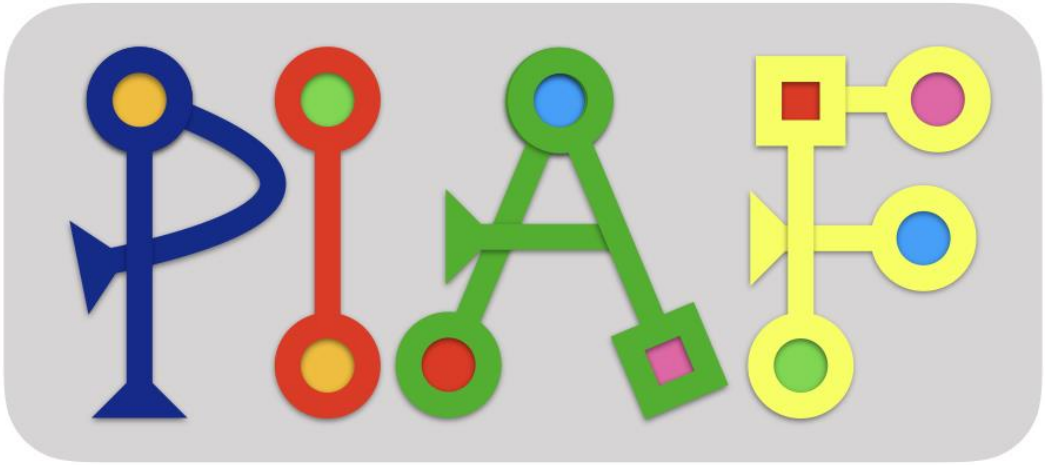

PIAF : développer la Pensée Informatique et Algorithmique dans l'enseignement Fondamental

Referenzrahmen

Anhang 1: Beschreibung und Beispiele

Erasmus+

LIÈGE université

UNIVERSITÉ DE LORRAINE

UNIVERSITÉ DU LUXEMBOURG

UNIVERSITÄT DES SAARLANDES

Version 1.2 – Juni 2021

Inhaltsverzeichnis

Kompetenz 1: Abstrahieren / Verallgemeinern

K 1.1 Objekte und Aktions-Sequenzen benennen

Definition:
In der Lage sein, Objekten, Aktionen und Aktions-Sequenzen einen Namen zu geben.

Hintergründe:
Es ist wichtig, sich anzugewöhnen bestehenden Objekten / Aktionen Namen zu geben, damit man darauf zurückgreifen kann und Lösungen einfach und unkompliziert ausgedrückt werden können.

Dabei ist zu beachten, dass einige Werte (Objekte) bekannt sein können und sich nicht ändern (z.B. ein Familienname, eine Hausnummer), und in diesem Fall als Konstanten bezeichnet werden. Andere Werte sind nicht bekannt und können sich je nach Kontext (z.B. Temperatur) ändern, und werden daher als Variablen bezeichnet. Dieses Konzept der Variablen ist Teil der Kompetenz K 1.6 weiter unten.

Beispiele:

1. Angenommen es gibt eine Geschichte mit mehreren ähnlichen (aber unterscheidbaren) Charakteren, sollte der Lernende in der Lage sein, ihnen Namen geben zu können, um sie eindeutig zu identifizieren und die Geschichte erzählen zu können.
2. Ein Lernender führt eine Aktion aus, ein zweiter sieht sie und muss sie einfach einem dritten erzählen, der den ersten Lernenden nicht sieht und die Aktion wiederholen muss.
3. Zwei Lernende einigen sich auf vier Wörter, die vier einfachen (Sequenzen von) Handlungen entsprechen. Ein dritter Lernender kommt herein, der erste Lernende sagt eines der vier Worte, der zweite führt die Befehle aus, und der dritte muss die Sprache lernen und dann erhaltene Befehle ausführen.
4. Definieren der notwendigen Schritte, um sich anzuziehen, und diese Aktions-Sequenz dann einfach "Anziehen" nennen.

K 1.2 Unterscheiden zwischen (i) Objekt und Aktion, und (ii) atomaren Aktionen und nicht-atomaren Aktionen

Definition:
In der Lage sein zu sagen, ob sich etwas (Bild, Satz, Phrase) auf ein Objekt, eine Aktion oder eine Aktions-Sequenz bezieht.

Hintergründe:
Objekte und Aktionen können als Metaphern für die in der Informatik verwendeten Konzepte des *Ausdrucks* und der *Anweisung* gesehen werden. Typischerweise haben Ausdrücke einen Wert, während Anweisungen dies nicht haben (sie ändern den Zustand der Umgebung [Speicher, Bildschirm, etc.]).

Diese Kompetenz bezieht sich auf die Fähigkeit, diejenigen Teile eines Algorithmus (Rezept, Bewegung, etc.), die Werte haben (typischerweise durch Substantive dargestellt), von jenen zu unterscheiden, die diese nicht haben (typischerweise durch Verben dargestellt).

