ALGE Verlag

ECDL

Farbausgabe

Computing

mit Python

ECDL Standard Modul

Österreichische Computer Gesellschaft

ECDL Computing

mit Python 3.x

OCG

Auflage 3 / Juli 2020

Österreichische Computer Gesellschaft (OCG)
Wollzeile 1/1/1
1010 Wien
Österreich
Telefon: +43 (0)1/51202350
E-Mail: info@ocg.at
Internet: www.ocg.at | www.ecdl.at | blog.ocg.at

ALGE EDV-Consulting GmbH
Sdl Neugebäude Weg 8/239
A-1110 Wien
Telefon: +43(01) 347 04 76
E-Mail: verlag@alge-edv.at
Internet: www.alge-edv.at

Autorenteam: ECDL Foundation/ OCG
Deckblattgestaltung: Ariane Kascha
Lekorat: Mag.ª Nora Paul

Bestellnummer: ALD-9820; unter ISBN: 978-3-903298-20-0 als Buch produziert

VORWORT

Sehr geehrte Leserin, sehr geehrter Leser,

der Europäische Computer Führerschein (ECDL) ist eine 25jährige Erfolgsstory über den ganzen Globus hinweg. Gerade in dieser Zeit wandelt der ECDL in ein lebensbegleitendes Programm – einem Internationalen Zertifikat für Digitale Kompetenz („International Certificate for Digital Literacy (ICDL)).

Mit diesem Lernmaterial für den ECDL bzw. ICDL haben Sie eine gute Wahl getroffen!

Die Österreichische Computer Gesellschaft (OCG) bestätigt mit der Approbation, dass dieses Lernmaterial den Qualitätskriterien der ECDL/ICDL-Initiative entspricht.

Der ECDL/ICDL ist das weltweit führende Zertifikat für IT-Kompetenz. Er wird in über 100 Ländern umgesetzt, aktuell nehmen mehr als 15 Millionen Menschen am Programm teil. In Österreich wird der ECDL bzw ICDL von der OCG koordiniert und organisiert.

Durch diese ECDL/ICDL Initiative wurde ein **international einheitliches Niveau** festgelegt. Ein effizienter und sicherer Einsatz der Programme erfordert fundierte praxisrelevante Kenntnisse. Genau dieses Wissen ist daher eine wesentliche Voraussetzung für den beruflichen Erfolg am Arbeitsmarkt, für Erfolg in der Schule oder beim Studium.

Ein **ECDL / ICDL -Zertifikat ist DER international anerkannte Nachweis** für Ihre Kenntnisse! Durch die **flexible modulare Struktur** des Zertifizierungsprogramms können Sie wählen, für welche Bereiche Sie ein ECDL/ICDL-Zertifikat erlangen möchten. Informieren Sie sich auf unserer Website www.ecdl.at über das Portfolio des ECDL/ICDL,
oder rufen Sie uns an – wir beraten Sie gerne!

Viel Freude beim Lernen und viel Erfolg bei der ECDL bzw ICDL -Prüfung wünscht Ihnen

Dr. Ronald Bieber
Generalsekretär

OESTERREICHISCHE COMPUTER GESELLSCHAFT®
AUSTRIAN COMPUTER SOCIETY

Wollzeile 1 | 1010 Wien
Tel +43 1 512 02 35 - 0
E-Mail info@ocg.at
URL www.ocg.at | www.ecdl.at | blog.ocg.at
www.facebook.com/ECDL.Austria

LEKTION 1 – DENKEN WIE PROGRAMMIERERINNEN

Wenn Sie mit dieser Lektion fertig sind, können Sie:

- die wichtigsten Begriffe im Bereich Computing erklären,
- bestimmte Denkweisen als Computational Thinking identifizieren,
- typische Methoden des Computational Thinking erläutern: Problemzerlegung, Mustererkennung, Abstraktion, algorithmisches Design,
- Methoden der Problemzerlegung verwenden, um Daten, Prozesse oder ein komplexes Problem in kleinere Teile zu zerlegen,
- den Begriff Programm erklären,
- die Rolle von Algorithmen beim Computational Thinking verstehen,
- Muster in den zerlegten Teilproblemen identifizieren,
- Abstraktion verwenden, um unnötige Einzelheiten bei der Problemanalyse beiseite zu lassen.

1.1 Computational Thinking

Begriffe

Unter **Computing** versteht man die Durchführung von Berechnungen oder die Verarbeitung von Daten, insbesondere unter Verwendung eines Computersystems.

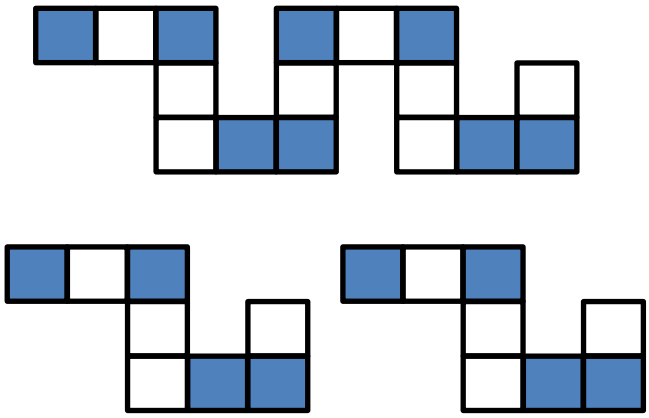
Computational Thinking (Informatisches Denken) ist der Prozess, in dem Herausforderungen und Probleme analysiert und mögliche Lösungen identifiziert werden.

Computational Thinking kann immer dann angewendet werden, wenn man vor komplexen Problemstellungen steht, also nicht nur im Zusammenhang mit Computern, sondern auch im täglichen Leben. Das kann zum Beispiel beim Kochen sein, bei der Planung einer Veranstaltung, der Reparatur eines Gegenstandes, dem Zusammenbauen von Möbeln, der Entwicklung eines neuen Produkts und in vielen anderen Situationen, die Problemlösungen erfordern.

Computational Thinking verwendet vier grundlegende Problemlösungstechniken. Diese können in jeder beliebigen Reihenfolge verwendet oder kombiniert werden.

Mustererkennung (Pattern Recognition)

Ein Muster ist eine wiedererkennbare Anordnung wie der Refrain eines Liedes oder das Motiv auf einer Fliese oder Tapete. Auch in Aktivitäten können Muster entdeckt werden: Man muss zum Beispiel bei einigen Kochrezepten das Backrohr auf eine bestimmte Temperatur vorheizen. Das nennt man ein **gemeinsames Muster.** Bei der Erkennung von Mustern werden Muster oder Wiederholungen in komplexen Problemen oder in kleineren, verwandten Problemen gefunden.



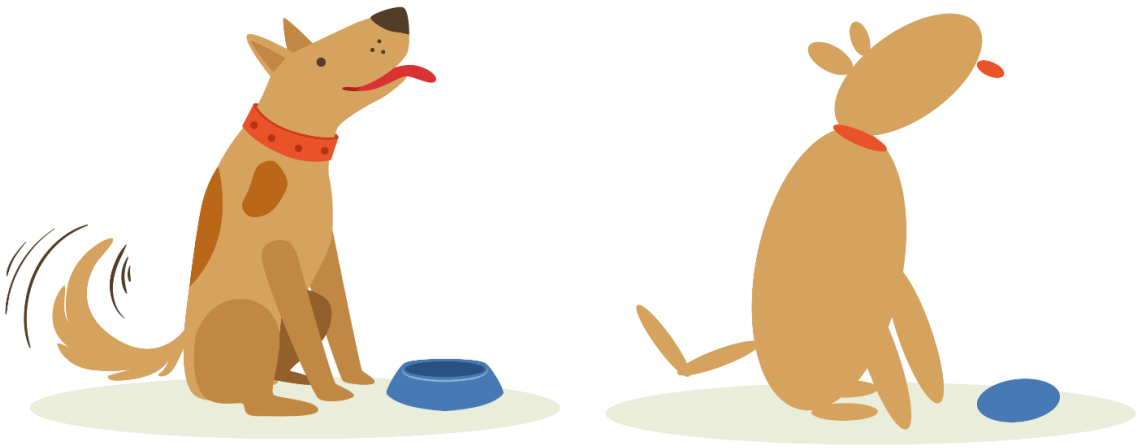
In der Figur oben wurde das erste Set von farbigen Blöcken in zwei Blöcke aufgeteilt, um deutlich zu zeigen, dass ein Muster wiederholt wird.

Abstraktion (Abstraction)

Abstraktion ist der Prozess, bei dem die wichtigsten oder definierenden Merkmale eines Problems oder einer Aufgabe extrahiert werden. Die extrahierten Merkmale stellen die notwendigen Informationen zur Verfügung, um mit der Untersuchung der Aufgabe zu beginnen und mögliche Lösungen zu finden.

Bei der Abstraktion filtert man unnötige Details heraus und konzentriert sich auf die Informationen, die für die Lösung des Problems wichtig sind. Bei der Aufgabe Kekse zu backen ist es nicht wichtig, ob man Links- oder Rechtshänder ist. Die für das Gelingen wichtigen Informationen sind zum Beispiel die Zutaten, die Reihenfolge, in der sie hinzugefügt werden, sowie die Backdauer und -temperatur.

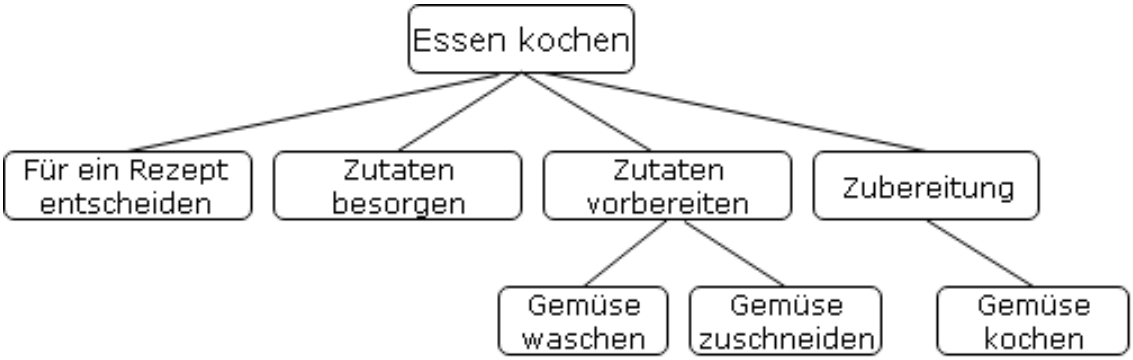
Im folgenden Beispiel wurden alle unnötigen Details vom zweiten Bild des Hundes entfernt, sodass nur die Informationen übrigbleiben, die für die Lösung des Problems wichtig sind – hier ist es ein Hund, der Futter möchte.



