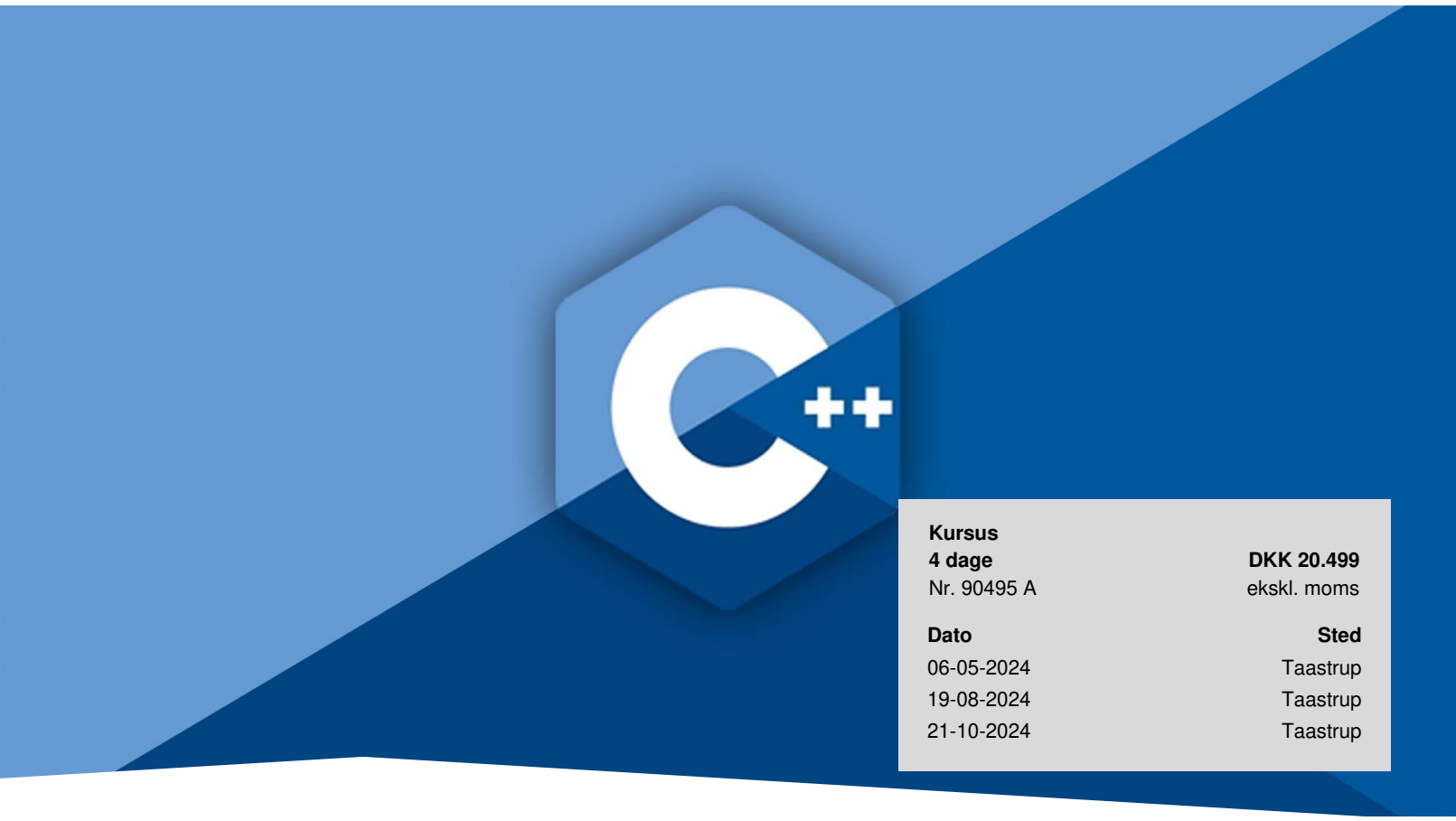
TEKNOLOGISK INSTITUT



Kursus	
4 dage	**DKK 20.499**
Nr. 90495 A	ekskl. moms

Dato	Sted
06-05-2024	Taastrup
19-08-2024	Taastrup
21-10-2024	Taastrup

Modern C++ Development

Bliv opdateret, og få et detaljeret indblik i alle de nye funktioner, der tilbydes i C++ 11. På kurset gennemgås også de løbende ændringer, der sker på C++ 14, C++ 17 og C++20. Undervisningen foregår på engelsk.