Einige Aktionen können in Teilaktionen zerlegt werden, in diesem Fall werden diese als nicht-atomare Aktionen bezeichnet. Aktionen, die nicht weiter zerlegt werden können, werden als atomare Aktionen bezeichnet.

Beispiele:

1. Angesichts von Wörtern/Wortgruppen sagen zu können, ob es sich um ein Objekt (ein Berg, ein Baum, ein Bild, die Zahl 5) oder eine Aktion (springen, etwas anziehen, singen, denken) handelt.
2. Wenn man Karten mit Wörtern bekommt, in der Lage zu sein, sie in zwei Gruppen einzuteilen (Objekte vs. Aktionen) und dann schauen welche Aktionen auf welche Objekte angewendet werden können.
3. Sich überlegen (i) welche Aktionen man mit einem Objekt ausführen kann oder (ii) mit welchen Objekten man eine Aktion ausführen kann.
4. Folgende Aktionen in atomare und nicht-atomare Aktionen unterteilen: Sich auf die Schule vorbereiten, Socken anziehen, tanzen, Tiere nach Größe ordnen, 90 Grad nach links drehen, eine Zahl mit zwei multiplizieren, den ersten Buchstaben des Alphabets angeben,...

K 1.3 Die Eingabeparameter einer Aktions-Sequenz identifizieren

Definition:
In der Lage sein, zu erkennen, was (Objekte oder Aktions-Sequenzen) erforderlich ist, um eine Aktions-Sequenz durchzuführen.

Hintergründe:
Aktions-Sequenzen können als Funktionen (wie in der Mathematik) betrachtet werden. Die Eingabeparameter werden oft einfach Eingabe genannt. Beim Erlernen der Algorithmik ist es ratsam, zunächst klar und präzise die Eingabeparameter und Resultate (Ausgabe, Ergebnis) eines Algorithmus aufzulisten.

Beispiele:

1. Wenn ich sagen will, wer der Größte von drei Personen ist, muss ich ihre Größen kennen.
2. Um mich anzuziehen, kann ich die Kleidungsstücke auflisten, die ich brauche.
3. Für ein Rezept kann ich die Zutaten auflisten.
4. Um jemandem die Haare zu schneiden, brauche ich eine Schere und jemanden mit Haaren.

K 1.4 Das Ergebnis einer Aktions-Sequenz beschreiben

Definition:

Nach der Ausführung einer Aktions-Sequenz (informell, Algorithmus, Programm), in der Lage sein zu sagen, was passiert ist. Der Lernende muss die endgültige Situation beschreiben (Veränderungen eines Wertes, der Position eines Objekts,...).

(Anmerkung: Die Erstellung der Sequenz ist nicht Teil dieser Kompetenz, sondern der Kompetenz K 2.3).

Hintergründe:
Diese Kompetenz ist sehr eng verbunden mit der Kompetenz K 1.3. Beide Kompetenzen sind erforderlich um effektive Aktions-Sequenzen zu definieren.

Beispiele:

1. Bei einem Programm (z.B. in Scratch) kann der Lernende sagen, dass die Katze dreimal auf und ab bewegt wurde, oder die Schildkröte im Auto ist, oder die Kiste 3 Stifte enthält .
2. Bei einem Rezept (ohne Titel) kann der Lernende erkennen, welches Gericht zubereitet wurde.
3. Hier ist eine Karte (Raster 3x3), hier ist der Startpunkt (Position (1,1)), und hier ist der Ankunftspunkt (Position (3,3)):

		Ankunftspunkt
Startpunkt		

Hier ist ein Algorithmus:

Nach oben bewegen
Nach oben bewegen
Nach rechts bewegen
Nach rechts bewegen

Angenommen, ein Roboter wird in Position (1,2) gebracht, erlaubt dieser Algorithmus dem Roboter, den Ankunftspunkt zu erreichen?

K 1.5 Das Ergebnis einer Aktions-Sequenz vorhersagen

Definition:
In der Lage sein, anhand einer Aktions-Sequenz zu erkennen, was passieren wird, wenn sie ausgeführt wird. Im Gegensatz zu Kompetenz 1.4 geht es bei dieser Kompetenz darum, eine Vorhersage zu machen, ohne die Aktions-Sequenz tatsächlich auszuführen.

Hintergründe
Diese Kompetenz ist wichtig, um zu verhindern, dass die Lernenden sich zu sehr auf einen " Versuch und Irrtum" Ansatz verlassen (d.h. Das Programm ändern solange es fehlschlägt, ohne zu versuchen zu verstehen/vorhersagen, warum dies der Fall ist).