Problemzerlegung

Problemzerlegung bedeutet die Zerlegung eines komplexen Problems in kleinere Probleme, die einfacher und leichter zu verstehen sind.

Auch die kleineren Probleme können immer weiter in noch kleinere Probleme zerlegt werden, bis sie leicht verständlich und lösbar sind. Bei der Aufgabe Kuchen zu backen kann zum Beispiel das Erreichen einer bestimmten Backrohrtemperatur eines der kleineren Probleme sein. Das Bild zeigt, wie ein komplexes Problem in immer kleinere Teile und Probleme zerlegt werden kann.

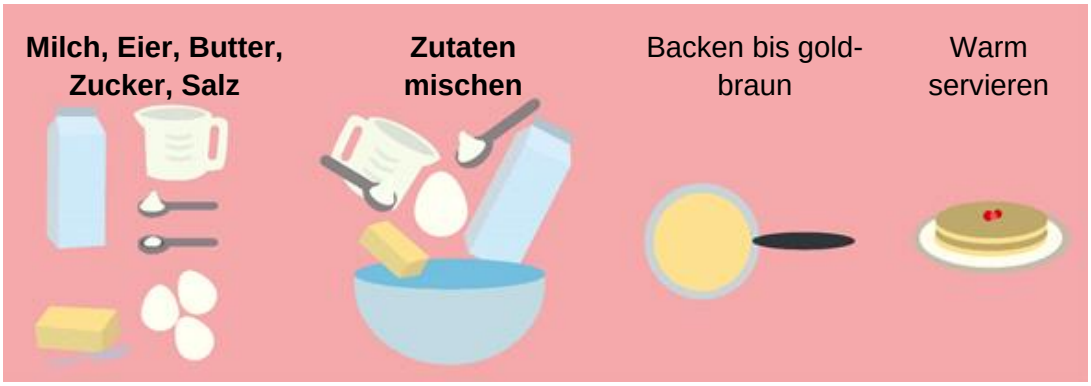


Algorithmisches Design

Ein Algorithmus ist eine eindeutige Handlungsvorschrift zur Lösung einer Aufgabe. Ein Algorithmus könnte zum Beispiel die Anleitung eines Kochrezeptes sein oder Berechnungen, die ein Computer durchführen soll.

Unter dem Entwurf von Algorithmen versteht man die Bereitstellung einer solchen eindeutigen Handlungsvorschrift, um eine bestimmte Aufgabe erfolgreich zu lösen.

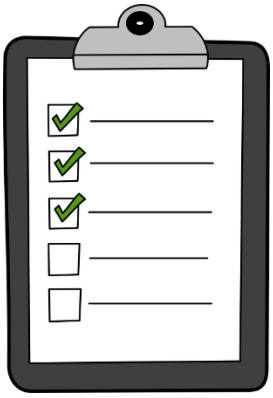
Ein möglicher Algorithmus zur Zubereitung von Palatschinken enthält folgende Schritte:



Zwei zusätzliche Ansätze werden oft zu Computational Thinking hinzugefügt: Evaluierung und Verallgemeinerung:

Evaluierung (Evaluation)

Die Evaluierung beinhaltet die Validierung (Überprüfung) des Entwurfs eines Produktes oder eines Algorithmus. Dabei wird überprüft, ob der gewünschte Zweck damit erfüllt beziehungsweise ob das Problem damit gelöst wird.



Verallgemeinerung (Generalisierung)

Eine Generalisierung ist die Vorgangsweise, aus einer Reihe von vorhandenen Fällen gemeinsame Merkmale zu identifizieren und daraus ein allgemeines Konzept zu formulieren.

Es gibt zum Beispiel viele verschiedene Blumen. Welche Eigenschaften müsste ein allgemeines Konzept für „Blume“ aufweisen? So eine generalisierte Blume hätte Blüten, einen Stängel und Blätter, aber über die genaue Form und Farbe könnte man noch nichts sagen.

Wie man sieht, hat die Generalisierung einige Gemeinsamkeiten mit der Abstraktion.

Auch den Versuch, die Wörter auf Bedienungselementen eines Geräts durch Symbole zu ersetzen (sodass international die gleiche Version eingesetzt werden kann), könnte man als Generalisierung betrachten, wenn man fragt: „Was haben der englische, der deutsche, der französische ... Begriff für ein Objekt gemeinsam?“ Im besten Fall lautet die Antwort: „Man kann sie alle durch das gleiche Symbol darstellen.“

Diese sechs Computational-Thinking-Methoden (vier grundlegende plus Evaluierung und Verallgemeinerung) können zur Lösung von komplexen Problemen miteinander kombiniert werden.

Schritte

Beispiel: Entwicklung einer Waschmaschine

In diesem Beispiel werden die Methoden des Computational Thinking für die Entwicklung einer Waschmaschine eingesetzt.

Abstraktion

Der erste Schritt bei der Entwicklung einer Waschmaschine besteht in der Festlegung der Ziele – sie sollte zum Beispiel auf Wunsch intensiv oder sanft waschen, mit einer niedrigen oder hohen Temperatur, und zu einem sauberen Waschergebnis führen. In der Entwurfsphase werden die Merkmale aufgelistet, die man zur Erreichung dieser Ziele braucht.

Die Entwickler verwenden Abstraktion, um wichtige von unwichtigen Eigenschaften zu unterscheiden. Zu den unwichtigen Eigenschaften gehören alle Details, die weder das Waschergebnis noch das Erscheinungsbild entscheidend beeinflussen.

Maßgebliche Besonderheiten betreffen die Gesamtfunktionalität. Dazu gehört zum Beispiel der Energieverbrauch, der Waschmittelverbrauch, das Waschergebnis, ob empfindliche Textilien gewaschen werden dürfen, wie lange ein Waschgang braucht und wie laut die Waschmaschine im Betrieb ist.

Problemzerlegung (Dekomposition)

Die Problemzerlegung kann verwendet werden, um Aufgaben in kleinere Aufgaben aufzuteilen, die einfacher zu lösen sind:

- Wie gibt man Wäsche in die Waschmaschine hinein und wieder heraus?
- Wie kommt Wasser in die Waschmaschine und wieder heraus?
- Wie vergewissert man sich, dass das Wasser die richtige Temperatur hat?

Diese kleinen Probleme können wiederum in noch kleinere Probleme zerlegt werden:

- Welcher Platz ist für die Tür der Waschmaschine gut geeignet?
- Wie kann man eine Tür bauen, die gut dichtet, sodass kein Wasser ausfließt?

Zerlegung des Problems bei der Entwicklung einer Waschmaschine

- Wo ist die Tür?
- Ist die Tür dicht?
- Wie wird sie verschlossen?
- Wie kommt das Wasser hinein?
- Wie kommt das Wasser heraus?
- Mit welcher Temperatur soll gewaschen werden?
- Wie kann man die korrekte Temperatur sicherstellen?

Algorithmen

Die Summe der Schritte jedes einzelnen Waschgangs kann als Algorithmus betrachtet werden sowie der ganze Herstellungsprozess der Waschmaschine selbst.

Evaluierung

Ein Produkt zu entwickeln ist eine große Herausforderung. Entwürfe in der frühen Entwicklungsstufe sollten laufend evaluiert werden. Durch die Evaluierung lernen Entwickler, wo und wie eine Planung verbessert werden kann. Ist zum Beispiel die Tür der Waschmaschine auch unter Bedingungen dicht, die in einem Waschgang tatsächlich auftreten können? Wenn nicht, muss die Planung schnellstmöglich revidiert werden.

Verallgemeinerung

Teile der Planung für die Waschmaschine könnten auch so verbessert werden, dass die Waschmaschine universell einsetzbar ist: Die Symbole zur sprachunabhängigen Bedienung wurden zuvor bereits erwähnt. Man könnte sie auch so gestalten, dass sie sowohl mit 110 Volt als auch mit 230 Volt – und damit in verschiedenen Ländern – betrieben werden kann. Wenn die Planung auf diese Weise geändert wurde, dann nennt man das Generalisierung (Verallgemeinerung).

Mustererkennung

Die Mustererkennung ist ein mächtiges Werkzeug, das zusammen mit den fünf anderen Methoden in verschiedenen Phasen des Entwicklungsprozesses eingesetzt werden kann. Beispiele dafür:

- Abstraktion:
  Ist ein Detail eines Objekts einer Klasse nicht Teil des gemeinsamen Musters, kann es in einem Abstraktionsschritt möglicherweise weggelassen werden.
- Verallgemeinerung:
  Wenn wir nach einer Verallgemeinerung von verschiedenen Fällen suchen, dann suchen wir genau nach deren gemeinsamen Mustern.
- Problemzerlegung:
  Die durch Zerlegung entstandenen kleineren Teilaufgaben weisen oft gemeinsame Muster auf.

Die sechs Methoden des Computational Thinking können in verschiedenen Entwicklungsphasen und in unterschiedlicher Reihenfolge verwendet werden.

Schritte

Beispiel: Organisation eines Musikfestivals

In diesem Beispiel werden die Methoden des Computational Thinking am Fallbeispiel der Organisation eines Musikfestivals eingesetzt.

Abstraktion

Zu Beginn der Planung eines Musikfestivals muss man klären, was ein Musikfestival überhaupt ist. Der erste Schritt könnte die Auflistung der Dinge sein, die für ein Musikfestival unbedingt erforderlich sind:

- Musik
- Veranstaltungsort
- Marketing und Öffentlichkeit
- Tickets
- Arbeitskräfte
- …

Andere Details bei der Organisation des Festivals sind nicht unbedingt notwendig, wie z. B. die Farbe eines Tickets oder die exakte Schreibweise auf den Tickets. Diese Details können später ausgearbeitet werden.

