Vorwort

Warum verwenden Sie C++? Warum lernen Sie nicht Java? Warum nicht ABAP, JavaScript oder PHP? Nun, JavaScript und PHP lernen Sie vielleicht sowieso, weil diese in einer bestimmten Domäne zu finden sind, nämlich wenn Sie Webprogrammierung betreiben. Ebenso kommen Sie um ABAP nicht herum, wenn Sie für SAP programmieren.

Anders Java und C++: Diese Sprachen werden überall eingesetzt. Aber so ist es mit Python, Ruby und Perl ebenfalls. Vielleicht lernen Sie eine diese Skiptsprachen auch noch, denn mit ihnen können Sie schnell einen Prototyp schreiben. Und Java? Diese exzellente Sprache schränkt Sie ein. Sie haben Objekte, eine virtuelle Maschine, Classloader – Sie haben Interfaces und schreiben auf jeden Fall objektorientiert. Nun, dann könnten Sie ja C als Ihre Sprache wählen, doch Sie wollen eine hohe Abstraktion. Sie wollen nicht maschinennah programmieren, sondern wollen von neuesten Technologien profitieren, die in den Bau von hochkomplexen Compilern einfließen.

Also wählen Sie C++. Sie wählen die Sprache, die Sie nicht auf das objektorientierte Paradigma festlegt, obgleich sie dieses komplett unterstützt. Sie wollen die Optionen der generischen Programmierung und der luxuriösen Fehlerbehandlung, und Sie wollen dynamische Typen bei gleichzeitiger totaler Typsicherheit. Sie wollen maximale Unterstützung des Compilers beim Vermeiden von Fehlern ebenso wie die Möglichkeit, für alle Plattformen – von der intelligenten Uhr bis zum Großcomputer – Programme zu entwickeln. Und das wollen Sie ohne Kompromisse in der Geschwindigkeit, mit der Ihre Programme laufen werden.

Dies alles ist in den letzten Jahren seit dem C++-Standard von 1998 jedes Jahr ein bisschen mehr wahr geworden. Die Compiler wurden immer besser – überhaupt wuchs die Zahl der verfügbaren Compiler immer weiter. Und es ist gut, die Wahl zu haben. Mit C++98 wurde ein Fundament geschaffen, auf dem man wirklich aufbauen kann. Und das ist 2011 mit C++11 geschehen: Nun haben Sie eine Sprache, die gutes Programmieren ermöglicht. Zu dieser Sprache gehört jetzt auch eine Standardbibliothek, die eine Programmiersprache erst zu einer solchen macht. Mit C++11 haben Sie das Werkzeug in der Hand, um gute Programme zu schreiben.

Sie holen sich aber auch Nachteile ins Haus: Insgesamt ist C++ eine komplexe Sprache. Die Kunst besteht darin, C++ so einzusetzen, dass die Programme dennoch einfach sind. Sie müssen von den vielen Möglichkeiten, die C++ Ihnen lässt, diejenigen auswählen, die Ihre Programme »gut« werden lassen. In diesem Buch lernen Sie deshalb nicht die staubigen Ecken von C++ kennen. Ich lege stattdessen Wert darauf, dass Sie einen möglichst runden, in sich geschlossenen Einblick in C++ bekommen. Ich bemühe mich,

die Dinge so zu erklären, dass Sie die Grundelemente in sich aufnehmen – nicht zuletzt durch wiederholtes Hinterfragen der »Warums« der Sprache. Denn daraus leiten sich die »Darums« ab. Wenn Sie die verstehen, dann können Sie in Zukunft sich selbst Fragen beantworten. Sie bekommen ein Gefühl für die Sprache und kombinieren die Elemente selbst, ohne dass Sie darüber besonders nachdenken müssen. Ich möchte nicht, dass Sie ein Listing nach dem anderen abtippen und sich die Ergebnisse anschauen. Verstehen Sie das »Dahinter«, probieren Sie aus, kombinieren Sie. Denn aus diesen Experimenten entsteht Verstehen und daraus Ihr Erfolg mit der Sprache.

