C++

Das umfassende Handbuch

» Hier geht's direkt zum Buch

DAS VORWORT

Vorwort

Vielen Dank, dass Sie dieses Buch erworben haben! In C++ gibt es immer wieder moderne Features, die neue Arbeitsweisen und neue Idiome ermöglichen. Fragt man »das Internet« nach Beispielen in C++, lehnen sich die meisten Quellen an den C-artigen Stil an, der aber nur noch wenig mit den Möglichkeiten des *modernen C++* zu tun hat: RAII vorneweg, nun auch Concepts und Module sowie Koroutinen und Ranges. Wie immer nehme ich mir vor, Ihnen besonders die Stärken von C++ aufzuzeigen – insbesondere dort, wo ich glaube, dass C++ anderen Sprachen voraus ist.

In dieser Auflage gehe ich von einem Compiler aus, der C++17 vollständig beherrscht. Da inzwischen alle verbreiteten Compiler auch C++20 unterstützen, werden Sie mit solchen Features ebenfalls keine Probleme haben. Im Buch weise ich auf die Verwendung von C++20 meistens hin, habe aber auf Wunsch des Verlags auf eine spezielle schriftbildliche Hervorhebung verzichtet. Ich beschreibe darüber hinaus C++23-Features, auf die ich aber immer hinweise und die ich auch besonders *hervorhebe*, denn bei diesen ist die Compilerunterstützung bei Weitem noch nicht vollumfänglich.

Ich will Sie kurz auf C++17-Features hinweisen, die ich nicht mehr hervorhebe, die ich aber in nahezu allen Listings verwende und die Sie verwirren könnten, falls Ihr Compiler sie nicht beherrscht. Die *Klassentemplate-Argumentdeduktion* für Konstruktoren in `vector data{1,2,3}` müssen Sie noch als `vector<int> data{1,2,3}` schreiben. `string_view` werden Sie vermissen und müssen `const string&` schreiben. Es gibt noch einige weitere Kleinigkeiten, die ich hier nicht alle aufliste. In C++20 sind *abgekürzte Funktionstemplates* hinzugekommen, die extrem nützlich sind, deren Erwähnung aber manchmal schwierig ist. Sollten Sie in einem Funktionsparameter einmal ein `auto` sehen, dann handelt es sich in Wirklichkeit um ein Funktionstemplate.

Es gibt jedoch auch Lücken: Module sind leider noch nicht so weit, sodass ich Ihnen dazu nicht viele Tipps an die Hand geben kann. Und während `format` breit unterstützt wird, ist dies für `print` leider noch nicht der Fall.

Bei den Recherchen zu den Neuerungen in C++20 und C++23 habe ich auch die Techniken der aktuellen Zeit verwendet: Das heißt, auch ich als Autor habe ab und zu einen KI-Assistenten zurate gezogen – so wie Sie es ebenfalls tun sollten, um aktuell zu bleiben. Daher ist es vielleicht nicht selbstverständlich, dass Sie sich entschlossen haben, zu einem Buch zu greifen. Was ich aber vor allem bei der Arbeit mit den ziemlich beeindruckenden KI-Tools festgestellt habe: Sie haben ihre Grenzen. Gerade was die Neuerungen in C++20 und C++23 angeht, konnten mir die KI-Tools selten helfen. Aktuell plappern diese Werkzeuge unreflektiert nach, was das Internet seit seinem Bestehen so hergibt.

Dennoch: Sie unterstützen bei der täglichen Arbeit. Und ja, sie erstaunen mich – trotz ihrer Naivität. Ich bin mir sicher, Sie wissen, dass Sie immer skeptisch bleiben müssen bei den Antworten, die sie Ihnen geben. Immer! Und das wird in der Zukunft nicht leichter werden. Ein sehr erhellender Vortrag von Andrei Alexandrescu, der aktuell bei Nvidia arbeitet, macht in der »Closing Keynote Code Europe 2023« Vorhersagen, die mir zeigen, wohin die Reise geht: KI-Tools werden uns immer mehr unterstützen, KI wird selbstverständlicher werden, und wir werden es immer weniger bemerken – und deshalb immer weniger hinterfragen. Aber das ist natürlich gefährlich. Daher: Bleiben Sie wachsam. Und viel Spaß mit diesem Buch!

April 2024, Bielefeld

Torsten T. Will

Vorwort zur 1. Auflage

Aber selbstverständlich spricht nichts dagegen, wenn Sie Ihr Programm »traditionell« schreiben. Das heißt, kurz zusammengefasst, für mich, dass Ihr Programm mehr nach C aussieht, als es aussehen könnte. Daran ist nichts falsch, natürlich nicht. Einige der besten Programme sind in C geschrieben. Dennoch, wenn Sie *heute* ein Projekt beginnen und sich für eine in Maschinencode übersetzte Programmiersprache entscheiden, dann nehmen Sie doch besser C++. Denn in der Sprache tut sich etwas – oder besser, hat sich was getan. Sie haben mit C++14 (und ganz frisch C++17) eine Sprache, die Sie auf aktuelle Art und Weise darin unterstützt, *gute* Programme zu schreiben. Das heißt, Ihre Programme sind schnell, fehlerresistent, wartbar. Sie können produktiv programmieren.

