mitp

V. Anton Spraul

THINK LIKE A PROGRAMMER

Typische Programmieraufgaben kreativ lösen am Beispiel von C++

Einführung

Bereitet es Ihnen Mühe, Programme zu schreiben, obwohl Sie davon überzeugt sind, Programmiersprachen eigentlich verstanden zu haben? Lesen Sie manchmal ein ganzes Kapitel in einem Buch über Programmierung und nicken dabei ständig wissend, aber es fällt Ihnen dennoch schwer, das Gelesene in Ihren eigenen Programmen umzusetzen? Können Sie im Internet gefundene Beispielprogramme so gut verstehen, dass Sie anderen die Bedeutung jeder einzelnen Codezeile erklären können, aber glauben, einen Krampf im Kopf zu haben, wenn Sie einer Programmieraufgabe gegenüberstehen und den leeren Bildschirm Ihres Texteditors sehen?

So geht es nicht nur Ihnen. Ich unterrichte seit mehr als 15 Jahren Programmierung, und den meisten meiner Studenten ist es irgendwann während ihrer Ausbildung ebenfalls so ergangen. Bei der hier fehlenden Fähigkeit handelt es sich darum, *Problemlösungen* zu finden, also anhand der Beschreibung einer bestimmten Aufgabe ein neues Programm zu schreiben, das diese Problemstellung löst. Nicht jede Art der Programmierung erfordert umfangreiches Lösen von Problemen. Falls Sie ein bereits bestehendes Programm nur geringfügig ändern, debuggen oder um Testcode ergänzen, geschieht die Programmierung möglicherweise so »mechanisch«, dass Ihr Erfindungsreichtum gar nicht erst auf die Probe gestellt wird. Früher oder später erfordern jedoch alle Programme das Lösen von Problemstellungen, und alle guten Programmierer können das.

Problemlösungen zu finden ist schwierig. Bei manchen Menschen, den »Naturtalenten« (in der Welt der Programmierung das Pendant zu begabten Athleten im Sport), scheint es so einfach zu sein. Diese wenigen Auserwählten übersetzen komplizierte Ideen wie im Schlaf in Quellcode. Wenn man ein Java-Gleichnis bemühen möchte, könnte man sagen, dass der Java-Code kompiliert im Gehirn abläuft, während wir anderen eine virtuelle Maschine verwenden müssen, die den Code nur interpretiert.

Als Programmierer kein Naturtalent zu sein, ist aber auch nicht weiter schlimm. Wäre dem so, gäbe es nur sehr wenige Programmierer auf der Welt. Dessen ungeachtet sind mir schon zu viele achtbare angehende Programmierer begegnet, die sich zu lange frustriert abmühen. Schlimmstenfalls geben sie das Programmieren komplett auf und sind davon überzeugt, ihr Ziel nie zu erreichen und dass nur Leute mit einer angeborenen Gabe gute Programmierer sind.

Warum aber ist es so schwierig, zu erlernen, Programmieraufgaben zu lösen?

Zum Teil liegt es daran, dass das Lösen von Problemen und das Erlernen der Syntax einer Programmiersprache ganz unterschiedliche Tätigkeiten sind und daher völlig andere »Muskeln« des Verstands beanspruchen. Beim Erlernen der Syntax einer Programmiersprache, dem Lesen von Quellcode und dem Auswendiglernen einer Programmierschnittstelle (*API*, englisch *Application Programming Interface*) handelt es sich um vornehmlich analytische Vorgänge in der linken Hirnhälfte. Ein originelles Programm unter Verwendung der vorher erlernten Werkzeuge und Fähigkeiten zu schreiben, ist hingegen eine Aktivität der rechten, kreativen Hirnhälfte.

Nehmen wir an, Sie möchten einen Zweig entfernen, der in die Regenrinne Ihres Hauses gelangt ist, aber Ihre Leiter ist zu kurz, um den Zweig zu erreichen. Sie begeben sich in Ihre Garage und suchen nach irgendeinem Objekt (oder einer Kombination von Objekten), das es Ihnen erlaubt, den Zweig aus der Regenrinne zu entfernen. Gibt es eine Möglichkeit, die Leiter zu verlängern? Gibt es ein Objekt, das Sie auf der Leiter stehend dazu verwenden könnten, den Zweig zu erreichen und fortzuschaffen? Vielleicht könnten Sie auch vom Inneren des Hauses aus einfach aufs Dach steigen und den Zweig von ober her zu fassen bekommen. Das ist mit Problemlösen gemeint, und es handelt sich dabei um eine kreative Tätigkeit. Ob Sie es glauben oder nicht, wenn Sie ein Programm erstmals entwerfen, sind Ihre Denkvorgänge und diejenigen der Person, die den Zweig entfernen möchte, ziemlich ähnlich und unterscheiden sich deutlich von den Denkvorgängen einer Person, die eine bereits vorhandene `for`-Schleife debuggt.

