Jürgen Lemke

Xpert.press

C++-Metaprogrammierung

Eine Einführung in die Präprozessor- und Template-Metaprogrammierung

Springer Vieweg

Xpert.press

Weitere Information zu dieser Reihe finden Sie auf
http://www.springer.com/series/4393

Die Reihe **Xpert.press** vermittelt Professionals in den Bereichen Softwareentwicklung, Internettechnologie und IT-Management aktuell und kompetent relevantes Fachwissen über Technologien und Produkte zur Entwicklung und Anwendung moderner Informationstechnologien.

Jürgen Lemke

C++-Metaprogrammierung

Eine Einführung in die Präprozessor- und Template-Metaprogrammierung

Springer Vieweg

Jürgen Lemke
Hilchenbach, Deutschland

ISSN 1439-5428
Xpert.press
ISBN 978-3-662-48549-1 ISBN 978-3-662-48550-7 (eBook)
DOI 10.1007/978-3-662-48550-7

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Springer Vieweg

Gedruckt auf säurefreiem und chlorfrei gebleichtem Papier

Springer-Verlag GmbH Berlin Heidelberg ist Teil der Fachverlagsgruppe Springer Science+Business Media (www.springer.com)

Dieses Buch widme ich meiner Frau Heike sowie unseren Kindern Annelie, Frieda und Torsten.

Vorwort

Kann man eine Programmiersprache wie C++ mit seinen Templates komplett beherrschen? Selbst Bjarne Stroustrup, der Erfinder von C++, musste sich vom Template-Mechanismus der Turing-Vollständigkeit eines Besseren belehren lassen. Auf dem C++-Standardisierungstreffen 1994 in SanDiego präsentierte Erwin Unruh zum ersten Mal ein rekursives C++-Metaprogramm zur Berechnung von Primzahlen.

Gibt es heute noch Grenzen von C++, wo liegen sie und was ist mit C++ möglich bzw. unmöglich? In meiner Arbeit mit C++-Schnittstellen (Interfaces) wurde ich mit der Behauptung konfrontiert, dass sich atomare (native) C++-Strukturen nicht automatisch innerhalb der Programmiersprache generieren lassen, im Gegensatz zu managed Strukturen von Java, C# und C++/CLI. Weil ich diese Aussage so nicht im Raum stehen lassen wollte, beschäftigte ich mich intensiv mit der Metaprogrammierung in C++. Das Ergebnis ist ein Konzept zur automatischen Generierung von atomaren Strukturen, einschließlich deren Serialisierung und Visualisierung, sowie eine große Sammlung von Template-Metafunktionen.

Doch wie kann nun der neue Stoff dem Leser vermittelt werden? Ein Nachschlagewerk für die C++-Metaprogrammierung zu schreiben ist kaum möglich, da die Metaprogrammierung stärker mit Programmiertechniken als mit C++-Sprachstandards verbunden ist. Eine Beschränkung auf die Darstellung von Programmiertechniken schränkt wiederum den Leserkreis zu stark ein, da bestimmte Vorkenntnisse der Präprozessor- und Templateprogrammierung vorhanden sein müssen. Aus diesem Grunde versuche ich einen Spagat zwischen beiden Ansätzen zu erreichen.

In den ersten Abschnitten des Buches werden die Grundlagen in Form eines Nachschlagewerkes sowohl für den Präprozessor als auch für den Templatemechanismus beschrieben. Diese Beschreibung zielt bereits auf die spätere Verwendung in der Metaprogrammierung ab, sodass diese Abschnitte in der Beschreibung der Grundlagen

bereits deutlich umfangreicher sind, als sie in Standard C++-Büchern sein können. In den weiteren Abschnitten werden dann Programmiertechniken und Anwendungsbeispiele vorgestellt, die auf die Grundlagenabschnitte aufbauen. Viele dieser Beispiele und Techniken ziehen sich dann wie ein roter Faden durch das gesamte Buch.