Beispiele
1. Wie im Beispiel von K 1.4, nur mit der Einschränkung, dass das Programm / der Algorithmus nicht tatsächlich ausgeführt wird.
2. Geben Sie Lernenden eine Startposition auf einem Gitter (z.B. 1,1 auf dem obigen Gitter) und ein Programm (z.B. Nach oben, Nach rechts, Nach unten, Nach links, Nach unten, Nach links) und fragen Sie sie, welche Figur die Schildkröte zeichnen wird, wenn dieses Programm von dieser Startposition aus abläuft.
3. Geben Sie den Lernenden ein Kriterium für den Vergleich von Personen (z. B. "größer als") und fragen Sie, wie eine Reihe von Zweier-Vergleichen einer Gruppe von Personen aussehen wird, so dass bei jedem Vergleich die Person mit dem größeren Wert beibehalten wird für den nächsten Vergleich.

K 1.6 Objekte verwenden, deren Wert sich ändern kann

Definition
In der Lage sein, auf Werte zu verweisen, die je nach Kontext variieren können, indem man benannte Abstraktionen (Identifikatoren) für sie verwendet.

Hintergründe
Diese Kompetenz führt den Begriff der Computervariablen ein, welcher der Zuordnung eines Namens zu einem Speicherplatz entspricht.

Beispiele
1. Die Lernenden bekommen Karten mit Zahlen. Der Lehrer sagt Stück für Stück eine Addition auf. Die Lernenden müssen das Ergebnis der Addition spontan berechnen (ohne ihre Elemente im Voraus zu kennen) und die Ergebniskarte auswählen / aktualisieren.
2. Ein Programm erzeugen, bei dem die Katze alle fünf Klicks miaut, oder bei dem die Punktzahl des Spielers die Anzahl der Klicks auf die Katze ist.

K 1.7 Vorhandene Objekte und Aktionen oder Aktions-Sequenzen erkennen, mit denen ein ähnliches Ziel erreicht werden kann

Definition

Angesichts eines neuen Problems erkennen können, dass es einem bereits bekannten Problem ähnlich ist, und dass eine bereits bekannte Lösung analog und teilweise wiederverwendbar ist, um das neue Problem zu lösen. Das konkrete Ändern einer bestehenden Lösung fällt allerdings in den Zuständigkeitsbereich von K 2.5.

Hintergründe

Zum informatischen Denken gehört auch, auf bereits vorhandenen Arbeiten aufbauen zu können, anstatt alles von Grund auf neu zu erfinden.

Beispiele

1. Sie haben eine Liste von Rezepten: Tomatensuppe, Erdbeerkuchen, Gemüsecurry. Wenn Sie eine Zwiebelsuppe zubereiten müssen, welches dieser Rezepte werden Sie verwenden?
2. Den Lernenden wird ein Algorithmus zur Berechnung des größten Wertes von drei Werten gegeben. Anschließend fragt die Lehrerin, welcher Teil wiederverwendet werden kann, um den kleinsten Wert von drei Werten zu berechnen.

Kompetenz 2: Eine Aktions-Sequenz kombinieren / zerlegen

K 2.1 Eine Aktions-Sequenz anordnen, um ein Ziel zu erreichen

Definition:
Angesichts einer ungeordneten Liste von Aktionen und einem Ziel in der Lage sein, diese Aktionen in eine gültigen Reihenfolge bringen, um eine Sequenz aufzubauen, mit der das Ziel erreicht wird.

Hintergründe:
Bevor eine ganze Abfolge von Aktionen definiert werden kann (siehe K 2.3), ist ein erster Schritt, um diese Kompetenz zu erreichen, bereits vorhandene Teile einer Aktions-Sequenz zu *sortieren*. Der Lernende muss also nicht alle Elemente identifizieren, die er zur Erreichung des Ziels benötigt, sondern sie nur in die richtige Reihenfolge bringen.

Beispiele:

1. Lassen Sie die Lernenden sich vorstellen, dass sie einen Kuchen backen müssen, und geben Sie ihnen ungeordnete Teile des Rezeptes, sie sollten in der Lage sein, diese Teile richtig zu ordnen (z.Bsp. 1. Eine Schüssel nehmen, 2. die Menge an Mehl messen, etc.).
2. Geben Sie den Kindern die verschiedenen Schritte zum Anziehen und bitten Sie sie, diese korrekt zu ordnen (mehrere sind akzeptabel).

K 2.2 Eine Aktions-Sequenz ergänzen, um ein einfaches Ziel zu erreichen

Definition:
Bei einer begrenzten Abfolge von Aktionen und einem einfachen Ziel (das sich nicht auf Konzepte wie Bedingungen und Schleifen beruft), kann die Aktions-Sequenz (durch Hinzufügen fehlender Aktionen) so ergänzt werden, dass das Ziel erreicht wird.

