C++ Deeply Embedded

HILF!GmbH Microcomputer-Consulting, Hofoldingerstr. 16 a, 82054 Sauerlach bei München Tel.: +49 (8104) 9085 250 Fax: +49 (8104) 9085 251
Web: www.hilf.de
Email: info@hilf.de

Zielgruppe

Dieser Kurs legt seinen Schwerpunkt auf die Vermittlung von Techniken, die zwar größtenteils aus dem embedded-Bereich kommen und dort seit Jahren erfolgreich eingesetzt werden, die aber natürlich auch in anderen Bereichen eingesetzt werden können - überall dort, wo es auf robuste Effizienz, Performance, ökonomischen Umgang mit Speicher und Zuverlässigkeit geht. Daher ist er für alle Programmierer geeignet, die diese fortgeschrittenen Techniken in eigenen Projekten einsetzen wollen.

Kursvoraussetzungen

Gute bis sehr gute Kenntnisse in C++.

Kursbeschreibung und Inhalt

Aus den Themen wird man entnehmen, dass es sich bei diesem Kurs nicht um einen Einsteigerkurs handelt. Vielmehr bedarf es schon einiger C++ Vorkenntnissen und einer gewissen Erfahrung mit C++. Die generische Programmierung mit Templates ist der „Zauberkasten" der C++ Programmierung. Sie erlaubt es, klassische Probleme einer neuen, erstaunlich flexiblen und wegen der besonderen Unterstützung durch den Compiler performanteren und speichersparenden Lösung zuzuführen.
Daher steht als erster und wichtigster Punkt die Template Programmierung im Vordergrund. Viele Programmierer benutzen zwar Templates, aber meist ohne sie wirklich bis ins Detail zu verstehen und selbst implementieren zu können.

Wir haben in diesem Kurs einige zentrale Probleme (nicht nur) der Programmierung für den embedded Bereich zusammengestellt. Fast alle werden durch Templates einer neuen, flexiblen, performanten und erfrischend neuen Lösung zugeführt.

Obwohl wir in diesem Kurs mit Windows als Betriebssystem und dem Visual Studio arbeiten, können alle Themen problemlos auf andere Plattformen umgesetzt werden. Sie sind mit verschiedenen Compilern und Betriebssystemen getestet. Und selbst in den Bereichen, wo kein Betriebssystem vorhanden ist, lassen sich die meisten dieser allgemeinen Konzepte nutzbringend anwenden.

Wegen der zum Teil hohen Komplexität der Beispiele können nur für einige der Themen Übungen durchgeführt werden, schwerpunktmäßig bei den ersten Kapiteln.
Die komplexen Sachverhalte werden dann durch eine Vielzahl von fundierten Beispielen dargestellt.

Die Beispiele werden auf Datenträger (DVD) den Teilnehmern zur Verfügung gestellt.

Der vorgesehene Zeitrahmen von 4 Tagen für diese Schulung bedingt bei der Vielzahl und Komplexität der Themen eine „straffe" Durchführung und stellt daher teils hohe Anforderungen an die Auffassungsfähigkeit der Teilnehmer. Dennoch ist dafür gesorgt, dass genügend Spielraum für zusätzliche Fragen und Erklärungen auch von Randproblemen vorhanden ist.
Der Kurs basiert auf dem Betriebssystem Microsoft Windows und dem Visual Studio als Entwicklungswerkzeug.

Kursmaterial

- Kursordner (deutsch)

Kursinhalt

1. Generische Programmierung mit C++ Templates
- Template Funktionen
  - Template Spezialisierungen
  - Umsetzung durch den Compiler / Linker
  - Kommt es durch Templates zur „Wucherung" (Proliferation) von Code?
  - Preconditions
- Template Klassen
  - Template Klasse als Wrapper
  - Template Klasse mit Mixin Design – Zwangsvererbung
- Traits
- Policies

2. Functionpointer
- Aufruf globaler Funktionen mit Function pointern wie in C
- Wie erfolgt der Zugriff auf Klassen- u. Objektmethoden?
- Aufruf von (statischen) Klassenmethoden
- Aufruf von Objektmethoden

3. Delegates
(Variante des Observer Design Patterns mit Template Klassen)
- Delegate Klasse für globale Funktionen und Klassenmethoden
- Delegate Klasse für Objektmethoden

4. Alternative zu STL Containern: intrusive Liste
- Betrachtung der STL Container
- Intrusive double ended queue

5. Multithreading und Thread-Synchronisation
- Erstellen einer eigenen Thread Klasse
  - Alternativen:
  - QThread Klasse
  - C++ 11 Thread Klasse
  - Posix Thread Klasse
- Notwendigkeit der Synchronisation
- Synchronisationswerkzeuge
  - Critical Section
  - Mutex Klasse
  - Semaphore Klasse
- Synchronisations-Design patterns
- Synchronisation scheinbar atomarer Anweisungen

HILF! GmbH
Microcomputer-Consulting

6. „gültiger“ pointer auf ungültiges Object
   Kann auf den Speicherbereich eines ungültigen (gelöschten) Objektes zugegriffen werden?
   Wie kann ich erkennen, ob an der Stelle, auf die der pointer zeigt, sich ein gültiges Objekt befindet?
      - Object Alive check
      - Object Alive ID Generator
   Möglichkeiten, wie verhindert werden kann, dass ein Objekt „abhanden kommt“
      - STL auto_pointer Klasse
      - Boost smartpointer Klassen
      - C++ 11 shared_pointer

7. Dynamisches Allokieren von Speicher
- Im embedded- und Echtzeit-Bereich
- malloc und free oder new und delete?
- manueller Konstruktor-/Destruktor-Aufruf
- Überladen von new-und delete Operatoren
      - Placement new Operator
      - delete Operator

8. Speicherlücken
- Entstehung und Auswirkung von Speicherlücken
- Gegenmaßnahmen
      - Null pointer / C++11 nullptr
      - Speicherlücken-Zähler
      - Speicherlücken-Listen Tool
      - Eigene operator new Variante
      - Transparentes Speichern zusätzlicher Objektinformationen

9. Eigener, echtzeitfähiger Speichermanager für dynamische Allokierung von Objekten fixer Größe
- Mit umgeleitetem new/delete Operator
- Verwendung als Allocator für die STL Container
- Variante für unterschiedliche Speichergrößen

10. Objects on demand – das Factory Design Pattern für Objekterzeugung
- Serialisierung und Deserialisierung
   Einfacher binärer Speicherabzug („flache“ Serialisierung)
   „Tiefe“ Serialisierung
   Zusätzliche Steuerinformationen in den Daten
   Zusätzliche Typinformationen, RTTI

11. Singleton Design Pattern
- Klassische Vorgehensweise
- Templates als Ersatz für fixe Singleton Implementierung
- Wrapper Template Variante