:28](#): [basics] Namespaces: Example
- ◇ [05:11](#): [basics] Memory Allocation: new and delete

54 Classes — Introduction [2016-03-05]

The following is a link to the full video:

- ◇ <https://youtu.be/8XIdrmAS4Aw> [duration: 00:02:10]

The following are links to particular offsets within the video:

- ◇ [00:00](#): [classes] Title
- ◇ [00:16](#): [classes] Section: Classes
- ◇ [00:25](#): [classes] Classes

55 Classes — Members and Access Specifiers [2016-03-05]

The following is a link to the full video:

- ◇ <https://youtu.be/ZdtqC6zASEI> [duration: 00:35:14]

The following are links to particular offsets within the video:

- ◇ [00:00](#): [classes] Title
- ◇ [00:16](#): [classes] Section: Members and Access Specifiers
- ◇ [00:25](#): [classes] Class Members
- ◇ [01:47](#): [classes] Access Specifiers
- ◇ [03:04](#): [classes] Class Example
- ◇ [06:19](#): [classes] Default Member Access
- ◇ [06:58](#): [classes] The struct Keyword
- ◇ [07:54](#): [classes] Data Members
- ◇ [09:23](#): [classes] Function Members
- ◇ [12:59](#): [classes] The this Keyword
- ◇ [16:34](#): [classes] const Member Functions
- ◇ [25:39](#): [classes] Definition of Function Members in Class Body
- ◇ [27:31](#): [classes] Type Members
- ◇ [29:56](#): [classes] Friends
- ◇ [31:59](#): [classes] Class Example

56 Classes — Constructors and Destructors [2016-03-06]

The following is a link to the full video:

- ◇ <https://youtu.be/9NVA6AGtccc> [duration: 00:30:27]

The following are links to particular offsets within the video:

- ◇ [00:00](#): [classes] Title
- ◇ [00:16](#): [classes] Section: Constructors and Destructors
- ◇ [00:27](#): [classes] Propagating Values: Copying and Moving
- ◇ [01:42](#): [classes] Propagating Values: Copying Versus Moving

- ◇ 04:31: [classes] Constructors
- ◇ 06:21: [classes] Default Constructor
- ◇ 09:30: [classes] Copy Constructor
- ◇ 13:32: [classes] Move Constructor
- ◇ 16:56: [classes] Constructor Example
- ◇ 21:02: [classes] Initializer Lists
- ◇ 22:32: [classes] Initialize List Example
- ◇ 26:23: [classes] Destructors
- ◇ 28:07: [classes] Destructor Example

57 Classes — Operator Overloading [2016-03-09]

The following is a link to the full video:

- ◇ <https://youtu.be/kpIJzSEIe4Y> [duration: 00:41:05]

The following are links to particular offsets within the video:

- ◇ 00:00: [classes] Title
- ◇ 00:16: [classes] Section: Operator Overloading
- ◇ 00:24: [classes] Operator Overloading
- ◇ 01:32: [classes] Operator Overloading (Continued 1)
- ◇ 06:02: [classes] Operator Overloading (Continued 2)
- ◇ 07:59: [classes] Operator Overloading (Continued 3)
- ◇ 09:50: [classes] Operator Overloading Example: Vector
- ◇ 14:17: [classes] Operator Overloading Example: Array10
- ◇ 22:34: [classes] Operator Overloading: Member vs. Nonmember Functions
- ◇ 24:07: [classes] Operator Overloading: Member vs. Nonmember Functions
- ◇ 28:29: [classes] Copy Assignment Operator
- ◇ 31:55: [classes] Self-Assignment Example
- ◇ 34:09: [classes] Move Assignment Operator
- ◇ 36:20: [classes] Copy/Move Assignment Operator Example: Complex

58 Classes — More on Classes [2016-03-22]

The following is a link to the full video:

- ◇ <https://youtu.be/esPOG-FQhdc> [duration: 00:12:25]

The following are links to particular offsets within the video:

- ◇ 00:00: [classes] Title
- ◇ 00:16: [classes] Section: Miscellany
- ◇ 00:23: [classes] Explicitly Deleted/Defaulted Special Member Functions
- ◇ 03:07: [classes] Static Data Members
- ◇ 05:16: [classes] Static Member Functions
- ◇ 08:22: [classes] Stream Inserters
- ◇ 10:31: [classes] Stream Extractors