Hintergründe:
In der Algorithmik beginnt das Lernen oft damit, sich von bestehenden Lösungen inspirieren zu lassen. In diesem Zusammenhang ist es wichtig, beurteilen zu können, wie weit eine Lösung (bzw. ein Problem) von einer bestehenden Lösung (bzw. einem Problem) entfernt ist, und diese bei Bedarf zu ergänzen.

Beispiele:

1. Eine Zeichnung ist fast fertig und der Lernende muss die fehlende(n) Aktion(en) hinzufügen.
2. In Anbetracht des Ziels, eine Figur an einen beliebigen Ort auf dem Bildschirm zu bringen, wobei ein vorhandenes Programm schon einen Teil des Weges beinhaltet, muss der Lernende die fehlenden Aktionen (möglicherweise Kopieren und Einfügen bereits vorhandener Aktionen) durchführen, um die Figur an den Zielort zu bringen.

K 2.3 Eine Aktions-Sequenz erstellen, um ein einfaches Ziel zu erreichen

Definition:
Ausgehend von einem einfachen Ziel (das sich nicht auf Konzepte wie Bedingungen und Schleifen beruft), alle notwendigen Aktionen identifizieren und in die richtige Reihenfolge bringen können.

Hintergründe:
Diese Fähigkeit kann als Erweiterung der Fähigkeit angesehen werden, eine bestehende Aktions-Sequenz zu sortieren (siehe K 2.1) oder ergänzen zu können (siehe K 2.2).

Beispiele:

1. Der Lernende wird in die Mitte eines leeren Raumes gesetzt, kriegt eine Reihe von Aktion welche erlaubt sind. Er soll die geordnete Liste der notwendigen Aktionen zum Öffnen einer Tür angeben, z.Bsp.: sich zur Tür drehen, zur Tür gehen, den Griff greifen, ihn nach unten drücken, die Tür zu sich hinziehen.
2. Ein Quadrat zeichnen: Vorwärts, rechts abbiegen, vorwärts, rechts abbiegen, ...

K 2.4 Eine Aktions-Sequenz erstellen, um ein komplexes Ziel zu erreichen

Definition:
In der Lage sein, Aktionen zu identifizieren und zu kombinieren, um ein komplexeres Ziel zu erreichen (ein Ziel, welches Bedingungen, Schleifen, etc. erfordert).

Hintergründe:
Diese Kompetenz ist eine Erweiterung der oben genannten Kompetenz K 2.3.

Beispiele:

1. Den Weg durch ein Labyrinth finden ohne die Wände zu berühren.
2. Die notwendigen Aktionen zum Addieren von zwei ganzen Zahlen beschreiben.
3. In einer Karte mit Anfangs- und Endpositionen (möglicherweise mehrere zu besuchende Kontrollpunkte) und zusätzlichen Regeln (Hindernisse,...) werden die Maßnahmen zur Erreichung des Ziels unter Einhaltung der Regeln beschrieben.

K 2.5 Aktions-Sequenzen kombinieren, um ein Ziel zu erreichen

Definition:
Angesichts bereits bekannter Aktions-Sequenzen, die bekannte Ziele erreichen, in der Lage sein, (i) in einem gegebenen Kontext (neues Ziel zu erreichen) relevante von irrelevanten

Sequenzen zu unterscheiden und (ii) sie in einer gültigen Reihenfolge miteinander verbinden können (d.h., die es ermöglicht, das neue Ziel zu erreichen).

Hintergründe:
Hier liegt der Fokus auf dem *Kombinieren* von Sequenzen. Es gibt noch eine weitere Ebene der Abstraktion. Es geht nicht nur darum, Aktionen oder Objekte zu kombinieren, sondern auch Sequenzen (welche allgemein mit einem Namen bezeichnet werden). Diese Kompetenz erweitert K 1.7 bei welcher es lediglich darum ging wiederverwendbare Teile zu identifizieren.

Beispiele:
1. Mit Aktions-Sequenzen einzelne Gerichte zubereiten und (in der richtigen Reihenfolge) zu einem kompletten Menü vom Aperitif bis zum Dessert kombinieren.
2. Ausgehend von Aktions-Sequenzen, die es ermöglichen, einfache geometrische Formen (z.B. Dreiecke, Quadrate) auf einem Gitter zu zeichnen, eine Sequenz vorschlagen, die es ermöglicht, eine komplexe Form (z.B. ein Sechseck) zu zeichnen.
3. Ein komplexes Spiel (mit mehreren Levels) aus kleineren, bereits bestehenden, Spielen erstellen.
4. Angesichts der Funktion max(a,b), die das Maximum von a und b berechnet, ist zu erkennen, dass max(max(a,b),c) das Maximum von a, b und c berechnet.

K 2.6 Ziele in einfachere Teilziele zerlegen

Definition:
Angesichts eines komplexen Ziels in der Lage sein, es in mehrere Teilziele aufteilen zu können, welche leichter erreicht werden. Diese Teilzielen können dann von verschiedenen Personen angegangen werden.