Durch Abstraktion können die unwesentlichen Dinge vorerst beiseitegelassen werden.

Problemzerlegung und algorithmisches Design

Die Aufgaben bei der Organisation eines Musikfestivals können in kleinere Unteraufgaben unterteilt werden. In der Praxis könnten diese zur Lösung an einzelne Personen delegiert werden:

- Jemand, der zum Veranstaltungsort des Festivals Kontakt aufnimmt
- Jemand, der sich um die Öffentlichkeitsarbeit kümmert

- Jemand, der sich um die Buchungen und Tickets kümmert

Diese kleineren Aufgaben können dann zur Lösung in noch kleinere Probleme aufgeteilt werden. Dies ist ein Beispiel für eine **Zerlegung** eines Problems und ein Beispiel für die Anwendung der Methoden des Computational Thinking.

Die Organisation des Veranstaltungsortes kann zum Beispiel noch weiter unterteilt werden:

- Wo wird der Ort sein?
- Wann wird er zur Verfügung stehen?
- Welche Anreisemöglichkeiten stehen zur Verfügung?

Ebenso könnte das Problem der Anreisemöglichkeiten in noch kleinere Prozesse **zerlegt** werden. Es kann ein einfacher Algorithmus entwickelt werden, der einige der folgenden Punkte enthält:

- Verhandlungen mit der Bahngesellschaft über Sonderzüge und Ermäßigungen
- Zubringerbusse
- Parkmöglichkeiten für Besucher
- reservierte Parkplätze für Musiker und Mitarbeiter

Das Problem der Öffentlichkeitsarbeit könnte wie folgt aufgeteilt werden:

- Wie kann man das Festival so interessant machen, dass es ein großes Publikum anspricht?
- Wie kann man das Zielpublikum kostengünstig bewerben?

Auch der Buchungs- und Ticketprozess könnte weiter **zerlegt** werden.

Zum Beispiel kann der Buchungsprozess in eine Reihe von Schritten zerlegt werden und ein **Algorithmus** geschaffen werden, der die Bereiche Design, Druck, Kauf, Lieferung, Ermäßigungen und Rückvergütungen beinhaltet.

Zerlegung eines Problems bei der Organisation eines Musikfestivals

- Wo?
- Wann ist der Ort verfügbar?
- Wie sind die Anreisemöglichkeiten?
- Wie wird das richtige Publikum erreicht?
- Wie werden die Kosten niedrig gehalten?
- Wie werden sie ausgestellt?
- Wie sieht es mit Ermäßigungen aus?
- Wie werden die Tickets kontrolliert?

Evaluierung

Nach der Veranstaltung hilft es Feedback einzuholen, um herauszufinden, was gut und was nicht so gut funktioniert hat. Auf diese Weise kann der Plan evaluiert werden. Wenn das Musikfestival wieder durchzuführen ist, können die gewonnenen Erkenntnisse zur Verbesserung der nächsten Veranstaltung beitragen.

Generalisierung (Verallgemeinerung)

Welche gewonnenen Erfahrungen lassen sich bezogen auf andere Veranstaltungen verallgemeinern? Welche Gemeinsamkeiten sind zu erwarten, wenn ein Marathon, ein Mittelalterfest, ein Theaterfestival oder eine Literaturlesung organisiert werden soll?

1.2 Computern Anweisungen geben

Begriffe

Computational Thinking ist ein allgemeiner Problemlösungsansatz. Es kann aber auch als Ausgangspunkt genützt werden, um Anweisungen für Computer zu erstellen.

Die Zerteilung von Problemen in leichtere Schritte führt zur Entwicklung eines oder mehrerer Algorithmen – einer Anzahl gut definierter, einfacher Schritte, denen man folgt, um ein Problem zu lösen. Algorithmen können dann so gestaltet werden, dass Computer sie verstehen können.

Schritte

Beispiel: Ein Algorithmus, um Menschen der Größe nach zu sortieren

In diesem Beispiel möchten Sie die Personen einer Klasse der Größe nach sortieren. Dazu könnten Sie die folgenden Schritte durchführen:

- Schritt 1: Lassen Sie alle Personen in einer Reihe aufstellen.
- Schritt 2: Entscheiden Sie, welches Ende der Reihe „klein“ und welches Ende „groß“ ist.
- Schritt 3: Vergleichen Sie wiederholt die Größe und tausche Personen, die auf der falschen Stelle stehen.

Diese Schrittfolge ist ein Algorithmus. Er ist jedoch nicht sehr detailliert. Schritt 3 kann zum Beispiel noch weiter in kleinere Probleme zerlegt werden:

- Bei welchem Ende der Reihe soll begonnen werden?
- Wie ist die Größe der Personen zu vergleichen?
- Was ist zu tun, wenn die Personen größer oder kleiner sind?

Wenn dieser Algorithmus zur Sortierung von Personen der Größe nach fertig ausgearbeitet ist, kann er modifiziert werden, um etwas andere Probleme zu lösen. Eine kleine Variation des Algorithmus könnte es sein, Personen nach ihrem Geburtsdatum zu sortieren. Eine weitere Modifikation könnte sein, herauszufinden, ob zwei Personen am selben Tag Geburtstag haben.

Begriffe

Algorithmen für Computer

Die Anweisungen im Algorithmus zur Sortierung der Personen nach der Größe sind für Menschen detailliert genug, nicht aber für einen Computer. Ein Computer muss seine Anweisungen in einer Computersprache erhalten. Im Gegensatz zu einer natürlichen Sprache ist die Bedeutung von Anweisungen einer Computersprache stets exakt definiert.

Damit also ein Computer den Instruktionen eines Algorithmus folgen kann, muss der Algorithmus in eine Sprache umgewandelt werden, die ein Computer ausführen kann. Anweisungen in dieser vom Computer ausführbaren Sprache werden als **Programm** bezeichnet.

Wenn ein Computer den Schritten eines Programms gehorcht, sagt man, „er führt das Programm aus“. Ein Computerprogramm wird also in einer Computersprache geschrieben und der Computer arbeitet es ab, indem er das Programm ausführt.

Im Gegensatz zu Menschen, bei denen man ein gewisses Maß an Vorwissen und gesunden Menschenverstand voraussetzen darf, besitzt ein Computer nur die Fähigkeiten, mit denen er vorher ausgestattet wurde. Fehlt ihm diese Befähigung auf einem gewissen Gebiet, dann sind selbst einfachste Aufgaben oder naheliegende Schlussfolgerungen für den Computer unlösbar und müssen erst aufwändig nachgerüstet werden. Durch moderne Methoden der künstlichen Intelligenz wird diese früher sehr strikte Grenze zwischen Mensch und Computer aber zusehends unschärfer.

1.3 Übungen zur Wiederholung

Erklärung der Symbole für den Schwierigkeitsgrad der Aufgaben:

☆★★: leicht
☆☆★: mittel
☆☆☆: schwer

1. **Computing umfasst viele Aktivitäten, dazu gehören:** ☆★★
   a) Ausführen von Berechnungen oder Verarbeitung von Daten.
   b) Folgen von Rezeptschritten für ein Biskuit.
   c) genaue, sorgfältige Sternenbeobachtung über einen längeren Zeitraum.
   d) genaue, sorgfältige Betrachtung chemischer Reaktionen.

2. **Computational Thinking bezeichnet Folgendes:** ☆★★
   a) Computer lösen ein schwieriges Problem.
   b) Der Prozess der Analyse von Problemen und Aufgaben und die Identifizierung von möglichen Lösungen.
   c) Einsatz von Computern, um viele Mathematikaufgaben zu lösen.
   d) Eine Aktivität, bei der ein Computer und eine Person zusammenarbeiten, um gemeinsam mehr zu erreichen.

3. **Welche Methode ist kein typischer Bestandteil des Computational Thinking?** ☆★★
   a) Abstraktion
   b) Auffassung
   c) Problemzerlegung
   d) Mustererkennung

4. **Welche der folgenden Antworten ist ein gutes Beispiel für die Zerlegung eines Problems?** ☆★★
   a) Zerlegen der Schritte, um einen Roboter zu entwickeln: Design der Hand, Wahl der Stromquelle, Auswahl und Position der Sensoren.
   b) Zerteilung einer Torte in 6 gleich große Stücke.
   c) Einen Apfel in ein Glasgefäß geben und täglich fotografieren, um zu beobachten, was passiert.
   d) Eine Suchmaschine verwenden, um Antworten auf eine Frage zu finden.

5. **Ein Computerprogramm ist:** ☆★★

   a) das detaillierte Regelwerk eines Spieles oder einer Sportart.

   b) ein Algorithmus, der in einer für den Computer geeigneten Form ausgedrückt ist.

   c) ein Gesetzeswerk, das Bauvorschriften beinhaltet.

   d) eine Reihe von Anweisungen für eine Person, zum Beispiel für ein Kuchenrezept.

6. **Auf dem Tisch steht eine Schale mit bunten Kugeln. Welche der folgenden Anweisungen erfüllt die Anforderung an einen Algorithmus nicht optimal?** ☆☆★

   a) Beginnen Sie, einzeln Kugeln aus der Schale zu nehmen. Legen Sie diese alternierend, links beginnend, einmal links und einmal rechts neben die Schale. Fahren Sie fort, bis die Schale leer ist.

   b) Nehmen Sie, ohne zu schauen, eine zufällige Kugel aus der Schale. Ist sie rot, dann legen Sie sie auf den Tisch, ansonsten legen Sie sie in die Schale zurück. Wiederholen Sie den Vorgang zehnmal. Mischen Sie die Kugeln in der Schale nach jedem Durchgang.

   c) Entfernen Sie alle roten Kugeln aus der Schale. Legen Sie die roten Kugeln in einer Reihe auf und legen Sie die erste, die dritte, die fünfte – und immer so weiter – wieder in die Schale zurück.

   d) Nehmen Sie ziemlich viele Kugeln aus der Schale, bis Sie glauben, es ist genug.

