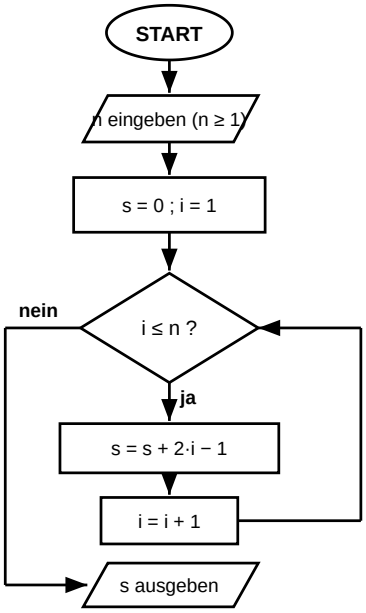
Expertenaufgaben: Flussdiagramme für Verzweigungen und Schleifen

Zusatzaufgaben für Schnelle – Transfer, Analyse und Knobeln

Diese Aufgaben sind Zusatzaufgaben für alle, die mit dem Basis-Arbeitsblatt fertig sind. Sie sind deutlich anspruchsvoller: Du musst Diagramme analysieren, Fehler finden, Ergebnisse begründen und eigene Lösungen entwerfen. Nimm dir Zeit, probiere Beispielwerte aus und notiere deine Überlegungen. Mehrere Lösungswege sind oft möglich.



Flussdiagramm X: Was berechnet dieser Algorithmus?

1 Verfolge Flussdiagramm X mit einer Tabelle (Spalten: i, s) für n = 4. Was wird ausgegeben? Welche mathematische Aussage vermutest du für beliebiges n? Begründe deine Vermutung, zum Beispiel am Zahlenmuster der addierten Werte.

2 Der Algorithmus aus Flussdiagramm X braucht n Schleifendurchläufe. Entwirf ein Flussdiagramm (oder beschreibe es genau) für denselben Zweck, das ganz ohne Schleife auskommt. Welche Vor- und Nachteile hat deine Lösung gegenüber Diagramm X? Überlege auch: Wann ist eine Schleife trotzdem sinnvoll?

3 Ein Flussdiagramm enthält eine Schleife mit der Bedingung 'i ≠ 10 ?' (ja → Schleifenrumpf, nein → Ende). Vor der Schleife steht i = 1, im Rumpf steht i = i + 2. Was passiert?

- ☐ Die Schleife läuft 5-mal und endet dann mit i = 10.
- ☐ Die Schleife läuft 4-mal und endet dann mit i = 9.
- ☐ Die Schleife endet nie (Endlosschleife), weil i nur ungerade Werte annimmt und 10 nie erreicht.
- ☐ Die Schleife wird gar nicht betreten.

4 Richtig oder falsch? Kreuze an.

	richtig	falsch
Jede Zählschleife (for-Schleife) lässt sich in eine bedingte Schleife (while-Schleife) umformen, aber nicht jede bedingte Schleife lässt sich in eine Zählschleife mit fest vorgegebener Wiederholungszahl umformen.	☐	☐

5 Knobelaufgabe (Collatz-Folge): Starte mit einer natürlichen Zahl n. Solange n ≠ 1 ist, gilt: Ist n gerade, wird n halbiert; ist n ungerade, wird n = 3·n + 1 berechnet. Bei jedem Schritt wird n ausgegeben. (a) Zeichne ein Flussdiagramm mit einer Schleife und einer Verzweigung innerhalb der Schleife. (b) Verfolge den Ablauf für n = 6 und gib alle ausgegebenen Zahlen an. (c) Warum kann man mit einem Flussdiagramm allein nicht beweisen, dass die Schleife für jedes n endet?

6 Analysiere dieses Diagramm: START → Eingabe z → Raute 'z > 0 ?'. Ja: Prozess 'z = z − 1', dann Rückführung zur Raute. Nein: Raute 'z < 0 ?'; ja → Ausgabe 'negativ', nein → Ausgabe 'null'; danach ENDE. (a) Welche Ausgabe entsteht für z = 3, z = 0 und z = −2? (b) Das Programm soll eigentlich das Vorzeichen der eingegebenen Zahl bestimmen. Welchen Fehler enthält es? (c) Korrigiere es mit möglichst wenigen Änderungen.

7 Entwirf ein Flussdiagramm für einen Zahlenraten-Algorithmus: Der Computer hat eine geheime Zahl zwischen 1 und 100 gewählt (Variable g). Der Nutzer rät so lange, bis er richtig liegt. Nach jedem Tipp t soll 'zu klein' oder 'zu groß' ausgegeben werden, außerdem soll mitgezählt werden, wie viele Versuche nötig waren. (a) Zeichne bzw. beschreibe das Diagramm (mit Schleife und Verzweigungen). (b) Wie viele Versuche braucht man höchstens, wenn man klug rät? Begründe mit dem Prinzip der Halbierung.