Hintergründe:
Beim Entwerfen umfangreicher Software sehen sich die Lernenden oft mit einem "weißen Blatt" konfrontiert und fragen sich, "wo soll ich anfangen?". Es ist wichtig, komplexe Probleme anzugehen, indem man sie zunächst in kleinere (und leichter zu lösende) Probleme zerlegt.

Beispiele:
1. Was sind die wichtigsten Schritte, um den Schulrucksack für morgen fertig zu machen? (z.B. Schultasche finden, die Fächer im morgigen Zeitplan überprüfen, Material für jedes Fach in die Tasche packen)
2. Eine Liste von Gerichten für eine Gruppe von Personen bestellen (z.B. alle Bestellungen erhalten, gruppieren, die zusammengestellte Liste in die Küche schicken).
3. Bei einer komplexen geometrischen Form den Flächeninhalt berechnen indem man diese in einfache Formen zerlegt.

Kompetenz 3: Eine Aktions-Sequenz steuern

K 3.1 Eine Aktions-Sequenz eine bestimmte Anzahl mal wiederholen

Definition:
In der Lage sein, Wiederholungen von Aktionen oder Aktions-Sequenzen *eine bestimmte Anzahl mal* strategisch zu nutzen um ein Ziel zu erreichen.

Hintergründe:
Eine erste bedeutende Familie von Wiederholungen in der Informatik sind diejenigen, die eine bestimmte (im Voraus bekannte) Anzahl mal ausgeführt werden (“For-Schleifen” in vielen Programmiersprachen; “for” (en.) = “für”).

Beispiele:

1. Einen Algorithmus schreiben, der es ermöglicht 3 Eigelb in eine Schale zu geben. Wenn die Lernenden den Nutzen einer Schleife nicht erkennen, bitten Sie sie, 25 Eigelb hinzuzufügen.
2. Bei einem Roboter, der sich nur 45° nach links drehen kann, erstellen Sie eine Reihe von Aktionen, die es ihm ermöglichen, einem bestimmten Weg zu folgen (jedes Mal, wenn der Roboter sich drehen muss, muss das Programm die Aktion "Links Drehen" so oft wie nötig wiederholen, um den gewünschten Winkel zu drehen).

K 3.2 Eine Aktions-Sequenz wiederholen, bis ein Ziel erreicht ist

Definition:
In der Lage sein, Wiederholungen von Aktionen oder Aktions-Sequenzen strategisch zu nutzen *bis eine Bedingung erfüllt ist* um ein Ziel zu erreichen.

Hintergründe:
Wiederholungen, die ausgeführt werden, bis eine Bedingung erfüllt ist (die Anzahl der Wiederholungen ist im Voraus nicht bekannt), sind ebenfalls eine bedeutende Familie in der Informatik (“While-Schleifen” in Programmiersprachen; “while” (en.) = “während”).

Beispiele:

1. Die Zutaten mischen, bis ein glatter Teig entsteht.
2. Nach vorne gehen, bis du die Wand erreichst (d.h. es gibt einen Sensor, um nahe Hindernisse zu erkennen).

K 3.3 Eine einfache Bedingung in eine Aktions-Sequenz einfügen

Definition:
In der Lage sein, einfache Bedingungen zu verwenden (kann mit einem einzigen Kriterium ausgedrückt werden), um gewisse Aktionen oder Aktions-Sequenzen zu erlauben (oder zu unterbinden).

Hintergründe:
Die Steuerung der Ausführung einiger Aktionen oder Aktions-Sequenzen mittels Bedingungen ist zentral im informatischen Denken, siehe auch Kompetenz K 4.1 unten.

Beispiele:

1. **Wenn** der Teig dick ist, **dann** gib 20 cl Wasser hinzu und mische durch.
2. **Wenn** der Roboter der Wand zugewandt ist, **dann** bewegt er sich rückwärts, **sonst** stoppt er.
3. Frag eine Person nach ihrem Alter und sag ihr (**je nach Antwort**) "Du bist jetzt erwachsen" **oder** "Warte ein wenig, bevor du ein Auto fährst".

K 3.4 Eine komplexe Bedingung in eine Aktions-Sequenz einfügen

Definition:
In der Lage sein, Bedingungen mittels logischer Operatoren zu kombinieren (und, oder, nicht), um gewisse Aktionen oder Aktions-Sequenzen zu erlauben (oder zu unterbinden).

Hintergründe:
Beim Entwickeln von Algorithmen ist es oft notwendig, eine Vielzahl von Bedingungen zu kombinieren, um eine Lösung für ein Problem zu erstellen.

