ANLEITUNG

Computational Thinking – Grundlagen

Was ist Computational Thinking – und warum ist es hilfreich?

Computational Thinking (CT) ist ein Prozess zur Problemlösung, der hilft selbst komplexe, »unlösbare« Probleme zu bewältigen. Dabei geht es darum, ein Problem ausdrücklich zu beschreiben und eine Lösung so zu formulieren, dass sie von einem Computer ausgeführt werden kann. So können Computer uns bei der Lösung von Problemen und der Automatisierung von Prozessen unterstützen.

Die vier Grundpfeiler des Computational Thinkings

CT beruht auf vier Fähigkeiten: der Zerlegung, der Mustererkennung, der Abstraktion und dem Algorithmischen Design. Zusammen bilden sie die Grundpfeiler des Computational Thinkings. Im folgenden Abschnitt werdet Ihr mit den vier Pfeilern vertraut gemacht.

Zerlegung: »Die Zerlegung eines komplexen Problems in kleinere, einfacher handhabbare Teilprobleme.«

Alltagsbeispiele

- Das Planen einer großen Geburtstagsparty kann zunächst überwältigend sein, aber wenn man das »Problem« in kleinere Teilaufgaben zerlegt, dann wird es bewältigbar. Für den Geburtstagskuchen sucht man zunächst ein Rezept aus, es werden die Zutaten eingekauft, dann der Kuchen gebacken und der Kuchen dann verziert. Ähnliche Zerlegungen werden für die anderen Snacks und Partyspiele angewandt.
- Die Morgenroutine kann in verschiedene Teil-Aufgaben zerlegt werden: Anziehen, Frühstücken, Zähne putzen. Diese Aufgaben können wiederum in kleinere Aufgaben zerlegt werden. Zum Beispiel kann Frühstücken zerlegt werden in die Aufgaben Kaffee machen, Brot belegen, Brot essen, Kaffee trinken.

Warum »Zerlegung«?

Durch das Zerlegen komplexer Aufgaben in einfachere Teilaufgaben werden schwierige Probleme bewältigbar. Nachdem die schlichteren Teilaufgaben gemeistert wurden, können ihre Lösungen so zusammengesetzt werden, dass das ursprüngliche komplexe Problem gelöst wird. Den Zerlegungsprozess nennt man auch »Analyse«, während die Verknüpfung der Teillösungen zur Lösung des Hauptproblems »Synthese« genannt wird.

Mustererkennung: »Die Suche nach Ähnlichkeiten zwischen verschiedenen und innerhalb eines einzelnen Problems.«

Alltagsbeispiel

Um eine Bowle auf der Geburtstagsparty zu haben, müssen ähnliche Schritte unternommen werden wie für den Kuchen: Ein Rezept wird ausgesucht, die Zutaten werden eingekauft, die Zutaten werden gemischt und die Bowle wird gekühlt. Wenn man diese Ähnlichkeit erkennt, kann man die Geburtstagsfeier effizienter vorbereiten: Statt einmal für den Kuchen einzukaufen, dann den Kuchen zu backen und dann noch einmal für die Bowle einkaufen zu gehen, kann man die Einkäufe zusammenlegen und Zeit sparen.

Warum »Mustererkennung«?

Die Mustererkennung zwischen einem aktuellen Problem und Problemen, die man bereits kennt und beherrscht, ermöglicht die Wiederverwendung des bereits bekannten Lösungsschemas und spart somit Zeit.
Das Gleiche gilt für die Mustererkennung innerhalb eines komplexen Problems bzw. zwischen dessen Teilproblemen. Wenn die Problemlösung nun automatisiert wird, indem ein Computerprogramm geschrieben wird, können Teile des Programmcodes wiederverwendet werden, wodurch der Code kürzer und in der Regel leichter verständlich wird.

Abstraktion: »Das Entfernen von nicht benötigten Details eines Problems, um sich auf die wichtigen Aspekte zu konzentrieren. So kann eine allgemeine Lösung entwickelt werden, die für das Problem (und ähnliche) funktioniert.«

Alltagsbeispiel

Je nach Kontext liegt der Fokus auf unterschiedlichen Informationen. Ein detailliertes Satellitenbild ist zum Beispiel nicht so hilfreich, wenn man mit dem Auto eine Reise machen möchte. Stattdessen ist eine Karte, die nur alle Autobahnen zeigt und andere Informationen auslässt, hilfreicher. Wenn Ihr hingegen mit der U-Bahn durch die Stadt fahren wollt, ist eine Karte nützlicher, die nur die Haltestellen der Bahn und ihre Verbindungen anzeigt und dabei Informationen wie die Position der Autobahnausfahrten weglässt.

Warum »Abstraktion«?

Das Entfernen (»Abstrahieren«) von unwichtigen Details ermöglicht eine Verallgemeinerung der konkreten Lösung zu einem Schema. Erst dann kann die Lösung wiederverwendet werden für ähnliche Probleme. Des weiteren ist ein gewisser Abstraktionsgrad vorausgesetzt, um dem Computer Anweisungen zu geben, die dieser ausführen kann, und um Prozesse zu automatisieren.
In der Regel setzt Abstraktion die Mustererkennung voraus: Erst durch das Erkennen von wiederkehrenden Mustern wird deutlich, welche Informationen konstant bleiben, welche sich ändern und welche unwichtig sind und somit vernachlässigt werden können.

Algorithmisches Design: »Die Entwicklung einer Schritt-für-Schritt-Lösung für das Problem.«
Bei der Formulierung der Lösung bietet sich Pseudocode an.

Alltagsbeispiele

- Anweisungen im Backbuch: 1. Heize den Ofen auf 180 °C 2.Mische die aufgelisteten Zutaten 3. Fette die Kuchenform ein 4. Fülle die Mischung in die Kuchenform …
- Wegbeschreibung von hier bis zur nächsten Toilette: »Gehe den Gang geradeaus runter bis zum Treppenhaus, dann gehe ein Stockwerk höher und dann gehe geradeaus bis zur zweiten Tür links, wo sich die Toiletten befinden«

Warum »Algorithmisches Design«?

Algorithmisches Design ermöglicht es, eine schrittweise Lösung so zu formulieren, dass auch ein Computer das Problem lösen kann, d. h., die Rechenleistung und Effizienz von Computern kann für die Lösung eigener Probleme genutzt werden. Auf diese Weise können ganze Prozesse automatisiert werden.

Vielen Dank!
Wir freuen uns
über Euer Feedback.