|  |
| --- |
| Logo  Description automatically generated with medium confidence  **Course Data Sheet** **Modern C++ for Developer**    Document ID: **CDS/Modern C++** |
| **CONFIDENTIALITY:**  This training content document is confidential and is intended, among other things, to present a general outline of eAge Technologies India Private Limited and its services. The contents are not to be reproduced or distributed to the public or press. Each person who has received a copy of this document (whether or not such person purchases any securities) is deemed to have agreed: (i) not to reproduce or distribute this document, in whole or in part, without the prior written consent of eAge Technologies India Private Limited, other than to legal, tax, financial and other advisors on a need to know basis, (ii) if such person has not purchased securities, to return this presentation to eAge Technologies India Private Limited upon its request, (iii) without the prior written consent of eAge Technologies India Private Limited, not to disclose any information contained in this presentation except to the extent that such information was (a) previously known by such person through a source (other than eAge Technologies India Private Limited) not bound by any obligation to keep such information confidential, (b) in the public domain through no fault of such person, or (c) lawfully obtained at a later date by such person from sources (other than eAge Technologies India Private Limited) not bound by any obligation to keep such information confidential, and (iv) to be responsible for any disclosure of this document, or the information contained herein, by such person or any of its employees, agents or representatives. All rights to this information is reserved. |
|  |

**Course Overview**

A perfectly designed and custom training which, focuses on designing and developing solving software requirement using C++ to enhance the skills required in every day programming challenge. This makes it significantly different from most C++ courses by focusing on C++ internals above design principles. This customized training is backed up by a comprehensive coverage demonstrating the best practices in C++ standards 98/03/11/14/20.

**Duration: 3 Days**

**Mode of training**

Online (4 Hours/day, 8 Hours/day)

**Course objectives**

* To provide an understanding of the core essentials of the C++ programming
* To give you detail exposure of writing C++ for challenging C++ software systems
* To explain the capability and limitations under C++ standards
* To understand an overview of design patterns
* To explain the extension under C++ stand

**Exercises**

* Will be shared to the participants on google drive/git link

**Pre‐requisites:**

* Working/development knowledge with C or CPP
* Strong knowledge of software life cycle
* Strong Object-Oriented understanding
* Usage of different C++ compilers

**Course Materials:** Online reference documents

**Lab Environment:**

* Every participant should have “g++”/visual c++ compiler pre-installed
* Compiler support with C++11/C++14/C++17/C++20 – compulsory
* Appropriate IDE / any text editor

**Course Outline**

**Day 1**

**C++ Refresher**

* Object Oriented concepts
* Classes, data members & methods
* Special methods – constructor, destructor
* Shallow copy & Deep copy
* Static data members & static methods
* Inheritance
* Overloading
* Overriding – virtual functions
* Exception Handling
* C++ Console & File Streams
* Template Functions & Classes
* STL – quick overview
* C++98 & 03 standards

**Day 2 is used for spill over from day 1 in case if any**

**C++11/14/17 language Features**

**SOLID Principles**

* **Evolution of SOLID Principles**
* Object Oriented Design thinking
* Robert C martin (Uncle Bob)
* Pros and cons
* **SOLID Principles**
* Single-Responsibility principle (SRP)
* Overview,
* auto Overview
* Purpose of auto
* Details of Using auto
* Lambda Expressions Overview
* Components of Lambda Expressions
* Storing Lambdas
* Capturing in C++11
* Rules of Thumb and Lambda Summary

**Day 3 ( focus on templates including relate back to STL)**

Class Design principles

* Open-closed principle (OCP)
* Liskov-Substitution Principle (LSP)
* Interface Segregation Principle (ISP)
* Dependency Inversion Principle (DIP)

**Templates, Classes, Initialization**

* Overview
* Variadic Templates
* Working with Parameter Packs
* More Variadic Template Features
* Template Aliases
* Template Arguments, extern Templates
* decltype and declval ( moved from day 2 to day 3)

Day 4

* In-class Initializers for Non-static Data Members
* Inheriting and Delegating Constructors
* Default and Deleted Methods
* override and final Keywords
* Friend Declarations, Nested Classes
* Uniform Initialization
* initializer\_list
* Narrowing Conversions, Initialization Caveats

**Move Semantics, Perfect Forwarding, constexpr**

* Overview
* Move Semantics Introduction
* lvalue/rvalue Revision
* rvalue References
* Move Semantics Implementation
* std::move
* Perfect Forwarding
* Reference binding rules
* Reference Collapsing, forward and move Templates

**Day 5**

* constexpr Mechanism
* constexpr Functions & Literal Types

**Range-based for nullptr, enums, literals, static\_assert, noexcept**

* Range-based for Loops
* Null Pointer
* Enum Changes
* Compile Time Assertions
* Unicode Support & New Literals
* Raw Literals
* User Defined Literals
* Literal Operators
* Exception Specifications
* Explicit Conversion Operators
* Inline Namespaces
* alignof, alignas, sizeof
* Memory Model
* Thread Local Storage
* Generalized Attributes
* POD Types
* Changed Restrictions on Unions

**Day 6**

**C++11/14/17 Library Features - Smart Pointers**

* Introduction
* The Pointer Landscape
* unique\_ptr
* Demo: unique\_ptr
* Demo: unique\_ptr deleter
* shared\_ptr
* Demo: shared\_ptr
* weak\_ptr
* Demo: weak\_ptr

**Modern Concurrency library**

* Threads
* Race conditions
* Mutexes
* Atomicity
* Asynchronous tasks
* Condition variables

**C++ Secure coding Guidelines**

**Day 7**

**STL – C++11/14/17 additions**

* Std::array
* Std::forward\_list
* Regex
* Chrono
* Str-class
* New algorithms

**C++11/14/17 language features**

* Aggregate classes
* binary literals
* generic lambda expressions
* lambda capture initializers
* function return type deduction
* decltype(auto)
* relaxing constraints on constexpr functions
* variable templates
* new/delete elision

**C++11/14/17 library features**

* user-defined literals for standard library types
* compile-time integer sequences
* std::make\_unique
* shared mutex & Locks

**C++11/14/17 language features**

* Easier nested namespace
* New rules for type detection from the braced initializer list
* Simplified static\_assert
* Std::invoke
* Exception specifications
* Fold expressions
* Explicit deduction guides
* Structured bindings
* if/switch initializers / scoped variables

**Day 8**

**C++11/14/17 library features**

* emplace\_back()
* std::string\_view()
* std::variant
* std::any
* std::apply

**C++11/14/17/20 features**

* Concepts – compile predicates
* Designated initializers
* Templatized lambdas
* Non-type template parameters
* range-for with initializer
* Const-expr for virtual functions
* Lambda capture parameter pack

**C++11/14/17/20 libraries**

* concept library
* std::midpoint
* std::bit\_cast
* std::to\_array