Beim Erwerb des informatischen Denkens ist es auch wichtig, jede Bedingung als ein Objekt wie jedes andere zu betrachten (in der Informatik werden sie als boolesche Objekte bezeichnet). Boolesche Objekte können konstant (wie der Ausdruck "2>1", der immer wahr ist) oder variabel sein ("es ist heute mehr als 10°" ist nicht immer wahr).

Beispiele:

1. Falls x streng genommen kleiner als 0 **und** größer als 24 ist, dann ist es keine gültige Stunde.
2. Wenn der Teig dick ist **und** seine Temperatur unter 10°C liegt **und** wir an einem Samstag **oder** Sonntag sind, dann begib dich in den 24/7-Shop **und** kaufe Milch.
3. Wenn der Roboter der Wand zugewandt ist **und** sich auch eine Wand auf der linken Seite befindet **oder** die Temperatur über 10°C liegt, dann stoppt der Roboter, sonst dreht er zweimal um 45°.

Kompetenz 4: Objekte oder Aktions-Sequenzen bewerten

K 4.1 Zwei Objekte anhand eines bestimmten Kriteriums vergleichen

Definition:
Bei zwei (oder mehreren) Objekten und einem Vergleichskriterium kann man diese Objekte in Bezug auf das Kriterium vergleichen.

Hintergründe:
In der Informatik ist es üblich, über Objekte mit einer Ordnungsbeziehung nachzudenken, wie z.B. über Zahlen (Reihenfolge nach ganzen oder reellen Zahlen) oder Wörter (lexikographische Reihenfolge). Die Fähigkeit, diesen Begriff der Ordnungsbeziehung auf verschiedene Objekte anwenden zu können (z.B. Größenordnung oder Altersordnung von Menschen), ist Teil des informatischen Denkens.

Beispiele:

1. Der Lehrer wählt das Kriterium (Größe, Gewicht, Breite, usw.) aus und die Kinder müssen die Objekte je nach Fall ordnen
2. Erstellen Sie Karten mit Objekten (Menschen, Tiere, Dinge, die Sie in einer Küche finden) und bitten Sie die Lernenden, sie in mehrere Kategorien einzuteilen, die sich nach einem von ihnen gewählten Kriterium richten (Größe, Anzahl der Beine, Dinge die sie essen, Hautfarbe, Material), und bitten Sie sie dann, das Kriterium und die Gruppierung zu ändern. (Anmerkung: Es gibt hier nicht immer eine einfache Reihenfolge).
3. Gewichte mit einer Tafelwaage vergleichen. Diese Art Waage ist ein gutes Werkzeug, da sie es ermöglicht, zwei Werte auf einmal zu vergleichen (wie ein Prozessor, der binäre Operationen durchführt). Es vermeidet die Gefahr, alle Werte gleichzeitig vergleichen zu wollen, wie es bei Anfängern oft der Fall ist.

K 4.2 Zwei Aktions-Sequenzen anhand eines bestimmten Kriteriums vergleichen

Definition:
In der Lage sein, Aktions-Sequenzen in Bezug auf Lesbarkeit, Anzahl der Zeilen, Ausführungszeit oder andere Kriterien zu vergleichen.

Hintergründe:
Wenn eine Lösung (eine Aktions-Sequenz) gefunden wurde, ist es verlockend, ihre Qualität nicht in Frage zu stellen. Beim informatischen Denken geht es auch darum, zu wissen, dass ein Problem möglicherweise keine, eine oder mehrere Lösungen hat, und es darum geht, eine kritische Sicht auf Lösungen aufzubauen (was erfordert sie vergleichen zu können).

Beispiele:

1. Wenn man von Nancy nach Lüttich reisen möchte, gibt es mehrere mögliche Routen. Welche ist die schnellste, die ökologischste, die billigste,....?
2. Angenommen man hat zwei Roboter: Einen, der sich schnell bewegt, aber langsam dreht, und einen anderen, der sich schnell dreht, aber langsam bewegt. Manchmal ist ein längerer Weg je nach Roboter effizienter als ein kürzerer. Geben Sie den Schülern mehrere Labyrinthe mit unterschiedlichen Merkmalen (z.B. Anzahl der erforderlichen Drehbewegungen) und lassen Sie sie entscheiden, welcher Roboter für welches Labyrinth geeignet ist.
3. Ich weiß, wie man radelt, fährt und geht. Jetzt soll ich eine Reise machen und entscheiden, welches Transportmittel am besten passt (basierend auf Parkplatzmöglichkeiten, Entfernung, Verkehr,...).

K 4.3 Eine Aktions-Sequenz anhand eines bestimmten Kriteriums verbessern

Definition:
In der Lage sein, darüber nachzudenken eine Aktions-Sequenz in Bezug auf ein Kriterium zu verbessern und die Aktions-Sequenz dementsprechend zu verbessern.

