EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Fachbereich Informatik

Programmiersprachen und Softwaretechnik

Prof. Dr. Klaus Ostermann

Julian Jabs
julian.jabs@uni-tuebingen.de

DENKEN VERSTEHEN LERNEN
Lehrerskript

Inhalt – WS 16

Tübingen, 9. Dezember 2016

Inhaltsverzeichnis

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Fachbereich Informatik

Programmiersprachen und Softwaretechnik

Prof. Dr. Klaus Ostermann

Julian Jabs
julian.jabs@uni-tuebingen.de

DENKEN VERSTEHEN LERNEN
Lehrerskript

Einführung – WS 16

Tübingen, 11. November 2016

Hinweis zur Grundlage des Materials

Alle Materialien dieses Curriculums basieren im Kern auf den “Unplugged”-Einheiten von “Course 2” des K-5 Curriculums von Code.org, verwendet unter der Creative Commons Attribution-NonCommercial-ShareAlike 4.0 Unported License.

Einführung

Grundidee

Das Herzstück der Informatik bildet etwas, das Algorithmus genannt wird. Das Wort “Algorithmus” mag sich nach etwas Kompliziertem anhören[1], es handelt sich dabei aber einfach um eine Liste von Anweisungen, denen jemand folgen kann um ein bestimmtes Ergebnis zu erreichen. Um eine stabile Grundlage für die weitere Informatik(aus)bildung der SchülerInnen zu schaffen, konzentriert sich dieses Curriculum darauf, ein sicheres Verständnis für Algorithmen aufzubauen. (Dieser Absatz ist im Wesentlichen eine Übersetzung der “Lesson Overview” der ersten Einheit von “Course 1” des Code.org K-5 Curriculums.)

Durch die Beschäftigung mit diesen grundlegenden algorithmischen Ideen werden wesentliche informatische Denkweisen (“Computational Thinking”) vermittelt. Ziel ist, dass die Bedeutung von “Computational Thinking” anhand dieser konkreten Beschäftigung mit Algorithmen klar wird. Dabei ist zu beachten, dass immer die wesentlichen gedanklichen Schritte bei Entwicklung und Ausführung (oder auch Debugging) der Algorithmen und Programme explizit herausgearbeitet werden. Anders herum gesagt: Es kommt nicht primär darauf an, dass die SchülerInnen *irgendwie* ein Programm schreiben können, dass die vorgegebene Aufgabe erfüllt. Viel wichtiger ist, dass sie verstehen, *warum* und in welchem Fall welche Vorgehensweise für die algorithmische Problemlösung geeignet ist. Die konkrete Implementierung innerhalb eines technologischen Rahmens zu erlernen, ist immer erst ein zweiter Schritt, dem das konzeptuelle Verständnis von Problem und Lösung vorhergeht.

[1] Für Interessierte: Die Bezeichnung “Algorithmus” geht auf die lateinische Form des Namens des persischen Gelehrten Muhammad ibn Musa al-Khwarizmi (ca. 780-850) zurück.

Aufbau des Curriculums

Das Curriculum ist in Unterrichtseinheiten aufgeteilt. Jede Einheit entspricht in etwa einer Dreiviertelstunde. Wir unterscheiden "Unplugged"-Einheiten, für die kein Computer benötigt wird, und Programmiereinheiten mit Computer. Prinzipiell sind die Einheiten aufgeteilt in (a) die des Grundgerüsts, die elementare algorithmische Konzepte vermittelt werden, und in (b) zusätzliche auf dem Grundgerüst aufbauende Einheiten. Die 7 "Unplugged-Einheiten" dieses Grundgerüsts sind:

1. Programmieren mit Kästchenpapier (C1)
2. Alltags-Algorithmen: Papierflieger (C2)
3. Schleifen (C3)
4. Staffel-Programmierung (C4)
5. Verzweigungen (C5)
6. Binäre Armbänder (C6)
7. Das große Ereignis (C7)

Dieses Grundgerüst wird dann mit weiteren Stunden ergänzt. In diesen wird entweder

- das in einer vorigen "Unplugged"-Einheit Gelernte am Computer ausprobiert, mithilfe einer graphischen Programmiersprache, oder
- eine Verbindung mit mathematischen Inhalten hergestellt, d.h. es werden elementare mathematische Konzepte und Praktiken algorithmisch betrachtet.

Vermittelte Konzepte

Dieser Abschnitt soll eine Übersicht über die Konzepte bieten, die neben allgemeineren Problemlösefähigkeiten in den einzelnen Unterrichtseinheiten vermittelt werden sollen. Es geht dabei nur um die übergeordneten Ideen, noch nicht um die feineren Aspekte und präzisen Lernziele jeder Einheit. (Hinweis 13.7.2016: Die Lernziele, die zu jeder Unterrichtsstunde aufgeführt sind, haben eher beschreibenden Charakter. Eine strukturiertere Fassung der Lernziele, die auch mit Standards abgeglichen ist, ist derzeit in Arbeit.) Die essentiellen behandelten algorithmischen Ideen, die den Aufbau von Algorithmen betreffen, sind:

- **Sequenz**
- **Schleife**
- **Verzweigung**
- (**Ereignisse**)[2]

[2] Die Vorstellung von "events" knüpft an einen weitreichenden theoretischen und praktischen Hintergrund an; insofern kann das Konzept durchaus als elementar angesehen werden (und es wird in Zukunft an Bedeutung gewinnen). Außerdem bieten sich viele "Alltags"-Zugänge zum Thema Ereignisse an.

Dazu kommt eine erste Annäherung an die Idee der **Datenstruktur**. (Dieser Gedanke wird dann in den mathematischen Transfer-Einheiten wieder aufgegriffen.)

Jede Unterrichtseinheit des Grundgerüsts hat zum Ziel, ein bestimmtes Konzept zu vermitteln. In der Regel ist dies eine algorithmische Idee; in einem Fall (C2) ist diese in einen Anwendungskontext gestellt, nachdem sie in der vorherigen Einheit (C1) vorgestellt wurde. In einer Einheit geht das übergeordnete Ziel über die bloße algorithmische Idee hinaus: C4 behandelt Debugging, also eine spezifische, aber besonders elementare Vorgehensweise im Zusammenhang mit Algorithmen bzw. Programmen. Eine Übersicht über diese Zuordnung bietet die Tabelle unten:

Einheit	Konzept	Anmerkung
C1	Sequenz	(+Alg. allgemein)
C2	Sequenz	+Anwendung
C3	Schleife	
C4	Debugging	Vorgehensweise
C5	Verzweigung	
C6	Datenstruktur	
C7	Ereignisse	

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Fachbereich Informatik

Programmiersprachen und Softwaretechnik

Prof. Dr. Klaus Ostermann

Julian Jabs
julian.jabs@uni-tuebingen.de

DENKEN VERSTEHEN LERNEN
Lehrerskript

Rezept: Problemlösen – WS 16

Tübingen, 8. Dezember 2016

Teilen Sie den SchülerInnen das Problemlösungs-Rezept (Schülerversion) aus und sprechen Sie mit Ihnen darüber. Wenn SchülerInnen um Hilfe fragen, können Sie ihnen die folgenden Suggestivfragen stellen.

Schritt 1: Verstehe die Aufgabe

- Verstehst du die Situation (was das Rätsel verlangt)?
- Kannst du das Problem in eigenen Worten beschreiben?
- Verstehst du den vorgegebenen Code und warum es vorgegeben ist? Was für eine Rolle spielt der Code?
- Weißt du, was das Ziel des Rätsels ist?
- Ist dieses Problem ähnlich wie ein anderes, das du schon gelöst hast?

Schritt 2: Mache einen Plan

Wähle eins oder mehrere aus:

- Können eine oder mehrere der folgenden Strategien eingesetzt werden?
  - Raten und überprüfen
  - Eine Karte zeichnen
  - Ein Bild malen
  - Nach einem Muster suchen
  - Mit einem vorher gelösten Rätsel vergleichen
  - Ein einfacheres Problem lösen
- Wie wäre es damit, ...

  - ...ein Diagramm zu zeichnen?
  - ...ein gleichbedeutendes Problem zu lösen?
  - ...Teilaufgaben zu identifizieren?
  - . . . rückwärts von der Aufgabenstellung aus zu arbeiten?

Schritt 3: Führe den Plan aus und verbessere ihn

- Hast du versucht, das Rätsel mithilfe deines Plans zu lösen?
- Wenn dein Plan fehlschlägt: Hast du dir die Rückmeldung bei den Fehlern angesehen? Modifiziere deinen Plan entsprechend.
- Hast du deine Strategie ausprobiert und wenn nötig geändert? Mache das oft und habe keine Angst davor, Lösungen auszuprobieren bevor du weiß dass sie perfekt sind.
- Hängst du am Problem fest? Mache für einen Moment etwas anderes (laufe ein wenig herum, schau weg vom Bildschirm/Arbeitsblatt,. . .). Denke an etwas anderes. Vllt. fällt dir die Lösung danach ein.
- Hast du schon mit anderen über das Problem gesprochen? Jemand hat vllt. einen Tipp oder kann dir sogar etwas erklären.

Schritt 4: Überprüfe dein Ergebnis

- Stimmt deine Lösung? Erfüllt deine Antwort alle Ziele? (z.B. Anzahl Blöcke, Verwendung eines bestimmten Blocks, . . .)
- Siehst du eine einfachere oder effizientere Lösung?
- Kannst du deine Lösung erweitern, so dass sie einem allgemeineren Muster folgt?

Nächste Seiten: Schülerversion (zum Austeilen)

DENKEN VERSTEHEN LERNEN
Rezepte und Tipps: Problemlösung (Schülerversion)

basiert auf "Course 2" des K-5 Curriculums von Code.org

1. September 2016

Diese Tipps werden dir helfen, wenn du beim Lösen der Aufgaben festhängst.

Schritt 1: Verstehe die Aufgabe

- Was verlangt die Aufgabe von dir?
- Kannst du über das Problem in deinen eigenen Worten sprechen?
- Wurde dir schon Code vorgegeben?
  - Was tut der Code?
  - Was glaubst du, wieso ist der Code da?
- Was ist das Ziel der Aufgabe?
- Hast du schon andere Aufgaben gelöst, die so ähnlich wie diese hier waren?

Schritt 2: Mache einen Plan

Wähle eins oder mehrere aus:

- Schreibe einen Algorithmus.
- Rate etwas und überprüfe es später.
- Male ein Bild von dem, was du tun willst.
- Versuche, ausgehend von dem was du bereits weißt rückwärts zu arbeiten.
- Löse immer nur einen kleinen Teil auf einmal.
- Vergleiche mit einer Aufgabe, die du bereits gelöst hast.

Schritt 3: Führe den Plan aus und verbessere ihn

- Hast du das Rätsel gelöst?
- Wenn nicht, behebe die Fehler, immer nur einen auf einmal.
- Teste deinen Plan nach jeder Änderung erneut.
- Wenn du langsam frustriert bist, atme tief durch, oder geh eine Minute vom Bildschirm weg. Wenn du zurückkommst siehst du vielleicht, was den Ärger verursacht hat!
- Stelle Fragen. Vielleicht kann dir einer deiner Freunde dabei helfen herauszufinden wo dein Plan schiefgeht.

Schritt 4: Überprüfe dein Ergebnis

- Löst deine Antwort die Aufgabe?
- Hast du alle vorgegebenen Ziele erfüllt?
- Jetzt, wo du einen Weg, die Aufgabe zu lösen, gefunden hast: Erkennst du einen einfacheren Weg, dies zu tun?
- Wenn du deine Lösung ein bisschen änderst, funktioniert sie dann für irgendeine andere Aufgabe?
- Könntest du deine Lösung erklären, um anderen zu helfen?

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Fachbereich Informatik

Programmiersprachen und Softwaretechnik

Prof. Dr. Klaus Ostermann

Julian Jabs
julian.jabs@uni-tuebingen.de

DENKEN VERSTEHEN LERNEN
Lehrerskript

Rezept: Debugging – WS 16

Tübingen, 8. Dezember 2016

Hier sind einige Hinweise aufgeführt, was Sie bei den einzelnen Schritten zu Fehlervermeidung und Debugging fragen und ansprechen können.

Fehlervermeidung

Schritt 1: Verstehe die Aufgabe

- Verstehst du die Situation oder die Aufforderung des Rätsels?
- Kannst du das Problem in deinen eigenen Worten ausdrücken?
- Verstehst du den vorgegebenen Code und warum er da ist? Was für eine Rolle spielt der Code?
- Weißt du, was das Ziel der Aufgabe (des Rätsels) ist?
- Ähnelt das Problem anderen, die du bereits gelöst hast?

Schritt 2: Auf Anweisungen achten & dein eigenes Tun kontrollieren

- Achte darauf, die Anweisungen einige Male durchzugehen, während du arbeitest. Es kann sein, dass du dabei etwas neues entdeckst sobald du die Aufgabe etwas mehr verstehst.
- Überprüfe deine Ergebnisse regelmäßig, um sicher zu gehen dass sich alles so verhält wie du es erwartest.

Schritt 3: Lass dir Zeit

- Wenn du eine Aufgabe hastig erledigst, wirst du eher Fehler machen die du hättest vermeiden können wenn du besser aufgepasst hättest.

Schritt 4: Schritt für Schritt

- Füge immer nur ein Element auf einmal hinzu, und achte darauf dass die Lösung immer noch funktioniert. Es wird viel schwieriger, Fehler zu finden, wenn du viele neue Dinge auf einmal hinzufügst.

Debugging

Schritt 1: Etwas geht schief!

- Achte genau auf deinen Fortschritt, damit du Fehler erkennst sobald diese passieren.

Schritt 2: Was hätte passieren sollen?

- Wenn du erkannt hast dass etwas falsch lief, hat deine Lösung vermutlich etwas anderes getan als was verlangt war.
  - Was tut deine Lösung?
  - Was sagt dir das?

Schritt 3: An welcher Stelle tritt der erste Fehler auf?

- Gehe Schritt für Schritt durch deine Lösung bis du die Stelle findest, an der der erste Fehler auftritt, dann behebe diesen Fehler.
- Gehe noch mal Schritt für Schritt durch um die nächste fehlerhafte Stelle zu finden, dann behebe diesen Fehler.
- Wiederhole das bis dein Programm funktioniert!

Schritt 4: Verborgene Bugs

- Wenn du deinen Fehler immer noch nicht finden kannst, versuche „Brotkrümel“ in deinem Programm zu hinterlassen. Du kannst Anweisungen an bestimmten Stellen setzen und sehen welche aktiviert werden und welche nicht. Das sollte dir die dringend benötigten zusätzlichen Informationen liefern.
- Wenn du immer noch festhängst, mache mal eine kleine Pause und komme dann zurück. Eine neue Perspektive kann Wunder bewirken!
- Manchmal hilft es auch, ein zusätzliches Augenpaar zu haben. Wenn dir die Ideen ausgehen, frag nach ob jemand mit dir zusammen einen Blick auf die Sache werfen kann.

Nächste Seiten: Schülerversion (zum Austeilen)

DENKEN VERSTEHEN LERNEN
Rezepte und Tipps: Debugging (Schülerversion)

basiert auf "Course 2" des K-5 Curriculums von Code.org

8. Dezember 2016

Diese Tipps werden dir dabei helfen, weiterzukommen wenn du festhängst.

Fehler vermeiden

- Lies die Anweisungen genau.
- Was ist das Ziel des Rätsels?
- Lass dir Zeit und nimm immer nur einen Schritt auf einmal.
- Kannst du über das Problem in deinen eigenen Worten sprechen?
- Wurde dir bereits Code vorgegeben?
  - Was macht der Code?
  - Was glaubst du, warum ist der Code wohl da?

Debugging

- Schau bei jedem Schritt nach Problemen.
- Beschreibe, was hätte passieren sollen.
- Beschreibe, was schiefgeht.
- Schau dir den Unterschied zwischen dem was passieren sollte und dem was passiert ist an. Siehst du da irgendwelche Hinweise, die dir helfen könnten das Problem zu beheben?
- Behebe immer nur eine Sache auf einmal und beschreibe dann wie sich das Ergebnis verändert hat.

- Versuche, “Brotkrümel” (breadcrumbs) in deinem Programm zu hinterlassen. Du kannst in deinen Code Hinweise einbauen (etwa kann dein Programm etwas “sagen”, mit dem “sage”-Block), damit du weißt wann welcher Teil des Programms abläuft.
- Versuche jede Einzelaufgabe für sich zu lösen und baue die Teile am Ende zusammen. So siehst du leichter, was jeder einzelne Teil tut.
- Probiere mindestens drei Wege aus, ein Problem zu beheben, bevor du um Hilfe fragst.
- Frage einen Freund. Vielleicht kann dir einer deiner Klassenkameraden helfen, herauszufinden wo dein Plan schiefgeht.

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Fachbereich Informatik

Programmiersprachen und Softwaretechnik

Prof. Dr. Klaus Ostermann

Julian Jabs
julian.jabs@uni-tuebingen.de

DENKEN VERSTEHEN LERNEN
Lehrerskript

Einheit C1 – WS 16

Tübingen, 29. November 2016

Übersicht

Indem sie sich gegenseitig “programmieren” ein Bild zu malen, beginnen die SchülerInnen zu verstehen, worum es beim Programmieren eigentlich geht. Zunächst weisen die Schüler sich gegenseitig dabei an, Kästchen so auszumalen, dass am Ende ein vorgegebenes Bild entsteht. Wenn Zeit übrig ist, kann das Ganze mit von den SchülerInnen selbst ausgedachten Bildern wiederholt werden.

Zusammenfassung

1. Einstieg (15 Min.)
   a) Neue Wörter
   b) *Einführung für Lehrer: Programmieren mit Kästchenpapier*
   c) Gemeinsames Üben: Programmieren mit Kästchenpapier
2. Partnerarbeit: Programmieren mit Kästchenpapier (15 Min.)
3. Abschluss (5 Min.)
   a) Kurzgespräch: Was haben wir gelernt?
   b) Wiederholung: Neue Wörter
4. Test: Programmieren mit Kästchenpapier (10 Min.)
5. Zusätzliche Lernangebote

Lernziele

SchülerInnen werden...

- ...verstehen, wie schwierig es sein kann, reale Probleme in Programme umzusetzen.

- ...lernen, dass Ideen klar und eindeutig erscheinen mögen, aber dennoch von einem Computer missverstanden werden können.
- ...üben, Ideen durch Codes und Symbole zu kommunizieren.

Materialien

Jeder Schüler benötigt die folgenden Materialien, die Sie zu Beginn der Stunde austeilen:

- Stift(e)
- 4x4-Kästchen-Arbeitsblätter (siehe Materialanhang)
- leeres Papier oder Karteikarten (für Programme)

Pro 2er-Gruppe wird benötigt (Austeilen vor Beginn der Partnerarbeit):

- das Partnerarbeitsblatt für Stunde 1 (siehe Materialanhang)

Dazu kommt das Testblatt (siehe Materialanhang), das Sie am Ende vor dem Test austeilen.

1 Einstieg (15 Min.)

1.1 Neue Wörter

Schreiben Sie die Begriffe und ihre Bedeutungen an die Tafel. Sprechen Sie sie vor und lassen Sie die Schüler wiederholen.

- **Algorithmus**: gesprochen Al - go - rith - mus
  "eine Liste von Schritten, denen man folgen kann um eine Aufgabe zu erfüllen"
- **Programm**: gesprochen Pro - gramm
  "ein Algorithmus der in etwas übersetzt wurde, das ein Computer ablaufen lassen kann"

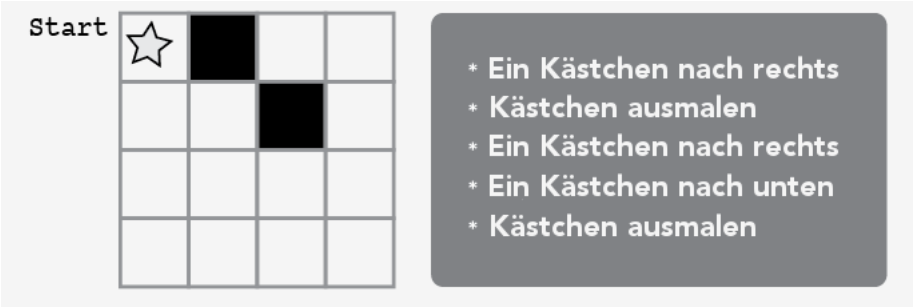
1.2 *Einführung für Lehrer: Programmieren mit Kästchenpapier*

In der folgenden Partnerarbeit leiten sich die Schüler gegenseitig an, ein Bild zu zeichnen ohne das ursprüngliche Bild zu kennen.

