C++ programmeren, gevorderd

Praktijkcursus van 4 dags - 28h
Ref : POP - Prijs 2026 : € 2 240 excl. BTW

De C++ taal evolueert voortdurend, van C++98 tot C++20, en biedt mechanismen voor robuust en zeer rijk ontwerp. Recente C++ standaarden hebben de Standard Template Library (STL) aanzienlijk verbeterd. Deze cursus stelt u in staat om meer te leren over ontwerpen in C++ door de nieuwste ontwikkelingen in de taal te behandelen en effectief gebruik te maken van de STL.

PEDAGOGISCHE DOELSTELLINGEN

Na afloop van de opleiding kan de cursist:

Ontdek wat er nieuw is in deze versies

Geheugen, pointers en referenties beheren

Genericiteit implementeren in C++

Ontdek de standaard STL-bibliotheek

Gebruik de bijdragen van de C++11 standaard

HANDS-ON WORK

De cursus vindt plaats op Windows/Visual C++ werkstations. Er zullen veel oefeningen worden gebruikt om de behandelde onderwerpen meer specifiek toe te passen vanuit het oogpunt van ontwerp.

HET PROGRAMMA

laatste update: 02/2025

1) Herinneringen

- Klassen voor geheugentoewijzing.
- Objecten bouwen, initialiseren en laden.
- Geheugenlekken.
- Constance, het sleutelwoord muteerbaar, Lazy Computation.
- Vriendschap C++ en toegangscontrole.
- Virtuele vernietiging.
- Strategie voor uitzonderingsbeheer.
- Naamruimten.

2) Nieuwe taalfuncties in C++11

- nullptr en andere literalen.
- De richtlijnen =delete, =default.
- Delegatie van fabrikant.
- Veilige opsommingen.
- Het trefwoord auto en loop over een interval.
- rvalue-verwijzing en invloed op de normale vorm van C++ klassen.
- Lambda-uitdrukkingen.

Bestaande C++-code herschrijven in C++11, waarbij de twee implementaties worden vergeleken.

3) Beheer operator

- Binaire en unaire operatoren.
- De indirectie operator, use case.
- De verwijzingsoperator.
- Prefixed en postfixed increment/decrement operatoren.
- Andere operatoren: vergelijking, toewijzing, enz.

DEELNEMERS

C++ applicatieontwerpers en -ontwikkelaars, projectmanagers, software-architecten.

VOORAFGAANDE VEREISTEN

Goede kennis van de ontwikkeling van C++ of kennis die gelijkwaardig is aan die van de cursus "Object Programming in C++" (ref. C++). Vereiste ervaring.

VAARDIGHEDEN VAN DE CURSUSLEIDER

De deskundigen die de cursus leiden zijn specialisten op het betreffende vakgebied. Zij werden geselecteerd door onze pedagogische teams zowel om hun vakkennis als hun pedagogische vaardigheden voor elke cursus die zij geven. Zij hebben minstens vijf tot tien jaar ervaring in hun vakgebied en oefenen of oefenden verantwoordelijke bedrijfsfuncties uit.

BEOORDELINGSMODALITEITEN

De cursusleider beoordeelt de pedagogische vooruitgang van de deelnemer gedurende de gehele cursus aan de hand van meerkeuzevragen, praktijksituaties, praktische opdrachten, …
De deelnemer legt ook van tevoren en naderhand een test af ter bevestiging van de verworven kennis.

PEDAGOGISCHE EN TECHNISCHE MIDDELEN

• De gebruikte pedagogische middelen en cursusmethoden zijn voornamelijk: audiovisuele hulpmiddelen, documentatie en cursusmateriaal, praktische oefeningen en correcties van de oefeningen voor praktijkstages, casestudies of reële voorbeelden voor de seminars.
• Na afloop van de stages of seminars verstrekt ORSYS de deelnemers een evaluatievragenlijst over de cursus die vervolgens door onze pedagogische teams wordt geanalyseerd.
• Na afloop van de cursus wordt een presentielijst per halve dag verstrekt, evenals een verklaring van de afronding van de cursus indien de stagiair alle sessies heeft bijgewoond.

TOEGANGSMODALITEITEN EN -TERMIJNEN

De inschrijving dient 24 uur voor aanvang van de cursus plaatsgevonden te hebben.