Hintergründe:
Als Nachfolger der Kompetenz K 4.2 ist es wichtig, nach Wegen zur Verbesserung suchen zu können, sobald erkannt wurde, dass eine Lösung (Aktions-Sequenz) in Bezug auf ein bestimmtes Kriterium suboptimal ist.

Beispiele:
1. Geben Sie ihnen einen Code, mit dem sich ein Roboter von Punkt A zu Punkt B bewegen kann, und fragen Sie nach einem Weg mit weniger Drehungen/Sprüngen/Schritten.
2. Geben Sie ein Rezept mit sich wiederholenden Aktionen (ein Ei hinzufügen, ein Ei hinzufügen, ein Ei hinzufügen) und bitten Sie darum, den Code lesbarer zu gestalten, indem Sie z.B. eine Schleife für die Eier hinzufügen.

Kompetenz 5: Umgang mit formalen Repräsentationen

K 5.1 Objekte oder Aktions-Sequenzen in einer formalen Repräsentation darstellen

Definition
In der Lage sein, ein Objekt oder eine Aktion darzustellen, indem man ein genau definiertes (d. h. nicht eindeutiges) Darstellungssystem verwendet.

Hintergründe:
Informatisches Denken bedeutet, eine genau definierte Sprache verwenden zu können, die von einem anderen Menschen oder einer Maschine effizient und eindeutig verarbeitet werden kann. Diese Kompetenz 5.1 bezieht sich daher auf die Fähigkeit, ein Wort, ein Bild oder eine Zahl (oder eine Kombination davon) durch einen genau definierten Code/Sprache (z. B. ein neues Alphabet, Symbole, usw.) darstellen zu können. Diese Darstellung wird in der Informatik allgemein als *Kodierung / Dekodierung* bezeichnet wird.

Beispiele:
1. Unterschiedliche Piktogramme mit genau definierter Bedeutung (bezogen auf eine Aktion) verwenden, um eine Aktions-Sequenz zu beschreiben (z. B. Bewegungen eines Roboters auf einem Raster), die mündlich und formlos genannt wird.
2. Einen Code (z. B. ASCII) anstelle eines Zeichens der Tastatur wiedergeben.
3. Eine geheime Nachricht an Jemanden senden durch Ver- und Entschlüsseln mit Codes (mit Hilfe einer Korrespondenztabelle).

K 5.2 Objekte oder Aktions-Sequenzen zwischen formalen Repräsentationen übersetzen

Definition:
In der Lage sein, eine andere formale Sprache/Code zu verwenden, um (eine) formal definierte Aktions-Sequenz(en) darzustellen.

Hintergründe:
Diese Kompetenz ist eine Erweiterung der oben genannten Kompetenz K 5.1, mit der zusätzlichen Einschränkung, dass die Darstellungssysteme für Eingangs- und Ausgangsinformationen auf genau definierten Regeln (formalen Repräsentationen) basieren.

Beispiele:
1. Die arabische Zahl 2 in eine andere formale Repräsentation (z. B. römische Ziffern, Zeichensprache, usw.) übersetzen.
2. Angenommen, man hat zwei verschiedene Sprachen für einen Roboter: Einerseits "Gehe nach Norden, Süden, Osten, Westen", und andererseits "nach vorne, biege nach links, biege nach rechts ab". Bitten Sie die Lernenden, ein Programm von einer dieser beiden Sprachen in die andere zu übersetzen.

Kompetenz 6: Eine Aktions-Sequenz iterativ erstellen

K 6.1 Überprüfen, ob eine Aktions-Sequenz ein vorgegebenes Ziel erreicht

Definition:
In der Lage sein zu überprüfen, ob eine bestimmte Aktions-Sequenz das erwartete Ergebnis erzeugt oder nicht.

Hintergründe:
Einige Aktions-Sequenzen beziehen sich ein spezifisches Problem während andere sich auf allgemeine Probleme (abhängig von Parametern) beziehen. Basierend auf dem letzteren Fall, ist die hier verlangte Kompetenz die Fähigkeit, Parameter überprüfen zu können, die eine Aktions-Sequenz (basierend auf dem erwarteten Ergebnis) erfolgreich oder nicht erfolgreich sein lassen.
Diese Kompetenz zielt neben anderen Aspekten daher darauf ab, beim Lernenden die Gewohnheit zu entwickeln, eine Reihe von Aktionen auf der Grundlage von Tests (Parametern und erwarteten Ergebnissen) zu überprüfen und iterativ zu ändern.