Allerdings legen die meisten Bücher über Programmierung ihren Schwerpunkt auf Syntax und Semantik. Nun sind Syntax und Semantik einer Programmiersprache natürlich unverzichtbar, aber sie sind nur der erste Schritt beim Lernen, wie man in der Sprache programmiert. Die meisten Bücher für Programmieranfänger führen im Wesentlichen vor, wie man ein Programm versteht, nicht, wie man eines schreibt. Diejenigen Bücher, die sich auf das Schreiben von Programmen konzentrieren, sind häufig tatsächlich »Kochbücher«, in denen »Rezepte« für bestimmte Gegebenheiten aufgeführt sind. Derartige Bücher können zwecks Zeitersparnis durchaus wertvoll sein, sind aber ungeeignet, um zu lernen, eigenständig Code zu schreiben. Denken Sie an Kochbücher im eigentlichen Sinn. Auch wenn hervorragende Köche Kochbücher besitzen, wird niemand, der sich nur auf Kochbücher verlässt, zu einem großartigen Koch. Ein exzellenter Koch kennt sich mit Zutaten, deren Vorbereitung und Garmethoden aus und weiß, wie er diese miteinander kombinieren muss, um fantastische Gerichte zuzubereiten. Um eine leckere Mahlzeit anzurichten, benötigt ein wirklich guter Koch nur eine vollständig ausgestattete Küche. In vergleichbarer Weise kennt sich ein ausgezeichneter Programmierer mit der Syntax der Sprache, Anwendungsframeworks, Algorithmen und

den Prinzipien der Softwareentwicklung aus und versteht es, diese miteinander zu verbinden, um tolle Programme zu erstellen. Geben Sie einem exzellenten Programmierer eine Liste mit Vorgaben und lassen Sie ihn dann auf eine vollständig ausgerüstete Programmierumgebung los; es werden großartige Dinge geschehen!

Bei der Ausbildung von Programmierern gibt es gegenwärtig im Bereich des Problemlösens kaum Orientierungshilfen. Stattdessen geht man davon aus, dass Programmierer, die Zugriff auf alle erforderlichen Programmierwerkzeuge besitzen, irgendwann schon lernen werden, gute Programme zu schreiben, wenn sie dazu aufgefordert werden. Das lässt sich auch nicht völlig von der Hand weisen, aber von jetzt bis »irgendwann« kann ein ziemlich langer Zeitraum sein. Vom Programmieranfänger bis zum abgeklärten Programmierer ist es ein langer und steiniger Weg, und zu viele der Leute, die sich auf den Weg gemacht haben, erreichen ihr Ziel nie.

Anstatt durch Versuch und Irrtum zu lernen, können Sie sich das Lösen von Problemen auch systematisch aneignen. Darum geht es in diesem Buch. Sie können Techniken zum Ordnen Ihrer Gedanken und Verfahren zum Auffinden von Lösungen und Strategien für bestimmte Arten von Problemen erlernen. Lassen Sie Ihrer Kreativität beim Studieren der verschiedenen Ansätze freien Lauf. Geben Sie sich hier keiner Täuschung hin: Programmieren und insbesondere das Lösen von Problemen sind kreative Tätigkeiten. Kreativität bleibt rätselhaft und niemand vermag genau zu sagen, wie ein kreativer Kopf funktioniert. Wenn wir aber imstande sind, das Komponieren von Musik zu erlernen und Ratschläge zum kreativen Schreiben zu befolgen, oder uns zeigen lassen können, wie man malt, dann können wir auch lernen, Aufgabenstellungen beim Programmieren kreativ zu lösen. Dieses Buch möchte Ihnen nicht vorschreiben, was genau Sie tun sollen. Es soll Ihnen vielmehr dabei helfen, Ihre verborgenen Problemlösungsfähigkeiten weiterzuentwickeln, sodass Sie dann schon wissen werden, was zu tun ist. Es geht hier vor allem darum, Ihnen dabei zu helfen, ein Programmierer zu werden, wie er im Buche steht.