TOEGANKELIJKHEID VOOR MINDERVALIDEN

Is voor u speciale toegankelijkheid vereist? Neem contact op met mevr. FOSSE, contactpersoon voor mindervaliden, via het adres psh-accueil@ORSYS.fr om uw verzoek en de haalbaarheid daarvan zo goed mogelijk te bestuderen.

- Overloading van de [] operator, de insertie (<<) en extractie (>>) operators.
- Functoren en operator overloading (), een voordeel ten opzichte van functies.

Creatie van functors en proxies (geheugenvrijgave, referentietelling) met de bestudeerde operatoren.

4) Conversie en RTTI

- Conversie operatoren. Impliciete constructies, het expliciete sleutelwoord.
- Casting operatoren const_cast, static_cast, reinterpret_cast.
- Dynamische conversie en Runtime Type-informatie.
- De typeid operator en gerelateerde uitzonderingen.
- De klasse type_info.
- Downcasting regelen met de dynamic_cast operator.

Implementatie van "is-a" en "is-kind-of" idiomen met dynamic_cast.

5) Algemeenheid

- Inleiding tot klassepatronen. Genericiteit en preprocessor.
- Generieke functie. Generieke klasse. Generieke samenstelling. Generieke generalisatie.
- Gedeeltelijke en volledige specialisatie.
- Inleiding tot metaprogrammeren.
- Genericiteit, het verenigende principe van de STL en Boost bibliotheken.

Start van de casestudie, die zal worden aangevuld met de STL. Implementatie van generieke compositie en generalisatie. Generieke plug-ins maken.

6) De STL (Standard Template Library)

- STL-componenten: complementaire types, containers, algoritmen, iterators, functieobjecten, adapters.
- STL tekenreeksen, de sjabloonklasse basic_string en zijn specialisaties.
- Sequentiële en associatieve containers: definitie, rol en selectiecriteria.
- Allocators en geheugenbeheer voor containers.
- Methoden voor invoegen, verwijderen, itereren en toegang tot de belangrijkste containers: Vector, Lijst, Set, Stapel, enz.
- Het iterator-concept. Navigeren door een container.
- De verschillende groepen STL-algoritmen: niet-mutant, mutant, sorteren en samenvoegen, numeriek.
- Omgaan met containers (manipulatie, zoeken naar waarden, enz.).
- Parametriseer generieke algoritmen met behulp van "functie"-objecten.
- Adapters" en het gedrag van een component wijzigen.
- STL en flowverwerking (bestanden, geheugen, enz.).
- Het RAII principe: automatische aanwijzers en de auto_ptr klasse.
- Standaard STL-uitzonderingen.

Implementatie van relaties met STL verzamelingen. Gebruik van standaard algoritmen.

7) Wat is er nieuw in de C++11 standaardbibliotheek?

- Historische evolutie: Boost --> TR1 --> C++11.
- Nieuwe containers: array, forward_list, unordered_set, unordered_map.
- De tuple klasse.
- Slimme pointers: shared_ptr, weak_ptr, unique_ptr.
- Nieuwe functoren en binders.
- Inleiding tot draadbeheer.
- Reguliere uitdrukkingen.

Robuustheid implementeren met slimme aanwijzers. Reguliere expressies gebruiken.

8) Boost en zijn principes

- De Pointer Container-bibliotheek (vernietiging van containerpointergegevens).
- De boost::any en boost::variant datastructuren.
- Event-driven programmeren (verbindingen en signalen).

- Procesbeheer, communicatiemechanismen tussen processen en gedeeld geheugen.
Verbeterde implementatie van de casestudy met behulp van de Pointer Container Library.

9) Geavanceerd gebruik van overerving

- Erfenis versus instappen. Privé-erfgoed. Beschermd erfgoed.
- Verborgen leden exporteren met de Using-clausule.
- Meervoudige overerving en beheer van botsingen tussen leden.
- Diamant overerving. Virtuele overerving en dynamic_cast.
- Ontwerpprincipes: Liskov substitutie, open/close principe, afhankelijkheidsinversie.
- Regels voor het implementeren van interfaces in C++.

Combineer meervoudige overerving, private overerving en export om robuuste, zeer schaalbare klassen te maken.

DATA

Neem contact met ons op