7. **Hier sind drei Rezepte:** ☆★★

Schokoladekuchen:

Backrohr auf 180°C vorheizen

Zwei Tortenformen mit ca. 20 cm Durchmesser einfetten

Zutaten eine Minute lang mixen

Teig in die vorbereiteten Tortenformen füllen

30 Min. lang backen

Abkühlen lassen

Kuchen glasieren und dekorieren

Lebkuchenmann:

Zutaten zusammenmischen

Backrohr auf 190°C vorheizen

Teig ca. einen halben cm dick ausrollen, Formen ausstechen

Backen, bis der Rand fest ist, ca. 10 Minuten

Heidelbeer-Muffins:

Backrohr auf 185°C vorheizen

18 Muffin-Formen einfetten

Butter und Zucker schaumig rühren

Übrige Zutaten hinzufügen

Masse in Muffin-Formen füllen

Topping darüber streuen

15 bis 20 Min. backen

Welches der folgenden Muster ist in allen drei Rezepten gleich?

a) 18 Muffin-Formen einfetten.

b) Glasieren und dekorieren.

c) Teig ausrollen.

d) Backrohr auf eine bestimmte Temperatur vorheizen.

LEKTION 2 – SOFTWARE-ENTWICKLUNG

Wenn Sie mit dieser Lektion fertig sind, können Sie:

- den Unterschied zwischen formalen und natürlichen Sprachen erklären,
- den Begriff Code erklären; zwischen Quellcode und Maschinencode unterscheiden,
- Programmbeschreibung und Programmspezifikation erläutern,
- erforderliche Schritte bei der Erstellung eines Programms erklären: Analyse, Entwurf, Programmierung, Testen, Erweiterung.

2.1 Präzision der Sprache

Begriffe

Arten von Sprachen:

Natürliche Sprache

Gesprochene Sprachen wie Deutsch, Englisch oder Chinesisch sind natürliche Sprachen. Natürliche Sprachen dienen der Kommunikation. Sie sind enorm vielseitig, aber eine typische Eigenschaft von natürlichen Sprachen ist ihre mögliche Mehrdeutigkeit. Dies ist meist kein Problem, denn die meisten Mehrdeutigkeiten können aus dem Zusammenhang aufgeklärt werden. Jedenfalls muss man feststellen, dass natürliche Sprachen in ihrem täglichen Gebrauch nicht ganz präzise sind. Aus dem ganz einfachen Grund, weil das auch meist gar nicht notwendig ist. Jedenfalls macht gerade die Mehrdeutigkeit auch einen besonderen Reiz der natürlichen Sprachen aus.

Beispiele für Mehrdeutigkeiten:

- Wir empfehlen Ihnen die frische Buttermilch mit Erdbeeren von unseren Kühen.
- Die Dame suchte den Mann mit dem Fernrohr.
- Dieser Bereich wird zur Verhütung von Straftaten durch die Polizei videoüberwacht.

Formale Sprache

Formale Sprachen werden u. a. in der Mathematik, in chemischen Gleichungen und in der Programmierung verwendet. Sie sind in ihrer Form streng definiert und die Bedeutung ihrer Aussagen ist meist eindeutig festgelegt.

Programmiersprachen sind ein Sonderfall innerhalb der formalen Sprachen, denn sie dienen vorrangig nicht der Kommunikation von Mensch zu Mensch, sondern von Mensch zu Maschine. Während die natürliche Sprache bis zu einem gewissen Grad fehlertolerant ist, „verzeiht" der Computer keine Fehler. Mehrdeutigkeiten wie in den Beispielen oben darf es nicht geben. Die für das Schreiben von Programmen benötigte Exaktheit ist für Anfänger oft ungewohnt, kann aber gut erlernt werden.

Einige Dinge, mit denen ein beginnender Programmierer konfrontiert ist, können erhellend sein. Manches ist zuerst aber auch verwirrend und der eigenen Intuition widersprechend.

Beispiel: Gesetz von De Morgan

Jemand, der gerne schwarzen, ungezuckerten Kaffee trinkt, möchte, dass sein Kaffee keine Milch **und** keinen Zucker enthält. Er will nicht, dass Milch **oder** Zucker in seinem Kaffee sind.

Beispiel: Exklusives Oder

Der Satz „Um die Prüfung zu schaffen, musst du entweder viel mehr lernen oder jemanden bitten, dass er dir die Kapitel erklärt, die du nicht verstehst.“ würde streng logisch bedeuten, dass man die Prüfung nicht schafft, wenn man viel mehr lernt **und** jemanden um Hilfe bittet.

2.2 Programmiersprachen

Begriffe

Programmiersprachen dienen dazu, Computerprogramme zu schreiben. Indem Computer diese Programme ausführen, können sie bestimmte Aufgaben erfüllen.

Die Grundlagen einer Computersprache und die wesentlichen Teile der Syntax sind meist schnell erlernt. Sich aber mit einer Programmiersprache ganz genau auszukennen, erfordert zum Beispiel auch die Kenntnis der üblicherweise verwendeten Erweiterungen (Bibliotheken, Klassenbibliotheken, Plugins, GUI-Frameworks).

Es gibt viele verschiedene Computersprachen. In dieser Lernunterlage wird die beliebte Computersprache Python verwendet. Beispiele für andere übliche Computersprachen sind JavaScript, Java, C++, C#, PHP.

Menschen, die Programme erstellen, nennt man Programmiererinnen, Programmierer, Software-Entwicklerinnen oder Software-Entwickler.

Begriffe

Programmcode

Der Text eines Computerprogramms heißt Programmcode (manchmal auch kurz: Code).

Verschiedene Programmiersprachen unterscheiden sich in den verwendeten Grundbausteinen und in der Syntax. Es gibt auch verschiedene sogenannte Programmiersprachenparadigmen wie zum Beispiel die funktionale oder die objektorientierte Programmierung. Eine Programmiersprache kann aber durchaus mehrere dieser Paradigmen realisieren. In Python kann im Sinne des funktionalen, des prozeduralen, des ereignisorientierten oder des objektorientierten Paradigmas entwickelt werden.

Quellcode

Ein von Entwicklern erstellter Programmcode wird auch Quellcode (englisch: Source Code) genannt. Nachdem du eine Programmiersprache erlernt hast, kannst auch du einen Quellcode schreiben wie zum Beispiel in Python. Beim Quellcode handelt es sich um einen lesbaren Text.

Maschinencode

Wurde, so wie oben beschrieben, ein Quellcode erstellt, dann gibt es zwei Möglichkeiten: Manche Programmiersprachen (Python ist ein Vertreter dieser Gruppe) werden interpretiert. Der Interpreter führt dann den Programmcode direkt aus. Im zweiten Fall wird der Quellcode kompiliert und aus dem Quellcode wird ein „Maschinencode" hergestellt. Dieser Maschinencode (man nennt ihn in diesem Zusammenhang auch „nativen Code") kann vom Prozessor direkt ausgeführt werden. Dateien mit Maschinencode sind keine Textdateien, sondern sogenannte „Binärdateien". Es ist nicht sinnvoll, diese mit einem Texteditor zu öffnen, denn es ist kein sinnvoller Inhalt erkennbar. Ein Maschinencode läuft meist schneller als ein interpretierter Code. Einer der Nachteile ist aber, dass ein Maschinencode für jede Zielplattform neu generiert werden muss. Ein für ein bestimmtes Betriebssystem oder für eine bestimmte Computerhardware erstellter Maschinencode läuft nur dort.

Das folgende Quellcodeprogramm

```
#include <stdio.h>
int main(){
 int this_number;
 this_number = 5;
 while(this_number > 0){
   this_number--;
   printf(".");
 }
}
```

wird zum Beispiel unter Windows in folgendes Maschinencodeprogramm umgewandelt (kurzer Auszug):

```
Offset(o)  00 01 02 03 04 05 06 07 10 11 12 13 14 15 16 17
00000000   4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00  MZ..........ÿÿ..
00000020   B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ¸.......@.......
00000040   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000060   00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 00  ............€...
00000100   0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ..°..´.Í!¸.LÍ!Th
00000120   69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F  is program canno
00000140   74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  t be run in DOS
00000160   6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00  mode....$.......
00000200   50 45 00 00 4C 01 0D 00 BC A0 6F 5A 00 60 00 00  PE..L...¼ oZ.`..
00000220   0C 04 00 00 E0 00 07 01 0B 01 02 1C 00 18 00 00  ....à...........
00000240   00 32 00 00 00 04 00 00 E0 14 00 00 00 10 00 00  .2......à.......
00000260   00 30 00 00 00 00 40 00 00 10 00 00 00 02 00 00  .0....@.........
00000300   04 00 00 00 01 00 00 00 04 00 00 00 00 00 00 00  ................
00000320   00 00 01 00 00 04 00 00 78 D9 00 00 03 00 00 00  ........xÙ......
00000340   00 00 20 00 00 10 00 00 00 00 10 00 00 10 00 00  .. .............
00000360   00 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00  ................
```

Der Quellcode ist für Menschen gut lesbar, der Maschinencode nicht. Wird das gleiche Programm für eine andere Rechnerarchitektur kompiliert, entsteht ein anderes Maschinenprogramm.

2.3 Kommentare im Programmcode

Begriffe

In Python gibt es, so wie in fast allen anderen Programmsprachen, die Möglichkeit Texte in das Programm zu schreiben, die bei der Ausführung des Programms ignoriert werden. Diese Texte werden „Kommentare“ genannt.

Oft haben Kommentare erklärende Bedeutung für das Programm, in dem sie sich befinden. Das muss aber keineswegs immer der Fall sein.

Gute Entwickler kommentieren ihre Programme. Gute Kommentare helfen anderen Menschen, den Programmcode schneller zu verstehen. Gute Kommentare sind aber auch als Stütze für den Entwickler selbst gedacht, um sich besser in ein Programm wieder einarbeiten zu können, das man länger nicht bearbeitet hat. Das könnte zum Beispiel bei einer Fehlerbehebung, einer Wartung oder einer Erweiterung relevant werden.