Mein Ziel ist es, dass Sie (und alle anderen Leser des Buchs) es lernen, beliebige Programmieraufgaben systematisch in Angriff zu nehmen und dabei zuversichtlich sind, die jeweilige Aufgabe schließlich lösen zu können. Ich wünsche mir, dass Sie nach der Lektüre des Buchs *wie ein Programmierer denken* und dass Sie sich *für einen Programmierer halten*.

Über dieses Buch

Nachdem die Notwendigkeit dieses Buches erläutert wurde, sind noch einige Bemerkungen dazu nötig, um was es sich beim vorliegenden Buch eigentlich handelt – und um was nicht.

Voraussetzungen

In diesem Buch wird davon ausgegangen, dass Ihnen die grundlegende Syntax und die Semantik der Programmiersprache C++ geläufig sind und Sie bereits damit begonnen haben, Programme zu schreiben. In den meisten Kapiteln wird von Ihnen erwartet, dass Sie bestimmte C++-Grundlagen kennen. In diesen Kapiteln finden Sie eingangs einen Überblick über diese Grundlagen. Machen Sie sich keine Sorgen, falls Sie noch im Begriff sind, die Sprache zu erlernen. Es gibt eine Vielzahl ausgezeichneter Bücher über C++ und Sie können das Lösen von Problemen und die Syntax gleichzeitig erlernen. Sie müssen jedoch die jeweils relevante Syntax erlernt haben, *bevor* Sie versuchen, die Aufgabenstellungen eines Kapitels in Angriff zu nehmen.

Ausgewählte Themen

Die im Buch behandelten Themen decken die Bereiche ab, bei denen ich Programmieranfänger am häufigsten mit Schwierigkeiten habe kämpfen sehen. Diese Bereiche stellen gleichzeitig einen umfassenden Querschnitt der für Anfänger und fortgeschrittene Anfänger maßgeblichen Themengebiete dar.

Ich muss jedoch betonen, dass es sich hier nicht um ein »Kochbuch« mit Algorithmen oder Verfahren zum Lösen bestimmter Problemstellungen handelt. Auch wenn in späteren Kapiteln erläutert wird, wie sich wohlbekannte Algorithmen oder Verfahren einsetzen lassen, sollten Sie dieses Buch nicht als »Spickzettel« verwenden, um bestimmte Aufgabenstellungen zu überwinden, oder sich ausschließlich auf die Kapitel konzentrieren, die in direktem Zusammenhang mit Ihren aktuellen Bemühungen stehen. Sie sollten vielmehr das gesamte Buch durcharbeiten und nur dann Lehrstoff überspringen, wenn Ihnen die erforderlichen Voraussetzungen fehlen, um der Themenbehandlung zu folgen.

Programmierstil

An dieser Stelle eine kurze Anmerkung zum im Buch verwendeten Programmierstil. Dieses Buch befasst sich weder mit Hochleistungs-Programmierung noch mit der Erstellung möglichst kompakter, effizienter Codes. Der in den Codebeispielen von mir verwendete Programmierstil soll vor allem gut lesbar und möglichst verständlich sein. In einigen Fällen werden mehrere Schritte ausgeführt, um ein Ziel zu erreichen, auch wenn dies in einem einzigen Schritt möglich wäre, damit das Prinzip deutlich wird, das ich demonstrieren möchte.

Es werden dennoch auch einige Aspekte des Programmierstils im Buch behandelt – allerdings nur bedeutsame, wie etwa, was eine Klasse beinhalten sollte und was nicht, nicht aber belanglose, wie beispielsweise wie der Code einzurücken ist. Als angehender Programmierer möchten Sie selbstverständlich einen einheitlichen, gut lesbaren Stil in allen Ihren Werken verwenden.