Für diese Übung verwenden wir 4x4-Kästchen-Blätter. Wir beginnen links oben und leiten die *Automatische Realisierungs-Maschine (ARM)* unseres Partners mit einfache Anweisungen an. Diese Anweisungen sind:

- ein Kästchen nach rechts bewegen
- ein Kästchen nach links bewegen
- ein Kästchen nach oben bewegen
- ein Kästchen nach unten bewegen
- das Kästchen ausmalen

Ein Beispiel: So würden wir einen Algorithmus schreiben, um einen Freund (der so tut, als ob er eine Zeichenmaschine wäre) anzuleiten, das leere Gitter so auszufüllen dass es wie das Bild unten aussieht:



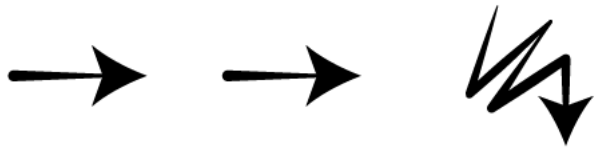
Das ist recht einfach, aber die Anweisungen für ein Bild wie das untere wären sehr viel Schreibarbeit:

Wir machen es uns viel leichter, wenn wir nur eine kleine Ersetzung vornehmen. Anstatt immer einen ganzen Satz für jede Anweisung zu schreiben, können wir einfach Pfeile verwenden:

In diesem Fall sind die Pfeilsymbole der “Programm”-Code und die Worte der “Algorithmus”. Das bedeutet dass wir als Algorithmus schreiben könnten:

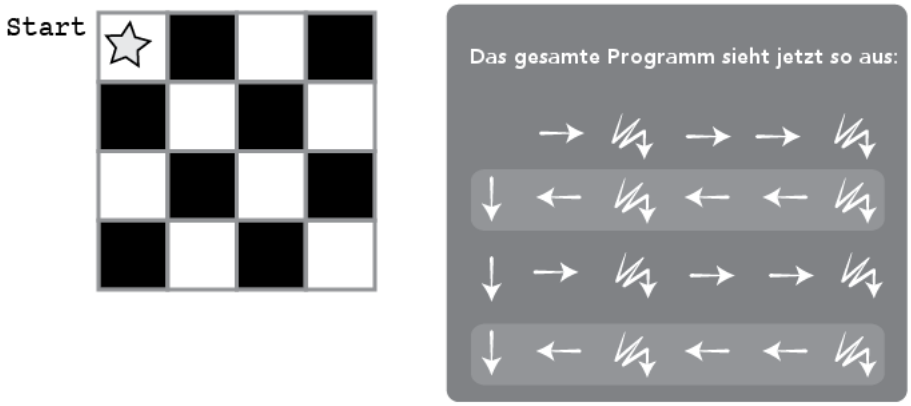
“ein Kästchen nach rechts bewegen, ein Kästchen nach rechts bewegen, das Kästchen ausmalen”

und das diesem Programm entsprechen würde:



Die Worte oben sind der “Algorithmus”, während die Pfeile unten ein “Programm” dafür sind. Der Algorithmus wurde in dieses Programm übersetzt, wie oben erklärt (siehe Symbolerklärungen).

Mit Pfeilen können wir den Programmcode des vorherigen Bildes jetzt viel einfach schreiben!

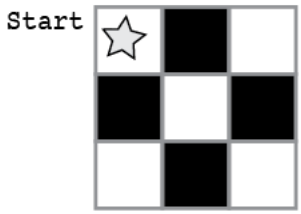


Versuchen Sie selbst, dem Programm mit dem Finger zu folgen.

1.3 Gemeinsames Üben: Programmieren mit Kästchenpapier

Führen Sie die Klasse in die Welt der Programmierung ein indem sie zunächst den Übersetzungsschlüssel Worte – Pfeile an die Tafel zeichnen oder projizieren.

Wählen Sie ein einfaches Bild wie das untere als Beispiel aus, und schreiben Sie sich einen passenden Algorithmus auf (siehe Einführung oben).



Das ist eine gute Gelegenheit, alle Symbole im Übersetzungsschlüssel einzuführen. Füllen Sie zunächst das Bild vor der Klasse aus – Kästchen für Kästchen – dann fordern Sie sie auf, Ihnen dabei zu helfen zu beschreiben, was Sie gerade getan haben. Wenn die Klasse dies bereits gut versteht: Nach ein paar Schritten können Sie die Klasse auch auffordern, Ihnen zu sagen, was der nächste Schritt sein soll. Sprechen Sie schließlich den vollständigen Algorithmus laut vor, danach können Sie diese verbalen Anweisungen in ein Programm übersetzen.

Ein Beispiel-Algorithmus:

“Nach rechts, ausmalen, nach rechts, nach unten, ausmalen, nach links, nach links, ausmalen, nach unten, nach rechts, ausmalen, nach rechts”

Einige in der Klasse werden bemerkt haben, dass da ein unnötiger Schritt ist, halten Sie sie aber zurück bis nach dieser Übungsphase.

Fragen Sie auch, ob jemandem aufgefallen ist, was wir zusätzlich zu den Anweisungen angeben müssen, damit wir den Algorithmus ausführen können: Bei welchem Kästchen wir anfangen (im Beispiel markiert mit einem Stern).

Gehen Sie mit der Klasse durch, wie der Algorithmus in ein Programm übersetzt werden kann.

Tipp: Dabei sollte klar werden, dass nur die vier Richtungsanweisungen und die Ausmalanweisung verwendet werden dürfen; man ist beschränkt auf diese Anweisungsmenge. Um das deutlich zu

machen, können Sie das etwa mit einem Computerspiel vergleichen, das mit der Tastatur gesteuert wird: Zur Bewegung darf man hier auch nur die vier Pfeiltasten verwenden.

An diesem Punkt könnte die Luft im Klassenzimmer vor Vorschlägen surren. Wenn die Klasse die wichtigsten Punkte verstanden hat, ist das ein guter Zeitpunkt, um mit ihr über andere Wege zu diskutieren, das gleiche Bild zu zeichnen. Wenn es noch Unklarheiten gibt, versuchen Sie alles nochmal mit einem anderen Beispiel zu erklären.

Wenn die Klasse Ihnen den Algorithmus zurufen kann und die richtigen Symbole für jeden Schritt definieren kann, sind sie bereit weiter zu machen. Abhängig von der Klasse und ihrem Alter können Sie sich zusammen ein komplizierteres Gitter probieren oder direkt zur Partnerarbeit mit dem 4x4-Arbeitsblatt übergehen.

2 Partnerarbeit: Programmieren mit Kästchenpapier (15 Min.)

Hierfür wird das Partnerarbeitsblatt benötigt. Teilen Sie die Schüler in Paare auf; die Partnerarbeit läuft nach den folgenden Schritten ab:

1. Jedes Paar wählt ein Bild vom Arbeitsblatt aus
2. Paar diskutiert mit welchem Algorithmus man dieses Bild malen kann
3. Paar übersetzt den Algorithmus in ein Programm mit Symbolen $\rightarrow$ Programm auf leeres Blatt/Karteikarte schreiben
4. Programme mit anderem Paar austauschen und das Bild anhand des Algorithmus malen
5. Anderes Bild wählen und von vorne beginnen

3 Abschluss (5 Min.)

3.1 Kurzgespräch: Was haben wir gelernt?

Diskutieren Sie mit den Schülern:

1. Was haben wir heute gelernt?
2. Was wenn wir die gleichen Pfeile für Programme verwenden, aber statt “Kästchen ausmalen” hätten wir “Baustein setzen”? Was könnten wir damit tun?
3. Was könnten wir noch programmieren, wenn wir einfach ändern was die Pfeile bedeuten?

3.2 Wiederholung: Neue Wörter

Fragen Sie: Für welche dieser Beschreibungen haben wir heute ein Wort gelernt?

- “Ein großer Papagei mit sehr langem Schwanz und schönen Federn”
- “Eine Liste von Schritten, denen man folgen kann um eine Aufgabe zu erfüllen”
- “Ein unglaublich stinkende Pflanze die nur einmal im Jahr blüht”

Welches dieser Dinge ist *am ehesten* wie ein “Programm”?

- “Ein Schuhkarton voller schöner Steine”
- “Zwölf rosa Blumen in einer Vase”
- “Noten für deinen Lieblingssong”

4 Test: Programmieren mit Kästchenpapier (10 Min.)

Teilen Sie die Tests für Stunde 1 aus (siehe Materialanhang). Die Schüler haben 10 Minuten, das Blatt zu bearbeiten.

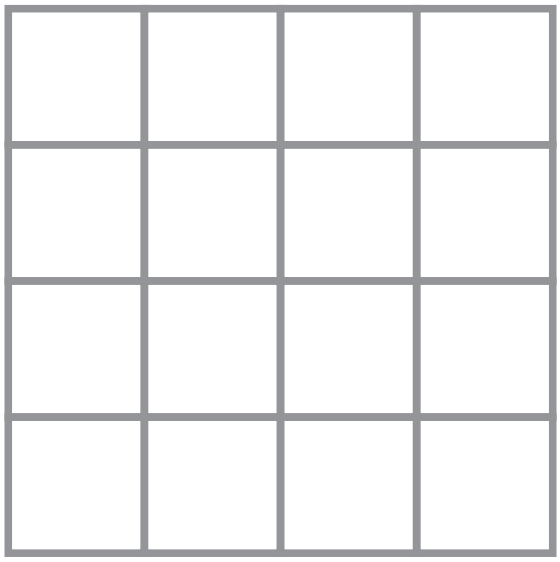
5 Zusätzliche Lernangebote

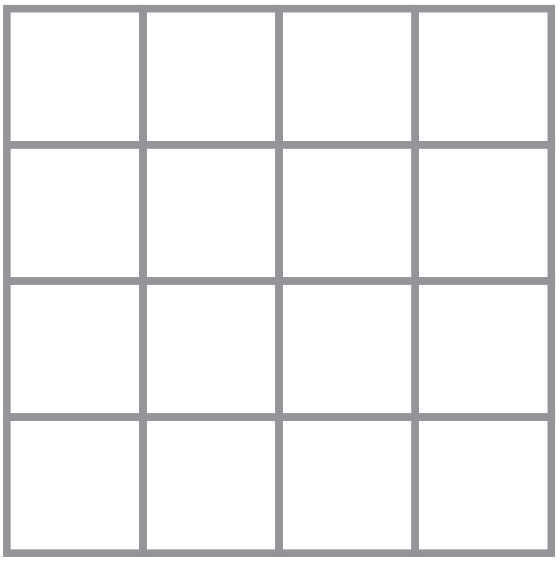
- Die Schüler denken sich andere Bilder aus (auf 4x4-Kästchen) und versuchen diese zu programmieren.
- Malen Sie ein 5x5-Bild an die Tafel und helfen Sie den Schülern, dieses zu programmieren.

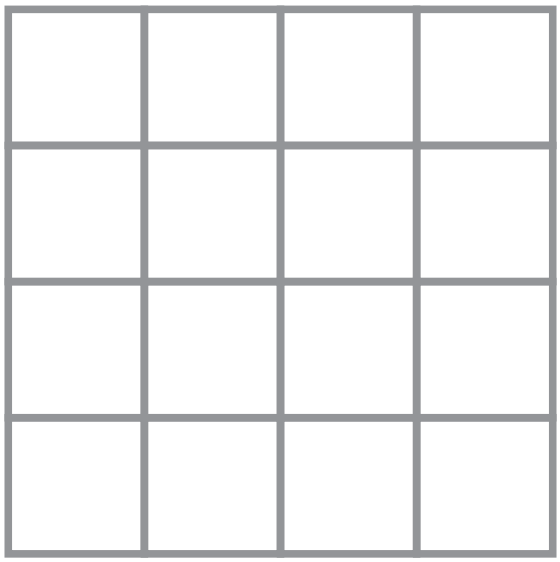
Materialanhang

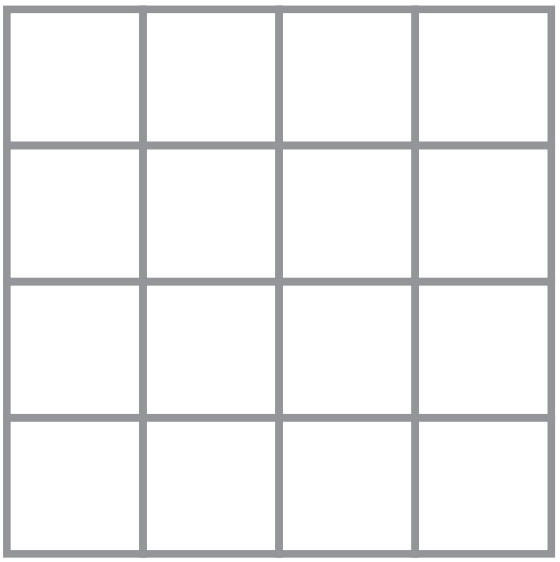
Es folgen in Reihenfolge:

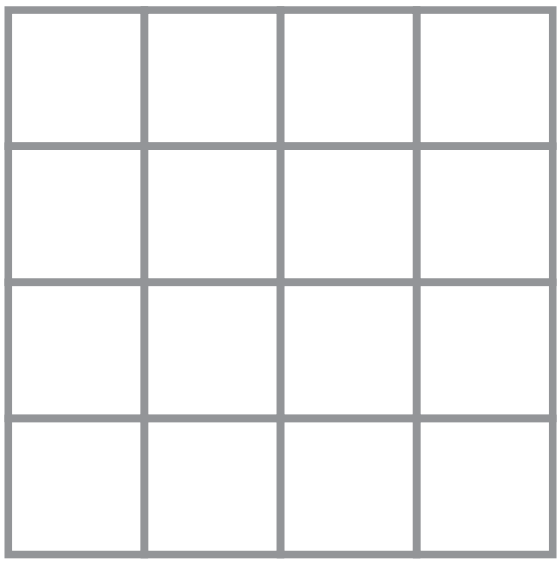
- das 4x4-Kästchen-Arbeitsblatt
- das Partnerarbeitsblatt für Stunde 1
- das Testblatt für Stunde 1

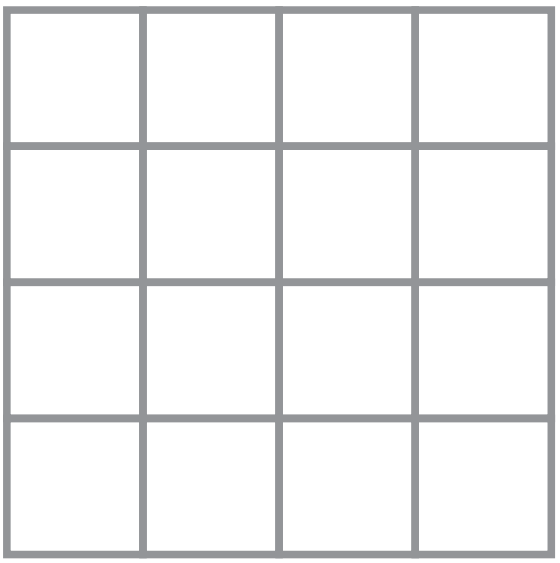












Name:_____ Datum:_____

Programmieren mit Kästchenpapier

Arbeitsblatt 4x4-Kästchen

Wähle eins der Bilder unten aus und lasse es von einem Freund/einer Freundin programmieren. Lass ihn/sie nicht sehen, welches du gewählt hast!

Schreibe ein Programm aus Pfeilen für eins der Bilder auf ein Stück Papier. Kann der andere das Bild zeichnen?

Verwende diese Symbole für ein Programm zum Zeichnen eines Bildes.

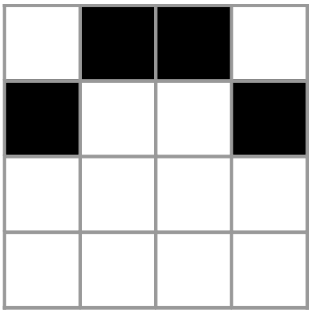


Bild 1

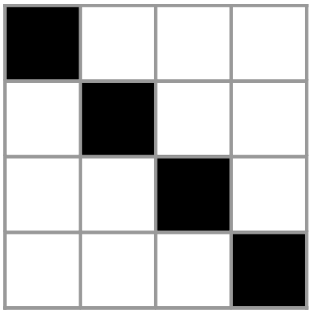


Bild 2

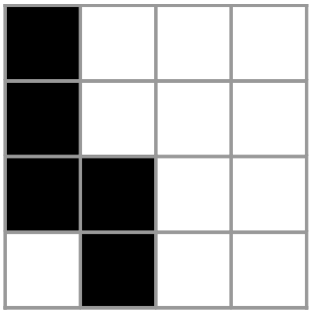


Bild 3

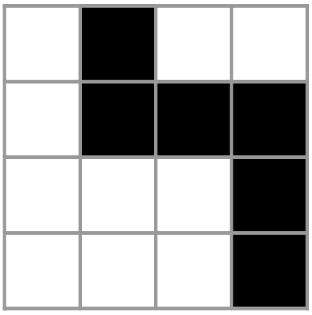


Bild 4

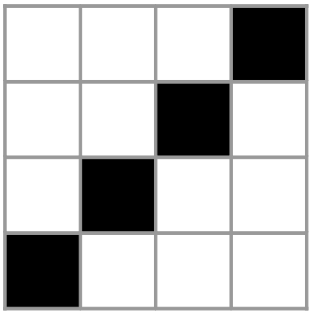


Bild 5

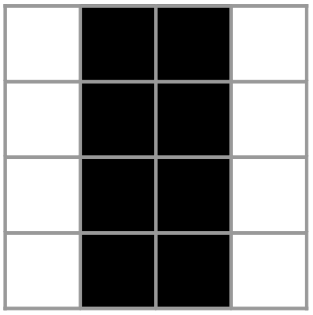


Bild 6

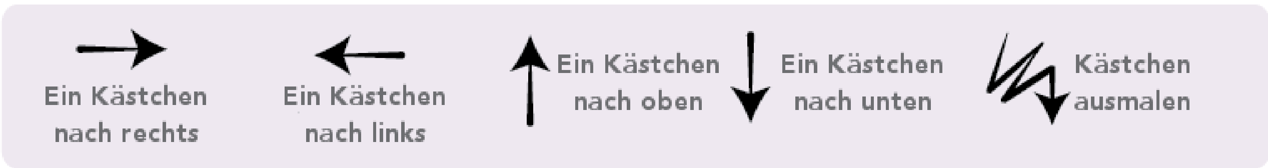
Name:______________________ Datum:____________

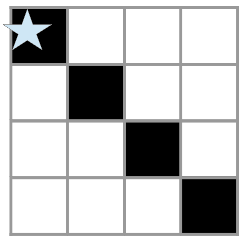
Programmieren mit Kästchenpapier

Übungsblatt

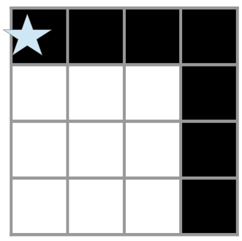
Heute habt ihr gelernt, wie man Algorithmen und Programme für Zeichnungen entwickeln kann. Im Unterricht habt ihr mit Anderen an Lösungen gearbeitet. Jetzt könnt ihr alleine mit den Bildern und Programmen unten üben.

Verwendet die Symbole unten um Programme zu schreiben, mit denen man die Bilder zeichnen kann. Beginnt immer links oben ☆.

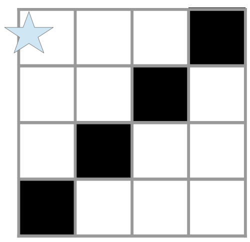




Schritt 1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20

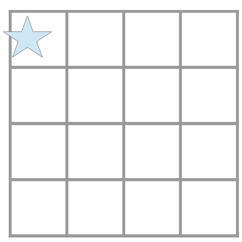


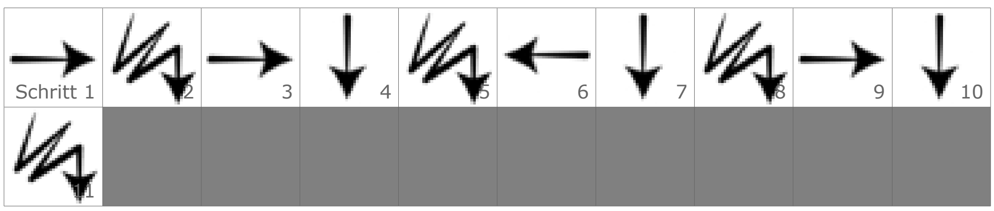
Schritt 1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20



Schritt 1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20

Jetzt schaut euch das Programm unten an. Malt das passende Bild.





EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Fachbereich Informatik

Programmiersprachen und Softwaretechnik

Prof. Dr. Klaus Ostermann

Julian Jabs
julian.jabs@uni-tuebingen.de

DENKEN VERSTEHEN LERNEN
Lehrerskript

Einheit C2 – WS 16

Tübingen, 29. November 2016

Übersicht

In dieser Unterrichtseinheit sollen die Schüler erkennen, wie sich das Konzept des Algorithmus im realen Leben wiederfindet. Dies soll am Beispiel des Baus von Papierfliegern deutlich werden. Ziel der Einheit ist es, dass die Schüler die Fähigkeit entwickeln, “Offline”-Situationen des Alltags in “Online”-Szenarien zu übersetzen und umgekehrt.

Zusammenfassung

1. Einstieg (15 Min.)
    a) Wiederholung
    b) Neue Wörter
    c) Algorithmen im Alltag
2. Einzel-/Gruppenarbeit: Papierflieger (15 Min.)
3. Abschluss (5 Min.)
    a) Kurzgespräch: Was haben wir gelernt?
4. Test: Alltags-Algorithmen (10 Min.)
5. Zusätzliche Lernangebote

Lernziele

Schüler werden...