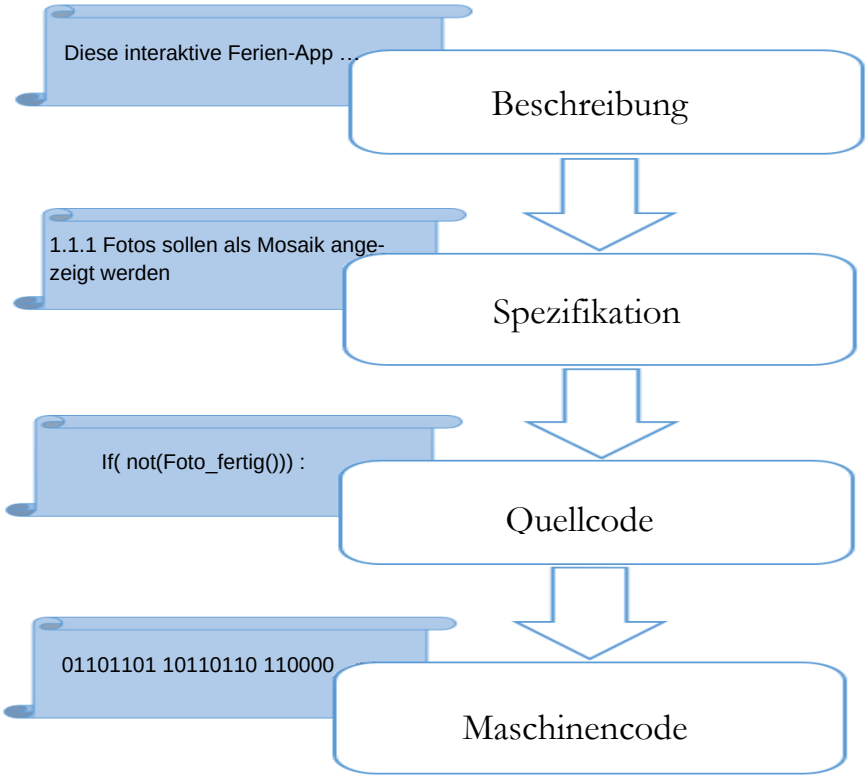
Dokumente in einem Softwareprojekt

Die folgenden Dokumente dienen der Vorbereitung und der Umsetzung eines Softwareprojekts. Außerdem werden diese auch in der Evaluierungsphase verwendet, um zu überprüfen, ob das Programm wunschgemäß umgesetzt wurde.

In einer **Programmbeschreibung** wird erläutert, was der Zweck eines Programms ist und wie es funktioniert. Diese Beschreibung kann von Mitarbeitern im Projekt verwendet werden, aber auch von Anwendern und von anderen Abteilungen wie zum Beispiel der Marketingabteilung, die diese Information für den Vertrieb nutzen kann.

Eine **Programmspezifikation** besteht aus den Anforderungen, die darlegen, was das Programm tun wird. Üblicherweise wird die Spezifikation vor dem Programm geschrieben. Eine Spezifikation ist detaillierter als eine Beschreibung und sie enthält Anforderungen, die überprüft werden können.

Eine Spezifikation kann zum Beispiel die folgende Anforderung enthalten: „Das Programm soll, um Strom zu sparen, den Bildschirm des Computers dimmen, wenn er länger als eine Minute lang nicht genutzt wird.“ Wenn das Programm fertiggestellt ist, kann man das überprüfen, indem man den Computer eine Minute lang laufen lässt und kontrolliert, ob das Programm tatsächlich den Bildschirm dimmt. Die folgende Zeichnung zeigt die Schritte bei der Erstellung einer Anwendung.



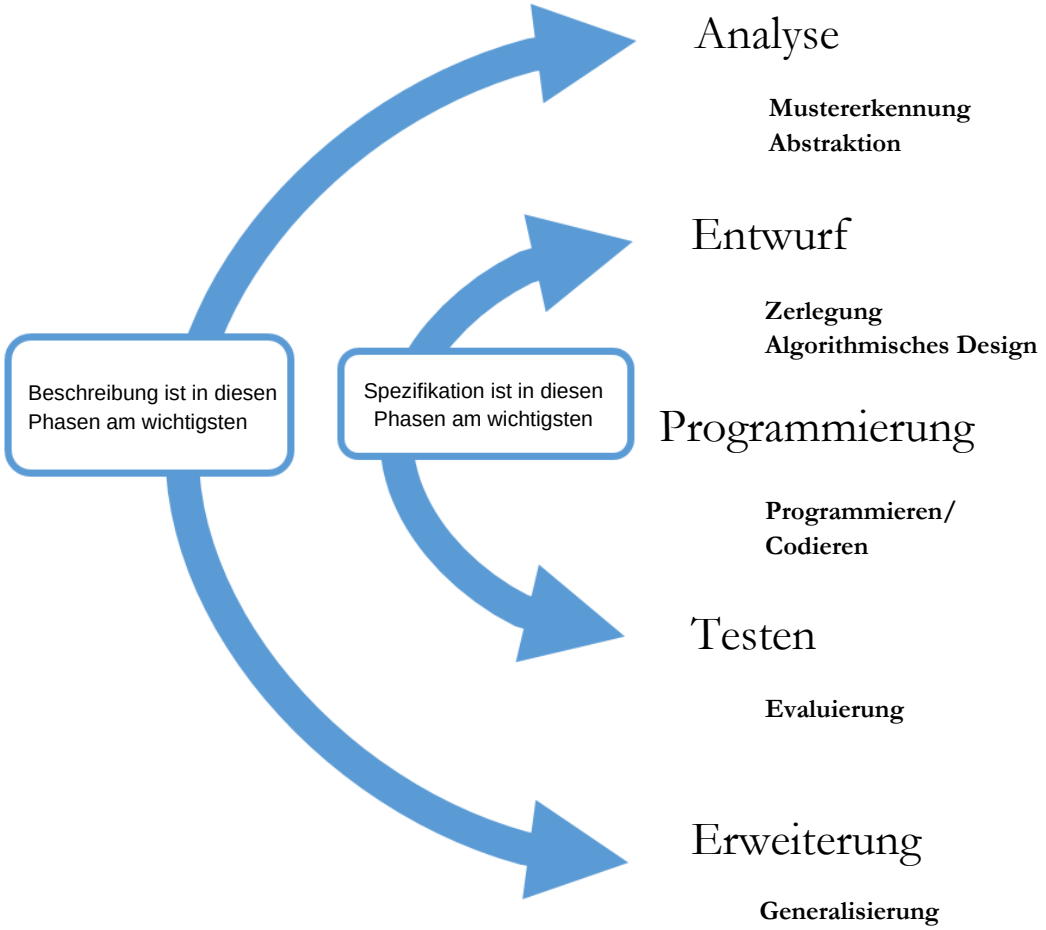
Schritte zur Erstellung einer Anwendung

2.4 Phasen der Programmentwicklung

Begriffe

Beschreibungen und Spezifikationen spielen eine wichtige Rolle bei der Programmentwicklung.

Das folgende Diagramm zeigt einige der wichtigsten Phasen und Tätigkeiten bei der Erstellung eines Programms.



Phasen der Programmentwicklung

Analyse

Dabei werden die zu lösenden Probleme präzisiert. Die Analyse ist in erster Linie ein Abstraktionsprozess, in dem alle Aspekte des Problems aufgelistet werden und die relevanten Aspekte und ihre Zusammenhänge identifiziert werden.

Entwurf

Dabei werden die Algorithmen (Folgen von Anweisungen) ausgearbeitet. Mithilfe der Zerlegung (Dekomposition) wird das Problem in kleinere Teile heruntergebrochen und dann der Algorithmus entworfen.

Programmierung

Der Algorithmus wird in einer ausgewählten Programmiersprache für den Computer verständlich formuliert.

Testen

Dabei wird überprüft, ob das Programm tatsächlich so arbeitet, wie es soll. Testen dient dazu, Fehler in der Logik eines Programms zu identifizieren, zu lokalisieren und zu beheben.

Erweiterung

Dabei werden dem Programm neue Funktionalitäten hinzugefügt, um die Leistungen des Programms zu verbessern oder um es zu generalisieren, damit es für unterschiedliche Situationen nutzbar ist.

Eine allgemeine **Programmbeschreibung** wird üblicherweise ganz am Anfang des Projektes abgefasst, und zwar als Teil der Entscheidung über den gewünschten Leistungsumfang des Programms.

Die **Programmspezifikation** wird üblicherweise in der Entwurfsphase geschrieben. Einige der Anforderungen in den Spezifikationen treten unter Umständen unmittelbar infolge der Zerlegung des Problems in kleinere Probleme auf. Die Programmspezifikation wird während der Testphase überprüft, da jede Aussage bezüglich der Programmleistung verifiziert werden muss.

Bei der Planung von Verbesserungen kann die Programmbeschreibung aktualisiert werden, um Pläne zur Ausweitung der Programmleistungen festzuhalten.

2.5 Übungen zur Wiederholung

1. **Eine formale Sprache ist:** ☆★★
   a) eine Sprache, in der man keinen Fehler machen kann.
   b) eine Sprache mit definierten Regeln und weitgehend ohne Mehrdeutigkeiten.
   c) die für Postkarten am besten geeignete Sprache.
   d) jede Sprache, welche die Worte „und“ bzw. „oder“ beinhaltet.

2. **Deutsch, Englisch, Arabisch und Chinesisch sind:** ☆★★
   a) formale Sprachen.
   b) Computersprachen.
   c) natürliche Sprachen.
   d) Quellcodes.

3. **Maschinencode** ☆★★
   a) wird in Quellcode übersetzt, damit ein Computer den Anweisungen folgen kann.
   b) ist eine Methode der sicheren Kommunikation zwischen zwei Computern.
   c) enthält Anweisungen, die vom Computer direkt ausgeführt werden können.
   d) ist ein Vertrag zwischen einer Programmiererin und einem Kunden, der den Leistungsumfang eines Programms spezifiziert.

4. **Ein in der Computersprache Python geschriebener Algorithmus ist ein Beispiel für** ☆★★
   a) Quellcode.
   b) Maschinencode.
   c) Programmspezifikation.
   d) Programmbeschreibung.