Beispiele:

1. Nimm das Wort "Toto", verstehe die Aktions-Sequenz, die es möglich macht, die Anzahl der Buchstaben zu zählen, die das Wort hat, und überprüfe, ob diese Sequenz „4“ zurückgibt.
2. Nimm irgendein Wort, eine Aktions-Sequenz, die die Anzahl der Buchstaben berechnet und überprüfe, ob das zurückgegebene Ergebnis dem erwarteten entspricht.
3. Hier ist eine Karte (Raster 3x3), hier ist der Ausgangspunkt (Position (1,1)) und hier ist der Ankunftspunkt (Position (3,3)):

		Ankunftspunkt
Ausgangspunkt		

Hier ist ein Algorithmus:

Nach oben bewegen
Nach oben bewegen
Nach rechts bewegen
Nach rechts bewegen

Vergewissere dich, dass der Roboter, wenn er sich am Ausgangspunkt befindet und der Algorithmus ausgeführt wird (der Roboter wird bewegt), nachher am Ankunftspunkt befindet.

K 6.2 Fehler in einer Aktions-Sequenz erkennen

Definition:
Im Falle einer fehlerhaften Aktions-Sequenz in der Lage sein, den/die Fehler zu identifizieren.

Hintergründe:
Tests sind nicht nur nützlich, um zu prüfen, ob eine Aktions-Sequenz gültig ist. Wenn diese gut gewählt ist, erleichtert sie es auch, herauszufinden, warum eine bestimmte Sequenz nicht gültig ist, und Hypothesen über die Ursachen des Fehlers aufzustellen.

Beispiele:

1. Rechne 120 Gramm in Kilogramm um.

   Die vorgeschlagene Sequenz: Nimm die Anzahl der Gramm, gehe zwei Stellen nach links und füge ein Komma ein → Das Ergebnis ist 1,2 kg.

   Der Lernende muss erkennen, dass er eine Korrektur machen muss.

K 6.3 Eine Aktions-Sequenz korrigieren, um ein gegebenes Ziel zu erreichen

Definition:
Ausgehend von einer bestimmten Aktions-Sequenz, die ein bestimmtes Ziel nicht erreicht, kann man sie so modifizieren, dass sie es tut.

Hintergründe:
Sobald die Tests es ermöglicht haben, den/die Fehler zu identifizieren, ist es wichtig, ihn/sie möglicherweise durch Wiederholung des Prozesses zu korrigieren. Es ist wichtig zu beachten, dass die Bereitstellung des erwarteten Ergebnisses an einem Beispiel weder eine Garantie dafür ist, dass die Abfolge der Aktionen korrekt/verallgemeinerbar ist noch, dass es sich um die beste(n) Lösung(en) handelt (z. B. in Bezug auf Zwischenschritte).

Beispiele:

1. Nehmen wir an, es gibt eine Karte (Raster 3x3) mit (1,1) als Ausgangspunkt und (3,3) als Ankunftspunkt:

		Ankunftspunkt

Ausgangspunkt		

Der folgende Algorithmus ermöglicht es nicht, vom Ausgangspunkt zum Ankunftspunkt zu gehen.
Repariere ihn.

Nach rechts bewegen
Nach links bewegen
Nach oben bewegen
Nach rechts bewegen

K 6.4 Eine Aktions-Sequenz erweitern oder modifizieren, um ein neues Ziel zu erreichen

Definition:
Von einer gegebenen Aktions-Sequenz, deren Ergebnis bekannt ist (die Sequenz ist in Bezug auf ein vorheriges Ziel vollständig), in der Lage sein, diese wiederzuverwenden und zu adaptieren, um ein anderes (neues) Ziel zu erreichen.

Hintergründe:
Wenn man in der Informatik komplexe Programme entwerfen möchte, ist es eine bewährte Praxis, mit einem einfacheren Ziel zu beginnen. Wir schreiben zuerst das entsprechende Programm, einfach, aber funktional, und erweitern es dann Schritt für Schritt um neue Funktionalitäten, wobei wir sicherstellen, dass die Änderungen nicht das bisher Erreichte aufheben. Diese Methode wird als iterativer Ansatz bezeichnet.

Man beachte, dass sich diese Kompetenz von C 2.2 unterscheidet, wo wir von einer unvollständigen Sequenz von Aktionen ausgegangen sind (die also ihr Ziel nicht erreicht hat). Hier erweitern wir schrittweise eine funktionierende Abfolge von Aktionen, um einem Endziel immer näher zu kommen.

Beispiele:

1. Nachdem du eine Zeichnung programmiert hast (z.B. ein Quadrat in Scratch), verwende diese in einem anderen Programm, um eine fortgeschrittenere Zeichnung zu programmieren (z.B. ein Haus bestehend aus dem Quadrat).
2. Die Lernenden beginnen mit einem (vorgegebenen oder selbst entwickelten) Programm, das ein Labyrinthspiel implementiert, bei dem der Spieler einen Charakter vom Eingang zum Ausgang bewegt, ohne die Wände zu berühren. Sie erweitern dann dieses Programm indem sie ein Gespenst hinzufügen, das sich auf dem Bildschirm bewegt und durch die Wände geht, und dem der Charakter entgehen muss, sonst ist das Spiel verloren.