- ...verschiedene Aktivitäten ihres Tages nennen können.
- ...üben, größere Aktivitäten in eine Reihe von kleineren aufteilen.
- ...üben, eine Reihe von Ereignissen in eine logische Reihenfolge zu bringen.

Materialien

Jeder Schüler benötigt die folgenden Materialien, die Sie zu Beginn der Stunde austeilen:

- Papier zum Falten des Papierfliegers
- Arbeitsblatt Papierflieger (siehe Materialanhang)
- Schere und evtl. Kleber

Dazu kommt das Testblatt (siehe Materialanhang), das Sie am Ende vor dem Test austeilen.

1 Einstieg (15 Min.)

1.1 Wiederholung

Rekapitulieren Sie zusammen mit den Schülern die letzte Stunde. Sie können dabei abwechselnd Fragen stellen und die Schüler ihre Antworten in kleinen Gruppen diskutieren lassen. Mögliche Fragen sind:

- Was haben wir letztes Mal gemacht?
- Was hättet ihr gerne noch gemacht?
- Sind euch nach der vorherigen Stunde noch Fragen eingefallen?
- Was hat euch an der vorherigen Stunde am besten gefallen?

1.2 Neue Wörter

Wiederholen Sie den Begriff des Algorithmus, der in dieser Einheit eine große Rolle spielt:

- **Algorithmus**: gesprochen Al - go - rith - mus
  "eine Liste von Schritten, denen man folgen kann um eine Aufgabe zu erfüllen"

1.3 Algorithmen im Alltag

- Fragen Sie die Schüler was sie heute morgen getan haben, um sich für die Schule fertig zu machen.
  - Schreiben Sie die Antworten an die Tafel.
  - Wenn möglich, schreiben Sie Nummern dazu um die Reihenfolge, in der die Schritte passieren, anzugeben. Wenn nötig, helfen Sie den Schülern die Schritte in eine logische Reihenfolge zu bringen. Weisen Sie sie außerdem darauf hin, wo die Reihenfolge wichtig ist und wo nicht.
- Führen Sie die Schüler in die Idee ein, dass es möglich ist, Algorithmen für alltägliche Aktivitäten zu schreiben. Nennen Sie einige Beispiele, wie Frühstück machen, Zähne putzen oder eine Blume zu pflanzen.
- Davon ausgehend, leiten Sie jetzt über zur folgenden Übung: Algorithmen für den Bau von Papierfliegern!

2 Einzel-/Gruppenarbeit: Papierflieger (15 Min.)

Diese Aufgabe kann von den Schülern einzeln oder in Gruppen bearbeitet werden. Sie verwenden dafür das Arbeitsblatt für Papierflieger, und bearbeiten es in den folgenden Schritten:

1. die einzelnen Schritte auf dem Arbeitsblatt ausschneiden
2. die richtigen sechs Schritte zum Bau des Papierfliegers auswählen
3. diese in der richtigen Reihenfolge auf ein Blatt Papier kleben
4. die fertigen Algorithmen mit einer anderen Person/Gruppe austauschen und den Papierflieger anhand des Algorithmus bauen

Hinweis 1: Wenn Sie denken, dass es für die Schüler zu schwierig ist, die richtigen Schritte auszuwählen: Machen Sie diesen Teil zusammen mit der Klasse vor dem Rest der Übung.
Hinweis 2: Falls Sie wegen der Verletzungsgefahr besorgt sind: Lassen Sie die Schüler die Spitze des Papierfliegers umknicken oder abreißen und mit Klebeband abkleben.

3 Abschluss

3.1 Kurzgespräch: Was haben wir gelernt?

Diskutieren Sie mit den Schülern:

1. Wer war in der Lage, einen Papierflieger anhand des Algorithmus seines Klassenkameraden zu bauen?
2. Fehlte ein Schritt in dieser Aufgabe?
   - Was hättest du hinzugefügt um den Algorithmus zu verbessern?
   - Was wäre, wenn der Algorithmus nur aus einem einzigen Schritt bestünde? Wäre das einfacher oder schwieriger? Was wenn es 40 Schritte wären?
3. Was hat euch an der Aufgabe am besten gefallen?

4 Test: Alltags-Algorithmen: Papierflieger (10 Min.)

Teilen Sie die Tests für Stunde 2 aus (siehe Materialanhang). Die Schüler haben 15 Minuten, das Blatt zu bearbeiten.

5 Zusätzliche Lernangebote

- Teilen Sie die Klasse in Teams auf.
- Jedes Team soll sich einige Schritte ausdenken, um eine Aufgabe zu erfüllen.
- Alle Teams treffen sich dann wieder, wobei ein Team allen seine Schritte mitteilt, ohne zu verraten um welche Aufgabe es geht.
- Die anderen sollen dann raten, um was für eine Aufgabe es sich handelt.

Materialanhang

Es folgen in Reihenfolge:

- das Arbeitsblatt für Stunde 2
- das Testblatt für Stunde 2

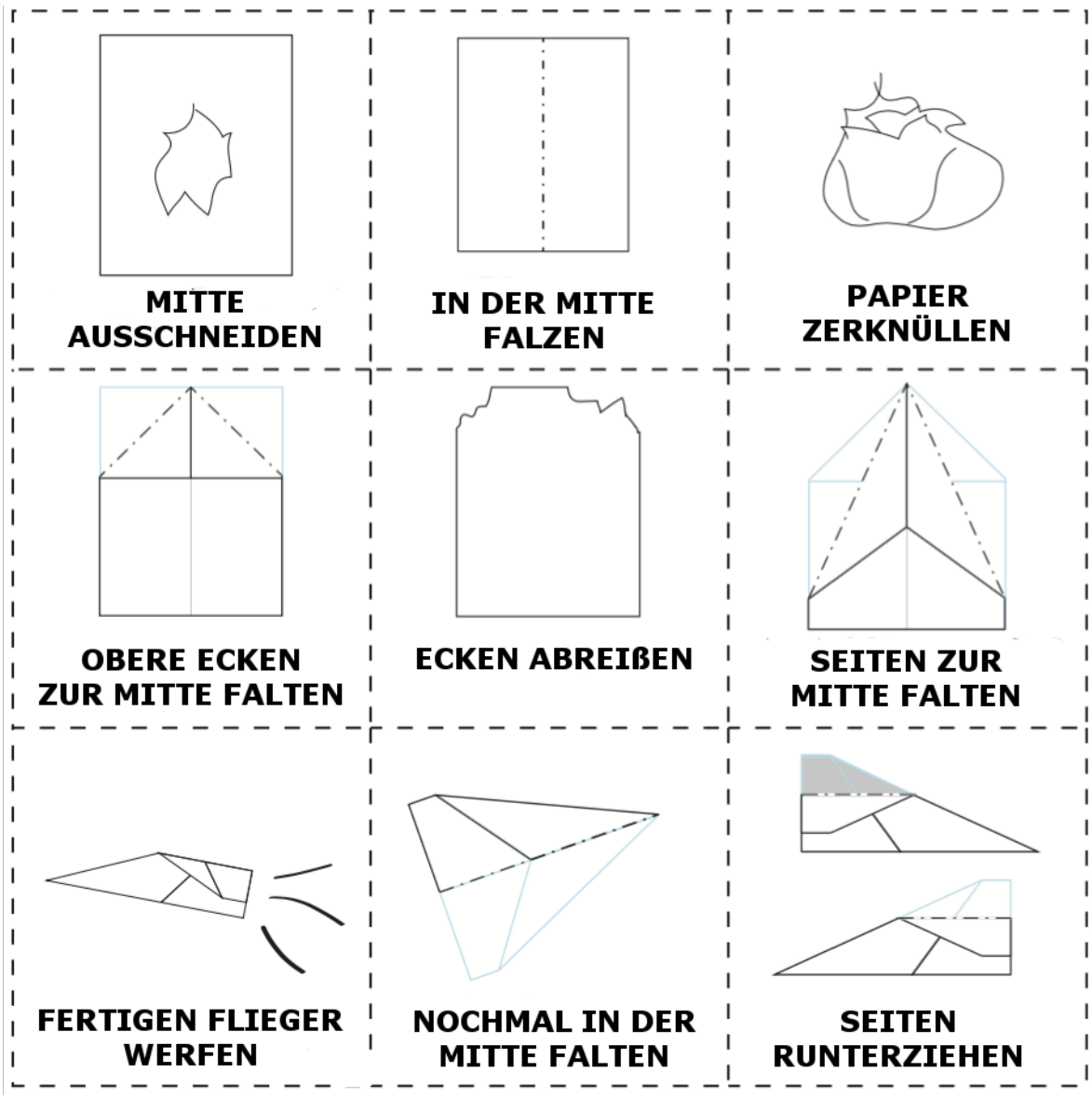
Name:_____ Datum:_____

Alltags-Algorithmen: Papierflieger

Arbeitsblatt Papierflieger

Man kann Algorithmen verwenden, um Dinge zu beschreiben die man jeden Tag macht. In dieser Übung entwickeln wir einen Algorithmus, um uns gegenseitig beim Basteln von Papierfliegern zu helfen.

Schneide unten die Schritte zum Basteln eines Fliegers aus. Klebe die sechs richtigen Schritte, in der richtigen Reihenfolge, auf ein extra Blatt Papier. Tausche den fertigen Algorithmus mit einer anderen Person oder Gruppe aus. Lass sie/ihn damit einen echten fliegenden Papierflieger basteln.



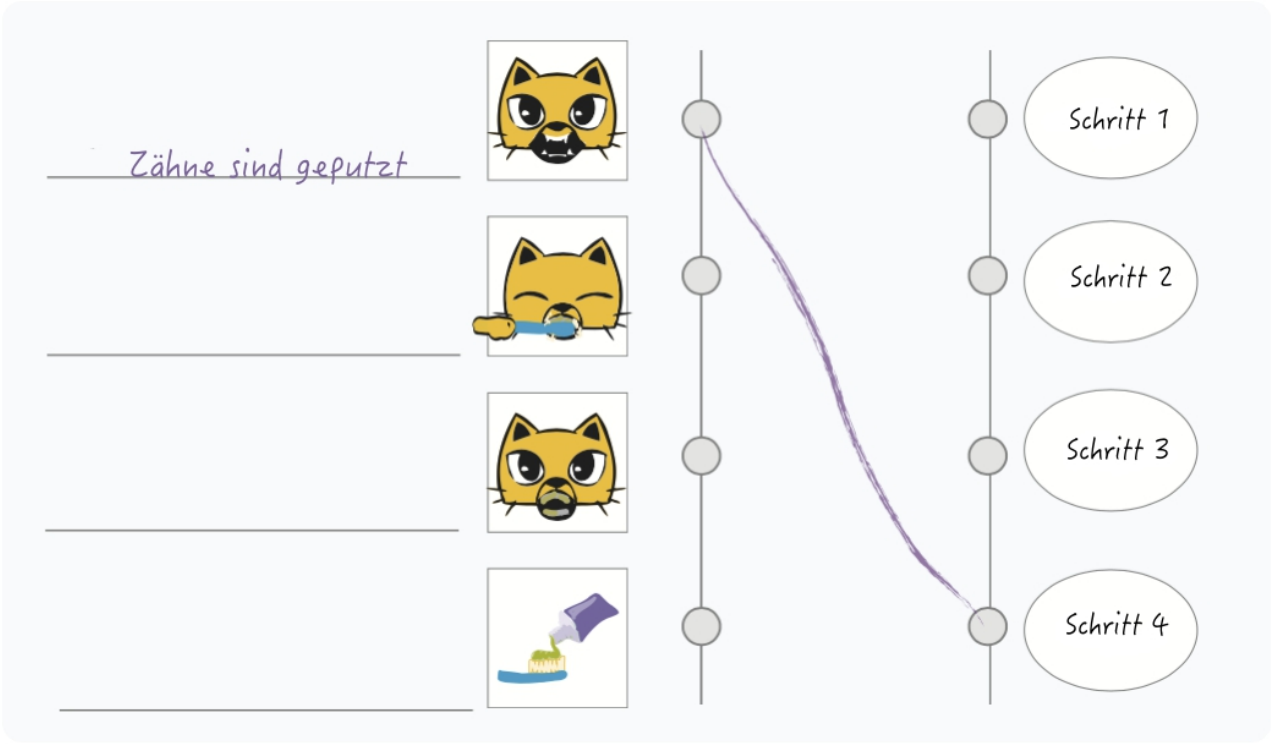
Name:_____ Datum:_____

Alltags-Algorithmen: Papierflieger

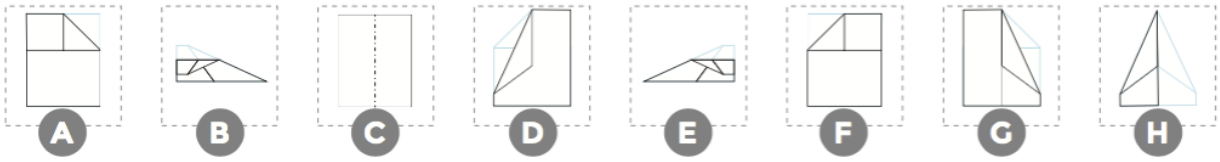
Übungsblatt

Ein Algorithmus ist eine Liste von Anweisungen um eine Aufgabe zu erfüllen. Wir befolgen jeden Tag Algorithmen, etwa wenn wir das Bett machen, Frühstück machen, oder sogar wenn wir uns morgens anziehen.

Die Bilder unten sind nicht in der richtigen Reihenfolge. Beschreibe zunächst auf der Linie links, was in jedem Bild passiert. Dann verbinde die Handlungen mit dem Schritt, in dem sie im Algorithmus vorkommen. Für die erste wurde das schon gemacht, als ein Beispiel.



Manchmal gibt es auch mehr als einen Algorithmus für die gleiche Tätigkeit. Wir können die Reihenfolge mancher Schritte ändern, ohne das Endergebnis zu ändern. Verwende die Buchstaben auf den Bildern unten um zwei Algorithmen zum Bau von Papierfliegern zu entwickeln.



ALGORITHMUS 1: ___ ___ ___ ___ ___ ___ ___ ___

ALGORITHMUS 2: ___ ___ ___ ___ ___ ___ ___ ___

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Fachbereich Informatik

Programmiersprachen und Softwaretechnik

Prof. Dr. Klaus Ostermann

Julian Jabs
julian.jabs@uni-tuebingen.de

DENKEN VERSTEHEN LERNEN
Lehrerskript

Einheit C2P – WS 16

Tübingen, 8. Dezember 2016

Übersicht

In dieser Reihe von Rätseln werden die Schüler auf dem grundlegenden Verständnis von Algorithmen in den beiden vorherigen “Unplugged”-Einheiten (Programmieren auf Kästchenpapier und Alltags-Algorithmen) aufbauen. Mit Charakteren aus dem Spiel “Angry Birds”^{TM} in den Hauptrollen werden die Schüler sequentielle Algorithmen entwickeln um einen Vogel durch das Labyrinth zum Schwein zu bewegen. Um dies zu erreichen, werden sie Blöcke zu einer linearen Sequenz – aus Bewegungen vorwärts geradeaus und Drehungen nach links und rechts – zusammenstecken.

Zusammenfassung

1. Einführung
2. Programmieren: Labyrinth: Sequenz
3. Zusätzliche Lernangebote

Lernziele

SchülerInnen werden...

- ...Bewegung als eine Reihe von Anweisungen ausdrücken.
- ...Bewegungsanweisungen als sequentielle Schritte in einem Programm anordnen.
- ...einen Algorithmus als Computerprogramm repräsentieren.
- ...zählen, wie oft eine Aktion ausgeführt werden soll und dies als Instruktionen in einem Programm repräsentieren.
- ...sich die Regeln des Pair Programming ins Gedächtnis rufen und diese anwenden.

- ...Pair Programming verwenden, um in Zusammenarbeit mit anderen Aufgaben mit und ohne Computer zu lösen.
- ...Situationen identifizieren, in denen die Regeln des Pair Programming nicht befolgt wurden.

1 Einführung

Fragen Sie Ihre Schüler, ob sie mit dem Spiel “Angry Birds”^{TM} vertraut sind. Erklären Sie, dass sie Programme schreiben werden, um einem Vogel zu helfen ein Schwein zu finden. Erläutern Sie:

- Damit der Vogel zum Schwein findet, müssen ihr eure Richtungsanweisungen in eine sehr bestimmte Reihenfolge bringen.
- Könnt Ihr die Rätsel mit so wenig Blöcken wie möglich lösen?

Hinweis: Einige Schüler haben vielleicht Probleme damit, den Vogel in die richtige Richtung zu drehen, insbesondere wenn der Vogel nicht nach oben ausgerichtet ist. Erinnern Sie die Schüler daran, dass, wenn wir “nach links drehen” oder “nach rechts drehen” sagen, dies aus der Perspektive des Vogels gemeint ist.

2 Programmieren: Labyrinth: Sequenz

Level 3 von Code.org Kurs 2

Während Ihre Schüler die Aufgaben bearbeiten, beobachten Sie wie sie den Weg für den Vogels planen. Identifizieren Sie verschiedene Strategien und bitten Sie die Schüler, diese mit der Klasse zu teilen. Das hilft den Schülern dabei zu erkennen, dass es viele Möglichkeiten gibt die Lösung dieser Probleme anzugehen. Sie können einige Beispiele auf dem Projektor zeigen und mit der Klasse durchgehen. Dabei können Sie einen Schüler bitten, dem Pfad auf dem Bildschirm zu folgen, während ein anderer die Anweisungen auf die Tafel/ein Whiteboard schreibt.

3 Zusätzliche Lernangebote

Eure eigenen Labyrinthe

Lassen Sie die Schüler in kleinen Gruppen Ihre eigenen Labyrinthe entwerfen und sich gegenseitig herausfordern, Programme zu schreiben um diese zu lösen. Noch mehr Spaß: Stellen Sie Labyrinthe in Lebensgröße zusammen, mit Schülern als Vogel und Schwein.

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Fachbereich Informatik

Programmiersprachen und Softwaretechnik

Prof. Dr. Klaus Ostermann

Julian Jabs
julian.jabs@uni-tuebingen.de

DENKEN VERSTEHEN LERNEN
Lehrerskript

Einheit C3 – WS 16

Tübingen, 29. November 2016

Übersicht

Mit Schleifen können auf praktische Weise Aktionen beschrieben werden die sich eine gewisse Zahl wiederholen. In dieser Einheit üben die Schüler eine Menge von Handlungen in eine einzelne Schleife zu übersetzen.

Zusammenfassung

1. Einstieg (15 Min.)
    a) Wiederholung
    b) Neue Wörter
    c) Wiederhole das nochmal!
2. Aktivität: Schleifen (15 Min.)
3. Abschluss (5 Min.)
    a) Kurzgespräch: Was haben wir gelernt?
4. Test: Schleifen (10 Min.)
5. Zusätzliche Lernangebote

Lernziele

Schüler werden...

- ...Aktionen des Lehrers wiederholen.
- ...ein Programm aus Bildern in einen Tanz umsetzen.
- ...üben, eine Reihe von Ereignissen in eine einzelne Schleife zu übersetzen.

Materialien

Als Lehrer benötigen Sie das Arbeitsblatt für die Aktivität. Die Schüler benötigen freien Raum, um sich zu bewegen. Dazu kommt das Testblatt (siehe Materialanhang), das Sie am Ende vor dem Test austeilen, und Stifte um dieses auszufüllen.

1 Einstieg (15 Min.)

1.1 Wiederholung

Rekapitulieren Sie zusammen mit den Schülern die letzte Stunde. Sie können dabei abwechselnd Fragen stellen und die Schüler ihre Antworten in kleinen Gruppen diskutieren lassen. Mögliche Fragen sind:

- Was haben wir letztes Mal gemacht?
- Was hättet ihr gerne noch gemacht?
- Sind euch nach der vorherigen Stunde noch Fragen eingefallen?
- Was hat euch an der vorherigen Stunde am besten gefallen?

1.2 Neue Wörter

Schreiben Sie den Begriff “Schleife” und seine Bedeutung an die Tafel. Sprechen Sie den Begriff vor und lassen Sie die Schüler wiederholen.

- **Schleife**:
  “die Tätigkeit, etwas wieder und wieder zu wiederholen”

1.3 Wiederhole das nochmal!

- Fragen Sie nach einem Freiwilligen und bitten Sie ihn aufzustehen.
  - Bitten Sie den Freiwilligen, um einen Tisch (Stuhl, Freund, ...) herumzulaufen.
  - Wenn er/sie damit fertig ist, bitten Sie ihn nochmal um das selbe, mit exakt den gleichen Worten wie zuvor.
  - Wenn er/sie fertig ist, wiederholen Sie nochmals die selben Worte.
  - Und nochmals.
- Fragen Sie: Wäre es nicht einfacher gewesen, dich zu bitten viermal um den Tisch zu gehen? Was wenn du es hättest zehnmal tun sollen?
- Erläutern Sie: Wenn ich will, dass du etwas zehnmal (viermal, zwanzigmal) wiederholst, nennt man das Schleife.
- Und weiter: Wenn ich bereits vorher weiß, wie oft du etwas tun sollst, ist es für uns beide einfacher wenn ich einfach sage: “Wiederhole das so-und-so oft.”
- Fragen Sie: Fallen euch andere Dinge ein, die man in Schleifen ablaufen lassen kann?