Bei der Untersuchung von Laufzeitfehlern fällt häufig auf, dass eigentlich schon der Compiler auf bestimmte Fehler hätte aufmerksam machen müssen, weil alle Informationen bereits zur Kompilationszeit zur Verfügung standen. Leider gehen zum Beispiel Längeninformationen von Feldern oder Typinformationen von Objekten bei der Parameterübergabe in Funktionen verloren. Mit Templates lassen sich solche Informationen in andere Programmcodes übertragen. Ein Schwerpunkt der Metaprogrammierung liegt daher auch in der Erzeugung von sicherem Programmcode.

Der interessierte Leser wird nach dem Studium des Buches einen erweiterten Blickwinkel auf die Möglichkeiten des C++-Compilers, der Sicherheit und der Effizienz von Programmcode erhalten. Er wird schon vor dem eigentlichen Schreiben von neuem Programmcode tiefergreifende Überlegungen anstellen, welche Programmteile generiert und welche Teile bereits zur Kompilationszeit erledigt werden können.

Auch dem .NET-Anwender steht die C++-Metaprogrammierung mit seinen Templates, über die erweiterte Programmiersprache C++/CLI, zur Verfügung. Damit können .NET-Anwendungen im Hinblick auf ihre Laufzeit beschleunigt werden. Mit Hilfe von Generatoren ist es leicht möglich, Strukturen nach .NET zu konvertieren und die Daten von unmanaged zu managed Objekten zu kopieren bzw. zu konvertieren.

In der vorliegenden Arbeit dreht sich aber nicht alles nur um die reine Metaprogrammierung. Das ist auch kaum möglich, denn die Metaprogrammierung ist immer auf Hilfsklassen angewiesen. Gerade dieses Zusammenspiel soll dargestellt werden, um den Leser für eigene Ansätze zu inspirieren.

Hilchenbach, 2015 Jürgen Lemke

Inhaltsverzeichnis

Teil II Techniken und Anwendungsfälle

Einleitung 1

Die Softwaretechnik ist einem ständigen Wandel unterzogen. Um heute effizient und wirtschaftlich Software entwickeln zu können ist es notwendig, einen großen Teil des Quellcodes aus allgemeingültigen Bibliotheken zu benutzen, um den kundenspezifischen Anteil möglichst gering zu halten. Software aus freien und tausendfach bewährten Bibliotheken, wie der Standard Template Library (STL) oder der Boost Bibliothek, sind dabei zu bevorzugen. Bei der Entwicklung von neuer Software sollte die erste Überlegung immer in Richtung einer allgemeingültigen Umsetzung gehen. So entstehen mit der Zeit Softwaresystemfamilien, die flexibel für unterschiedlichste Anwendungen eingesetzt werden können.

Auf die Entwicklung der Softwaresystemfamilien baut die generative Programmierung auf. Sie zielt dabei auf die nahezu automatische Erstellung von Softwaresystemen auf Grundlage der flexiblen Softwaresystemfamilien. Hierbei kommen vor allem Generatoren zum Einsatz, die auf Grundlage von Konfigurationswissen das Softwaresystem erzeugen.

Die Metaprogrammierung lässt sich wiederum als Programmierung des Generators auffassen, deren Ergebnis ein neuer Quellcode ist. Aber auch eine Erweiterung der Programmiersprache um neue Funktionalitäten oder optimierten Quellcode ist mit ihr möglich. Die griechische Vorsilbe *meta* steht dabei für *über* oder *hinter* und deutet auf eine übergeordnete Programmierung hin. Die hier vorgestellte C++-Metaprogrammierung beschreibt die Technik der Metaprogrammierung nur mit Mitteln der Programmiersprache C++ selbst. Hierfür stehen in C++ zwei unterschiedliche Mechanismen zur Verfügung, die Präprozessor- und die Template-Metaprogrammierung.

Die Präprozessor-Metaprogrammierung befasst sich mit dem Generieren und Einsetzen von Codesegmenten in den vorhandenen Quellcode vor dem Kompilieren. Es erfolgt dabei weder eine Typ- noch eine Syntaxprüfung, weshalb der Einsatz des Präprozessors auf diejenigen Einsatzfälle zu beschränken ist, die mit der Template-Metaprogrammierung nicht abgedeckt werden können.

J. Lemke, *C++-Metaprogrammierung*, DOI 10.1007/978-3-662-48550-7_1