SENG 475 Video Lectures

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

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

The following is a link to the full video:

- `https://youtu.be/JOUZZVLMJvI` [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: [data_structures] 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

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)

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 Multimap ADTs
    - [data_structures] Remarks on Implementation of Sets and Maps
- 38:04: [data_structures] Priority Queue ADT
- 41:01: [data_structures] Remarks on Priorty 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)

The following is a link to the full video:

- `https://youtu.be/1nDMJrwta24` [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

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

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

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

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: [lrvalues] Value Categories of Expressions
- 36:39: [lrvalues] Value Categories of Expressions (Continued)
- 40:36: [lrvalues] Lvalues
- 43:39: [lrvalues] Lvalues (Continued 1)

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

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

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

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

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

The following is a link to the full video:

- ◇ https://youtu.be/yoA7fFfBRII [duration: 00:52:24]

The following are links to particular offsets within the video:

- ◇ 00:00: [exericses] [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

The following is a link to the full video:

- https://youtu.be/_jyR6uel2k4 [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

The following is a link to the full video:

- https://youtu.be/xMZl2vghJF4 [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

The following is a link to the full video:

- https://youtu.be/EcOOzgwRPw4 [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

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

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

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

The following is a link to the full video:

- ◇ https://youtu.be/xKObs70kzC8 [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

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

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 Identificiation
- 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

The following is a link to the full video:

- ◇ `https://youtu.be/ZV3LOrsHuV0` [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

The following is a link to the full video:

- ◇ `https://youtu.be/BC-eOhw6kAQ` [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

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

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

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

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

The following is a link to the full video:

- `https://youtu.be/nHll640_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

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

The following is a link to the full video:

- ◇ `https://youtu.be/DeLPO3S_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

The following is a link to the full video:

- https://youtu.be/_VV1BlJ97ug [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

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

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

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

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

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

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

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)

39 Extra (2019-06-16) — Meshlab/Geomview Demo

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: Meshlab
- ◇ 01:23: Geomview