2 Aktivität: Schleifen (15 Min.)

“Heute tanzen wir Schleifen!”

Erklären Sie: Manchmal, wenn man weiß dass man etwas wiederholt tun möchte, kann es helfen vorher zu wissen, wie oft. Auf diese Weise kann man währenddessen im Kopf behalten, wie oft man noch wiederholen muss.

Ein Beispiel:

Wenn deine Mutter will, dass du ihren Lieblingssong wieder und wieder spielst, würde sie nicht sagen: “Bitte spiel meinen Song, bitte spiel meinen Song, bitte spiel meinen Song, bitte spiel meinen Song.”

Sie würde wohl eher sagen: “Bitte spiel meinen Song viermal.”

Nach dieser Einleitung können Sie jetzt mit der Aktivität anfangen (davor am besten Platz zum Bewegen schaffen):

1. Sehen Sie sich die Tanzbewegungen auf dem “Schleifen”-Arbeitsblatt (siehe Materialanhang) an.
2. Zeigen Sie der Klasse wie der gesamte Tanz bei voller Geschwindigkeit aussieht.
3. Zeigen Sie dann langsam jeden einzelnen Schritt und lassen Sie die Klasse die Schritte wiederholen.
4. Können die Schüler die Schleife finden? Fragen Sie außerdem:
   - Wie würde der Tanz aussehen wenn wir den Hauptteil nur zweimal wiederholen würden?
   - Und wenn wir den Hauptteil viermal wiederholen würden?
5. Können die Schüler noch etwas anderes im Tanz finden, für das eine Schleife hilfreich wäre?

3 Abschluss (5 Min.)

3.1 Kurzgespräch: Was haben wir gelernt?

Diskutieren Sie mit den Schülern:

- Denkt ihr dass es einfacher ist, auf dem Bildschirm mehr Bilder zu zeigen, oder die Anzahl zu ändern, wie oft wiederholt wird? Wäre eure Antwort dieselbe wenn das Bild 100-mal wiederholt würde?
- Könnten wir die gleichen Schleifen für andere Tanzbewegungen verwenden?
- Kennt ihr irgendwelche Tänze mit Wiederholungen (Schleifen)?
- Was hat euch heute am besten gefallen?
- Fehlte ein Schritt in dieser Aufgabe?

4 Test: Schleifen (10 Min.)

Teilen Sie die Tests für Stunde 5 aus (siehe Materialanhang). Die Schüler haben 10 Minuten, das Blatt zu bearbeiten.

5 Zusätzliche Lernangebote

Bewegend

- Geben Sie den Schülern Bildern von Tanzbewegungen oder anderen Aktionen die sie durchführen können. Lassen Sie die Schüler die Bewegungen anordnen und Schleifen hinzufügen, so dass sie ihren eigenen Tanz choreographieren.
- Die Schüler können dann ihre eigenen Tänze mit der übrigen Klasse teilen.

Transfer

- Suchen Sie nach einigen Videos von beliebten Tänzen die sich selbst wiederholen.
- Kann die Klasse die Schleifen finden?
- Versucht das Gleiche mit Songs!

Kästchen ausmalen mit Schleifen

Lassen Sie die Schüler Programme zum Ausmalen von Kästchen wie in Einheit C1 entwickeln, aber diesmal sollen sie auch Schleifen verwenden (z.B. “Tue das 3 mal: Gehe nach rechts, male Kästchen aus”).

Materialanhang

Es folgen in Reihenfolge:

- das Arbeitsblatt für Stunde 5
- das Testblatt für Stunde 5

Name:_____ Datum:_____

Die Wiederholung

Wiederhole diesen Teil 3 mal!


Klatschen


Klatschen


Klatschen


Hinterm Kopf


Hüfte


Hinterm Kopf


Hüfte


Klatschen


Klatschen


Klatschen


Linke hoch


Rechte hoch


Linke hoch


Rechte hoch


Klatschen


Klatschen


Klatschen

Dann mach das:


Bauch-Lachen

Name:_____ Datum:_____

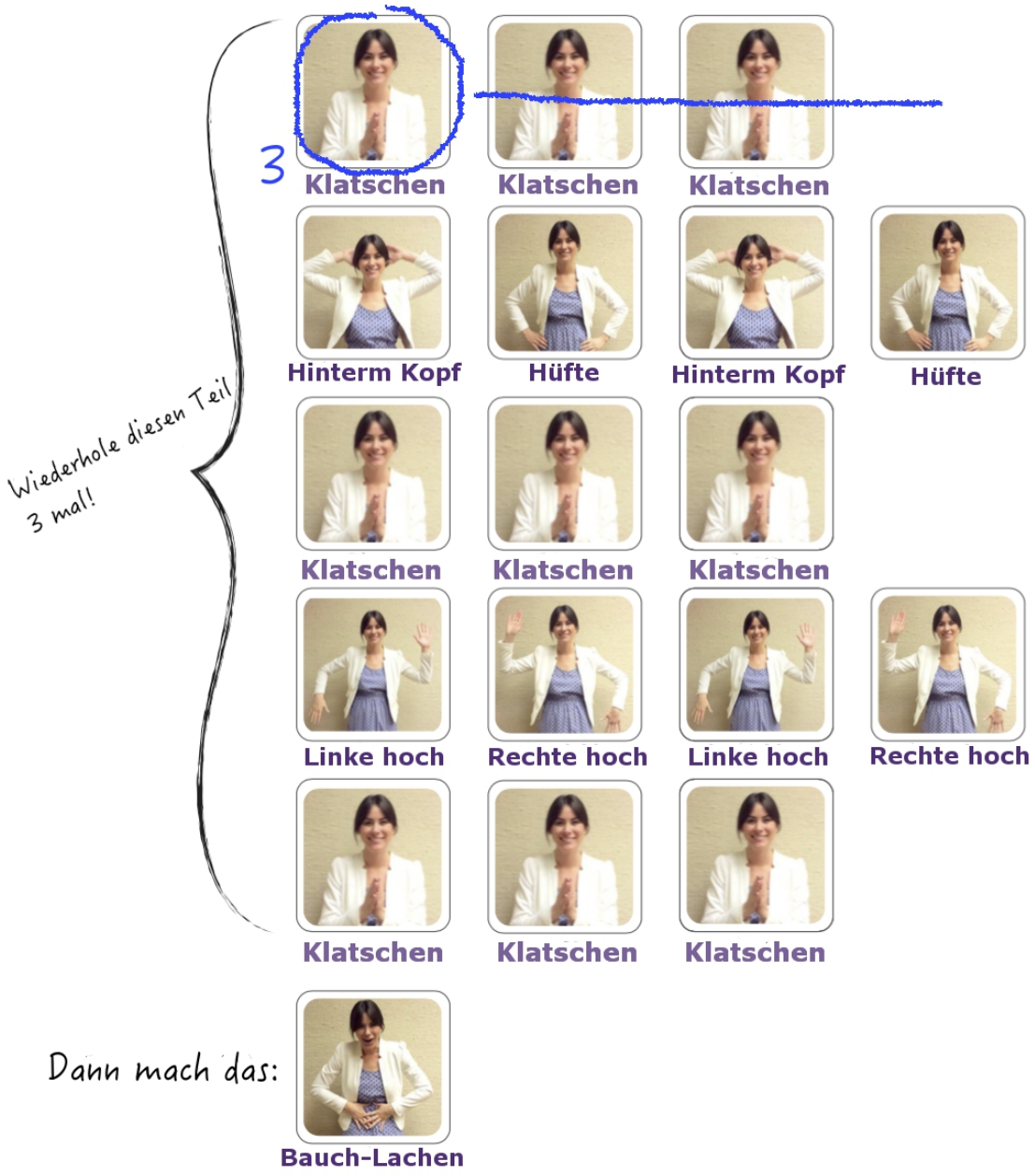
Schleifen

Übungsblatt

Schleifen können Platz sparen!

Was wäre wenn wir in den „Die Wiederholung"-Tanz unten mehr Schleifen einbauen wollen? Kannst du die Aktionen einkreisen, die wir in eine Schleife gruppieren können, und diejenigen durchstreichen, die wie dann nicht mehr brauchen? Schreibe eine Zahl neben jeden Kreis, damit wir wissen wie oft die Aktion wiederholt wird.

Die erste Zeile wurde schon bearbeitet.



EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Fachbereich Informatik

Programmiersprachen und Softwaretechnik

Prof. Dr. Klaus Ostermann

Julian Jabs
julian.jabs@uni-tuebingen.de

DENKEN VERSTEHEN LERNEN
Lehrerskript

Einheit C3P1 – WS 16

Tübingen, 9. Dezember 2016

Übersicht

Auf dem Konzept sich wiederholender Anweisungen aus der letzten Einheit (Schleifen) aufbauend, werden die Schüler in diesem Abschnitt Schleifen verwenden, um das Labyrinth effizienter zu durchqueren.

Zusammenfassung

1. Einführung
2. Programmieren: Labyrinth: Schleifen
3. Zusätzliche Lernangebote

Lernziele

SchülerInnen werden...

- ...Vorteile dafür identifizieren, eine Schleifenstruktur anstelle von manueller Wiederholung zu verwenden.
- ...ein Programm für eine bestimmte Aufgabe schreiben, das eine Schleife für einen einzelnen Befehl beinhaltet.
- ...eine lange Sequenz von Anweisungen in die kleinstmögliche wiederholbare Sequenz herunterbrechen.
- ...ein Programm für eine gegebene Aufgabe schreiben, das eine Schleife für eine Sequenz von Befehlen beinhaltet.
- ...eine Kombination von sequentiellen Befehlen und Befehlen in Schleifen einsetzen, um den Ausgang eines Labyrinths zu erreichen.

1 Einführung

Wiederholen Sie mit den Schülern die letzte Stunde (Einheit C3: Schleifen):

- Was sind Schleifen?
- Warum verwenden wir sie?

2 Programmieren: Labyrinth: Schleifen

Level 6 von Code.org Kurs 2

Während die Schüler die Aufgaben bearbeiten, lassen Sie sie herausfinden, wie viel weniger Blöcke sie bei Verwendung einer Schleife benötigen, verglichen damit wie viele sie ohne Schleife benötigen würden.

3 Zusätzliche Lernangebote

Wenn Sie diese noch nicht ausprobiert haben, können Sie hier die zusätzlichen Angebote für Einheit C3 (Schleifen) nutzen.

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Fachbereich Informatik

Programmiersprachen und Softwaretechnik

Prof. Dr. Klaus Ostermann

Julian Jabs
julian.jabs@uni-tuebingen.de

DENKEN VERSTEHEN LERNEN
Lehrerskript

Einheit C3P2 – WS 16

Tübingen, 9. Dezember 2016

Übersicht

Im letzten Abschnitt haben die Schüler Schleifen verwendet um einfache Bewegungen zu wiederholen. Jetzt werden sie Schleifen von Aktionen dazu nehmen, die einer Biene helfen, mehr Nektar zu sammeln und mehr Honig zu produzieren.

Zusammenfassung

1. Einführung
2. Programmieren: Biene: Schleifen
3. Zusätzliche Lernangebote

Lernziele

SchülerInnen werden...

- ...ein Programm für eine bestimmte Aufgabe schreiben, das eine Schleife für einen einzelnen Befehl beinhaltet.
- ...identifizieren, wann eine Schleife verwendet werden kann um eine wiederholte Aktion zu vereinfachen.
- ...eine Kombination von sequentiellen Befehlen und Befehlen in Schleifen einsetzen, um Bewegungen und Aktionen auszuführen.

1 Einführung

An diesem Punkt haben die SchülerInnen Schleifen bereits im Kontext des Labyrinth-Levels eingesetzt. Dieser Abschnitt konzentrierte sich auf Schleifen für Bewegungsanweisungen. Fragen Sie:

- Was gibt es für andere Elemente in unseren Programmen, bei denen Schleifen nützlich wären?
- Was denkt ihr, wie könnten wir die Schleifen einsetzen, um die Bienen-Programme effizienter zu machen.

2 Programmieren: Biene: Schleifen

Level 8 von Code.org Kurs 2

Wenn die Schüler eine Schleife einsetzen um eine Aktion (wie Nektar sammeln) zu wiederholen, ermuntern Sie sie dazu, über die Bewegungen vor und nach der Aktion nachzudenken: Können diese mit in die Schleife genommen werden?

3 Zusätzliche Lernangebote

Wenn Sie diese noch nicht ausprobiert haben, können Sie hier die zusätzlichen Angebote für Einheit C3 (Schleifen) nutzen.

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Fachbereich Informatik

Programmiersprachen und Softwaretechnik

Prof. Dr. Klaus Ostermann

Julian Jabs
julian.jabs@uni-tuebingen.de

DENKEN VERSTEHEN LERNEN
Lehrerskript

Einheit C4 – WS 16

Tübingen, 9. Dezember 2016

Übersicht

Nach einer kurzen Wiederholung von Einheit C1 (Programmieren mit Kästchenpapier) wird diese Stunde zu einem Wettlauf gegen die Zeit: Die Schüler arbeiten in Teams um ein Programm zu entwickeln, Anweisung für Anweisung.

Zusammenfassung

1. Einstieg (15 Min.)
    a) Wiederholung
    b) Neue Wörter
    c) Wiederholung Einheit C1
2. Aktivität: Staffel-Programmierung (15 Min.)
3. Abschluss (5 Min.)
    a) Kurzgespräch: Was haben wir gelernt?
4. Test: Debugging (10 Min.)
5. Zusätzliche Lernangebote

Lernziele

Schüler werden...

- ...üben, Ideen durch Codes und Symbole zu kommunizieren.
- ...im Team arbeiten um eine Aufgabe zu erfüllen.
- ...die Arbeit ihrer Teamkollegen überprüfen, um ein richtiges Ergebnis sicherzustellen.

Materialien

Jeder Schüler benötigt die folgenden Materialien, die Sie zu Beginn der Stunde austeilen:

- Stift(e)
- leeres Papier oder Karteikarten (für Programme)

Dazu kommt das Testblatt (siehe Materialanhang), das Sie am Ende vor dem Test austeilen.
Für die Aktivität wird ein großer offener Raum benötigt (etwa draußen oder in der Turnhalle) und die Bilder, für die dann Programme entwickelt werden sollen (siehe Materialanhang).

1 Einstieg (15 Min.)

1.1 Wiederholung

Rekapitulieren Sie zusammen mit den Schülern die letzte Stunde. Sie können dabei abwechselnd Fragen stellen und die Schüler ihre Antworten in kleinen Gruppen diskutieren lassen. Mögliche Fragen sind:

- Was haben wir letztes Mal gemacht?
- Was hättet ihr gerne noch gemacht?
- Sind euch nach der vorherigen Stunde noch Fragen eingefallen?
- Was hat euch an der vorherigen Stunde am besten gefallen?

1.2 Neue Wörter

Schreiben Sie den Begriff “Debugging” und seine Bedeutung an die Tafel. Sprechen Sie den Begriff vor und lassen Sie die Schüler wiederholen. Es gibt hier keine deutsche Übersetzung in einem Wort: Debugging bedeutet in etwa “Fehler finden/beheben”. Sie können die Klasse aber auch darauf hinweisen, dass “bug” Käfer bedeutet und eine scherzhafte Bezeichnung für Fehler in Algorithmen oder Programmen ist; “debugging” bezeichnet entsprechend das Beseitigen von Käfern oder eben Fehlern.

- **Debugging**: Die - bagg - ging
  “Probleme in einem Algorithmus finden und beheben”

1.3 Wiederholung Einheit C1

Erinnern Sie die Klasse an Einheit C1 (Programmieren mit Kästchenpapier) am Anfang des Kurses:

- Beim “Programmieren mit Kästchenpapier” haben wir unseren Teampartner mit Pfeilen angeleitet, ein Bild zu malen. Wir haben ihn also wie eine Art Maschine gesteuert.
- Hier ist nochmal ein Bild und ein Programm mit dem man dieses malen kann (bereits bekannt oder neu).
- Wir werden heute etwas ähnliches tun, aber anstatt einander wie Maschinen zu steuern, werden wir zusammenarbeiten um Symbol für Symbol ein Programm zu entwickeln.

2 Aktivität: Staffel-Programmierung (15 Min.)

Nach dieser Erinnerung an Einheit C1 werden wir jetzt aktiv: Wir schreiben wieder ein Programm das ein Bild beschreibt, aber dieses Mal in Staffelteams, Symbol für Symbol. Die Regeln für dieses Spiel lauten:

1. Die Schüler treten in Teams der Größe 3-5 an.
2. Jedes Team reiht sich als Staffel auf.
3. Wählen Sie ein zu beschreibendes Bild aus. Platzieren Sie pro Team eine Kopie des Bildes auf der anderen Seite der Halle/des Feldes, dem Team gegenüber.
4. Der erste Schüler in der Reihe rennt mit einem leeren Blatt oder einer Karteikarte zum Bild und schreibt ein erstes Symbol des zugehörigen Programms auf.
5. Der Schüler rennt zurück, gibt das Programm an den nächsten in der Reihe, und stellt sich hinten wieder an.
6. Der nächste Schüler in der Reihe rennt ebenfalls zum Bild. Er betrachtet das Bild und ergänzt entweder den Algorithmus um ein Symbol oder debuggt das Programm wenn ein Symbol fehlerhaft ist, in dem er das Symbol durchstreicht.
7. Der Schüler rennt zurück, übergibt an den nächsten, und das geht dann so weiter bis eine Staffel ihr Programm fertig hat.
8. Die Staffel, die als erstes fertig ist, gewinnt!

Hinweise:

- Aus jeder Staffel darf immer nur eine Person gleichzeitig beim Bild sein.
- Das Vorgehen in der Gruppe zu diskutieren ist erlaubt, auch um zu planen, wer was schreibt wenn er beim Bild ist.
- Wenn ein Schüler das Programm debuggt indem er eine falsche Anweisung durchstreicht, muss er auch den Rest des Programms nach dieser Anweisung streichen. Dies zählt als ein Durchgang. Der nächste Schüler in der Staffel setzt die Arbeit am Programm dann nach der letzten *korrekten* Anweisung fort.

3 Abschluss (5 Min.)

3.1 Kurzgespräch: Was haben wir gelernt?

Diskutieren Sie mit den Schülern:

- Was haben wir heute gelernt?
- Was wenn wir bei jedem Durchgang fünf Pfeile hinzufügen dürften?
  - Wie wichtig wäre es, unsere eigene Arbeit und die der vorherigen Programmierer zu debuggen?
  - Wie ist es bei 10 Pfeilen?

  - Und wie bei 10 000? Wäre es wichtiger oder weniger wichtig?
- Macht es die Arbeit leichter oder schwieriger, wenn mehrere Leute am selben Programm arbeiten?
- Denkt ihr dass Leute mehr Fehler oder weniger Fehler machen wenn sie in Eile sind?
- Wenn du einen Fehler findest, musst du das gesamte Programm wegschmeißen und von vorn anfangen?

4 Test: Debugging (10 Min.)

Teilen Sie die Tests für diese Einheit aus (siehe Materialanhang). Die Schüler haben 10 Minuten, das Blatt zu bearbeiten.

5 Zusätzliche Lernangebote

5.1 Papier herumreichen

- Wenn Sie keine Zeit oder keinen Raum für die Staffelaktivität haben: Lassen Sie die Schüler das Papier mit dem Programm in ihrer Tischgruppe herumreichen und der Reihe nach mit Pfeilen ergänzen oder debuggen.

5.2 Ausfüllen und Weitergeben

- Teilen Sie die Klasse in Gruppen ein. Malen Sie pro Gruppe ein Bild mit so viel ausgefüllten Kästchen wie es Schüler in der Gruppe gibt.
- Jeden Schüler soll so viele Pfeile fürs Programm schreiben wie nötig um *ein* Kästchen auszufüllen, und das Programm dann weitergeben.

5.3 Gemeinsames Debuggen

- Malen Sie ein Bild an die Tafel.
- Jeder Schüler soll ein Programm für dieses Bild schreiben.
- Dann soll er das Programm mit seinem Nebensitzer austauschen und die beiden sollen gegenseitig ihre Programme debuggen.
- Der Nebensitzer soll den ersten falschen Schritt einkreisen und das Programm zurückgeben.
- Dann haben die Schüler die Möglichkeit, ihr eigenes Programm nochmal zu überprüfen und zu debuggen.
- Fragen Sie nach einem Freiwilligen, dessen Programm die Klasse sehen darf.
  - Wie viele Schüler haben das gleiche Programm?
  - Hat jemand etwas anderes?

Bug einbauen

- Wählen Sie ein Bild und ein Programm, das dieses zeichnet.
- Lassen Sie einen Schüler einen Bug in das Programm einbauen.
- Können die anderen den Bug finden?