59 Classes — Temporary Objects [2016-03-24]

The following is a link to the full video:

- ◇ <https://youtu.be/TT0TcIUo88E> [duration: 00:14:01]

The following are links to particular offsets within the video:

- ◇ 00:00: [classes] Title
- ◇ 00:16: [classes] Section: Temporary Objects

- ◇ 00:27: [classes] Temporary Objects
- ◇ 04:10: [classes] Temporary Objects (Continued)
- ◇ 06:56: [classes] Temporary Objects Example
- ◇ 07:37: [classes] Temporary Objects Example (Continued)
- ◇ 10:08: [classes] Prefix Versus Postfix Increment/Decrement

60 Classes — Functors [2016-03-24]

The following is a link to the full video:

- ◇ https://youtu.be/qM2kvcSW4_E [duration: 00:08:14]

The following are links to particular offsets within the video:

- ◇ 00:00: [classes] Title
- ◇ 00:16: [classes] Section: Functors
- ◇ 00:22: [classes] Functors
- ◇ 01:50: [classes] Functor Example: Less Than
- ◇ 04:05: [classes] Functor Example With State

61 Templates — Introduction [2016-03-14]

The following is a link to the full video:

- ◇ <https://youtu.be/q9Wx-kB7MRw> [duration: 00:01:25]

The following are links to particular offsets within the video:

- ◇ 00:00: [templates] Title
- ◇ 00:16: [templates] Section: Templates
- ◇ 00:24: [templates] Templates

62 Templates — Function Templates [2016-03-17]

The following is a link to the full video:

- ◇ <https://youtu.be/hD9MN9aVfi4> [duration: 00:25:49]

The following are links to particular offsets within the video:

- ◇ 00:00: [templates] Title
- ◇ 00:16: [templates] Section: Function Templates
- ◇ 00:30: [templates] Motivation for Function Templates
- ◇ 02:29: [templates] Function Templates
- ◇ 06:05: [templates] Function Templates (Continued)
- ◇ 10:47: [templates] Function Template Examples
- ◇ 13:14: [templates] Template Function Overload Resolution
- ◇ 18:29: [templates] Qualified Names
- ◇ 19:27: [templates] Dependent Names
- ◇ 20:29: [templates] Qualified Dependent Names
- ◇ 23:12: [templates] Why typename is Needed

63 Templates — Class Templates [2016-03-17]

The following is a link to the full video:

- ◇ <https://youtu.be/NXUR5tffiHtE> [duration: 00:18:56]

The following are links to particular offsets within the video:

- ◇ 00:00: [templates] Title

- ◇ 00:16: [templates] Section: Class Templates
- ◇ 00:30: [templates] Motivation for Class Templates
- ◇ 03:13: [templates] Class Templates
- ◇ 06:14: [templates] Class Templates (Continued)
- ◇ 10:36: [templates] Class Template Example
- ◇ 13:16: [templates] Class-Template Default Parameters
- ◇ 15:22: [templates] Qualified Dependent Names

64 Templates — Variable Templates [2016-03-14]

The following is a link to the full video:

- ◇ <https://youtu.be/tble6t8uFGk> [duration: 00:04:04]

The following are links to particular offsets within the video:

- ◇ 00:00: [templates] Title
- ◇ 00:16: [templates] Section: Variable Templates
- ◇ 00:26: [templates] Variable Templates
- ◇ 02:05: [templates] Variable Template Example: pi

65 Templates — Alias Templates [2016-03-14]

The following is a link to the full video:

- ◇ <https://youtu.be/mzM0MHQIQcI> [duration: 00:04:51]

The following are links to particular offsets within the video:

- ◇ 00:00: [templates] Title
- ◇ 00:16: [templates] Section: Alias Templates
- ◇ 00:26: [templates] Alias Templates
- ◇ 02:56: [templates] Alias Template Example

66 Standard Library — Introduction [2016-03-30]

The following is a link to the full video:

- ◇ https://youtu.be/-TY7_GniLig [duration: 00:10:16]

The following are links to particular offsets within the video:

- ◇ 00:00: [lib] Title
- ◇ 00:16: [lib] Section: C++ Standard Library
- ◇ 00:22: [lib] C++ Standard Library
- ◇ 00:57: [lib] C++ Standard Library (Continued)
- ◇ 03:11: [lib] Commonly-Used Header Files
- ◇ 04:24: [lib] Commonly-Used Header Files (Continued 1)
- ◇ 05:48: [lib] Commonly-Used Header Files (Continued 2)
- ◇ 07:14: [lib] Commonly-Used Header Files (Continued 3)
- ◇ 07:53: [lib] Commonly-Used Header Files (Continued 4)
- ◇ 09:19: [lib] Commonly-Used Header Files (Continued 5)