5. **Eine Programmspezifikation ist:** ☆★★
   a) der vom Computer ausgeführte Code.
   b) der Kommentar im Code.
   c) das tatsächliche elektronische Signal in einem Computer, wenn ein Programm läuft.
   d) die genaue Beschreibung dessen, was ein Programm tun soll.

6. **Schreiben Sie zu jeder Aktivität bei der Programmerstellung den richtigen Zweck (a bis e) hinzu:** ☆☆★

a)Testen b) Entwurf c) Programmierung d) Analyse e) Erweiterung

Ein bestehendes Programm verbessern	
Das Problem klar definieren	
Überprüfen, ob das Programm korrekt läuft	
Den Algorithmus in der gewählten Computersprache ausdrücken	
Einen algorithmischen Ansatz zur Problemlösung ausarbeiten	

LEKTION 3 – ALGORITHMEN

In dieser Lektion werden Sie:

- über die Bedeutung von Anweisungsfolgen (Sequenzen) in Algorithmen erfahren,
- Flussdiagramm und Pseudocode kennenlernen, die einem Programmieranfänger helfen können, den Programmcode besser zu verstehen,
- wichtige Elemente eines Flussdiagramms finden wie: Start/Stop, Prozess, Entscheidung, Ein-/Ausgabe, Verbinder, Pfeil,
- sehen, wie man einen Algorithmus mit einem Flussdiagramm oder mit Pseudocode beschreiben kann,
- Fehler in einem Algorithmus beseitigen wie: fehlendes Element, falsche Reihenfolge, falsches Entscheidungskriterium bei der Verzweigung.

3.1 Grundelemente eines Algorithmus

Begriffe

Anweisungsfolge (Sequenz)

Eine Anweisungsfolge ist eine Anzahl von Anweisungen, die hintereinander in der gegebenen Reihenfolge auszuführen sind. Grundsätzlich ist die Reihenfolge der Anweisungen von Bedeutung: Man muss zum Beispiel den Kuchenteig zuerst in die Kuchenform geben, bevor man ihn in das Backrohr stellt. Es gibt aber auch Fälle, wo die genaue Reihenfolge von Schritten keine Rolle spielt. Anweisungsfolgen sind ein grundlegender Baustein von Computerprogrammen. Das Erstellen von Anweisungsfolgen ist eine wichtige Tätigkeit beim Programmieren. Programmiererinnen und Programmierer müssen sicherstellen, dass alle erforderlichen Aktionen in der richtigen Reihenfolge ausgeführt werden, um eine vorgegebene Aufgabe zu erfüllen.

Eingabe und Ausgabe (Input und Output)

Anweisungen verwenden oft Informationen, die ihnen von außen zugeführt werden. Computerprogramme können Informationen, mit denen sie arbeiten sollen, über **Eingaben** (Inputs) erhalten und Resultate über **Ausgaben** (Outputs) liefern.

Eingabe (Input)

Eine Eingabe ist ein Wert, der dem Computer von außen zugeführt wird. Der übergebene Wert wird dann innerhalb des Programms verwendet und dabei unter Umständen auch weiterverarbeitet. Beispiele sind ein Temperaturmesswert oder eine Eingabe über die Tastatur.

Ausgabe (Output)

Ein von einem Computerprogramm berechneter Wert oder eine ausgeführte Aktion, die nach außen angezeigt wird. Beispiele sind das Ein- bzw. Ausschalten von Licht oder die Anzeige einer Nachricht auf einem Bildschirm.

Schritte

Beispiel: Eingaben und Ausgaben bei einem Smartphone

Ein Smartphone nimmt Eingaben (oder Inputs) auf, verarbeitet sie und liefert Ergebnisse (oder Outputs).

- Der Touchscreen ist zum Beispiel ein wichtiges Mittel zur Eingabe, wenn er genützt wird, um eine Nachricht einzutippen oder eine App zu öffnen. Weitere Eingabemittel sind alle Tasten, das Mikrophon, die Kamera, alle Sensoren des Smartphones und die Schnittstellen (WLAN, Bluetooth …) nach außen.
- Die Bildschirmanzeige ist ein wichtiges Ausgabemittel, wenn sie zum Beispiel die Liste der Kontakte anzeigt. Weitere Ausgaben sind der Lautsprecher, Status-LED-

Anzeigen, der Vibrationsmotor und die Schnittstellen (WLAN, Bluetooth …) nach außen.

Verzweigungen

In den vorherigen Lektionen wurde erklärt, wie wichtig die Anweisungsfolgen für Programme sind. Anweisungsfolgen werden, wie oben erwähnt, hintereinander durchlaufen, aber an manchen Stellen im Programm sind Knotenpunkte, wo abhängig von Eingabe, Zuständen oder Messwerten in verschiedene Richtungen verzweigt werden kann.

Eine Verzweigung ist eine spezielle Anweisung, die einen Eingabe-, Zustands- oder Messwert abfragen kann und abhängig vom Ergebnis in verschiedenen Richtungen verzweigt.

Schritte

Beispiel: Warten, bis das Backrohr aufgeheizt ist

Bevor der Kuchenteig ins Backrohr gestellt wird, muss das Backrohr die richtige Temperatur erreicht haben. Ein Algorithmus könnte dazu eine Entscheidung beinhalten. „Hat das Backrohr die richtige Temperatur?“ Ein Temperaturfühler liefert den Input – *Ja* oder *Nein*. Wenn die Antwort „*Ja*“ ist, dann kann der Kuchenteig ins Backrohr gestellt werden. Wenn die Antwort „*Nein*“ ist, wartet der Algorithmus eine festgelegte Zeit und stellt die Frage später erneut.

3.2 Flussdiagramme und Pseudocode

Begriffe

Pseudocode und Flussdiagramme sind Methoden, die Programmieranfängern bestimmte Aspekte der Programmierung besser veranschaulichen können. Beide Techniken können einen Algorithmus recht gut beschreiben, einige kleine Details können unter Umständen aber noch fehlen. Deshalb eignen sich diese Formalismen nicht zur Ausführung durch den Computer.

Pseudocode

Pseudocode ist eine informelle Art, einen Algorithmus zu formulieren, die in natürlicher Sprache (Deutsch, Englisch …) erfolgt und sich an eine tatsächliche oder idealisierte Programmiersprache anlehnt, aber nicht so strengen Regeln unterliegt wie ein Programm. Der Algorithmus muss nicht ins letzte Detail ausformuliert sein. Manche Teile können noch fehlen. Ein in Pseudocode geschriebener Algorithmus ist für Menschen gedacht und nicht für Maschinen, er beinhaltet aber dennoch Programmcode-Elemente. Ein in Pseudocode formulierter Algorithmus kann als Grundlage für die spätere Programmierung dienen.

Ein Algorithmus fürs Zähneputzen könnte in Pseudocode so aussehen:

START

Wenn Zahnpasta in Tube?

Putzen!

Sonst: Wenn Frische Tube vorhanden?

Neu Tube öffnen!

Putzen!

Sonst: Wenn Geschäfte geöffnet?

Schnell eine Tube kaufen!

Putzen!

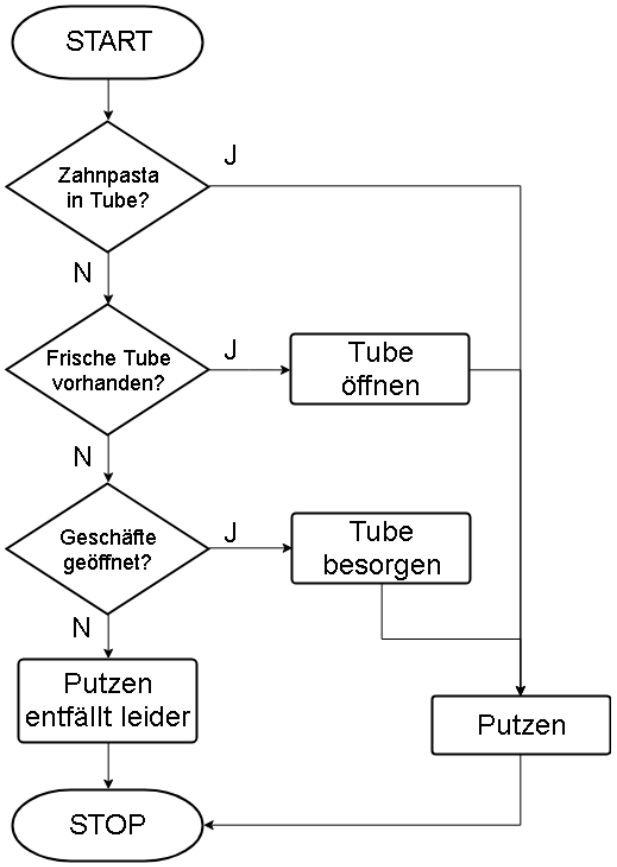
Sonst: Zähneputzen muss heute leider ausfallen.

STOP

Flussdiagramm

Ein Flussdiagramm stellt einen Algorithmus bildlich dar. Flussdiagramme bestehen aus Formen, die durch gerichtete Pfeile miteinander verbunden sind. Die Formen repräsentieren Zustände, die Pfeile zeigen an, in welcher Reihe diese durchlaufen werden. Flussdiagramme werden in der Softwareerstellung nur mehr selten verwendet, einem Programmieranfänger können sie aber helfen, den logischen Aufbau von Programmen zu visualisieren.

Der Algorithmus fürs Zähneputzen kann auch als Flussdiagramm veranschaulicht werden:

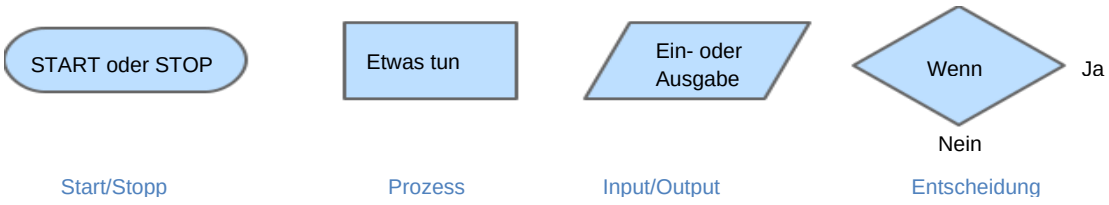


Flussdiagramm fürs Zähneputzen

3.3 Flussdiagramme

Begriffe

Hier sind einige der in einem Flussdiagramm verwendeten Formen:



Flussdiagrammsymbole

Die Formen werden folgendermaßen verwendet:

Start oder Stop

Der Algorithmus beginnt beim Start-Feld und läuft bis zum Stop-Feld.

