1 Einführung

»Und was kann ich alles programmieren, wenn ich das Buch durchgelesen habe?«

Eine beliebte Frage auch in Schulungen, wenn die Sprache nicht mit einem speziellen Ziel im Hinterkopf gelernt wird, sondern als erster Einstieg in die Welt des Programmierens gewählt wurde und die genaue Reiseroute noch nicht feststeht.

Um diese Frage zu beantworten, geht dieses Buch einen besonderen Weg. Anstatt die einzelnen Elemente der Sprache isoliert zu behandeln und es dem Leser zu überlassen, wo sich das Gelernte in einem größeren Programm wiederfindet, werden wir uns in diesem Kapitel das Ergebnis eines größeren Projekts anschauen und uns dann schrittweise die notwendigen Kenntnisse aneignen, um das Programm verstehen, erweitern und sogar selbst programmieren zu können.

Abgesehen von diesem großen buchumspannenden Projekt gibt es noch Beispiele, die über mehrere Kapitel entwickelt werden. Zusätzlich wird jedes Thema in genügend kleineren Beispielen erläutert, um die Anwendung der Sprachelemente zu verstehen.

1.1 Über das Buch

Bevor ich Ihnen das große Projekt vorstelle, möchte ich noch kurz abgrenzen, um was es in diesem Buch geht und an wen es gerichtet ist.

Zielgruppe

Dieses Buch richtet sich an Einsteiger in die Programmierung sowie Personen, die bereits erste Erfahrungen mit C++ gemacht haben.

Jedes Thema beginnt mit einer einfachen Einführung und vielen Beispielen, die immer weiter ausgebaut werden, bis auch Details von C++ zur Sprache kommen.

Je nach Vorwissen kann Ihnen eventuell die Einführung zu ausführlich oder die Details zu detailliert erscheinen. Das ist überhaupt kein Problem, übersprin-

gen Sie die Abschnitte einfach. Immer, wenn ein bereits im Buch behandeltes Thema an anderer Stelle Anwendung findet, steht ein Querverweis im Text, sodass Sie die tiefer gehenden Abschnitte auch erst bei Bedarf lesen können.

C++14

Das in diesem Buch verwendete C++ entspricht dem C++14-Standard, der bei Drucklegung dieses Buches kurz vor der offiziellen Verabschiedung steht. Die Erweiterungen von C++11 zu C++14 sind aber nur minimal und beziehen sich in vielen Fällen auf fortgeschrittene Themen, die in diesem Buch nicht behandelt werden. Alle Programmcodes im Buch lassen sich sowohl mit einem C++14-Compiler übersetzen als auch mit einem C++11-Compiler. Im Buch wird an vielen Stellen auf den internationalen Standard ISO/IEC 14882-2011 in der Form [C++ #] verwiesen, wobei # für eine (Kapitel-)Nummer steht.

Objektorientierte Programmierung

Natürlich beschäftigt sich das Buch mit der objektorientierten Programmierung (OOP). Nur: Im Gegensatz zu anderen Sprachen wie zum Beispiel Java erzwingt C++ keine objektorientierte Programmierung. Viele Sprachmittel setzen keine Elemente der objektorientierten Programmierung ein oder sind sogar flexibler, wenn sie außerhalb einer Klasse stehen.

Als ehemaliger Java-Programmierer erscheint Ihnen daher die OOP vielleicht zu spät im Buch, als C-Programmierer kommt sie unter Umständen zu früh.

Um das Ganze mit Zahlen zu hinterlegen: Über 40% des Buches handeln ausschließlich von den Mechanismen der Klassen und der Vererbung und weitere 30% setzen diese Mechanismen ein. Trotzdem gebe ich dem Leser die Zeit, die Sprache zunächst mit ihren weniger abstrakten prozeduralen Mechanismen kennenzulernen, bevor wir die höheren Abstraktionsebenen der objektorientierten Programmierung betreten, die in C++ anspruchsvoller sind als in anderen Sprachen.

1.2 Vorstellung des Projekts

Der Mensch lernt am leichtesten und effektivsten spielerisch. Was liegt bei der Erlernung einer Programmiersprache also näher, als ein Computerspiel zu entwerfen?

