MICRO CONSULT

Stand 27.01.2026

Clean Code für C-Programme: Softwareentwicklung als Handwerkskunst - Der Weg zum Clean Code - Live-Online-Training

**Ziele -
Ihr Nutzen**

Sie lernen die wichtigsten Prinzipien, Regeln und Praktiken für die Erstellung von praxisgerechter, wartbarer Softwaren nach den Ideen des "Clean Code" kennen.

Mithilfe von Refactoring können Sie die Codestruktur optimieren und die Komplexität Ihrer Software senken.

Die Qualität von vorhandenem Quellcode wird verbessert und die Qualität neuer Software-Projekte wird von Anfang an gesichert.

Teilnehmer

Software-Entwickler, Software-Architekten

Voraussetzungen

Grundlegende Kenntnisse der Programmiersprache C

Live Online Training

20.07. – 22.07.2026	1.950,00 €	3 Tage

* Preis je Teilnehmer, in Euro zzgl. USt.

Anmeldecode: L-CLEANC

Präsenz-Training - Deutsch

Termin	Dauer
14.04. – 16.04.2026	3 Tage

Clean Code für C-Programme: Softwareentwicklung als Handwerkskunst - Der Weg zum Clean Code - Live-Online-Training

Inhalt

Softwareentwicklung als Handwerkskunst - 'Software Craftsmenship' - Der Weg zum 'Clean Code'

Guter Code

- Warum ist guter Code wichtig?
- Was zeichnet guten Code aus?
- Welche Probleme verursacht schlechter Code?
- Was sind die Ursachen für schlechten Code?
- Warum ist es sinnvoll, auf guten Code Wert zu legen?
- Wie entsteht guter Code?

Objektorientierte Programmierung mit C

- Die Programmierung von Klassen in C
- Wie werden die üblichen Beziehungen (Assoziation, Komposition und Vererbung) in C umgesetzt
- Warum ist auch in C die objektorientierte Programmierung sinnvoll
- Übung: Erstellen einer Klasse

- Übung: Implementieren von Beziehungen (Komposition, Vererbung)

Grundregeln zur Erstellung guten Codes

- DRY - Don't Repeat Yourself
- KISS - Keep it simple, stupid
- Geheimnisprinzip
- Programming to an Interface in C
- Modularisierung
- Prinzip der losen Kopplung
- Prinzip der hohen Kohäsion
- Vorsicht vor Optimierungen
- POLS - Principle of Least Surprise
- Übungen zum besseren Verständnis der Prinzipien

Die SOLID-Prinzipien

- Single-Responsibility-Prinzip
- Open-Closed-Prinzip
- Liskovsches Substitutionsprinzip
- Interface-Segregation-Prinzip
- Dependency-Inversion-Prinzip
- Übungen zum besseren Verständnis der Prinzipien

Weitere Prinzipien

- SLA - Single Level of Abstraction
- Tell don't ask
- Law of Demeter
- YAGNI - You Ain't Gonna Need It
- Nutze Source Code Konventionen

Refactoring von Code

- Was ist Refactoring?
- Welche Arten gibt es?
- Wie wird eine Refaktorisierung durchgeführt?
- "Smells", die auf die Notwendigkeit einer Refaktorisierung hinweisen
- Refactoring-Patterns
- Übung: Finden von 'Smells' im Code
- Übungen zum Refactoring: Einsatz ausgewählter Patterns

Hinweise zur Verbesserung der Codequalität im Projekt

- Wie lässt sich Bewusstsein für guten Code schaffen?
- Wie lässt sich Code kontinuierlich verbessern?

Praktische Übungen

- Übungen zur Realisierung von Klassen
- Übungen zum besseren Verständnis der Programmierprinzipen
- Finden von "Code-Smells"
- Übungen zum Einsatz von Refactoring-Patterns

MicroConsult PLUS

- Sie erhalten von uns Ihre Übungsverzeichnisse und Lösungsbeispiele für alle Übungsaufgaben.