Materialanhang

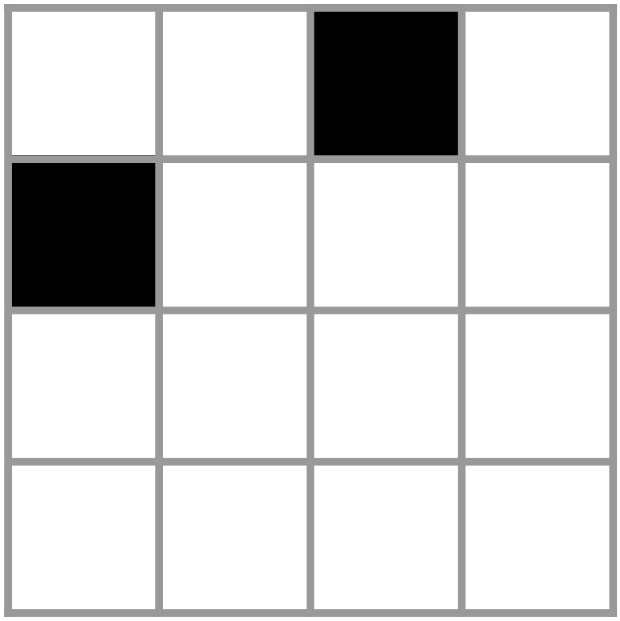
Es folgen in Reihenfolge

- die 4x4-Kästchen-Arbeitsblätter
- das Testblatt

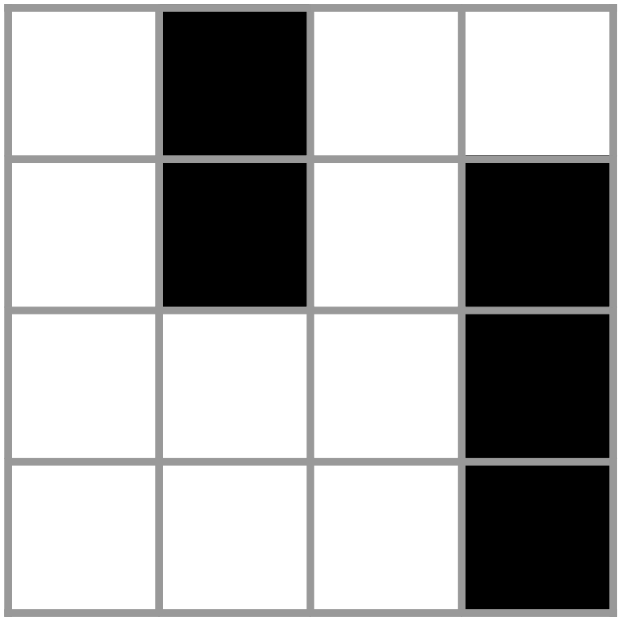
Staffel-Programmierung

Bilder für Programme

Staffel-Bild 1



Staffel-Bild 2



Name:_____ Datum:_____

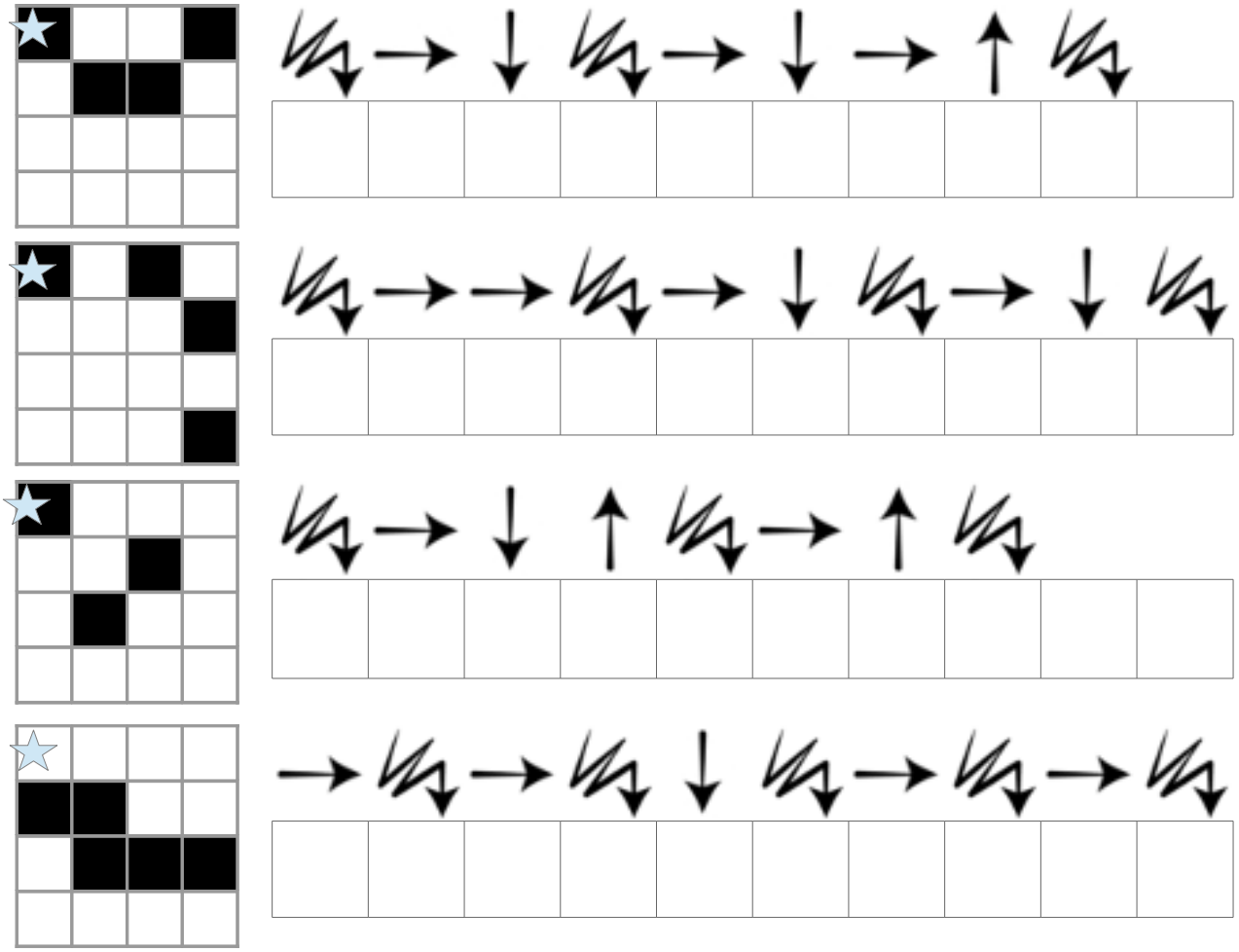
Debugging

Übungsblatt

Wenn ihr in Gruppen programmiert, macht manchmal jemand einen Fehler von dem alle betroffen sind.

Jemand hat bereits Programme für die Bilder unten geschrieben, aber jedes Programm hat einen Fehler! Finde heraus, wie die Programme aussehen *sollten* und umkreise den Fehler in jedem Programm. Dann malt das richtige Symbol in das Kästchen darunter.

Jedes Programme verwendet die Symbole unten um Bilder zu zeichnen und beginnt immer links oben ☆.



EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Fachbereich Informatik

Programmiersprachen und Softwaretechnik

Prof. Dr. Klaus Ostermann

Julian Jabs
julian.jabs@uni-tuebingen.de

DENKEN VERSTEHEN LERNEN
Lehrerskript

Einheit C4P – WS 16

Tübingen, 8. Dezember 2016

Übersicht

Beim Programmieren lernen ist Debugging ein wesentliches Element. In dieser Einheit werden die Schüler auf Rätsel stoßen, die falsch gelöst wurden. Sie werden dann den existierenden Code durchgehen müssen um Fehler zu finden, was falsche Schleifen sowie fehlende, überflüssige oder falsch angeordnete Blöcke beinhaltet.

Zusammenfassung

1. Einführung
2. Programmieren: Biene: Debugging
3. Zusätzliche Lernangebote

Lernziele

SchülerInnen werden...

- ...vorhersagen, wo ein Programm fehlschlagen wird.
- ...ein vorher existierendes Programm modifizieren, um Fehler zu beheben.
- ...einen Algorithmus identifizieren, der fehlschlägt wenn seine Schritte in der falschen Reihenfolge sind.
- ...über den Prozess des Debuggings auf altersgemäße Weise reflektieren.

1 Einführung

Lassen Sie die Schüler über Probleme nachdenken, die sie im Alltag lösen müssen:

- Wie repariert ihr etwas, das nicht funktioniert?
- Folgt ihr einer bestimmten Reihenfolge von Schritten?
- Die Rätsel in diesem Abschnitt wurden schon für euch gelöst (juhu!), aber sie scheinen nicht richtig gelöst worden zu sein (buuh!).
- Wir nennen die Probleme in diesen Programmen “Bugs” (deutsch: Käfer, siehe die Erläuterung aus der letzten Stunde (Debugging)), und es wird euer Job sein, sie zu “debuggen”.

2 Programmieren: Biene: Debugging

Level 10 von Code.org Kurs 2

Während die Schüler die Aufgaben bearbeiten, beobachten Sie wie sie nach Bugs suchen. Identifizieren Sie verschiedene Strategien und bitten Sie die Schüler, diese der ganzen Klasse mitzuteilen. Dies hilft den Schülern dabei zu erkennen, dass es viele Möglichkeiten gibt, diese Probleme anzugehen. Lassen Sie die Schüler den vom Code beschriebenen Pfad mit ihren Fingern folgen, um potentielle Fehler zu finden.

3 Zusätzliche Lernangebote

Bugs einbauen

Lassen Sie die Schüler frühere Level noch einmal durchgehen, wobei sie absichtlich Fehler in ihre Lösung einbauen. Sie können dann andere Schüler ihre Arbeit debuggen lassen. Das kann auch mit Rätseln auf Papier gemacht werden.

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Fachbereich Informatik

Programmiersprachen und Softwaretechnik

Prof. Dr. Klaus Ostermann

Julian Jabs
julian.jabs@uni-tuebingen.de

DENKEN VERSTEHEN LERNEN
Lehrerskript

Einheit C5 – WS 16

Tübingen, 9. Dezember 2016

Übersicht

Wir wissen nicht immer im Voraus, wie manche Dinge sein werden wenn wir unsere Programme laufen lassen. Verschiedene Benutzer haben verschiedene Wünsche, und manchmal soll etwas passieren, das ein Benutzer will, was bei einem anderen aber so nicht geschehen soll. Hier kommen Verzweigungen ins Spiel. In dieser Stunde wird gezeigt, wie Verzweigungen verwendet werden können um ein Programm auf spezifische Informationen zuzuschneidern.

Zusammenfassung

1. Einstieg (15 Min.)
    a) Wiederholung
    b) Neue Wörter
    c) Unter einer Bedingung
2. Gruppenarbeit: Verzweigungen (20 Min.)
3. Abschluss (5 Min.)
    a) Kurzgespräch: Was haben wir gelernt?
    b) Wiederholung: Neue Wörter
4. Test: Verzweigungen (5 Min.)
5. Zusätzliche Lernangebote

Lernziele

Schüler werden...

- ...definieren, unter welchen Umständen bestimmte Teile eines Programms laufen sollten und unter welchen Umständen diese nicht laufen sollten.
- ...basierend auf bestimmten Kriterien entscheiden, ob eine Bedingung erfüllt ist.
- ...ein Programm durchgehen und das Ergebnis für eine gegebene Eingabe bestimmen.

Materialien

Jeder Schüler benötigt die folgenden Materialien, die Sie zu Beginn der Stunde austeilen:

- Stift(e)
- Spielkarten
- Papier, um zu festzuhalten wie ein Programm auf eine Spielkarte reagiert

Dazu kommt das Testblatt (siehe Materialanhang), das Sie am Ende vor dem Test austeilen. Außerdem brauchen Sie noch die Beispielprogramme (siehe Materialanhang).

1 Einstieg (15 Min.)

1.1 Wiederholung

Rekapitulieren Sie zusammen mit den Schülern die letzte Stunde. Sie können dabei abwechselnd Fragen stellen und die Schüler ihre Antworten in kleinen Gruppen diskutieren lassen. Mögliche Fragen sind:

- Was haben wir letztes Mal gemacht?
- Was hättet ihr gerne noch gemacht?
- Sind euch nach der vorherigen Stunde noch Fragen eingefallen?
- Was hat euch an der vorherigen Stunde am besten gefallen?

1.2 Neue Wörter

Schreiben Sie den Begriff “Verzweigung” und seine Bedeutung an die Tafel. Sprechen Sie den Begriff vor und lassen Sie die Schüler wiederholen.

- **Verzweigung**: Ver - zwei - gung
  “Anweisungen, die nur unter bestimmten Bedingungen ausgeführt werden”

1.3 Unter einer Bedingung

- Beginnen Sie mit: “Wir können sofort mit der Stunde anfangen...”
  - Bieten Sie der Klasse an: Falls sie es schafft für 30 Sekunden völlig still zu sein, werden Sie...
    - ...eine Opernarie singen, oder...
    - ...noch fünf weitere Minuten warten, bevor die Stunde anfängt, oder...
    - ...einen Handstand machen.
  - Fangen Sie sofort an zu zählen.
  - Schaffen es die Schüler so lange still zu sein, sagen Sie Ihnen ausdrücklich dass sie erfolgreich waren und deshalb die Belohnung *bekommen*.
  - Andernfalls, sagen Sie ihnen ausdrücklich dass sie nicht die vollen 30 Sekunden völlig still waren und deshalb die Belohnung *nicht bekommen*.
- Fragen Sie die Klasse: Was war die Bedingung für die Belohnung?
  - Die Bedingung war “*FALLS* ihr für 30 Sekunden still seid”. Die englische Bezeichnung für “*FALLS*” ist “*IF*”, und das Wort “if” findet man deswegen in den meisten Programmiersprachen.
    - Falls die Klasse das gewesen wäre, wäre die Bedingung wahr, und sie bekäme die Belohnung.
    - Falls nicht, wäre die Bedingung falsch, und es gäbe auch keine Belohnung.
  - Fallen uns noch andere Verzweigungen ein?
    - “Falls du mein Alter richtig errätst, applaudiert dir die Klasse.”
    - “Falls ich die Antwort weiß, kann ich meine Hand heben.”
    - Fallen der Klasse noch andere Beispiele ein?
- Manchmal möchten wir auch angeben, was passiert falls die “IF”-Bedingung nicht wahr ist.
  - Diese zusätzliche Anweisung wird “ELSE”-Anweisung genannt. Das englische Wort “ELSE” heißt soviel wie “SONST”.
  - Sollte die “IF”-Bedingung nicht zutreffen, können wir beim “ELSE” nachschauen, was zu tun ist.
    - Beispiel: FALLS (IF) ich eine 7 ziehe, klatscht jeder, und SONST (ELSE) sagt jeder “Ooohhh.”
    - Probieren Sie es aus. (Ziehen Sie eine Karte und sehen Sie ob die Klasse entsprechend reagiert.)
  - Fordern Sie die Klasse auf, zu analysieren was gerade passiert ist.
    - Was war das “IF”?
    - Was das “ELSE”?
    - Wurde die “IF”-Bedingung erfüllt? Welche Anweisung wurde ausgeführt (die bei “IF” oder die bei “ELSE”)?
  - Nicht zu glauben, aber es gibt noch eine weitere Möglichkeit. Erläutern Sie: Was wenn ich will, dass ihr klatscht falls ich eine 7 ziehe, und sonst, wenn ich etwas kleiner als 7 ziehe ruft ihr “JUHU!”, und sonst sagt ihr “Ooohhh”?

- Dafür haben wir die Ausdrücke “IF” (“falls ...”), “ELSE IF” (“sonst, falls ...”) und “ELSE” (“sonst...”).
- “IF” ist die erste Bedingung.
- “ELSE IF” wird nur betrachtet, wenn die erste Bedingung falsch ist.
- Das “ELSE” am Ende wird nur betrachtet, wenn nichts davor wahr ist.

2 Gruppenarbeit: Verzweigungen (20 Min.)

1. Entwickeln Sie mit der Klasse einige Algorithmen oder Programme die von Farbe, Wert, oder der Unterscheidung Schwarz/Rot einer Spielkarte abhängig sind, und dementsprechend dann Punkte vergeben oder abziehen. Sie können diese wahlweise als Algorithmus oder Pseudocode (s.u.) schreiben.

   Ein Beispielalgorithmus, bei dem die Bedingung in Klammern steht:

```
Falls (KARTE ist ROT)
    Schreibe DEINEM Team 1 Punkt gut
Sonst
    Schreibe ANDEREM Team 1 Punkt gut
```

   Hinweis: Die deutsche Sprache eignet sich nicht so gut wie die englische für eine direkte Übertragung in etwas, was Programmcode ähnelt. Im Englischen lässt sich der Algorithmus (fast) ebenso gut in natürlicher Sprache lesen wie als Code. Unten ist daher auch die englische Version angegeben, als Basis für das danach folgende Beispiel des gleichen Programms in Pseudocode.

```
If (CARD is RED)
    Award YOUR team 1 point
Else
    Award OTHER team 1 point
```

   Unten haben wir noch ein Beispiel für das gleiche Programm in Pseudocode. Dies soll vor allem als Beispiel dafür dienen, wie “if” und “else”-Schlüsselwörter in echten Programmiersprachen aussehen können. Lassen Sie die Klasse versuchen, den Pseudocode zu verstehen. Wenn die Klasse daran interessiert ist, können Sie im folgenden auch mit Pseudocode-Beispielen üben.

```
If (card.color == RED) {
     points.yours = points.yours + 1;
}

Else {
    points.others = points.others + 1;
}
```

2. Teilen Sie die Klasse in Teams ein.
3. Jedes Team braucht einen Stapel Karteikarten/leere Blätter (mindestens so viele wie Teammitglieder).

4. Zeigen Sie der Klasse eines Ihrer Programme an der Tafel, so dass es alle sehen können.
5. Lassen Sie die Teams reihum Karten ziehen und, dem Programm folgend, die erreichten Punkte erfassen.
6. Spielen Sie das mit verschiedenen Programmen durch, damit die Klasse Verzweigungen wirklich versteht.

Sobald die Klasse einige Übung mit Verzweigungen hat, können Sie sie dazu ermuntern, mit verschachtelten Verzweigungen weiterzumachen.

Ein Beispielalgorithmus:

```
Falls (KARTE ist ROT)
    Schreibe DEINEM Team 1 Punkt gut
Sonst
    Falls (KARTE hat Wert höher als 9)
        Schreibe ANDEREM Team 1 Punkt gut
    Sonst
        Schreibe DEINEM Team die Anzahl Punkte auf der Karte gut
```

Englische Version:

```
If (CARD is RED)
    Award YOUR team 1 point
Else
    If (CARD is higher than 9)
        Award OTHER team 1 point
    Else
        Award YOUR team the same number of points on the card
```

Und das gleiche Programm in Pseudocode:

```
If (card.color == RED) {
    points.yours = points.yours + 1;
}
Else {
    If (card.value > 9) {
        points.others = points.others + 1;
    }
    Else {
        points.yours = points.yours + card.value;
    }
}
```

3 Abschluss (5 Min.)

3.1 Kurzgespräch: Was haben wir gelernt?

Diskutieren Sie mit den Schülern:

- Wenn ihr das in Blockly programmieren würdet, was müsstet ihr um die Verzweigungen herum hinzufügen damit der Code mehr als einmal abläuft? *Hinweis: Nur relevant, wenn die Klasse ausreichend Übung sowohl mit Schleifen als auch mit Blockly hat.*
- Was macht ihr im Alltag nur unter bestimmten Bedingungen?
- Wenn du etwas tun sollst wenn der Wert einer Karte größer 5 ist, und du ziehst eine 5, erfüllst du dann die Bedingung?
- Beachtet, dass Bedingungen entweder “wahr” oder “falsch” sind. Es gibt keine Bewertung einer Bedingung bei der “Banane” heraus kommt.
- Wenn es um mehrere Kombinationen von Bedingungen geht, können wir etwas verwenden das “verschachtelte Verzweigungen” heißt.
  - Was denkt ihr was das bedeutet?
  - Könnt ihr ein Beispiel geben, wo wir das während des Spiels vorher gesehen haben?
- Welchen Teil des Spiels fandet ihr am besten?

3.2 Wiederholung: Neue Wörter

Fragen Sie: Für welche dieser Beschreibungen haben wir heute ein Wort gelernt?

- “Am Anfang einer Textzeile zusätzlichen Leerraum einfügen”
- “Eine Kombination aus Gelb und Grün”
- “Anweisungen, die nur unter bestimmten Bedingungen ausgeführt werden”

...und wie lautet dieses Wort?

4 Test: Verzweigungen (5 Min.)

Teilen Sie die Tests für diese Einheit aus (siehe Materialanhang). Die Schüler haben 5 Minuten, das Blatt zu bearbeiten.

5 Zusätzliche Lernangebote

5.1 Wahr/Falsch-Fangen

- Die Schüler stellen sich auf wie beim Spiel “Ochs vorm Berg”: einer steht vor den anderen als Rufer, die anderen bleiben hinter einer Grundlinie zurück.
- Der Rufer wählt eine Bedingung. Jeder, der die Bedingung erfüllt, darf einen Schritt vorwärts machen.
  - “Falls du einen Gürtel trägst, mache einen Schritt vorwärts.”
  - “Falls du Sandalen trägst, mache einen Schritt.”
- Oder etwas verändert: Bedingungen wie in “Falls du *nicht* blond bist, schreite vor.”