67 Standard Library — Containers, Iterators, and Algorithms [2016-04-05]

The following is a link to the full video:

- ◇ <https://youtu.be/TxufBysSPK0> [duration: 00:38:57]

The following are links to particular offsets within the video:

- ◇ 00:00: [lib] Title
- ◇ 00:16: [lib] Section: Containers, Iterators, and Algorithms
- ◇ 00:22: [lib] Standard Template Library (STL)
- ◇ 01:05: [lib] Containers
- ◇ 02:13: [lib] Sequence Containers and Container Adapters
- ◇ 03:18: [lib] Associative Containers
- ◇ 04:50: [lib] Typical Container Member Functions
- ◇ 05:54: [lib] Container Example
- ◇ 07:36: [lib] Motivation for Iterators
- ◇ 09:47: [lib] Motivation for Iterators (Continued)
- ◇ 12:43: [lib] Iterators
- ◇ 15:03: [lib] Abilities of Iterator Categories
- ◇ 17:03: [lib] Input Iterators
- ◇ 17:45: [lib] Output Iterators
- ◇ 18:34: [lib] Forward Iterators
- ◇ 19:17: [lib] Bidirectional Iterators
- ◇ 19:36: [lib] Random-Access Iterators
- ◇ 21:17: [lib] Iterator Example
- ◇ 24:17: [lib] Iterator Gotchas
- ◇ 25:13: [lib] Algorithms
- ◇ 26:43: [lib] Functions
- ◇ 27:19: [lib] Functions (Continued 1)
- ◇ 28:03: [lib] Functions (Continued 2)
- ◇ 28:27: [lib] Functions (Continued 3)
- ◇ 29:13: [lib] Functions (Continued 4)
- ◇ 29:34: [lib] Functions (Continued 5)
- ◇ 29:42: [lib] Functions (Continued 6)
- ◇ 30:04: [lib] Functions (Continued 7)
- ◇ 30:37: [lib] Algorithms Example
- ◇ 33:40: [lib] Prelude to Functor Example
- ◇ 35:52: [lib] Functor Example

68 Standard Library — The vector Class Template [2016-03-30]

The following is a link to the full video:

- ◇ <https://youtu.be/T8uaiYTIwjc> [duration: 00:27:44]

The following are links to particular offsets within the video:

- ◇ 00:00: [lib] Title
- ◇ 00:16: [lib] Section: The vector Class Template
- ◇ 00:23: [lib] The vector Class Template
- ◇ 01:35: [lib] Member Types
- ◇ 04:08: [lib] Member Functions
- ◇ 05:36: [lib] Member Functions (Continued 1)
- ◇ 07:48: [lib] Member Functions (Continued 2)
- ◇ 09:14: [lib] Invalidation of References, Iterators, and Pointers
- ◇ 11:23: [lib] Iterator Invalidation Example
- ◇ 14:55: [lib] vector Example: Constructors
- ◇ 16:50: [lib] vector Example: Iterators
- ◇ 21:32: [lib] vector Example
- ◇ 24:40: [lib] vector Example: Emplace

69 Standard Library — The `basic_string` Class Template [2016-04-01]

The following is a link to the full video:

- ◇ <https://youtu.be/J6P0JIHactU> [duration: 00:15:16]

The following are links to particular offsets within the video:

- ◇ 00:00: [lib] Title
- ◇ 00:16: [lib] Section: The `basic_string` Class Template
- ◇ 00:24: [lib] The `basic_string` Class Template
- ◇ 02:01: [lib] Member Types
- ◇ 04:47: [lib] Member Functions
- ◇ 05:51: [lib] Member Functions (Continued 1)
- ◇ 07:46: [lib] Member Functions (Continued 2)
- ◇ 08:47: [lib] Member Functions (Continued 3)
- ◇ 10:37: [lib] Member Functions (Continued 4)
- ◇ 10:51: [lib] Non-Member Functions
- ◇ 11:48: [lib] `string` Example
- ◇ 14:07: [lib] Numeric/String Conversion Example

70 Standard Library — Time Measurement [2016-04-02]

The following is a link to the full video:

- ◇ <https://youtu.be/UeCiNGRZAyA> [duration: 00:04:58]

The following are links to particular offsets within the video:

- ◇ 00:00: [lib] Title
- ◇ 00:16: [lib] Section: Time Measurement
- ◇ 00:24: [lib] Time Measurement
- ◇ 01:35: [lib] `std::chrono` Types
- ◇ 02:47: [lib] `std::chrono` Example: Measuring Elapsed Time

71 Concurrency — Preliminaries [2015-02-12]

The following is a link to the full video:

- ◇ <https://youtu.be/oM1VxfrTQWg> [duration: 01:01:50]

The following are links to particular offsets within the video:

- ◇ 00:00: [concurrency] Title
- ◇ 00:22: [concurrency] Disclaimer
- ◇ 00:32: [concurrency] Disclaimer
- ◇ 01:17: [concurrency] Section: Concurrency
- ◇ 01:41: [concurrency] Section: Preliminaries
- ◇ 02:08: [concurrency] Processors
- ◇ 03:58: [concurrency] Processors (Continued)
- ◇ 05:15: [concurrency] Memory Hierarchy
- ◇ 08:55: [concurrency] Examples of Multicore Processors
- ◇ 09:47: [concurrency] Examples of Multicore SoCs
- ◇ 10:57: [concurrency] Why Multicore Processors?
- ◇ 13:28: [concurrency] Section: Multithreaded Programming
- ◇ 13:35: [concurrency] Concurrency
- ◇ 17:58: [concurrency] Memory Model
- ◇ 21:00: [concurrency] Sequential Consistency (SC)
- ◇ 24:15: [concurrency] Sequential-Consistency (SC) Memory Model
- ◇ 27:51: [concurrency] Load/Store Reordering Example: Single Thread

- ◇ 30:28: [concurrency] Load/Store Reordering Example: Multiple Threads
- ◇ 35:10: [concurrency] Store-Buffer Example: Store Buffer
- ◇ 36:54: [concurrency] Store-Buffer Example: Without Store Buffer
- ◇ 40:18: [concurrency] Store-Buffer Example: With Store Buffer (Not SC)
- ◇ 46:21: [concurrency] Atomicity of Memory Operations
- ◇ 48:19: [concurrency] Data Races
- ◇ 51:40: [concurrency] Torn Reads
- ◇ 54:29: [concurrency] Torn Writes
- ◇ 56:53: [concurrency] SC Data-Race Free (SC-DRF) Memory Model
- ◇ 01:00:31: [concurrency] C++ Memory Model

72 Concurrency — Threads [2015-02-17]

The following is a link to the full video:

- ◇ <https://youtu.be/fqG8BgVbmcM> [duration: 00:33:45]

The following are links to particular offsets within the video:

- ◇ 00:00: [concurrency] Title
- ◇ 00:22: [concurrency] Disclaimer
- ◇ 00:32: [concurrency] Disclaimer
- ◇ 01:17: [concurrency] Section: Thread Management
- ◇ 01:40: [concurrency] The std::thread Class
- ◇ 04:38: [concurrency] The std::thread Class (Continued)
- ◇ 07:14: [concurrency] std::thread Members
- ◇ 07:52: [concurrency] std::thread Members (Continued)
- ◇ 09:52: [concurrency] Example: Hello World With Threads
- ◇ 12:24: [concurrency] Example: Thread-Function Argument Passing (Copy Semantics)
- ◇ 14:49: [concurrency] Example: Thread-Function Argument Passing (Reference Semantics)
- ◇ 17:58: [concurrency] Example: Thread-Function Argument Passing (Move Semantics)
- ◇ 18:43: [concurrency] Example: Moving Threads
- ◇ 21:46: [concurrency] The std::this_thread Namespace
- ◇ 23:12: [concurrency] Example: Identifying Threads
- ◇ 25:38: [concurrency] Example: Lifetime Bug
- ◇ 28:47: [concurrency] Thread Local Storage
- ◇ 30:47: [concurrency] Example: Thread Local Storage

73 Concurrency — Mutexes [2015-02-23]

The following is a link to the full video:

- ◇ <https://youtu.be/Vm-Jsno58Y0> [duration: 01:25:24]

The following are links to particular offsets within the video:

- ◇ 00:00: [concurrency] Title
- ◇ 00:22: [concurrency] Disclaimer
- ◇ 00:32: [concurrency] Disclaimer
- ◇ 01:17: [concurrency] Section: Sharing Data Between Threads
- ◇ 01:30: [concurrency] Shared Data
- ◇ 02:46: [concurrency] Race Conditions
- ◇ 09:01: [concurrency] Critical Sections
- ◇ 11:02: [concurrency] Data-Race Example
- ◇ 12:46: [concurrency] Example: Data Race (Counter)
- ◇ 14:09: [concurrency] Example: Data Race and/or Race Condition (IntSet)
- ◇ 17:49: [concurrency] Section: Mutexes

- ◇ 18:16: [concurrency] Mutexes
- ◇ 22:01: [concurrency] The `std::mutex` Class
- ◇ 23:32: [concurrency] `std::mutex` Members
- ◇ 24:39: [concurrency] Example: Avoiding Data Race Using Mutex (Counter) (`mutex`)
- ◇ 27:38: [concurrency] Example: Avoiding Data Race Using Mutex (Counter) (`mutex`)
- ◇ 28:54: [concurrency] The `std::lock_guard` Template Class
- ◇ 32:15: [concurrency] `std::lock_guard` Members
- ◇ 32:54: [concurrency] Example: Avoiding Data Race Using Mutex (Counter) (`lock_guard`)
- ◇ 36:47: [concurrency] Example: Avoiding Data Race Using Mutex (IntSet) (`lock_guard`)
- ◇ 42:05: [concurrency] The `std::unique_lock` Template Class
- ◇ 45:11: [concurrency] `std::unique_lock` Members
- ◇ 46:48: [concurrency] Example: Avoiding Data Race Using Mutex (Counter) (`unique_lock`)
- ◇ 49:36: [concurrency] The `std::lock` Template Function
- ◇ 50:07: [concurrency] Example: Acquiring Two Locks for Swap (Incorrect)
- ◇ 55:51: [concurrency] Example: Acquiring Two Locks for Swap
- ◇ 58:32: [concurrency] The `std::timed_mutex` Class
- ◇ 59:42: [concurrency] Example: Acquiring Mutex With Timeout (`std::timed_mutex`)
- ◇ 01:03:10: [concurrency] Recursive Mutexes
- ◇ 01:06:08: [concurrency] Recursive Mutex Classes
- ◇ 01:07:13: [concurrency] Shared Mutexes
- ◇ 01:09:49: [concurrency] The `std::shared_timed_mutex` Class
- ◇ 01:10:04: [concurrency] `std::shared_timed_mutex` Members
- ◇ 01:11:53: [concurrency] The `std::shared_lock` Template Class
- ◇ 01:13:29: [concurrency] Example: `std::shared_timed_mutex`
- ◇ 01:18:18: [concurrency] `std::once_flag` and `std::call_once`
- ◇ 01:19:12: [concurrency] Example: One-Time Action
- ◇ 01:20:57: [concurrency] Example: One-Time Initialization
- ◇ 01:23:39: [concurrency] Static Local Variable Initialization and Thread Safety

74 Concurrency — Condition Variables [2015-02-27]

The following is a link to the full video:

- ◇ <https://youtu.be/Wsk56vrKONG> [duration: 00:17:37]

The following are links to particular offsets within the video:

- ◇ 00:00: [concurrency] Title
- ◇ 00:22: [concurrency] Disclaimer
- ◇ 00:32: [concurrency] Disclaimer
- ◇ 01:17: [concurrency] Section: Condition Variables
- ◇ 01:43: [concurrency] Condition Variables
- ◇ 05:27: [concurrency] The `std::condition_variable` Class
- ◇ 08:43: [concurrency] `std::condition_variable` Members
- ◇ 09:14: [concurrency] `std::condition_variable` Members (Continued)
- ◇ 10:12: [concurrency] Example: Condition Variable (`IntStack`)
- ◇ 16:38: [concurrency] The `std::condition_variable_any` Class

75 Concurrency — Promises and Futures [2015-04-02]

The following is a link to the full video:

- ◇ https://youtu.be/hic5W_UMjqU [duration: 00:47:45]

The following are links to particular offsets within the video:

- ◇ 00:00: [concurrency] Title

- ◇ 00:22: [concurrency] Disclaimer
- ◇ 00:32: [concurrency] Disclaimer
- ◇ 01:17: [concurrency] Section: Promises and Futures
- ◇ 01:25: [concurrency] Promises and Futures
- ◇ 04:35: [concurrency] Promises and Futures (Continued)
- ◇ 06:28: [concurrency] The std::promise Template Class
- ◇ 09:39: [concurrency] std::promise Members
- ◇ 10:03: [concurrency] std::promise Members (Continued)
- ◇ 11:44: [concurrency] The std::future Template Class
- ◇ 14:05: [concurrency] std::future Members
- ◇ 16:00: [concurrency] Example: Promises and Futures (Without std::async)
- ◇ 20:26: [concurrency] The std::shared_future Template Class
- ◇ 21:54: [concurrency] Example: std::shared_future
- ◇ 25:11: [concurrency] The std::async Template Function
- ◇ 28:44: [concurrency] The std::async Template Function (Continued)
- ◇ 29:54: [concurrency] Example: Promises and Futures (With std::async)
- ◇ 34:11: [concurrency] Example: Futures and Exceptions
- ◇ 37:50: [concurrency] The std::packaged_task Template Class
- ◇ 39:40: [concurrency] std::packaged_task Members
- ◇ 41:09: [concurrency] Example: Packaged Task
- ◇ 44:01: [concurrency] Example: Packaged Task With Arguments

76 CGAL — Introduction [2015-06-29]

The following is a link to the full video:

- ◇ https://youtu.be/Mk-NH2-_hMo [duration: 00:08:04]

The following are links to particular offsets within the video:

- ◇ 00:00: [cgal] Title
- ◇ 00:24: [cgal] Section: Computational Geometry Algorithms Library (CGAL)
- ◇ 00:33: [cgal] Computational Geometry Algorithms Library (CGAL)
- ◇ 02:29: [cgal] CGAL (Continued)
- ◇ 03:35: [cgal] Handles
- ◇ 04:27: [cgal] Linear Sequences Versus Circular Sequences
- ◇ 06:14: [cgal] Circulators

77 CGAL — Polygon Meshes [2015-07-02]

The following is a link to the full video:

- ◇ <https://youtu.be/R8h1JCR4x00> [duration: 00:33:54]

The following are links to particular offsets within the video:

- ◇ 00:00: [cgal] Title
- ◇ 00:24: [cgal] Section: Polygon Meshes
- ◇ 00:30: [cgal] Polyhedron_3 Class
- ◇ 02:53: [cgal] Polyhedron_3 Type Members
- ◇ 04:27: [cgal] Polyhedron_3 Type Members (Continued 1)
- ◇ 06:13: [cgal] Polyhedron_3 Type Members (Continued 2)
- ◇ 07:34: [cgal] Polyhedron_3 Function Members
- ◇ 09:40: [cgal] Polyhedron_3 Function Members (Continued 1)
- ◇ 10:31: [cgal] Polyhedron_3 Function Members (Continued 2)
- ◇ 12:51: [cgal] Polyhedron_3::Facet
- ◇ 14:43: [cgal] Facet Function Members

- ◇ 16:11: [cgal] Polyhedron_3::Vertex
- ◇ 17:54: [cgal] Vertex Function Members
- ◇ 19:21: [cgal] Polyhedron_3::Halfedge
- ◇ 21:04: [cgal] Polyhedron_3::Halfedge (Continued)
- ◇ 22:36: [cgal] Halfedge Function Members
- ◇ 23:52: [cgal] Halfedge Function Members (Continued 1)
- ◇ 25:32: [cgal] Halfedge Function Members (Continued 2)
- ◇ 26:25: [cgal] Adjacency Example
- ◇ 31:12: [cgal] Polyhedron_3 I/O
- ◇ 31:37: [cgal] Polyhedron_3 Gotchas

78 CGAL — Subdivision Surface Methods [2015-06-29]

The following is a link to the full video:

- ◇ https://youtu.be/t_zvp9dTTBY [duration: 00:03:23]

The following are links to particular offsets within the video:

- ◇ 00:00: [cgal] Title
- ◇ 00:24: [cgal] Section: Surface Subdivision Methods
- ◇ 00:29: [cgal] Subdivision Methods
- ◇ 01:23: [cgal] Subdivision Functions

79 CGAL — Example Programs [2015-07-01]

The following is a link to the full video:

- ◇ <https://youtu.be/sTfG7tStFvI> [duration: 00:34:03]

The following are links to particular offsets within the video:

- ◇ 00:00: [cgal] Title
- ◇ 00:24: [cgal] Section: Example Programs
- ◇ 00:31: [cgal] Mesh Generation Program: makeMesh
- ◇ 00:55: [cgal] meshMake Source-Code Walkthrough
- ◇ 09:40: [cgal] Mesh Information Program: meshInfo
- ◇ 10:20: [cgal] meshInfo Source-Code Walkthrough
- ◇ 25:36: [cgal] Mesh Subdivision Program: meshSubdivide
- ◇ 26:12: [cgal] meshSubdivide Source-Code Walkthrough

80 Text Formatting in C++20 [2021-02-03]

The following is a link to the full video:

- ◇ https://youtu.be/E8456triH_g [duration: 00:55:23]

The following are links to particular offsets within the video:

- ◇ 00:00: [format] Text Formatting in C++20
- ◇ 01:02: [format] Text Formatting
- ◇ 02:27: [format] Motivating Example 1: sprintf Family Functions
- ◇ 05:58: [format] Motivating Example 2: I/O Streams
- ◇ 09:52: [format] Text Formatting and std::format Family Functions
- ◇ 12:07: [format] std::format Family of Functions
- ◇ 13:46: [format] Format Strings
- ◇ 17:31: [format] Format String (Continued)
- ◇ 21:09: [format] Format Specifiers
- ◇ 22:48: [format] Type Options for Integer Types

- ◇ 25:07: [format] Type Options for Character Types
- ◇ 26:12: [format] Type Options for Boolean Types
- ◇ 27:14: [format] Type Options for Floating-Point Types
- ◇ 30:03: [format] Type Options for String Types
- ◇ 30:17: [format] Sign Options
- ◇ 32:13: [format] Field Width and Precision Options
- ◇ 36:36: [format] Fill Characters and Alignment Options
- ◇ 38:08: [format] Locale-Specific Formatting
- ◇ 39:12: [format] Locale-Specific Formatting Example
- ◇ 41:06: [format] Example: Formatting to a Buffer [format_to, format_to_n, formatted_size]
- ◇ 46:21: [format] Formatting User-Defined Types
- ◇ 48:11: [format] Point Formatter Example: custom_1.hpp [1]
- ◇ 48:15: [format] Point Formatted Example: User [1]
- ◇ 48:19: [format] Point Formatter Example: custom_1.hpp [2]
- ◇ 48:59: [format] Point Formatted Example: User [2]
- ◇ 49:52: [format] Point Formatter Example: custom_1.hpp [3]
- ◇ 53:16: [format] Point Formatted Example: User [3]
- ◇ 53:47: [format] References
- ◇ 54:37: [format] Questions

81 Meshlab/Geomview Demo [2019-06-16]

The following is a link to the full video:

- ◇ https://youtu.be/X7A_7REjrsk [duration: 00:02:08]

The following are links to particular offsets within the video:

- ◇ 00:00: [misc] meshlab/geomview Demo

82 Accessing the SDE Using VM Software [2020-04-26]

The following is a link to the full video:

- ◇ <https://youtu.be/Sv6dpmZWxgE> [duration: 00:04:59]

The following are links to particular offsets within the video:

- ◇ 00:00: [sde] Demonstration: Accessing the SDE Using a VM

83 Assertions and CMake Build Types Demonstration [2020-04-30]

The following is a link to the full video:

- ◇ <https://youtu.be/lwp7BZpHrog> [duration: 00:08:12]

The following are links to particular offsets within the video:

- ◇ 00:00: [cmake] Assertions and CMake Build Types Demonstration

84 Address Sanitizer (ASan) Demonstration [2020-04-26]

The following is a link to the full video:

- ◇ <https://youtu.be/nkxGxWo2THo> [duration: 00:07:28]

The following are links to particular offsets within the video:

- ◇ 00:00: [asan] Address Sanitizer Demo

85 Undefined-Behavior Sanitizer (UBSan) Demonstration [2020-04-26]

The following is a link to the full video:

- ◇ <https://youtu.be/HvYn5pHgVsg> [duration: 00:05:19]

The following are links to particular offsets within the video:

- ◇ **00:00**: [ubsan] Undefined-Behavior Sanitizer Demo

86 Lcov Demonstration [2020-04-30]

The following is a link to the full video:

- ◇ https://youtu.be/_KM0rDQYFSg [duration: 00:14:48]

The following are links to particular offsets within the video:

- ◇ **00:00**: [lcov] Lcov Demonstration