Prozess

Diese Felder beinhalten einfache Anweisungen, die beim Erreichen der Form ausgeführt werden.

Ein- oder Ausgabe

Diese Felder dienen der Interaktion mit der Welt außerhalb des Computers. Eine Eingabe könnte zum Beispiel von einem Bewegungs- oder Temperatursensor kommen. Eine Ausgabe könnte eine Ausgabe auf einem Bildschirm, ein Lichtsignal oder ein Ton sein.

Entscheidung

Entscheidungsfelder stellen eine Wahlmöglichkeit für den nächsten Schritt eines Algorithmus dar. Sie beinhalten oft eine Frage, die eine Ja- oder Nein-Antwort hat. Wenn die Antwort *Ja* ist, wird eine bestimmte Richtung eingeschlagen, wenn die Antwort *Nein* ist, eine andere. So können Algorithmen über eine einfache Schrittreihenfolge hinausgehen.

Verbinden der Felder

Es gibt noch zwei weitere Flussdiagrammsymbole:

Pfeil Verbinder

Der Pfeil und der Verbinder werden verwendet, um Flussdiagrammfelder zu verbinden.

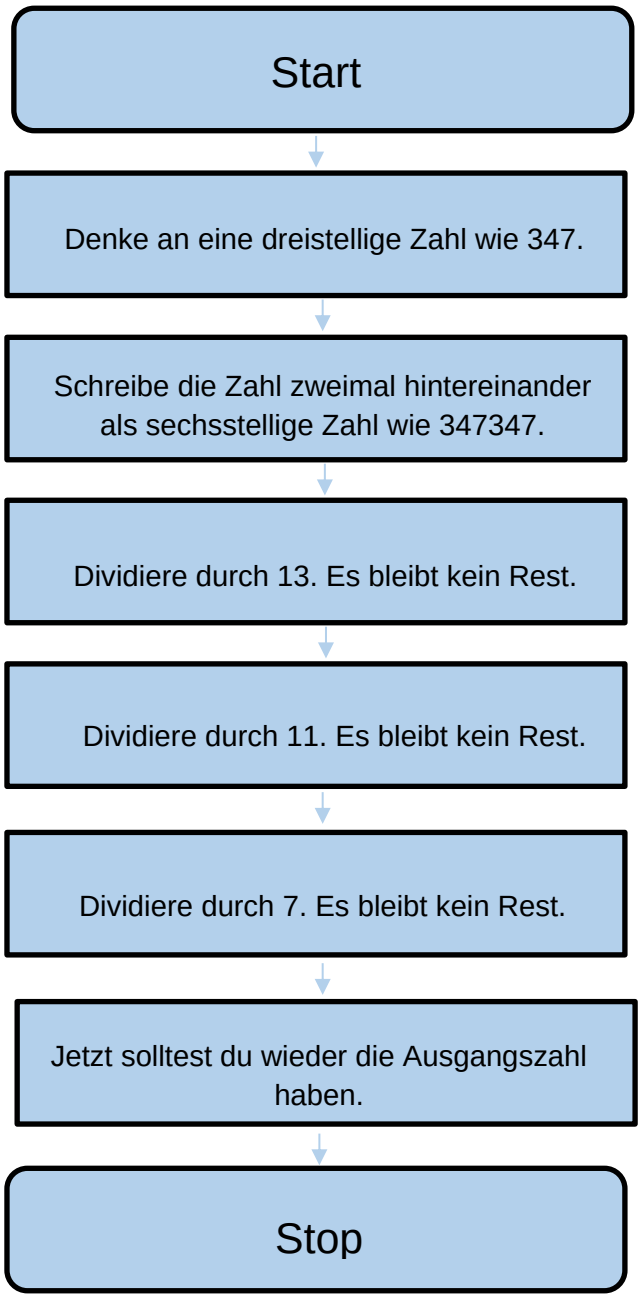
Pfeil

Ein Pfeil ist eine Richtungslinie zur Verbindung von zwei Feldern in einem Flussdiagramm. Der Pfeil zeigt die Flussrichtung in einem Algorithmus. Er wird auch oft als Flusslinien bezeichnet. So sind zum Beispiel die Anweisungen in einer Sequenz mit Pfeilen verbunden.

Verbinder

Ein Verbinder ist ein kleiner Kreis in einem Flussdiagramm, der verwendet wird, um zwei Flusslinien miteinander zu verbinden. Er zeigt einen Sprung zwischen einem Punkt im Prozessfluss zu einem anderen, zum Beispiel bei der Beantwortung einer Ja/Nein-Frage.

Schritte



Flussdiagramm für einen Zahlentrick

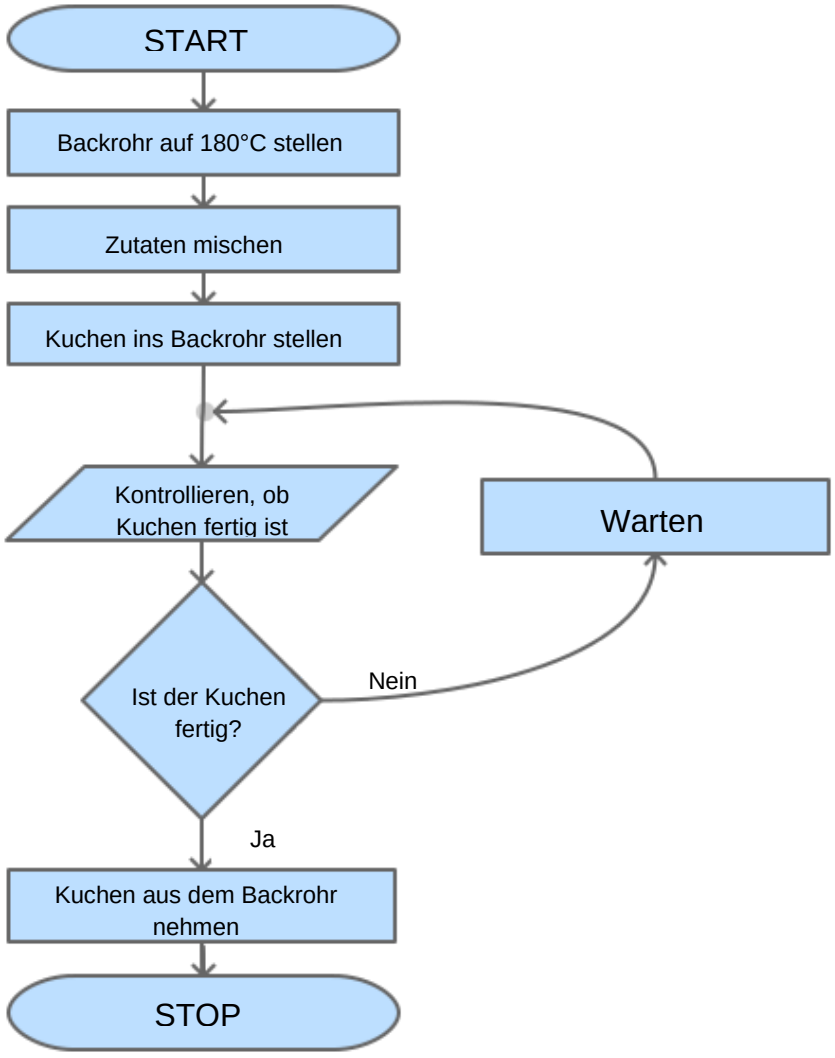
Beispiel: Flussdiagramm für den 347347-„Zahlentrick“

Das ist ein Flussdiagramm für einen Zahlentrick (siehe vorige Seite). Dieses Flussdiagramm zeigt eine einfache Sequenz ohne Entscheidungsfelder. Wir beginnen den Prozess beim Startfeld und folgen den Flusslinien, wobei wir die Anweisungen in jedem Feld befolgen, bis wir zum Stopfeld kommen.

Beispiel: Flussdiagramm für das Backen eines Kuchens

Das folgende Flussdiagramm stellt einen Algorithmus für das Backen eines Kuchens dar. Es beinhaltet das Entscheidungsfeld „Ist der Kuchen fertig“. Das Entscheidungsfeld legt fest, ob der Kuchen fertig ist, um ihn aus dem Backrohr zu nehmen, und – falls nicht – weist es den Koch an zu warten.