5.2 Verschachteln

- Teilen Sie die Schüler in Paare oder kleine Gruppen ein.
- Lassen Sie sie “Falls”-Anweisungen zum Kartenspielen auf Papierstreifen schreiben, wie etwa:
  - Falls die Farbe der Karte Pik ist
  - Falls die Karte rot ist
- Lassen Sie die Schüler dann noch Papierstreifen mit möglichen Ergebnissen beschreiben:
  - Füge einen Punkt hinzu
  - Ziehe einen Punkt ab
- Sobald sie damit fertig sind, sollen die Schüler drei von jeder Art von Papierstreifen (Bedingungen und Ergebnisse) und drei Spielkarten auswählen. Sie sollen sich dabei die Reihenfolge merken, in der sie die Karten gezogen haben.
- Die Schüler nehmen drei Blätter und schreiben darauf drei verschiedene Programme, wobei sie nur die Anweisungen auf den ausgewählten Papierstreifen verwenden.
  - Ermuntern Sie die Schüler, einige “Falls”-Anweisungen innerhalb anderer “Falls”-Anweisungen zu verwenden.
- Jetzt können die Schüler alle drei Programme mit den Spielkarten, die sie gezogen haben, ablaufen lassen. Dabei sollten sie immer die gleiche Reihenfolge der Karten einhalten.
  - Haben irgendwelche zwei Programme dasselbe Ergebnis?
  - Haben irgendwelche Programme verschiedene Ergebnisse?

Materialanhang

Es folgen in Reihenfolge:

- Beispielprogramm Nr. 1
- Beispielprogramm Nr. 2
- das Testblatt

Verzweigungen

Beispielprogramm 1

Beispielprogramm als Algorithmus:

```
FALLS (KARTE ist ROT)
     Schreibe DEINEM Team 1 Punkt gut
SONST
     Schreibe ANDEREM Team 1 Punkt gut
```

Das Programm verlangt, dass du eine Karte ziehst. Ist die Karte rot, bekommt dein Team einen Punkt, sonst bekommt das andere Team einen Punkt.

Obiges Beispielprogramm in Pseudocode (wie Code, aber in keiner bestimmten Sprache):

```
If (card.color == RED) {
     points.yours = points.yours + 1;
}
Else {
     points.other = points.other + 1;
}
```

Verzweigungen

Beispielprogramm 2

Beispielprogramm als Algorithmus:

```
FALLS (KARTE ist ROT)
     Schreibe DEINEM Team 1 Punkt gut
SONST
     FALLS (KARTE hat Wert größer als 9)
          Schreibe ANDEREM Team 1 Punkt gut
     SONST
          Schreibe DEINEM Team die Anzahl Punkte auf der Karte gut
```

Das Programm verlangt, dass du eine Karte ziehst. Ist die Karte rot, erhält dein Team einen Punkt. Sonst muss die Karte schwarz sein. Falls deine schwarze Karte einen Wert größer als 9 hat, bekommt das andere Team einen Punkt, ansonsten muss deine Karte schwarz sein und einen Wert kleiner oder gleich 9 haben, und du bekommst so viele Punkte wie auf der Karte stehen.

Obiges Beispielprogramm in Pseudocode (wie Code, aber in keiner bestimmten Sprache):

```
If (card.color == RED) {
     points.yours = points.yours + 1;
}
Else {
     If (card.value > 9) {
          points.other = points.other + 1;
     }
     Else {
          points.yours = points.yours + card.value;
     }
}
```

Name:_____ Datum:_____

Verzweigungen

Übungsblatt

Sieh dir das Programm unten an.

Die Schritte unten zeigen, wie zwei Teams das Verzweigungs-Spiel spielen. Sie sind dabei immer abwechselnd an der Reihe. Versuche herauszufinden, was nach jeder gezogenen Karte passiert. Schreibe dabei die Punktzahl während jeder Runde auf. Umkreise den Sieger nach drei Runden.

```
FALLS (KARTE hat Wert kleiner als 5)
     FALLS (KARTE ist SCHWARZ)
          Schreibe DEINEM Team die Anzahl Punkte auf der Karte gut
     SONST
          Schreibe ANDEREM Team 1 Punkt gut
SONST
     FALLS (KARTE ist HERZ)
          Schreibe DEINEM Team 1 Punkt gut
```

So lief das Spiel ab:

	TEAM 1	PUNKTE	TEAM 2	PUNKTE
RUNDE 1		_____		_____
RUNDE 2		_____		_____
RUNDE 3		_____		_____

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Fachbereich Informatik

Programmiersprachen und Softwaretechnik

Prof. Dr. Klaus Ostermann

Julian Jabs
julian.jabs@uni-tuebingen.de

DENKEN VERSTEHEN LERNEN
Lehrerskript

Einheit C5P – WS 16

Tübingen, 8. Dezember 2016

Übersicht

Bis zu diesem Punkt sollten alle Programme, die die Schüler geschrieben haben, jedes Mal auf die exakt gleiche Weise ablaufen – verlässlich, aber nicht sehr flexibel. In diesem Abschnitt führen wir Verzweigungen und bedingte Anweisungen ein, also Code der verschiedenartig funktioniert, abhängig von den Bedingungen unten denen er abläuft.

Zusammenfassung

1. Einführung
2. Programmieren: Künstler: Debugging
3. Zusätzliche Lernangebote

Lernziele

SchülerInnen werden...

- ...Werte mit dem “==”-Operator vergleichen.
- ...bedingte Aussagen in mündlicher Sprache in ein Programm übersetzen.
- ...identifizieren, wann Verzweigungen verwendet werden können, um mit unbekannten Variablen umzugehen.
- ...einen Algorithmus mit einer bedingten Aussage ausführen.
- ...Rätsel unter Verwendung einer Kombination aus Sequenzen in Schleifen und Verzweigungen lösen.

1 Einführung

Wiederholen Sie die Übung mit den Spielkarten aus der letzten Stunde (Verzweigungen):

- Was ist eine Verzweigung?
- Wann ist sie nützlich?
- Was gab es für Bedingungen in der Übung?

Erläutern Sie: Jetzt werden wir Verzweigungen für unsere Biene verwenden, um mit einigen mysteriösen lila Blumen umzugehen. Wir wissen nicht ob diese Blumen Nektar haben oder nicht, daher müssen wir Verzweigungen einsetzen um sicher zu gehen dass wir nur Nektar einsammeln wenn welcher da ist, aber nicht versuchen Nektar von einer Blume einzusammeln die keinen hat.

2 Programmieren: Biene: Verzweigungen

Level 13 von Code.org Kurs 2

3 Zusätzliche Lernangebote

Wenn Sie diese noch nicht ausprobiert haben, können Sie hier die zusätzlichen Angebote für Stunde 12 (Verzweigungen) nutzen.

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Fachbereich Informatik

Programmiersprachen und Softwaretechnik

Prof. Dr. Klaus Ostermann

Julian Jabs
julian.jabs@uni-tuebingen.de

DENKEN VERSTEHEN LERNEN
Lehrerskript

Einheit M1 – WS 16

Tübingen, 9. Dezember 2016

Übersicht

In dieser Stunde werden geometrische Formen auf algorithmische Weise betrachtet. Die Schüler werden lernen, für einfache geometrische Formen Algorithmen zu entwickeln. Damit soll ihnen ein alternativer Zugang zu Geometrie eröffnet werden, bei dem sie für die einzelnen Formen wesentlichen Größen (Längen und Winkel) “in Aktion” erleben können.

Zusammenfassung

1. Einstieg (15 Min.)
   a) Wiederholung
   b) Links, rechts, geradeaus
2. Gruppenarbeit: Formen-Sprinter (15 Min.)
3. Abschluss (5 Min.)
   a) Kurzgespräch: Was haben wir gelernt?
4. Test: Formen (10 Min.)
5. Zusätzliche Lernangebote

Lernziele

Schüler werden...

- ...verschiedene geometrische Formen wiederholen.
- ...diese mit Algorithmen beschreiben, die diese Formen zeichnen.
- ...mithilfe dieser Algorithmen die wesentlichen Eigenschaften (Größen) der Formen nachvollziehen und diese von den unwesentlichen unterscheiden.
- ...Argumente als Möglichkeit kennenlernen, Anweisungen genauer anzugeben.

Materialien

Jeder Schüler benötigt die folgenden Materialien, die Sie zu Beginn der Stunde austeilen:

- Stift(e)
- Geodreieck
- kariertes Papier
- das Arbeitsblatt “Formen” (siehe Materialanhang)

Dazu kommt das Testblatt (siehe Materialanhang), das Sie am Ende vor dem Test austeilen.

1 Einstieg (15 Min.)

1.1 Wiederholung

Rekapitulieren Sie zusammen mit den Schülern die letzte Stunde. Sie können dabei abwechselnd Fragen stellen und die Schüler ihre Antworten in kleinen Gruppen diskutieren lassen. Mögliche Fragen sind:

- Was haben wir letztes Mal gemacht?
- Was hättet ihr gerne noch gemacht?
- Sind euch nach der vorherigen Stunde noch Fragen eingefallen?
- Was hat euch an der vorherigen Stunde am besten gefallen?

1.2 Neue Wörter

Wiederholen Sie den Begriff des Algorithmus, der in dieser Einheit eine große Rolle spielt:

- **Algorithmus**: gesprochen Al - go - rith - mus
  “eine Liste von Schritten, denen man folgen kann um eine Aufgabe zu erfüllen”

1.3 Links, rechts, geradeaus

Führen Sie in die Einheit ein: Ihr kennt verschiedene Formen wie Rechtecke, Quadrate und Dreiecke. Stellt euch vor, ihr habt eine bestimmte Form auf Kästchenpapier gezeichnet. Wie würdet ihr jemand anderem die Form beschreiben, so dass er die gleiche Form zeichnen kann?

- Angenommen, der andere weiß bereits dass es sich um ein Rechteck handelt. Was musst du ihm dann noch sagen, damit er das gleiche Rechteck zeichnen kann, dass auch du gezeichnet hast?
- Wenn der andere gar nicht weiß, um was für eine Form es sich handelt: Was könnt ihr ihm dann für Anweisungen geben, um die Form zu zeichnen?
  - Stellt euch denjenigen, der die Form nachzeichnen soll, wie einen Sprinter vor, der am Rand der Form entlangläuft.
  - Überlegt euch, wie der andere dann laufen, also den Stift führen muss.

– Bedenkt, dass der andere im Voraus wissen möchte, wie lange die Linie ist, die er entlanglaufen (zeichnen) soll.

- Wenn ihr euch an die erste Stunde (Einheit C1) zurückerinnert: Da hatten wir auch mehrere Anweisungen, die jemand anderes beim Ausmalen von Kästchen anleiten sollten. Damals waren das fünf Anweisungen (links, rechts, oben, unten, und ausmalen). Was können wir für eine Liste von Anweisungen zusammenstellen, mit denen wir Formen wie Dreiecke und Rechtecke zeichnen lassen können? Schafft die Klasse es, die notwendigen Anweisungen zusammenzustellen?

  Mögliche Anweisungen sind:

  – eine gerade Linie von einer bestimmten Länge (Anzahl Kästchen oder Zentimeter) ziehen
  – um xyz Grad nach links/rechts drehen, wobei xyz = 45 oder 90 (beliebige Auswahl ist eine Alternative für Fortgeschrittene)
  – eine bestimmte Länge überspringen (Stift absetzen)

  Siehe auch: die Alternativen für Fortgeschrittene am Ende dieses Abschnitts.

- Die Längen, Gradzahlen, und die Unterscheidung links/rechts nennt man *Argumente* der Anweisungen. Damit wird dem Zeichner genauer mitgeteilt, wie er vorgehen soll.

  – Beispiele:

    * Wenn ihr jemand eine Linie von 5 Kästchen Länge ziehen lassen wollt, so verwendet ihr die Anweisung “Ziehe Linie der Länge 5 Kästchen”.
    * Wenn es nur 3 Kästchen sein sollen, so verwendet ihr “Ziehe Linie der Länge 3 Kästchen.”

  – Es ändert sich also immer nur diese eine Angabe, ansonsten ist die Anweisung zum Ziehen einer Linie gleich.

- Um es uns einfacher zu machen, werden wir von jetzt an statt Text-Anweisungen wie “Ziehe Linie der Länge 5” Symbole verwenden, wenn wir unsere Programme schreiben. (Erinnerung: In der ersten Stunde haben wir das auch schon so gemacht.)

  – Die 5 ist hier jeweils das Argument. Stattdessen könnt ihr jede andere Zahl verwenden, die ihr gerade braucht. Auch die Gradangaben $45°$. und $90°$ kann man als Argumente der Drehanweisung sehen (dies wird noch bedeutsamer, wenn die Winkel frei gewählt werden können).

2 Gruppenarbeit: Formen-Sprinter (15 Min.)

1. Teilen Sie die Klasse in Gruppen ein, und teilen Sie das Arbeitsblatt für diese Stunde aus (siehe Materialanhang).
2. Jede Gruppe wählt einige der auf dem Blatt zur Auswahl stehenden Formen aus. Achtung: Form 6 ist für Fortgeschrittene, die sich an Winkelmessung versuchen wollen; weisen Sie die Klasse darauf hin.

3. In der Gruppe entwickeln die SchülerInnen dann ein Programm für jede dieser Formen. Dabei orientieren sie sich an den auf dem Blatt vorgegebenen Schritten.
4. Lassen Sie die Gruppen ihre Programme austauschen und versuchen, die Programme der jeweils anderen auszuführen. Können die SchülerInnen damit die beabsichtigten Formen zeichnen?

3 Abschluss (5 Min.)

Diskutieren Sie mit der Klasse:

- Was war schwierig beim Entwickeln der Programme, was einfach?
- Wie war das beim Ausführen beim Programme?
- Fallen euch noch andere Formen ein, die man anhand von unseren Anweisungen zeichnen könnte?
- Was für Formen kann man mit unseren Anweisungen nicht zeichnen?

4 Test: Formen (10 Min.)

Teilen Sie die Tests für Einheit M1 aus (siehe Materialanhang). Die Schüler haben 10 Minuten, das Blatt zu bearbeiten, wofür sie ein Geodreieck verwenden sollten. Weisen Sie die Klasse darauf hin, dass es nicht darauf ankommt, dass Zentimeter- und Grad-Angaben sehr genau sind.

5 Zusätzliche Lernangebote

5.1 Zusammengesetzte Formen

Lassen Sie die SchülerInnen Programme für komplexere Formen entwickeln, die aus mehreren Rechtecken und Dreiecken zusammengesetzt sind (Beispiel: Dreieck und Quadrat als Haus).

Alternative Anweisungsmengen (für Fortgeschrittene)

Freie Winkelauswahl

Wer mit Winkel und Messung von Winkeln umgehen kann, kann auch die Anweisungsmenge aus der Unterrichtseinheit (Drehen und Linien ziehen) verwenden, und dabei beliebige Argumente (also nicht nur 45 und 90 Grad wie bisher) zum Drehen verwenden.

Diagonale Bewegung

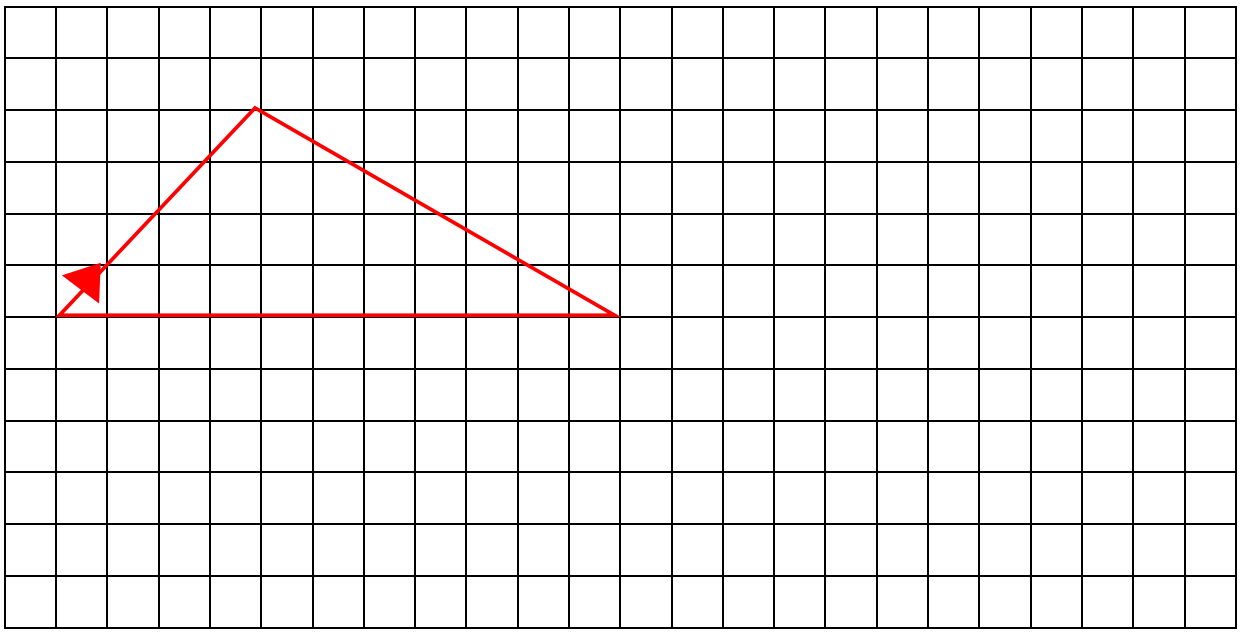
Alternative Anweisungen zum Zeichnen von Formen; für diese Anweisungen braucht man sich nicht die aktuelle Richtung zu merken:

- eine gerade Linie von einer bestimmten Länge (Anzahl Kästchen) nach oben/unten/links/rechts ziehen

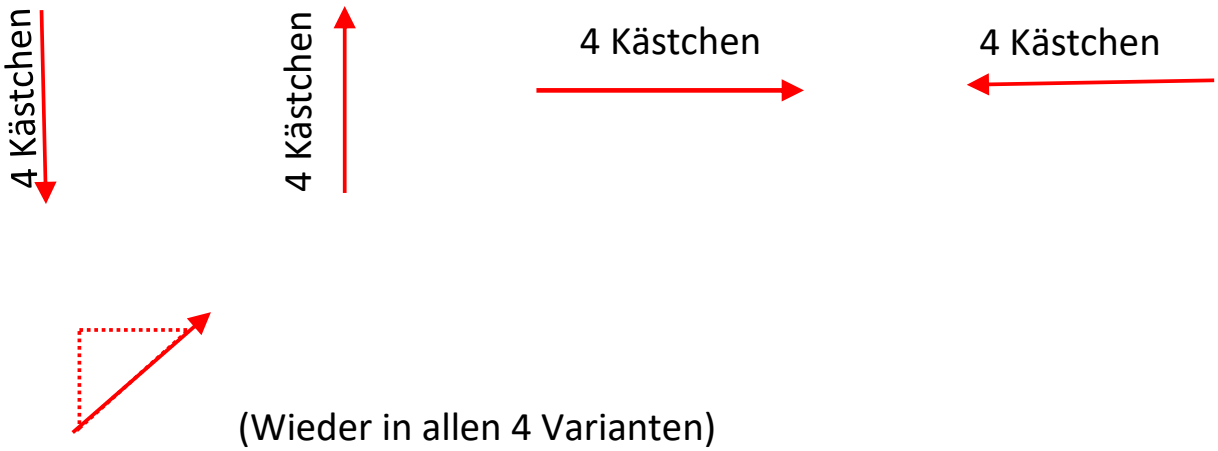
- eine diagonale Linie ziehen zum eine bestimmte Anzahl Kästchen nach links/rechts und oben/unten entfernten Ziel

Auf der folgenden Seite wird eine symbolische Darstellung für diese Anweisungen illustriert (Credit: J. Dominik Krauss):

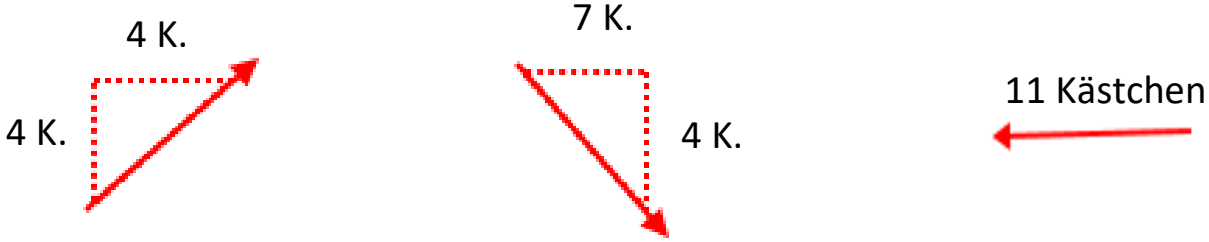
Alternative Aufgabe:



Mit Anweisungsarten:



Also Lösung fürs Dreieck:



Materialanhang

Es folgen in Reihenfolge

- das Arbeitsblatt für Einheit M1
- das Testblatt für Einheit M1

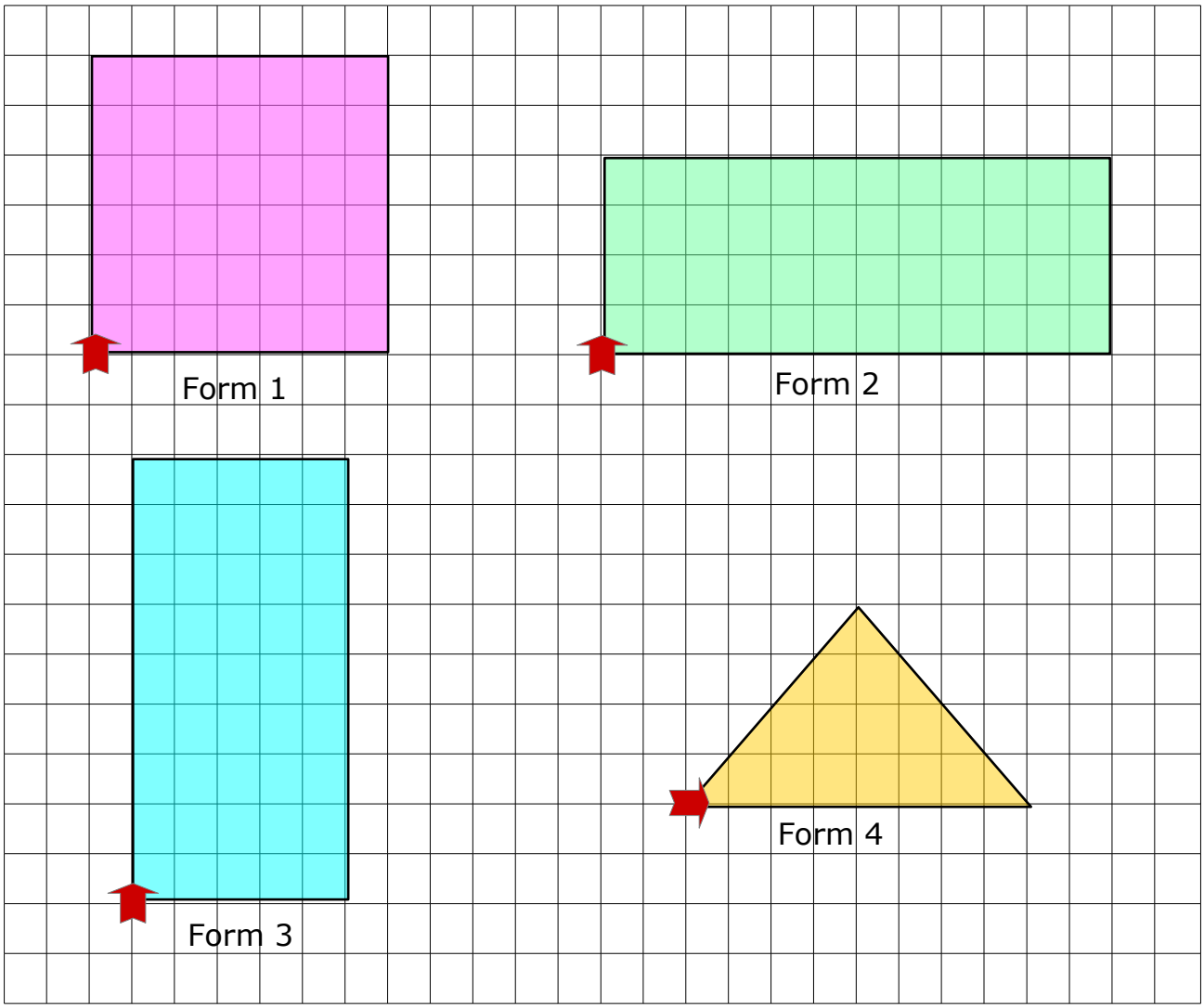
Name:_____ Datum:_____

Formen-Sprinter

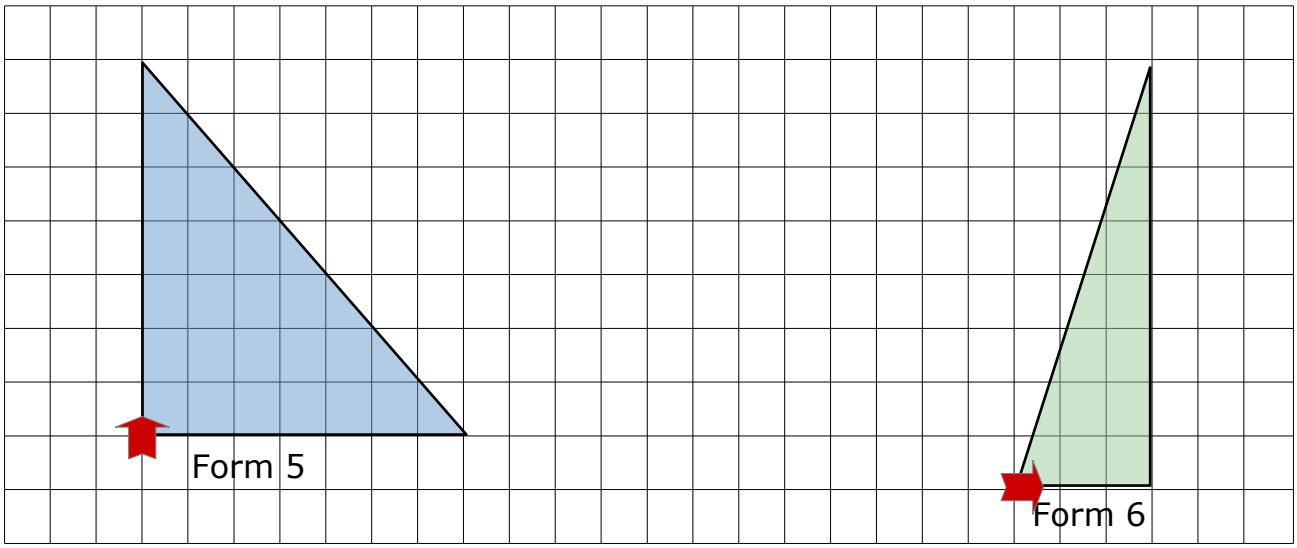
Arbeitsblatt Formen

Wählt euch einige der Formen unten aus und geht für jede Form folgendermaßen vor:

1. Schreibt euch zunächst auf, welche Größen, also Längen und Winkel, in der Form vorkommen.
2. Versucht dann, ein Programm zu schreiben, das nur die vorgegebenen Anweisungen sowie Schleifen verwendet. Eine Möglichkeit, wo ihr anfangen könnt und wie gedreht, ist eingezeichnet.
3. Kommen in dem Programm alle Größen vor, die ihr euch aufgeschrieben hattet? Kommen noch andere Größen vor? Wenn ja, überprüft ob euer Programm oder eure Liste einen Fehler hat.
4. Wenn ihr noch Zeit übrig habt: Versucht, möglichst wenige Anweisungen in eurem Programm zu verwenden, so dass es aber immer noch die richtige Form zeichnet.

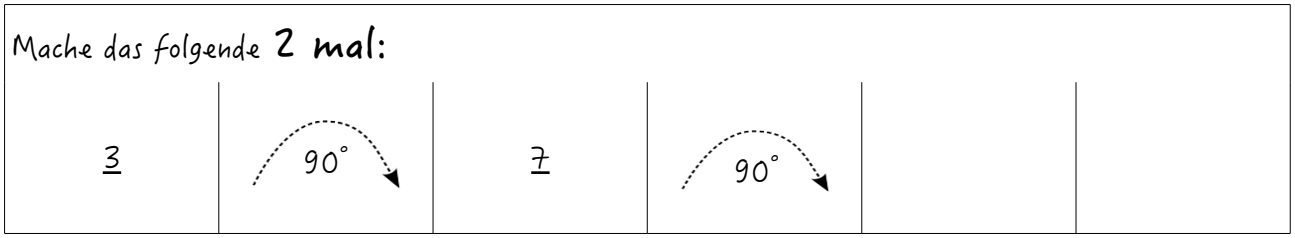


Name:_____ Datum:_____

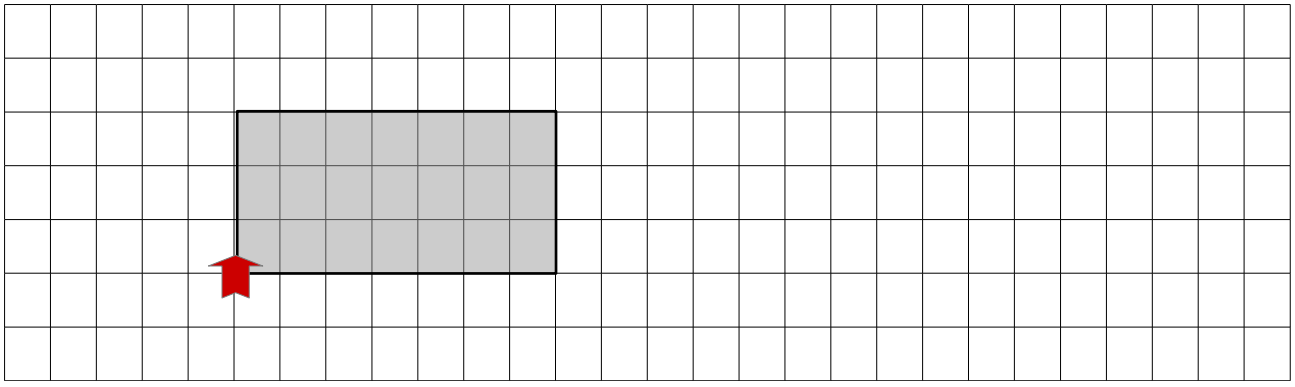


Anweisungen:

Ein Beispielprogramm:



Das zeichnet das unten gezeigte Rechteck.



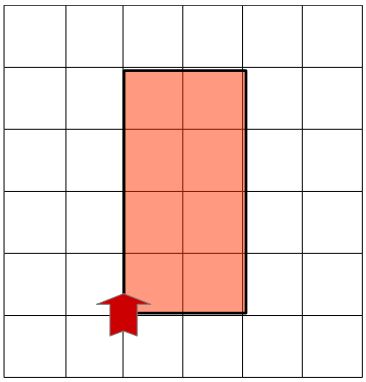
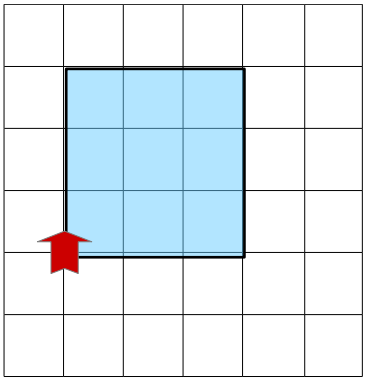
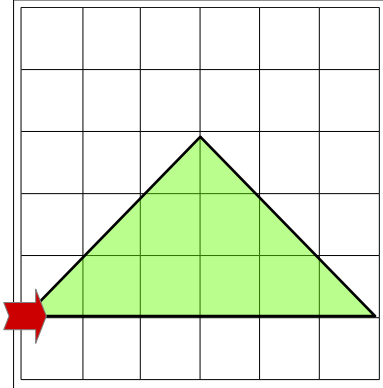
Name:_____ Datum:_____

Formen-Sprinter

Übungsblatt

Heute habt ihr Programme für verschiedene Formen entwickelt. Das könnt ihr jetzt alleine üben.

Schreibt jeweils ein möglichst kurzes Programm für die unten gezeigten Formen. Das erste wurde als Beispiel schon gemacht.

<table>
<tr>
<td>  </td>
<td>Mache das folgende 2 mal:
4 | 90° | 2 | 90°</td>
</tr>
<tr>
<td>  </td>
<td></td>
</tr>
<tr>
<td>  </td>
<td></td>
</tr>
</table>

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Fachbereich Informatik

Programmiersprachen und Softwaretechnik

Prof. Dr. Klaus Ostermann

Julian Jabs
julian.jabs@uni-tuebingen.de

DENKEN VERSTEHEN LERNEN
Lehrerskript

Einheit M1P1 – WS 16

Tübingen, 8. Dezember 2016

Übersicht

In dieser Einheit werden die Schüler die Kontrolle über einen Künstler übernehmen um einfache Zeichnungen auf dem Bildschirm fertigzustellen.

Zusammenfassung

1. Einführung
2. Programmieren: Künstler: Sequenz
3. Zusätzliche Lernangebote

Lernziele

SchülerInnen werden...

- ...ein Programm erschaffen, um ein Bild mit sequentiellen Schritten fertigzustellen.
- ...für einen bestimmten Befehl ein Argument auswählen.
- ...zwischen definierenden und nicht definierenden Eigenschaften von Dreiecken, Quadraten und Rechtecken unterscheiden.
- ...Dreiecke, Quadrate und Rechtecke zeichnen, um definierende Eigenschaften widerzuspiegeln.
- ...den Unterschied zwischen Quadraten und Rechtecken erklären und diesen anhand der Befehle, die zum Zeichnen der verschiedenen Formen benötigt werden, nachweisen.
- ...Quadrate und Rechtecke anhand der Anzahl ihrer Seiten und der Länge ihrer Seiten vergleichen und einander gegenüberstellen.

- ...zweidimensionale Formen (Rechtecke, Quadrate, Trapezoide, Dreiecke) zusammensetzen um eine zusammengesetzte Form zu erschaffen, zum Beispiel zwei Quadrate um ein Rechteck zu bilden oder zwei Rechtecke um ein Quadrat zu bilden.
- ...neue Formen aus bereits zusammengesetzten Formen zusammensetzen.
- ...Zerlegungen in ein Rechteck einzeichnen und die Zerlegungen unter Verwendung der Worte “Halbe”, “Viertel”, “Hälfte von” und “Viertel von” beschreiben.
- ...ein gesamtes Rechteck als zwei Halbe oder vier Viertel beschreiben.
- ...erklären, dass die Unterteilung in gleichmäßigere Anteile zu kleineren Anteilen führt.

1 Einführung

Wiederholen Sie mit der Klasse kurz den Inhalt der vorherigen Einheit (M1):

- Letztes Mal haben wir Programme entwickelt, mit denen wir bestimmte Formen zeichnen konnten.
- Wir haben die Programme dann “von Hand” ausgeführt, um die Formen zu zeichnen.
- In dieser und den folgenden Stunden werden wir das am Computer machen. Dabei werden wir auch etwas schwierigere Zeichnungen programmieren, etwa welche die aus mehreren Formen bestehen (z.B. ein Haus).

2 Programmieren: Künstler: Sequenz

Level 4 von Code.org Kurs 2

Geodreiecke können den Schülern dabei helfen, sich die Winkel die sie benötigen besser vorzustellen.

3 Zusätzliche Lernangebote

TODO?

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Fachbereich Informatik

Programmiersprachen und Softwaretechnik

Prof. Dr. Klaus Ostermann

Julian Jabs
julian.jabs@uni-tuebingen.de

DENKEN VERSTEHEN LERNEN
Lehrerskript

Einheit M1P2 – WS 16

Tübingen, 8. Dezember 2016

Übersicht

Wir kehren zurück zum Künstler (siehe vorherige Einheit M1P1), wobei die Schüler dieses Mal lernen, komplexere Bilder zu zeichnen indem sie für einfache Sequenzen von Anweisungen Schleifen verwenden.

Zusammenfassung

1. Einführung
2. Programmieren: Künstler: Schleifen
3. Zusätzliche Lernangebote

Lernziele

SchülerInnen werden...

- ...zählen, wie oft eine Aktion wiederholt werden soll und dies als Schleife repräsentieren.
- ...eine Form in ihre kleinste wiederholbare Sequenz zerlegen.
- ...ein Programm schreiben, dass komplexe Formen durch die Wiederholung einfacher Sequenzen zeichnet.

1 Einführung

Erinnern Sie die Klasse an die vorherige “Unplugged”-Einheit M1 und wie dabei Schleifen zum Einsatz kamen, um die Programme kürzer zu machen.

- Lassen Sie die Schüler so viele einfache Formen wie möglich benennen, wobei sie sich auf Formen mit gleichen Seiten und Winkeln konzentrieren sollen.

- Fragen Sie bei jeder Form:
  - Wie würdet ihr jemandem erklären, wie man diese Form zeichnen kann?
  - Wie könntet ihr die Form unter Verwendung einer Schleife zeichnen?

2 Programmieren: Künstler: Schleifen

Level 7 von Code.org Kurs 2

Geodreiecke können den Schülern dabei helfen, sich die Winkel die sie benötigen besser vorzustellen.

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Fachbereich Informatik

Programmiersprachen und Softwaretechnik

Prof. Dr. Klaus Ostermann

Julian Jabs
julian.jabs@uni-tuebingen.de

DENKEN VERSTEHEN LERNEN
Lehrerskript

Einheit M1P3 – WS 16

Tübingen, 8. Dezember 2016

Übersicht

In diesem Abschnitt werden die Schüler weiter ihre Debugging-Fähigkeiten üben, indem sie dem Künstler (siehe die vorherigen Einheiten M1P1 und M1P2) helfen, Bilder zu korrigieren, die nicht ganz richtig sind.

Zusammenfassung

1. Einführung
2. Programmieren: Künstler: Debugging
3. Zusätzliche Lernangebote

Lernziele

SchülerInnen werden...

- ...vorhersagen, wo ein Programm fehlschlagen wird.
- ...ein vorher existierendes Programm modifizieren, um Fehler zu beheben.
- ...einen Algorithmus identifizieren, der fehlschlägt wenn seine Schritte in der falschen Reihenfolge sind.
- ...über den Prozess des Debuggings auf altersgemäße Weise reflektieren.

1 Einführung

Mittlerweile sollten die Schüler daran gewöhnt sein, tief zu schürfen und Bugs zu finden. Das ist ein guter Zeitpunkt, an dem sich die Klasse über Debugging-Taktiken und -Schwierigkeiten austauschen kann:

- Welche Art Bugs sind für euch am einfachsten zu finden? Warum?
- Was für Bugs waren am schwersten zu finden? Wie habt ihr sie schließlich behoben?
- Was ist das erste, wonach ihr in einem fehlerhaften Programm schaut?

2 Programmieren: Künstler: Debugging

Level 11 von Code.org Kurs 2

Manche Schüler ist es zuwider, ein Programm laufen zu lassen, ehe sie alle Fehler behoben haben. Manchmal ist aber der einfachste Weg herauszufinden, was in einem Programm fehlerhaft ist, zu beobachten wie es fehlschlägt. Es ist also nichts falsch daran, ein Programm laufen zu lassen bevor wir damit fertig sind, alle Fehler zu beheben. (Wir sind nur bei den Test-Leveln daran interessiert, es beim ersten Mal gleich richtig zu haben.)

3 Zusätzliche Lernangebote

Bugs einbauen

Lassen Sie die Schüler frühere Level noch einmal durchgehen, wobei sie absichtlich Fehler in ihre Lösung einbauen. Sie können dann andere Schüler ihre Arbeit debuggen lassen. Das kann auch mit Rätseln auf Papier gemacht werden.

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Fachbereich Informatik

Programmiersprachen und Softwaretechnik

Prof. Dr. Klaus Ostermann

Julian Jabs
julian.jabs@uni-tuebingen.de

DENKEN VERSTEHEN LERNEN
Lehrerskript

Einheit C6 – WS 16

Tübingen, 9. Dezember 2016

Übersicht

Binär ist enorm wichtig in der Computerwelt. Die Mehrzahl der heutigen Computer speichern alle möglichen Arten von Informationen in Binärform. In dieser Stunde wird gezeigt, wie es möglich ist, etwas uns bekanntes in eine Reihe von “ans” und “aus” zu übersetzen.

Zusammenfassung

1. Einstieg (15 Min.)
    a) Wiederholung
    b) Neue Wörter
    c) An und Aus
2. Aktivität: Binäre Armbänder (15 Min.)
3. Abschluss (5 Min.)
    a) Kurzgespräch: Was haben wir gelernt?
4. Test: Binär (10 Min.)
5. Zusätzliche Lernangebote

Lernziele

Schüler werden...

- ...Buchstaben in Binär codieren.
- ...Binär zurück in Buchstaben decodieren.
- ...die Idee, Initialen auf einem Armband zu “speichern” mit der Idee in Verbindung bringen, Informationen in einem Computer zu speichern.

Materialien

Jeder Schüler benötigt die folgenden Materialien, die Sie zu Beginn der Stunde austeilen:

- Stift(e)
- Schere
- das Arbeitsblatt “Binäre Armbänder” (mit Binär-Decodier-Schlüssel) (siehe Materialanhang)

Dazu kommt das Testblatt (siehe Materialanhang), das Sie am Ende vor dem Test austeilen.

Außerdem brauchen Sie noch Bilder von einem geöffneten Computer, oder noch besser: bringen Sie einen geöffneten Computer mit. Sie können sich auch eine kurze binäre Nachricht überlegen (siehe “And und Aus” unten) und vor Beginn an die Tafel schreiben.

1 Einstieg (15 Min.)

1.1 Wiederholung

Rekapitulieren Sie zusammen mit den Schülern die letzte Stunde. Sie können dabei abwechselnd Fragen stellen und die Schüler ihre Antworten in kleinen Gruppen diskutieren lassen. Mögliche Fragen sind:

- Was haben wir letztes Mal gemacht?
- Was hättet ihr gerne noch gemacht?
- Sind euch nach der vorherigen Stunde noch Fragen eingefallen?
- Was hat euch an der vorherigen Stunde am besten gefallen?

1.2 Neue Wörter

Schreiben Sie den Begriff “Binär” und seine Bedeutung an die Tafel. Sprechen Sie den Begriff vor und lassen Sie die Schüler wiederholen.