Vielleicht fällt Ihnen auf, dass ich mich sehr eingehend mit den Grundprinzipien der Sprache beschäftige. Ich halte Typen, Variablen, Ausdrücke und Anweisungen für derart wichtige und zentrale Elemente einer Programmiersprache, insbesondere von C++, dass ich sicherstellen will, dass Sie diese durch und durch verstehen. Es ist – wie immer – schwierig, das eine ohne das andere zu erklären. Daher, und das möchte ich nur kurz erwähnen, ist der erste Teil des Buches eine Serie von unterschiedlich großen »Schleifen«, die Sie durch die Programmiersprache fliegen werden. Mit jedem Durchgang hole ich weiter aus, erkläre immer mehr Details. Das wird in etwa bis zum Einblick in Funktionen, Exceptions oder Modularisierung anhalten. Am Ende dieser Rundflüge steigen wir hoch auf, und Sie lernen Klassen und Objekte kennen, bis es im Parabelflug zu Makros und Funktionstemplates geht. Sie werden diese Teile meistern, weil Sie die Grundkonzepte der Sprache bis dahin verinnerlicht haben werden.

In gewisser Hinsicht beneide ich Sie dafür, dass Sie jetzt C++ lernen dürfen. Mit C++11 haben Sie die Chance, viel sauberere und besser strukturierte Programme zu schreiben, als es mir vor über 20 Jahren vergönnt gewesen ist. Ich will Ihnen deshalb genau dies als Vorteil mit auf den Weg geben: Ich fokussiere in diesem Buch auf die »hübschen« Elemente, die hauptsächlich mit C++11 den Weg in die Sprache gefunden haben. Es wird Ihnen leichter fallen, die Sprache zu erlernen, wenn Sie mit diesen starten dürfen. Ach, hätte es zu meiner Zeit schon verallgemeinerte Initialisierung oder verlässliches Exceptionhandling gegeben!

Oder hätte es auch nur eine gute Standardbibliothek gegeben, wenn wir schon dabei sind. Sie haben das Glück, zum Beispiel reguläre Ausdrücke und parallelläufige Programme schon mit der Standardbibliothek schreiben zu können. Allerdings geht dieses Buch auf die mehr als umfangreiche Standardbibliothek nur dort ein, wo es hilft, die Sprache zu erlernen. Parallel werden Sie eine Referenz zurate ziehen müssen, gedruckt oder online. Mit diesem Buch will ich erreichen, dass Sie die Referenz auch verstehen und das dort Erklärte effektiv einsetzen können.

Außerdem werden Sie, wenn Sie jetzt mit C++ starten, wahrscheinlich gerade so richtig mit dem Programmieren loslegen, wenn es auch mit C++ so richtig losgeht: C++14 ist gerade verabschiedet und bügelt ein paar kleine Fehler und Falten glatt. Das nächste C++ steht vor der Tür (Arbeitsname: C++17), und mit ihm werden Sie vor allem Erweiterungen der Standardbibliothek sehen: Netzwerkprogrammierung, effiziente Ein- und Ausgabe

und eine noch modernere Parallelisierung. Aber auch an der Sprache wird gefeilt: Sie werden erleben, wie sogenannte Konzepte die Templateprogrammierung bereichern, und Fehlermeldungen des Compilers werden leichter zu verstehen sein. Sie beginnen mit C++ zu einem wahrhaft spannenden Zeitpunkt.

Mir ist nicht nur wichtig, dass Sie die Sprache C++ lernen. Ich möchte auch, dass Sie sie »richtig« einsetzen. Das versuche ich durch die Wahl und Reihenfolge der Themen zu erreichen, die ich Ihnen präsentiere, aber auch dadurch, dass ich nützliche Idiome und Tipps aus der Praxis einstreue. Das geschieht aber, ohne Sie einschränken zu wollen, denn C++ ist und bleibt eine Sprache, mit der Sie selbst wählen können: objektorientiert, typsicher, generisch und so weiter. Also geht es auch bei diesen Tipps um das Verstehen. Wie sollten Sie Programme modularisieren? Womit sollten Sie Klassen ausrüsten? Worauf sollten Sie achten, wenn Sie Performance wollen? Wenn dies auch nicht universelle Hinweise sind, so sind sie diesen doch ziemlich nah.

Mit diesem Buch, so hoffe ich, lernen Sie C++. Und Sie lernen nicht nur, Quellcode zu lesen oder Zeilen aneinanderzureihen mit Programmen die aus einer Eingabe eine Ausgabe machen (obwohl auch dieses trivial erscheinende Konzept wichtig ist). Sie lernen, es so zu machen, dass Sie weniger Fehler machen werden – weniger *Bugs* produzieren. Insbesondere, wenn Sie Programme schreiben wollen, die Bestand haben, die Sie oder andere noch in Jahren weiterentwickeln oder auch nur nutzen, dann sollte Ihnen diese Eigenschaft am Herzen liegen.

Schreiben Sie *nachhaltigen* Code.

Bielefeld,

Torsten T. Will