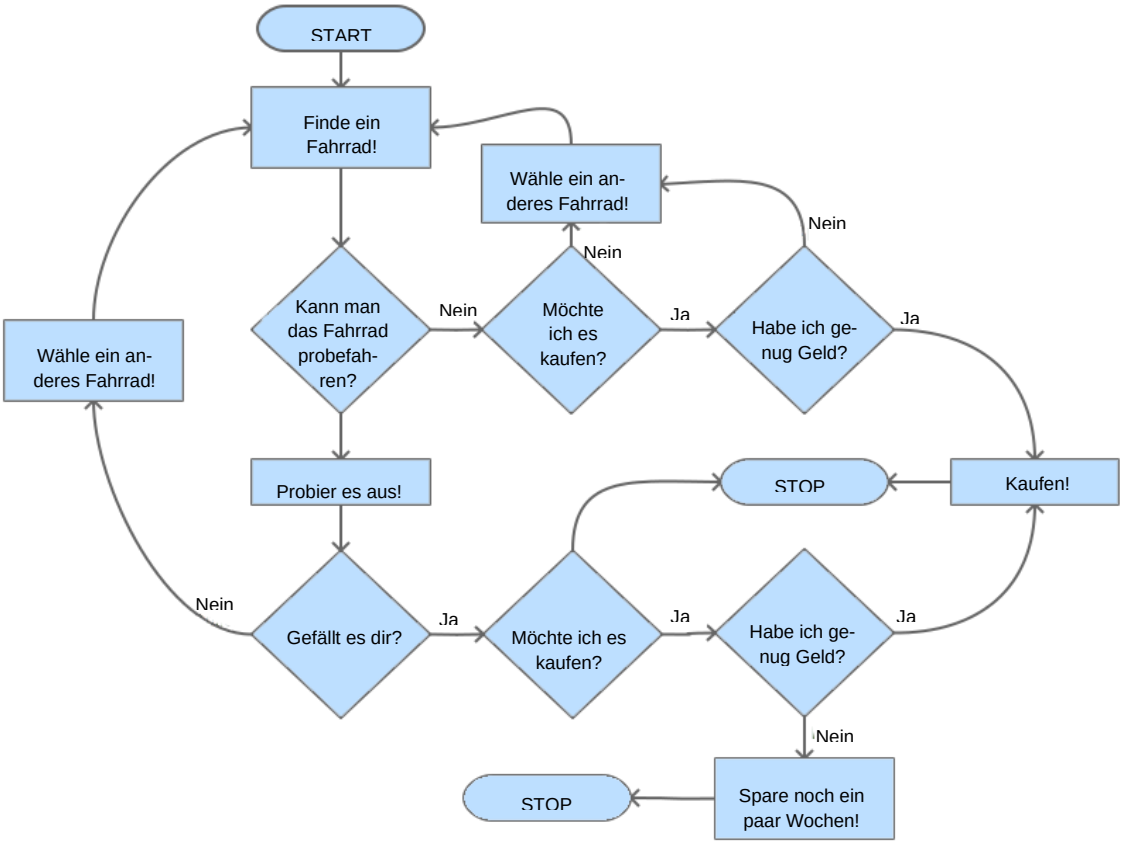
„Kontrollieren, ob Kuchen fertig ist“ ist ein Input für den Algorithmus und steht in einem Feld mit Input/Output-Form.



Flussdiagramm für das Backen eines Kuchens

Beispiel: Flussdiagramm, um ein Fahrrad zu kaufen

Das folgende Flussdiagramm zeigt die Aktionsfolge, um ein Fahrrad auszusuchen. Jedes rautenförmige Feld beinhaltet eine Frage. Die Antwort auf die Frage kann *Ja* oder *Nein* sein. Die Antwort bestimmt, zu welchem Feld man weitergehen muss.



Flussdiagramm, um ein Fahrrad zu kaufen

Übung: Flussdiagramm zum Aufstehen am Morgen

Erstellen Sie ein Flussdiagramm für das Aufstehen, Frühstücken und den Schulweg.

Hinweise:

1. Beginnen Sie mit einer Abfolge von Aktionen und schreiben Sie diese Schritte als Pseudocode in einer geordneten Liste auf.

2. Nehmen Sie einige dieser Schritte und zerlegen Sie sie in kleinere Detailschritte. Wenn Sie zum Beispiel einen Schritt „Frühstück machen“ habe, kann er in kleinere Schritte wie „Toast in Toaster geben“ zerlegt werden.

3. Stellen Sie diese detailliertere Sequenz als Flussdiagramm dar.

4. Machen Sie das Flussdiagramm interessanter, indem Sie Variationen einbauen. Zum Beispiel: Welchen Einfluss hat das Wetter auf den Schulweg? Diese Variationen können von verschiedenen Fragen abhängen wie:

- Regnet es?
- Bin ich schon spät dran?

5. Fügen Sie Entscheidungsfelder für diese Fragen ein und unterschiedliche Schritte – je nachdem, ob die Antwort *Ja* oder *Nein* ist.

3.4 Pseudocode

Begriffe

Statt in einem Flussdiagramm kann ein Algorithmus auch als Pseudocode dargestellt werden.

Pseudocode ist eine Möglichkeit, um informell zu beschreiben, wie ein Algorithmus funktioniert. Er verbindet informelle natürliche Sprache (zum Beispiel Deutsch) mit Strukturteilen einer Programmiersprache.

Schritte

Beispiel: Pseudocode für den 347347-„Zahlentrick"

Das Flussdiagramm für den Zahlentrick könnte wie folgt als Pseudocode geschrieben werden:

START
Denken Sie an eine dreistellige Zahl wie 347.
Schreiben Sie die Zahl zweimal hintereinander als sechsstellige Zahl, wie 347347.
Dividieren Sie durch 13.
Dividieren Sie durch 11.
Dividieren Sie durch 7.
Du solltest wieder die Ausgangszahl haben.
STOP

Beispiel: Pseudocode für das Backen eines Kuchens

START
Backrohr auf 180 0C stellen
Zutaten mischen
Kuchen ins Backrohr stellen
Kontrollieren, ob Kuchen fertig ist, wenn nicht
    Warten
Kuchen aus Backrohr nehmen
STOP

Beim Schreiben von Pseudocode werden die Struktur und einige Konventionen von Programmiersprachen verwendet. Das Wort „Warten“ ist **eingerückt**, was darauf hinweist, dass es nur dann geschieht, wenn der Kuchen nicht fertig ist.

3.5 Fehler in Algorithmen beseitigen

Begriffe

Beim Schreiben von Algorithmen können leicht Fehler passieren, wie das Auslassen von Schritten, die falsche Reihung von Schritten oder das Treffen inkorrekter Entscheidungen. Diese Fehler müssen korrigiert werden.

Hier sind einige der häufigsten Fehler angeführt und Wege, sie zu korrigieren.

1. Falsche Reihenfolge

Das Schreiben von Anweisungen in der falschen Reihenfolge wird als falsche Sequenz bezeichnet.

Der folgende Algorithmus fürs Zähneputzen ist als Pseudocode geschrieben. Die Anweisungen sind in einer falschen Reihenfolge. Der Algorithmus kann korrigiert werden, indem „Putzen“ und „Schnell eine Tube kaufen“ vertauscht werden.

```
START
Wenn Zahnpasta in Tube?
        Putzen
Sonst: Wenn frische Tube vorhanden?
                Neue Tube öffnen
                Putzen
        Sonst: Wenn Geschäfte geöffnet?
                                Putzen
                                Schnell eine Tube kaufen
                Sonst: Zähneputzen muss heute leider ausfallen.
STOP
```

2. Falsches Entscheidungskriterium bei der Verzweigung

Das ist ein Algorithmus, um einen Kuchen zu backen und ihn aus dem Backrohr zu nehmen:

```
START
Backrohr auf 180 °C stellen
Zutaten mischen
Kuchen ins Backrohr stellen
Wenn Kuchen fertig?
    Warten
Sonst:
    Kuchen herausnehmen
STOP
```

Bei der Entscheidung wurden die beiden Optionen vertauscht angegeben, was man so korrigierten kann:

START
Backrohr auf 180 °C stellen
Zutaten mischen
Kuchen ins Backrohr stellen
Wenn: Kuchen fertig?
    Kuchen herausnehmen
Sonst:
    Warten
STOP

3. Fehlendes Programmelement

Im folgenden Algorithmus für das Backen eines Kuchens fehlt ein wichtiger Schritt.

START
Backrohr auf 180 °C stellen
Zutaten mischen
Wenn: Kuchen fertig?
    Kuchen herausnehmen
Sonst:
    Warten
STOP

Es wurde vergessen den Kuchen in das Backrohr zu stellen. Der Algorithmus kann verbessert werden, indem man den fehlenden Schritt hinzufügt:

START
Backrohr auf 180 °C stellen
Zutaten mischen
Kuchen ins Backrohr stellen
Wenn: Kuchen fertig?
    Kuchen herausnehmen
Sonst:
    Warten
STOP

3.6 Übungen zur Wiederholung

1. **Als Anweisungsfolge bezeichnet man:** ☆★★

   a) eine bestimmte Anzahl von Anweisungen, die der Reihe nach ausgeführt werden können.

   b) eine bestimmte Anzahl von Anweisungen, die eine nach der anderen zu befolgen sind.

   c) eine Analyse eines zu lösenden Problems.

   d) eine Liste von Empfehlungen zur Verbesserung eines Computerprogramms.

2. **Was dient nicht zur Darstellung eines Algorithmus?** ☆★★

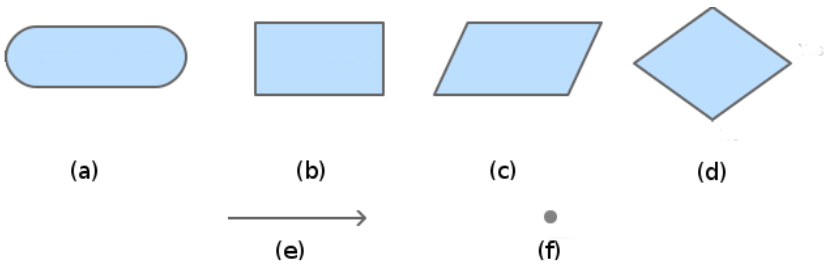
   a) Computerprogramm

   b) Flussdiagramm

   c) Pseudocode

   d) Tortendiagramm

3. **Ordne die folgenden Symbole ihren richtigen Bezeichnungen zu:** ☆☆★



Pfeil	
Entscheidung	
Start oder Stop	
Prozess	
Ein- oder Ausgabe	
Verbinder	

4. **Welche Anweisung wird im folgenden Pseudocode unmittelbar nach „Zutaten mischen“ ausgeführt?** ☆★★

```
START
Backrohr auf 180 °C stellen
Zutaten mischen
Kuchen ins Backrohr stellen
Wenn Kuchen fertig?
    Kuchen herausnehmen
Sonst:
    Warten
STOP
```

a) STOP
b) Warten
c) Lebkuchen aus Backrohr nehmen
d) Kuchen ins Backrohr stellen

5. **Das folgende Programm soll Wasser aus der Waschmaschine befördern. Wo liegt der Fehler im Programm?** ☆☆☆

```
START
Pumpe einschalten
Wenn: Kein Wasser mehr da?
        1 Sekunde warten
Pumpe ausschalten
STOP
```

a) Fehlende Anweisung
b) Falsche Reihenfolge
c) Falsches Entscheidungskriterium
d) Inkorrekte Einrückung