Troede du, at du kendte C++?

Tro om igen! C++11-standarden er et kæmpestort skridt frem for programmeringssproget og STL.

Med generel inspiration fra Boost understøtter C++11-standarden lamda-udtryk, multithreading, gennemgribende forbedret objektkonstruktionsmekanismer og meget mere.

Dette kursus går i dybden med alle de nye funktioner i C++11 samt de gradvise yderligere ændringer i C++14, C++17 og C++ 20.

Deltagerprofil

Kurset henvender sig til udviklere med mindst 3-6 måneders erfaring med C++-programmering.

Udbytte

- Working with C++11 smart pointers
- Functional programming in C++11

Teknologisk Institut
kurser@teknologisk.dk
+45 72 20 30 00

- Defining and using lambda expressions
- Using C++11 container classes
- Implementing code applications in C++11
- Using miscellaneous C++ language features
- What's new in C++ 14, C++17, and C++20

Det får du på arrangementet

- Kursusbevis
- Erfaren underviser
- Fuld forplejning
- Gratis parkering
- Materiale på engelsk
- Undervisning på engelsk

Indhold

General Language Enhancements in C++11

- Auto variables
- Using auto in template definitions
- Using decltype
- New return syntax
- Range-based for loops
- Making your own classes iterable
- Generalised constant expressions
- Strongly-typed enums
- Null pointers
- Explicit overrides
- Static asserts

Additional Language Features in C++11

- Lvalues, rvalues, and rrvalues
- Movability
- Reference binding rules
- Support for movability in the STL
- Improved initialization syntax
- Inheriting and delegating constructors
- Regular expressions
- Date and time
- Chrono
- Explicit conversions
- Variadic templates

Miscellaneous New Language Features in C++14

- Function return type deduction
- Alternate type deduction in declarations
- Relaxed constexpr restrictions
- Variable templates
- Aggregate member initialization
- Standard user-defined literals

Smart Pointers

- Recap of smart pointer concepts
- Shared pointers

Teknologisk Institut
kurser@teknologisk.dk
+45 72 20 30 00

- Weak pointers
- Unique pointers
- Techniques and patterns

Introduction to Functional Programming

- Overview of functional programming
- Using std bind to bind parameters
- Using placeholders with for_each()
- Passing by reference
- Using std function to represent free functions and member functions

Lambda Expressions

- Overview of lambda expressions
- Lambda syntax in C++11
- Defining lambdas with arguments and a return value
- Variable capture
- Using lambdas with the STL
- Performance considerations
- Generic lambdas and lambda capture expressions in C++14

C++11 and C++14 Containers

- Overview of new STL features
- Using std array and forward_list
- Using unordered containers
- Understanding hashing
- Defining a custom hash function
- Understanding buckets
- In-place construction
- Heterogeneous lookup in associative containers in C++ 14

C++11 and C++14 Multithreading

- Creating simple threads using std thread
- Using lambda expressions with threading
- Accessing the current thread
- Using mutexes
- Lock management and lock strategies
- Atomic variables
- Condition variables
- Calling functions asynchronously
- Working with future values
- Shared mutexes and locking in C++14

What's New in C++17

- Nested namespaces
- Attributes
- Fold expressions in variadic templates
- Aggregate initialization with inheritance
- Lambda enhancements
- Template class type deduction
- Inline variables
- Library enhancements
- Parallel algorithms
- Miscellaneous enhancements and additions

Teknologisk Institut
kurser@teknologisk.dk
+45 72 20 30 00

What's new in C++20

- Concepts
- Ranges
- Lambda improvements
- The spaceship operator
- Atomic smart pointers
- Concurrency and synchronization improvements
- Co-routines
- Modules
- The consteval and constinit keywords
- Miscellaneous enhancements and additions

UNDERVISER

Andy Olsen

Andy kommer fra Storbritannien og har mere end 30 års erfaring inden for IT. Andy begyndte sin professionelle karrierer som C/C++ udvikler og senere udvikling i Java, C# og andre sprog. Andy er aktivt involveret i en bred vifte af teknologier herunder Full-Stack Development, Cloud Native applikationer, Data Science og meget mere. Andy er meget engageret og brænder for at undervise.

Har du faglige spørgsmål så kontakt

Mette Rosenløv Vad
+45 72202432
mva@teknologisk.dk