Die Herausforderung bestand darin, ein Spielkonzept zu entwickeln, das mit reinem ISO-C++ realisiert werden kann, ohne plattformspezifische Bibliotheken einsetzen zu müssen. Grafische Darstellungen fielen damit schon mal weg. Es musste ein Spiel auf Textebene werden. Weil das Genre der Point&Click-Adventure eines meiner liebsten ist, habe ich mich dazu entschieden, ein Spiel im Stile seiner Vorgänger zu programmieren: ein Text-Adventure.

Den gesamten Programmcode des Spiels, zusätzliche Informationen sowie eine plattformunabhängige Java-Version zum direkten Ausprobieren finden Sie auf meiner Homepage unter *http://cpp.andrewillms.de*.

Ziel des Spiels ist es, aus einem Haus herauszukommen. Dazu muss das Haus erkundet, neue Räume erschlossen und Gegenstände eingesammelt und kombiniert werden. Grundsätzlich gilt: Sie können auch verlieren, entweder direkt, weil Sie beispielsweise einen Stromschlag bekommen haben und deshalb ohnmächtig wurden, bis der Hausbesitzer zurückkommt, oder indirekt, weil Sie sich den Weg verbaut haben, zum Beispiel, weil Sie einen notwendigen Gegenstand zerstört, verloren oder erst gar nicht eingesammelt haben. Die vielen unterschiedlichen und mitunter auch skurrilen Arten des Verlierens machen aber einen Teil des Spielspaßes aus. Sie sollten daher Folgendes beachten:

- Regelmäßig Spielstände speichern.
- Alle Gegenstände untersuchen und gegebenenfalls benutzen.
- Auch verschiedene Aktionen probieren. Es macht oft einen Unterschied, ob Gegenstände benutzt, gezogen, gedrückt, geöffnet oder geschlossen werden.
- Erst einmal nichts tun, was Sie auch in Ihrem eigenen Haus unterlassen würden, wie zum Beispiel alles abfackeln. (Es sei denn, es muss sein.)

Um die Programmierung einfach zu halten, ist die Syntax der Eingabemöglichkeiten rudimentär. »gehe treppe«, »verwende haarknäuel + toilette« gelten innerhalb des Spiels als gutes Deutsch. Präpositionen und Artikel sind nicht erlaubt.

Es reicht immer aus, so viel von einem Wort anzugeben, dass es eindeutig ist. Bei den Befehlen können bis auf »Speichern« alle mit nur einem Buchstaben abgekürzt werden. Damit ist »g v« eine gültige Schreibweise für »gehe vor«.

Es ist für das Lesen des Buches nicht notwendig, das Spiel durchgespielt oder überhaupt gespielt zu haben. Es erleichtert aber das Verständnis, wenn Sie die Programmbeispiele den Aktionen im Spiel zuordnen können.

Als grobe Richtschnur gilt: Sie haben einen guten Einblick in das Spiel, wenn Sie es unfallfrei in den Keller geschafft haben. Haben Sie gar den Vorratsraum erreicht, kennen Sie alle programmtechnischen Spielelemente.

1.3 Identifizieren der Programmteile

An dieser Stelle wollen wir das Spiel einer genaueren Betrachtung unterziehen und typische Elemente eines Computerprogramms ausmachen. Diese werden dann in den weiteren Kapiteln im Detail besprochen.

1.3.1 Objekte

Im Spiel springen zwei Elemente direkt ins Auge, die Räume und die darin befindlichen Gegenstände. Solche klar abgegrenzten und unterscheidbaren Entitäten nennt man in der objektorientierten Programmierung Objekte.

Objekte besitzen üblicherweise zu jedem Zeitpunkt einen eindeutigen Zustand. Das Handtuch besitzt zu Beginn den Zustand »In der Toilette befindlich«. Nachdem der Spieler es genommen hat, wechselt der Zustand zu »Im Inventar befindlich«. Das Feuerzeug ist zunächst unsichtbar. Wenn der Spieler die Jacken untersucht, wechselt der Zustand zu »Sichtbar, im Inventar und 25% gefüllt«. Die zustandsbeschreibenden Elemente eines Objekts nennt man Attribute oder Objektdaten.

Die vorangegangenen Beispiele offenbaren eine weitere Eigenschaft von vielen Objekten; die Fähigkeit, ihren Zustand zu verändern. Die zustandsverändernden Elemente eines Objekts werden als Methoden, Nachrichten und speziell in C++ auch als Elementfunktionen bezeichnet.