Für dieses Buch habe ich lange überlegt, wie man C++ am besten vermittelt. Bjarne Stroustrup hat auf der C++Con 2017 eine Keynote gehalten, die genau dieses Thema zum Kern hatte. Und er sagte dort Dinge, die, so finde ich, weltbewegend sind. Zumindest, was die C++-Welt angeht. Denn er sagte: »Wir (Lehrenden) haben bis C++98 darin schlechte Arbeit geleistet, Menschen C++ beizubringen.« Und er habe sich Gedanken gemacht, warum das so war. Er schließt sich dabei mit ein und resümiert, dass die meisten C++-Bücher lang, eintönig und langsam sind. Sie brächten »bottom-up 1990-C++« bei und benutzten dabei C++11-Syntax. Und das sei verkehrt. Nun habe ich dieses Buch zu schreiben begonnen, lange bevor Bjarne Stroustrup diese Keynote gehalten hat. Und gerade deshalb fühle ich mich im ausklingenden Jahr 2017 in der Art und Weise, wie dieses Buch am Ende des Arbeitsprozesses nun aussieht, bestätigt. Denn ich sehe das genauso und habe mich von Grund auf bemüht, es anders zu machen.

Zum Beispiel werden Sie in diesem Buch Zeiger erst weit hinten erklärt bekommen. Das ist ziemlich gewagt. Zeiger sind wichtig, in C++ dreht sich vieles um Adressen – aber seit C++11 eben nicht alles. Viel wichtiger ist es, das Konzept hinter Zeigern zu

verstehen, manifestiert in Iteratoren. Denn wenn man den Mechanismus versteht, kann man das Detail mit anderen Dingen kombinieren und Neues erschaffen. Ich möchte immer das Warum in den Vordergrund gestellt sehen.

Stroustrup sagt in seiner Keynote, dass das neue C++ unter anderem Ressourcensicherheit in den Vordergrund stelle. Er fragt danach, welches Buch RAII deswegen in den Vordergrund stelle? Es seien wenige. Der Begriff RAII wird Ihnen in diesem Buch mehrmals begegnen. Er kritisiert, dass viele Bücher Typsicherheit, Abstraktion, Klassendesign und generische Programmierung nicht einmal erwähnen. Dieses Buch tut es.

Mir sind aber auch Genauigkeit und Sorgfalt wichtig, und darum gibt es hier auch einen eher technischen Teil, der sich um Syntax und Semantik der kleinen und großen C++-Konstrukte kümmert. Diese kann man in einem Handbuch nicht überspringen. In einem einzelnen Projekt reicht es vielleicht, eine einzelne Regel dazu zu kennen, welche Defaultoperationen man für eine Klasse definieren sollte. In einer Architektur und für das Verständnis des Warums muss man aber wissen, welche Defaultoperationen es gibt und wie sie miteinander interagieren. Mein Ansatz ist daher, dass ich Ihnen die Dinge in drei Bissgrößen vermittle: Der erste Überblick ist in wenigen Seiten erledigt und gibt Ihnen das erste Gefühl für ein C++-Programm. Es folgt die größere Schleife, in der ich auf nahezu jedes Sprachelement kurz eingehe, damit Sie die Interaktionen verstehen: Sie lernen Ausdrücke, Typen, Anweisungen, Variablen und die Standardbibliothek kennen. Erst in der dritten Runde gehe ich in einzelnen Kapiteln auf alle diese Elemente im Detail ein. Dort finden Sie die Dinge mit Hintergrund und Interaktion mit der Welt erklärt: Bits, Bytes, Big-Endian, Fließkommaformate, Exceptions, Klassen und so weiter und so fort.

Besonders am Herzen liegen mir dabei die Kapitel über `vector`, `map` und Konsorten – also das Thema Container. Die Container der Standardbibliothek werden unterschätzt und durchweg zu wenig eingesetzt. Warum? Zurück zu Bjarne: Weil wir es nicht gut genug vermittelt haben. Ich bemühe mich hier um einen anderen Ansatz. Statt nur aufzuzählen, welche Container es gibt und was für Schnittstellen und Eigenschaften sie haben, möchte ich Ihre Aufmerksamkeit mehr die Konzepte lenken, besonders auf die Gemeinsamkeiten und Unterschiede zwischen den Containern. Sie sollen sich nicht von Beginn an alle Methoden von `vector` merken, sondern, was die Container können, was sie nicht können und wann Sie welchen wählen sollten. Und genau zu Letzterem habe ich deshalb noch ein eigenes Kapitel geschrieben. Wenn ein Problem gegeben ist, können Sie mit einer Referenz doch nur *dann* den richtigen Container finden, wenn Sie alle Containerbeschreibungen gelesen und verinnerlicht haben. Ich beschreibe also typische Probleme und stelle Ihnen Kriterien vor, nach denen Sie die passenden Container auswählen können.

Und das stellt mich immer noch nicht zufrieden. Wer C++ kann, kann nicht automatisch programmieren. Wer das neue C++ aber richtig anwendet, hat verstanden, worum es geht. Und weil »worum es geht« nicht nur in C++ wichtig ist, sondern auch in

anderen Programmiersprachen, war es mir wichtig, dass Sie auch über den Tellerrand hinausschauen. Die eingestreuten Dan-Kapitel beschäftigen sich mit Dingen, die überall in der Softwareentwicklung vorkommen. Egal, ob Sie in Java, PHP oder SQL programmieren. Sie müssen testen, Ihren Code modularisieren, und in den meisten Fällen schadet es nicht, OOP zu kennen. Auf diese Punkte gehe ich ein und wende sie mit C++ an, nehmen Sie sie davon unabhängig mit auf Ihren Weg in Sachen Programmierung und Architektur.

Ich hoffe, dass ich Ihnen mit diesem Buch bei der Arbeit mit C++ helfe. Und wenn dieses Buch Ihnen hilft, C++ anzuwenden, dann wird durch Ihre Arbeit auch C++ besser.

November 2017, Bielefeld

Torsten T. Will