Übungen

Das Buch enthält eine Reihe von Programmierübungen, ist jedoch kein Schulbuch und Sie werden am Ende des Buches keine Antworten auf die Übungsaufgaben finden. Die Übungen bieten Ihnen die Möglichkeit, die in den einzelnen Kapiteln vorgestellten Konzepte anzuwenden. Ob Sie davon Gebrauch machen, steht Ihnen natürlich frei, es ist aber ganz entscheidend, dass Sie diese Konzepte in die Tat umsetzen. Mit dem einfachen Durchlesen des Buchs wird nichts erreicht. Wie bereits erwähnt, möchte Ihnen dieses Buch nicht vorschreiben, was genau Sie in einer bestimmten Situation tun sollen. Durch Anwendung der in diesem Buch vorgestellten Techniken werden Sie selbst die Fähigkeit entwickeln, herauszufinden, was zu tun ist. Ein weiteres Hauptziel dieses Buches, die Steigerung Ihres Vertrauens in die eigenen Fähigkeiten, erfordert darüber hinaus Erfolgserlebnisse. Tatsächlich handelt es sich hier um eine bestens geeignete Methode, um festzustellen, ob Sie bereits genug Übungen eines bestimmten Themenbereichs durchgearbeitet haben: wenn Sie sich sicher sind, andere Aufgabenstellungen aus diesem Themenbereich lösen zu können. Und zu guter Letzt sollten die Übungsaufgaben auch *Spaß* machen. Auch wenn es sicherlich Momente geben wird, in denen Sie sich lieber mit etwas anderem beschäftigen möchten, sollte die Ausarbeitung der Lösung einer Problemstellung bei der Programmierung eine lohnenswerte Herausforderung sein.

Stellen Sie sich dieses Buch als eine Art Hindernislauf für Ihren Verstand vor. Hindernisläufe sorgen für Kraft, Ausdauer und Geschicklichkeit sowie für Zutrauen beim Ausbilder. Durch die Lektüre der einzelnen Kapitel und die praktische Umsetzung möglichst vieler Übungen werden Sie Selbstvertrauen aufbauen und Problemlösungsfähigkeiten entwickeln, die bei allen möglichen Aufgabenstellungen Anwendung finden. Falls Sie künftig auf eine schwierige Problemstellung stoßen, werden Sie wissen, wie die Aufgabe in Angriff zu nehmen ist.

Warum C++?

Die Programmbeispiele dieses Buches sind in C++ programmiert. Davon einmal abgesehen geht es in diesem Buch um Problemlösungen durch Programme – nicht speziell um C++. Sie werden hier also kaum Tipps und Tricks zu C++ finden, denn die im Buch vorgestellten allgemeinen Ideen sind in jeder Programmiersprache einsetzbar. Gleichwohl kann man nicht über Programmierung reden, ohne auch Programme zu erwähnen, und daher musste eine konkrete Programmiersprache ausgewählt werden.

Es gibt mehrere Gründe für die Auswahl von C++. Erstens ist C++ in vielen verschiedenen Bereichen weit verbreitet. Zweitens stammt C++ von der streng prozeduralen Sprache C ab, sodass sich C++-Programme sowohl prozedural als auch objektorientiert schreiben lassen. Die objektorientierte Programmierung ist inzwischen so verbreitet, dass sie bei der Erörterung von Problemlösungen nicht wegge-

lassen werden darf, aber viele der fundamentalen Konzepte bei der Lösung von Problemen lassen sich bestens durch Begriffe der streng prozeduralen Programmierung beschreiben, was sowohl den Code als auch die Erläuterungen vereinfacht. Drittens erlaubt es C++ als maschinennahe (*low-level*) Sprache mit komplexen (*high-level*) Bibliotheken, beide Ebenen der Programmierung zu untersuchen. Die besten Programmierer können bei Bedarf maschinennahe Lösungen implementieren und gleichzeitig die komplexen Bibliotheken und APIs nutzen, um die zur Programmierung erforderliche Zeit zu verringern. Und schließlich – teilweise auch als Resultat der anderen genannten Gründe – ist C++ eine ausgezeichnete Wahl, weil Sie Problemstellungen in allen anderen Programmiersprachen lösen können, wenn Sie gelernt haben, diese in C++ zu lösen. Viele Programmierer haben festgestellt, dass sich die in einer bestimmten Programmiersprache erworbenen Kenntnisse auch in anderen Sprachen als nützlich erweisen. Für C++ trifft dies aufgrund des zweigleisigen Ansatzes (prozedural und objektorientiert) in besonderem Maße zu. Dazu trägt – offen gestanden – auch der Schwierigkeitsgrad beim Erlernen von C++ bei. C++ ist nun mal der »wahre Jakob« – Programmieren ohne Netz und doppelten Boden. Anfangs ist dies zwar abschreckend, aber sobald Sie die ersten Erfolge in C++ erzielen, wird Ihnen klar werden, dass Sie nicht nur ein wenig Code zusammenstückeln können, sondern zu einem Programmierer werden.