1.3.2 Kontrollstrukturen

Es reicht aber nicht aus, dass die Objekte einen Zustand besitzen, den sie ändern können. Sie müssen zu dieser Zustandsänderung auch aufgefordert werden, aber nicht unkontrolliert, sondern nur, wenn bestimmte Bedingungen gelten.

Falls der Spieler in der Toilette steht und das Handtuch im Raum liegt und der Spieler »nimm handtuch« als Befehl angibt, genau dann wechselt das Handtuch seinen Zustand zu »Im Inventar befindlich«. Dieses bedingte Abarbeiten von Anweisungen wird in der Programmierung Verzweigung genannt.

Eine andere Form der Kontrollstruktur ist bei der Eingabe des Anwenders involviert. Dort hat der Anwender die Möglichkeit, einen Befehl und betroffene Objekte einzugeben. Gibt er »ende« ein, dann ist das Spiel beendet. In allen anderen Fällen wird der Text zerlegt, der Befehl und die Objekte identifiziert, der Befehl ausgeführt und der Anwender erneut nach einem Befehl gefragt.

Die Anweisung lässt sich so formulieren: Solange der Anwender nicht »ende« eingegeben hat, wird der Befehl ausgeführt und erneut nach einem Befehl gefragt. Diese Art der Kontrollstruktur wird Wiederholung oder Schleife genannt.

1.4 Abstraktion

Ein weiteres Thema bei der Programmierung des Spiels ist der Weg von der Idee zum Programm. In begrenztem Umfang spiegelt das Spiel die reale Welt wider. Genau genommen ist das Spiel eine vereinfachte und abstrahierte Form der Realität.

1.4.1 Datenabstraktion

Bei der Datenabstraktion werden die Attribute, die den Zustand in der realen Welt beschreiben, auf die für das Programm absolut notwendigen Elemente reduziert. So einfach wie möglich, aber nicht einfacher.

Nehmen wir wieder das Handtuch. In der Realität besitzt es eine Breite, eine Länge, eine Dicke und ein Gewicht. Es besitzt eine Farbe, die je nach Qualität nach jedem Waschen mehr oder weniger ausbleicht. Das Handtuch besteht aus 100% Baumwolle mit 50%iger Polyesterbeimischung (frei nach Loriot), es ist mehr oder weniger mit Weichspüler gesättigt und daher weicher oder kratziger. Ich könnte diese Liste bis zum Ende des Buches fortführen.

Im Spiel kann sich das Handtuch nur auf der Toilette befinden oder im Inventar sein, es kann nass oder trocken sein. Mehr muss es nicht können. Die Reduktion aller möglichen Zustände eines realen Handtuchs auf diese beiden Attribute im Spiel nennt man Datenabstraktion.

1.4.2 Algorithmische Abstraktion

Dass die Zustände der Objekte innerhalb des Programms abgebildet werden können, ist aber nur die halbe Miete. Sinnvoll sind Zustände nur, wenn sie sich auch ändern können.

Jedes Programm besitzt einen Startzustand (definiert über die zu Beginn des Programms vorhandenen Objekte und deren Zustände) und einen Endzustand (definiert über die am Ende des Programms vorhandenen Objekte und deren Zustände). Die Regeln, wann und wie sich die Zustände während des Programmlaufs ändern, werden über den Algorithmus definiert.

Nehmen wir exemplarisch die Berechnung des größten gemeinsamen Teilers zweier positiver ganzer Zahlen. Dieser wird unter anderem beim Kürzen von Brüchen verwendet. Der ggT (die Abkürzung für »größter gemeinsamer Teiler«) von 6 und 12 ist beispielsweise 6, der ggT von 6 und 9 ist 3. Konkrete ggT zu bestimmen, fällt nicht sehr schwer.

Schwieriger wird es, eine allgemeingültige Lösung zur Bestimmung des ggT zu formulieren.

Algorithmus

Ein Algorithmus ist eine Menge von Regeln, durch deren Befolgung in festgelegter Reihenfolge ein bestimmtes Problem gelöst wird.

Eine einfache Beschreibung einer Lösung könnte so aussehen:

Der ggT zweier Zahlen kann naturgemäß nicht größer sein als die kleinere der beiden Zahlen. Deshalb beginnen wir mit ihr als potenziellem ggT.

Ist die so gefundene Zahl nicht ggT der beiden Zahlen, dann vermindere sie um 1 und wiederhole diesen Schritt.