8 Transfer: Ein Geldautomat soll eine PIN-Eingabe prüfen. Regeln: Der Nutzer hat maximal 3 Versuche. Bei richtiger PIN erscheint 'Zugang erlaubt', nach 3 falschen Versuchen 'Karte gesperrt'. Entwirf ein Flussdiagramm. Erkläre, warum hier die Schleifenbedingung aus zwei Teilen besteht, und formuliere sie.

9 Beurteilen und begründen: Zwei Schüler lösen die Aufgabe 'Gib alle Zahlen von 1 bis 5 aus'. Schüler A nutzt eine Schleife (wie im Basis-Arbeitsblatt). Schüler B zeichnet ein Diagramm mit fünf hintereinander stehenden Ausgabe-Rechtecken ohne Schleife. Beide Diagramme sind korrekt. Welche Lösung ist besser? Was ändert sich, wenn die Zahlen von 1 bis 1000 ausgegeben werden sollen oder wenn die Obergrenze vom Nutzer eingegeben wird?

10 Ordne jedem Problem die am besten passende Kontrollstruktur zu.

1. Alle Zahlen von 1 bis 50 addieren	____	A) Bedingte Schleife (Anzahl vorher unbekannt)
2. Passwort so lange abfragen, bis es stimmt	____	B) Verzweigung (Entscheidung mit mehreren Bedingungen)
3. Entscheiden, ob ein Jahr ein Schaltjahr ist	____	C) Zählschleife (Anzahl vorher bekannt)
4. Ein Wort Buchstabe für Buchstabe ausgeben	____	D) Zählschleife über die Wortlänge

Lösungen

Expertenaufgaben: Flussdiagramme für Verzweigungen und Schleifen · Informatik · Klasse 9

Für dich als Lehrkraft: leichtere Fassung in einer Minute

QR scannen oder lehreule.de/anpassen/flussdiagramme-klasse-9 öffnen: kostenloses Konto, dann schreibt Lehreule die leichtere Fassung mit Hilfen — oder Zusatzaufgaben für Schnelle, eine Word-Datei, einen QR-Code für die Klasse.

Aufgabe 1: Ablaufverfolgung für n = 4 (Start: s = 0, i = 1):

```
i = 1: s = 0 + 2·1 − 1 = 1
i = 2: s = 1 + 3 = 4
i = 3: s = 4 + 5 = 9
i = 4: s = 9 + 7 = 16
```

Danach i = 5, Bedingung i ≤ 4 ist falsch, Ausgabe: 16.

Vermutung: Der Algorithmus gibt n^2 aus. Begründung: Pro Durchlauf wird die i-te ungerade Zahl (1, 3, 5, 7, ...) addiert. Die Summe der ersten n ungeraden Zahlen ist n^2. Anschaulich: Ein Quadrat der Seitenlänge i−1 wird durch einen winkelförmigen Rand aus 2i−1 Kästchen zu einem Quadrat der Seitenlänge i erweitert. (Algebraisch: $(i-1)^2 + 2i - 1 = i^2$.)

Aufgabe 2: Schleifenlose Lösung: START → Eingabe n → Prozess: s = n · n → Ausgabe s → ENDE. Es gibt keine Raute und keine Rückführung.

Vorteile: Nur ein Rechenschritt, unabhängig von der Größe von n (konstanter Aufwand); kürzer, übersichtlicher und weniger fehleranfällig.
Nachteile: Man muss die mathematische Gesetzmäßigkeit (Summe der ersten n ungeraden Zahlen = n^2) kennen und beweisen können. Diagramm X funktioniert dagegen auch, ohne dass man die Formel kennt.

Eine Schleife ist sinnvoll, wenn es keine einfache geschlossene Formel gibt, wenn jeder Zwischenschritt benötigt wird (z. B. Ausgabe aller Zwischenwerte) oder wenn Werte während des Ablaufs eingelesen oder geprüft werden müssen.

Aufgabe 3: Die Schleife endet nie (Endlosschleife), weil i nur ungerade Werte annimmt und 10 nie erreicht.

Aufgabe 4: Jede Zählschleife (for-Schleife) lässt sich in eine bedingte Schleife (while-Schleife) umformen, aber nicht jede bedingte Schleife lässt sich in eine Zählschleife mit fest vorgegebener Wiederholungszahl umformen. → **richtig**

Aufgabe 5: (a) Aufbau: START → Eingabe n → Raute 'n ≠ 1 ?'. Ja-Pfad: Raute 'n % 2 == 0 ?'; ja → Prozess 'n = n / 2', nein → Prozess 'n = 3·n + 1'; beide Pfade werden zusammengeführt → Ausgabe n → Rückführung zur ersten Raute. Nein-Pfad der ersten Raute: ENDE. (Optional kann n vor der Schleife einmal ausgegeben werden.)

(b) n = 6: 6 → 3 → 10 → 5 → 16 → 8 → 4 → 2 → 1. Ausgegeben werden (nach jedem Schritt): 3, 10, 5, 16, 8, 4, 2, 1 (bei Ausgabe des Startwerts zusätzlich 6 am Anfang). Das sind 8 Schleifendurchläufe.