Spätestens bei 1 terminiert diese Schleife, denn die 1 ist Teiler von allem.

Um die Lösung zu visualisieren, verwende ich das Aktivitätsdiagramm der UML. Die Unified Modeling Language, kurz »UML«, ist eine grafische Sprache, mit deren Hilfe Sachverhalte der Softwareentwicklung (Programmfluss, Klassen- und Objektbeziehungen, Zustände etc.) dargestellt werden.

In Abbildung 1–1 wird das Aktivitätsdiagramm der UML eingesetzt, um einen Algorithmus grafisch darzustellen.

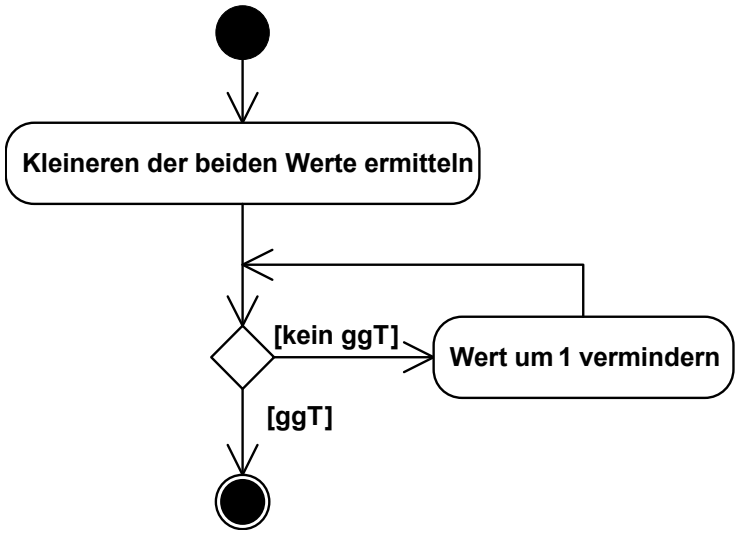

Abb. 1–1 *Der größte gemeinsame Teiler als Aktivitätsdiagramm*

Der ausgefüllte Kreis definiert den Start der Aktivität, sie endet am ausgefüllten Kreis mit Ring.

Einzelne Schritte oder Anweisungen werden in Form von Rechtecken mit abgerundeten Ecken dargestellt. Die Abarbeitungsreihenfolge ist über die Pfeile definiert, die aus den Symbolen heraustreten beziehungsweise zu ihnen hin führen.

Das auf der Spitze stehende Quadrat definiert eine Verzweigung. Aus einer Verzweigung tritt immer mehr als ein Pfeil aus. An den Pfeilen stehen in eckigen Klammern die sogenannten Wächter. Die bestimmen, unter welcher Bedingung dem jeweiligen Pfeil gefolgt wird.

Jedem Programm liegt zwangsläufig ein Algorithmus zugrunde. Wir werden im alltäglichen Leben andauernd mit Algorithmen konfrontiert. Alle nach Regeln ablaufenden Tätigkeiten sind gewissermaßen Algorithmen. Typische Algorithmen sind zum Beispiel einen Kuchen backen oder einen Fahrschein aus dem

Automaten ziehen. Selbst das Abschließen der Wohnungstür ist ein Algorithmus, wenn auch ein recht primitiver.

Wo immer bestimmte Tätigkeiten ein Problem lösen oder eine Aufgabe bewältigen, haben wir es mit Algorithmen zu tun. Dabei werden Algorithmen in der Weise formuliert, die auch Personen ohne Fachkenntnis das Lösen des Problems ermöglicht. Dies setzt eine Beschreibung mit möglichst klein gehaltenem Vokabular voraus.

Nehmen wir als Beispiel ein Kuchenrezept. Selbst wenn absolut kein Wissen um die Kunst des Kuchenbackens vorliegt, ist man in der Lage, einen Kuchen nach Rezept zu backen, weil sich die Anweisungen einer Sprache bedienen, die jedem Nichtbäcker verständlich ist (z. B. rühren, kneten, in die Backform füllen, Zucker abwiegen).

Und wenn man einen Algorithmus so formuliert, dass ein Computer ihn befolgen kann, dann spricht man von einem Programm.

Programm

Einen Algorithmus, der für den Computer verständlich formuliert wurde, nennt man Programm.