(c) Ein Flussdiagramm beschreibt nur den Ablauf, nicht, ob die Abbruchbedingung für alle Eingaben erreicht wird. Bei ungeraden Zahlen wächst n zunächst (3n+1), bei geraden sinkt es – es gibt kein erkennbares Muster, das ein Ende garantiert. Man kann für einzelne Zahlen testen, dass die Folge bei 1 endet, aber für alle natürlichen Zahlen ist das bis heute unbewiesen (Collatz-Vermutung). Für ein Programm bedeutet das: Es ist nicht allgemein bewiesen, dass es immer terminiert.

Aufgabe 6: (a) z = 3: Die Schleife läuft dreimal (3 → 2 → 1 → 0). Dann ist z > 0 falsch, z < 0 ebenfalls falsch, Ausgabe: 'null'. z = 0: Die Schleife wird nicht betreten, Ausgabe: 'null'. z = −2: z > 0 ist falsch, z < 0 ist wahr, Ausgabe: 'negativ'.

(b) Fehler: Der Ja-Pfad der ersten Raute verändert z (z wird heruntergezählt), statt 'positiv' auszugeben. Dadurch gehen die Information über das ursprüngliche Vorzeichen verloren und jede positive Zahl wird fälschlich als 'null' gemeldet. Es wurde eine Schleife verwendet, wo eine einfache Verzweigung nötig ist.

(c) Korrektur: Statt 'z = z − 1' mit Rückführung wird im Ja-Pfad direkt 'positiv' ausgegeben und dann zu ENDE geführt (die Rückführung entfällt). Es entsteht eine verschachtelte Verzweigung: z > 0? → ja: 'positiv'; nein: z < 0? → ja: 'negativ', nein: 'null'.

Aufgabe 7: (a) START → Prozess 'v = 0' → Eingabe t → Prozess 'v = v + 1' → Raute 't ≠ g ?'. Ja-Pfad: Raute 't < g ?'; ja → Ausgabe 'zu klein', nein → Ausgabe 'zu groß'; beide Pfade zusammenführen → Rückführung zur Eingabe t. Nein-Pfad (t = g): Ausgabe 'Richtig nach v Versuchen' → ENDE. Es handelt sich um eine bedingte Schleife (while bzw. repeat-until), weil die Anzahl der Durchläufe vorher unbekannt ist. Der erste Tipp wird vor der Schleife bzw. am Schleifenanfang eingelesen, damit die Bedingung geprüft werden kann.

(b) Bei kluger Strategie rät man jeweils die Mitte des noch möglichen Bereichs und halbiert so mit jedem Tipp die Anzahl der Kandidaten: 100 → 50 → 25 → 13 → 7 → 4 → 2 → 1. Nach höchstens 7 Tipps ist der Bereich auf eine Zahl eingegrenzt, der siebte Tipp trifft also sicher ($2^7 = 128 \geq 100$, aber $2^6 = 64 < 100$). Es genügen also maximal 7 Versuche.

Aufgabe 8: Diagramm: START → Prozess 'versuche = 0; ok = falsch' → Raute 'versuche < 3 UND ok == falsch ?'. Ja-Pfad: Eingabe PIN → Prozess 'versuche = versuche + 1' → Raute 'PIN == richtig ?' (ja → Prozess 'ok = wahr'; nein → weiter) → Zusammenführung → Rückführung zur Schleifenraute. Nein-Pfad der Schleifenraute: Raute 'ok == wahr ?'; ja → Ausgabe 'Zugang erlaubt'; nein → Ausgabe 'Karte gesperrt'; danach ENDE.

Begründung: Die Schleife kann auf zwei verschiedene Arten enden – entweder wurde die richtige PIN eingegeben (Erfolg) oder die Versuche sind aufgebraucht (Misserfolg). Beide Abbruchgründe müssen in der Bedingung stehen: Die Schleife läuft nur weiter, solange noch Versuche übrig sind UND noch keine richtige PIN eingegeben wurde. Nach der Schleife muss man unterscheiden, welcher Grund zum Ende geführt hat, hier mit der Variable ok.

Aufgabe 9: Für 1 bis 5 sind beide Lösungen korrekt; Schüler B ist sogar sehr anschaulich und leicht nachvollziehbar. Schüler A ist aber bereits hier flexibler und kürzer.

Bei 1 bis 1000 wäre B unpraktikabel: 1000 Rechtecke, hoher Aufwand, hohe Fehleranfälligkeit. A muss nur die Grenze von 5 auf 1000 ändern, das Diagramm bleibt gleich groß.

Wenn die Obergrenze vom Nutzer eingegeben wird, ist die Lösung von B grundsätzlich unmöglich, weil die Anzahl der benötigten Ausgaben beim Zeichnen noch nicht bekannt ist. Nur eine Schleife (mit der Bedingung i ≤ Obergrenze) kann das lösen.

Fazit: Schleifen machen Algorithmen kurz, änderbar und für variable Eingaben verallgemeinerbar. Für sehr kleine, feste Wiederholungszahlen kann eine Aneinanderreihung aber verständlicher sein.

Aufgabe 10: 1 → C, 2 → A, 3 → B, 4 → D