- **Binär**: Bi - när
  “Ein Weg Informationen darzustellen, unter Verwendung von nur zwei Auswahlmöglichkeiten”

1.3 An und Aus

- Wenn Sie eine binäre Nachricht an die Tafel geschrieben haben: Machen Sie die Schüler darauf aufmerksam und fragen Sie ob jemand weiß was das ist oder was es bedeutet.
- Fragen Sie die Klasse, ob sie schon mal das Innere eines Computers gesehen hat.
  - Fragen Sie, was sich darin befindet.
  - An dieser Stelle passt es gut, wenn Sie einen geöffneten Computer zeigen (in Bildern oder einen echten).
- Erläutern Sie: Kabel transportieren Informationen durch die Maschine, in Form von Elektrizität.

  - Die zwei Auswahlmöglichkeiten, die ein Computer bezogen auf diese elektrische Information verwendet sind “an” und “aus”. Wenn Computer Informationen nur mit zwei Auswahlen darstellen, nennt man das “binär”.
  - Auch wenn die Information ihr Ziel erreicht hat, werden für ihre Darstellung weiter nur zwei Auswahlmöglichkeiten verwendet.

- Computer *speichern* Informationen auch binär.
  - Binär bedeutet nicht immer nur “aus” und “an”.
    * Festplatten speichern Informationen als magnetisch positiv und magnetisch negativ.
    * DVDs speichern Informationen als entweder reflektierend oder nicht-reflektierend.
  - Fragen Sie: Wie denkt ihr sollen wir die Dinge, die wir in einem Computer speichern, in Binär übersetzen?
    * Fangen Sie mit Buchstaben an: Verwenden Sie den Binär-Decodier-Schlüssel um zu zeigen, wie ein Computer Großbuchstaben darstellen könnte.
      · Jetzt wäre ein guter Zeitpunkt, um zu erwähnen, dass jede Stelle an der eine binäre Auswahl möglich ist, Binärzifffer oder auch kurz Bit heißt.
      · Fragen Sie ob jemand weiß wie eine Gruppe von acht Bits genannt wird (Byte).
      · Nebenbei: Eine Gruppe von vier Bits heißt Nibble.
    * Gehen Sie ein paar Beispiele durch, bei denen Buchstaben in Binär und zurück übersetzt werden.
    * Schreiben Sie dann einen codierten Buchstaben und lassen Sie die Klasse herausfinden, welcher es ist.
    * Wenn die Klasse es schafft, herauszufinden welche Buchstaben Sie codiert haben, können Sie mit der Gruppenarbeit fortfahren.

2 Aktivität: Binäre Armbänder (15 Min.)

1. Lassen Sie jede/n Schüler/in den ersten Buchstaben eines von ihr/ihm gewählten Namens oder Begriffs im Binär-Dekodier-Schlüssel finden. (Sie können einige Vorschläge machen, was für ein Name/Begriff das sein könnte; er sollte nur nicht zu einfach von Mitschülern zu erraten sein, wie z.B. der eigene Vorname.)

2. Sie können dann die Quadrate des bereitgestellten Armbands so ausmalen, dass sie dem Muster der Quadrate neben dem von ihnen ausgewählten Buchstaben entsprechen.

3. Danach können sie das Armband ausschneiden und am Handgelenk befestigen (z.B. die Enden mit Klebeband verbinden).

4. Lassen Sie die Schüler ihre Armbänder untereinander austauschen und versuchen, die Buchstaben herauszufinden.

5. Zum Abschluss wenden wir uns nochmal der binären Nachricht an der Tafel zu: Fragen Sie die Klasse ob sie die Nachricht jetzt, mit dem was sie gelernt hat, entschlüsseln kann.

3 Abschluss (5 Min.)

3.1 Kurzgespräch: Was haben wir gelernt?

Diskutieren Sie mit den Schülern:

- Was wird in einem Computer sonst noch binär dargestellt?
- Wie könnten wir Binär noch darstellen, außer als gefüllte und nicht gefüllte Kästchen?
- Was hat euch an der Aktivität am besten gefallen?

4 Test: Verzweigungen (10 Min.)

Teilen Sie die Tests für diese Einheit aus (siehe Materialanhang). Die Schüler haben 10 Minuten, das Blatt zu bearbeiten.

5 Zusätzliche Lernangebote

Binäre Bilder

- Im Web gibt es einige Ressourcen mit denen man an die heutige Stunde anknüpfen kann, etwa über das binäre Codieren von Bildern oder Musik.

Materialanhang

Es folgen in Reihenfolge

- das Arbeitsblatt für Einheit C6
- das Testblatt für Einheit C6

Name:_____ Datum:_____

Binäre Armbänder

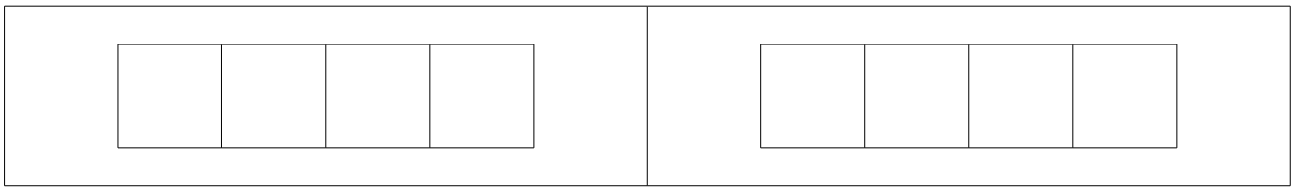
Binär-Decodier-Schlüssel

A	■□■■	■■■□	N	■□■■	□□□■
B	■□■■	■■□■	O	■□■■	□□□□
C	■□■■	■■□□	P	■□■□	■■■■
D	■□■■	■□■■	Q	■□■□	■■■□
E	■□■■	■□■□	R	□■■□	■□■□
F	■□■■	■□□■	S	■□■□	■■□□
G	■□■■	■□□□	T	■□■□	■□■■
H	■□■■	□■■□	U	■□■□	■□■□
I	■□■■	□■■□	V	■□■□	■□□■
J	■□■■	□■□□	W	■□■□	■□□□
K	■□■■	□■□□	X	■□■□	□■■■
L	■□■■	□□■■	Y	■□■□	□■■□
M	■□■■	□□■□	Z	■□■□	□■□■

Finde den ersten Buchstaben des Namens, den du gewählt hast.

Fülle die Kästchen des Armbands unten aus, so dass das Muster dem neben dem Buchstaben entspricht.

Schneide das Armband aus und klebe es um dein Handgelenk!



Name:_____ Datum:_____

Binäre Armbänder

Übungsblatt

Verwende diesen Binär-Decodier-Schlüssel um die Nachricht ganz unten zu entschlüsseln:

A	■□■■	■■■□	N	■□■■	□□□■
B	■□■■	■■□■	O	■□■■	□□□□
C	■□■■	■■□□	P	■□■□	■■■■
D	■□■■	■□■■	Q	■□■□	■■■□
E	■□■■	■□■□	R	□■■□	■□■□
F	■□■■	■□□■	S	■□■□	■■□□
G	■□■■	■□□□	T	■□■□	■□■■
H	■□■■	□■■□	U	■□■□	■□■□
I	■□■■	□■■□	V	■□■□	■□□■
J	■□■■	□■□□	W	■□■□	■□□□
K	■□■■	□■□□	X	■□■□	□■■■
L	■□■■	□□■■	Y	■□■□	□■■□
M	■□■■	□□■□	Z	■□■□	□■□■

Kannst du herausfinden, was die Nachricht bedeutet?

■□■■ □■■□ _____

■□■■ ■■■□ _____

■□■■ □□■□ _____

■□■■ □□■□ _____

■□■■ □□□□ _____

■□■□ ■□□□ _____

■□■■ ■□■□ _____

■□■■ □□■■ _____

■□■□ ■□■■ _____

_____ _____ _____ _____ _____

_____ _____ _____ _____

!

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Fachbereich Informatik

Programmiersprachen und Softwaretechnik

Prof. Dr. Klaus Ostermann

Julian Jabs
julian.jabs@uni-tuebingen.de

DENKEN VERSTEHEN LERNEN
Lehrerskript

Einheit M2 – WS 16

Tübingen, 8. Dezember 2016

Übersicht

Auch in dieser mathematischen Transfereinheit geht es darum, eine Verbindung der bisher gelernten algorithmischen Ideen mit mathematischen Konzepten herzustellen. Diesmal beschäftigen wir uns mit einem Additionsalgorithmus. Den SchülerInnen soll dabei klar werden, dass Addition letztenendes auch nur ein Verfahren ist, dass man in mehrere Schritte aufteilen und ausführen kann. Das kann ihnen etwa dabei helfen, Fehler beim Addieren zu vermeiden.

Zusammenfassung

1. Einstieg (10 Min.)
    a) Wiederholung
    b) Neue Wörter
    c) Teilweises Addieren
2. Aktivität: Addier-Maschine (20 Min.)
3. Abschluss (5 Min.)
    a) Kurzgespräch: Was haben wir gelernt?
4. Test: Addier-Algorithmus (10 Min.)
5. Zusätzliche Lernangebote

Lernziele

Schüler werden...

- ...ein Additionsverfahren wiederholen.
- ...lernen, wie dieses in seine einzelnen Schritte unterteilt werden kann.
- ...lernen, diesen Algorithmus sicher durchführen.

Materialien

Jeder Schüler benötigt die folgenden Materialien, die Sie zu Beginn der Stunde austeilen:

- Stift(e)
- das Arbeitsblatt “Addieren” (siehe Materialanhang)

Dazu kommt das Testblatt (siehe Materialanhang), das Sie am Ende vor dem Test austeilen.

1 Einstieg (10 Min.)

1.1 Wiederholung

Rekapitulieren Sie zusammen mit den Schülern die letzte Stunde. Sie können dabei abwechselnd Fragen stellen und die Schüler ihre Antworten in kleinen Gruppen diskutieren lassen. Mögliche Fragen sind:

- Was haben wir letztes Mal gemacht?
- Was hättet ihr gerne noch gemacht?
- Sind euch nach der vorherigen Stunde noch Fragen eingefallen?
- Was hat euch an der vorherigen Stunde am besten gefallen?

1.2 Neue Wörter

Wiederholen Sie den Begriff des Algorithmus, der in dieser Einheit eine große Rolle spielt:

- **Algorithmus**: gesprochen Al - go - rith - mus
  “eine Liste von Schritten, denen man folgen kann um eine Aufgabe zu erfüllen”

1.3 Teilweises Addieren

Eine recht einfache Möglichkeit, zwei zweistellige Zahlen zu addieren, bietet das unten beschriebene Verfahren. Es nutzt die Struktur einer zweistelligen Zahl, die aus einem “Zehner”- und einem “Einser”-Bestandteil besteht.

1. Teile die beiden Zahlen in ihre “Zehner”- und ihre “Einser”-Bestandteile: Zum Beispiel besteht 24 aus dem Zehner-Bestandteil 20 und dem Einser-Bestandteil 4, und 13 besteht aus dem Zehner-Bestandteil 10 und dem Einser-Bestandteil 3.
2. Addiere jeweils die Zehner-Bestandteile und die Einser-Bestandteile.
3. Dann addiere die beiden Ergebnisse.

Gehen Sie das Verfahren mit der Klasse zunächst mit einem einfachen Beispiel durch (gefolgt von einem etwas schwierigeren):

$$24 + 13$$

Um die Einser- und Zehner-Bestandteile besser zu erkennen, können wir das Beispiel wie folgt aufschreiben:

$$20 + 4 \quad + \quad 10 + 3$$

Wir können auch die Zehner- und Einser-Teile schon mal jeweils nebeneinander schreiben:

$$20 + 10 \quad + \quad 4 + 3$$

(Lassen Sie jeweils um das mittlere Pluszeichen etwas mehr Platz als um die anderen, damit die beiden Eingabe der Addition deutlicher erkennbar sind.)

Wir addieren zunächst die Einser-Bestandteile, die hinten stehen, und merken uns das Ergebnis:

$$4 + 3 = 7$$

Dann addieren wir die Zehner-Bestandteile und merken uns das Ergebnis; praktisch heißt das, dass wir die vorderen Ziffern addieren und sonst nichts ändern:

$$20 + 10 = 30$$

Zum Schluss addieren wir diese beiden Ergebnisse. Für dieses Beispiel reicht es dafür, die Einser- und Zehner-Bestandteile, die bei den beiden Rechnungen herauskamen, zusammenzuschreiben:

$$30 + 7 = 37$$

Diskutieren Sie mit der Klasse:

- Was war besonders einfach an diesem Beispiel?
  - Beim Addieren der Einser- und der Zehner-Bestandteile kamen wieder Einser- und Zehner-Bestandteile heraus.
  - Wir haben also wieder die beiden Bestandteile der Zahl, daher brauchen wir nichts mehr zu tun und können die Ziffern einfach hintereinander schreiben.
- Was wäre ein schwierigeres Beispiel? Wie könnten wir dabei vorgehen?

Wenn der Klasse ein schwierigeres Beispiel eingefallen ist, machen Sie damit weiter, sonst zum Beispiel mit dem unten vorgeschlagenen:

$$14 + 57$$

Wir trennen die beiden Zahlen wieder in ihre Einser-und Zehner-Bestandteile auf, und schreiben Einser und Zehner nebeneinander:

$$10 + 50 \quad + \quad 4 + 7$$

Wir addieren zunächst wieder die Einser, dann die Zehner:

$$4 + 7 = 11$$

$$10 + 50 = 60$$

Fragen Sie die Klasse, ob ihr etwas auffällt: Im Unterschied zum vorherigen Beispiel ist das Ergebnis der ersten Rechnung nicht nur ein Einser-Bestandteil! Stattdessen hat das Ergebnis auch wieder einen Einser- und einen Zehner-Bestandteil:

$$11 = 10 + 1$$

Was können wir jetzt tun, um die beiden Ergebnisse 11 und 60 zu addieren? Wir können wieder das gleiche Verfahren anwenden! Um 11 und 60 zu addieren, müssen wir also zunächst wieder deren Einser und Zehner addieren.

$$1 + 0 = 1$$

und

$$60 + 10 = 70$$

Hier haben wir jetzt wieder einen Einser- und einen Zehner-Bestandteil, also können wir wieder ganz einfach die Ziffern zusammenschreiben:

$$70 + 0 = 70$$

Erläutern Sie: Wir haben uns in einer der vorherigen Stunden mit Schleifen beschäftigt. Eine andere Möglichkeit, um etwas zu wiederholen, ist, dem Computer zu sagen, dass er wieder von vorne anfangen soll.

2 Aktivität: Addier-Maschine (20 Min.)

Jetzt entwickeln wir eine Machine zum Addieren von Zahlen, die uns die Arbeit abnimmt!

Erläutern Sie: Stellt euch vor, dass wir für unsere Maschine die folgenden Bauteile hätten:

- ein Bauteil zum **Umordnen**: schreibt die beiden Einser und die beiden Zehner jeweils nebeneinander
- ein **Einser-Rechner**: addiert die Einser-Teile
- ein **Zehner-Rechner**: addiert die Zehner-Teile
- ein Bauteil zum **Zusammenbauen**: baut einen Einser- und einen Zehner-Teil wieder zu einer Zahl zusammen

Führen Sie vor, wie die Maschine funktioniert, indem Sie so tun, als ob Sie selbst die Maschine wären. Schreiben Sie einen Additionsterm wie $24 + 13$ an die Tafel und ändern Sie ihn dann schrittweise wie folgt:

1. Umordnen: $20 + 10 \quad + \quad 3 + 4$
2. Einser-Rechner: $20 + 10 \quad + \quad 7$
3. Zehner-Rechner: $30 + 7$
4. Zusammenbauen: 37

Sie können die einzelnen Schritte untereinanderschreiben und z.B. mit horizontalen Strichen voneinander trennen. Erläutern Sie, dass man sich die Maschine so vorstellen kann, dass sie das an der Tafel stehende umschreibt: Ähnlich wie der Arm des Mitschülers in der ersten Unterrichtsstunde als Maschine bedient wurde, um Kästchen auszumalen, kann diese Maschine die Zeichen an der Tafel umschreiben.

Wie müssen wir die Bauteile zusammenbauen, um so zu addieren, wie wir es vorher gemacht haben?

Teilen Sie die Klasse in Gruppen ein und lassen Sie jede Gruppe eine solche Addier-Maschine entwickeln:

1. Teilen Sie das Arbeitsblatt aus.
2. Die Schüler schneiden die Anweisungen aus und überlegen sich, wie diese auf der Vorlage miteinander verbunden werden sollen.
3. Sie probieren die Beispiel-Additionen aus und sehen nach, ob ihre Maschine immer zum richtigen Ergebnis kommt.
   - Wenn ja, können Sie überprüfen, ob es tatsächlich die richtige Maschine ist.
   - Wenn nein, sollen die Schüler zunächst selbst versuchen, den Fehler zu finden.

Eine mögliche Kombination:
Aufspalter $\rightarrow$ Einser-Rechner $\rightarrow$ Zehner-Rechner $\rightarrow$ Zusammenbauer / von vorne beginnen

3 Abschluss (5 Min.)

3.1 Kurzgespräch: Was haben wir gelernt?

Diskutieren Sie mit den Schülern:

- Hätte man die Maschine auch anders zusammensetzen können, so dass sie den gleichen Zweck erfüllt?
  - Ist es wichtig, ob man zuerst die Einser oder zuerst die Zehner zusammenrechnet?
- Gibt es zweistellige Zahlen bei denen die Maschine nicht funktioniert? (Immer wenn das Ergebnis der Addition eine dreistellige Zahl ist.)
  - Hat jemand eine Idee, was wir an der Maschine verändern müssten, damit die Maschine auch bei solchen Zahlen funktioniert?
- Was hat euch heute am besten gefallen?

4 Test: Addier-Algorithmus (10 Min.)

Teilen Sie die Tests für Stunde M-1 aus (siehe Materialanhang). Die Schüler haben 10 Minuten, das Blatt zu bearbeiten.

5 Zusätzliche Lernangebote

Verbesserte Plus-Maschine Entwickeln Sie zusammen mit den Schülern die verbesserte Version der Addiermaschine, die auch bei dreistelligen Ergebnissen funktioniert. Dazu können Sie das Zusatz-Arbeitsblatt (siehe Materialanhang) verwenden.

Materialanhang

Es folgen in Reihenfolge

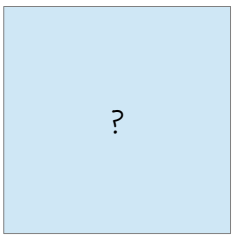
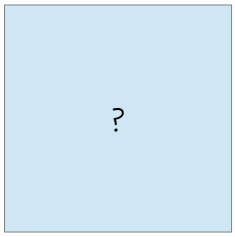
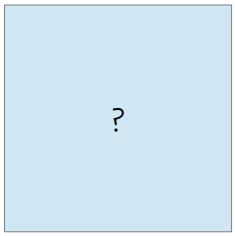
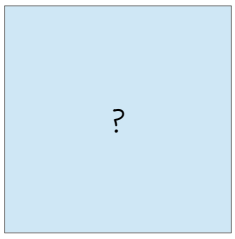
- das Arbeitsblatt für Einheit M2
- das Testblatt für Einheit M2
- das Zusatz-Arbeitsblatt

Name:_____ Datum:_____

Die Plus-Maschine

Arbeitsblatt Plus-Maschine

Schneidet die vier Teile der Plus-Maschine unten aus und legt sie auf das passende freie Feld mit einem ?. Geht die Schritte anhand des Beispiels durch und probiert so aus, ob die Maschine das Richtige tut.

1. 	Beispiel: 24 + 13 wird zu 20 + 10 + 4 + 3
2. 	Beispiel: 20 + 10 + 4 + 3 wird zu 20 + 10 + 7
3. 	Beispiel: 20 + 10 + 7 wird zu 30 + 7
4. **FALLS** nur 1 Einser und 1 Zehner:  **SONST** BEGINNE VON VORNE.	Beispiel 1: 30 + 7 wird zu 37 Beispiel 2: Bei 30 + 11 beginne von vorne.

Name:_____ Datum:_____

Einser-Rechner 1+1 = 2 1+2 = 3 ...	**Zehner-Rechner** 10 + 10 = 20 10 + 20 = 30 ...	**Aufspalten** ... 11+42 = 10 + 40 + 1 + 2 ...
	Zusammenbauen ... 10 + 1 = 11 ...	

Name:_____ Datum:_____

Die Plus-Maschine

Übungsblatt

Heute habt ihr einen Algorithmus für die Addition zweier Zahlen kennengelernt. Im Unterricht habt ihr diesen Algorithmus aus mehreren Schritten zusammengebaut. Jetzt könnt ihr alleine üben, Zahlen mit dem Algorithmus zu addieren.

Addiert die Zahlen unten mit dem Algorithmus. Schreibt dabei jeden Schritt auf. Die erste Addition wurde schon für euch durchgeführt.

45 + 13	1. Schritt: Aufspalten: **40** + **10** + **5** + **3** 2. Schritt: Einser addieren: 40 + 10 + **8** 3. Schritt: Zehner addieren: **50** + 8 4. Schritt: Nur Einser und Zehner? **Ja!** ☐ Zusammenbauen: **58** Fertig!
27 + 61	
56 + 29	
13 + 48	

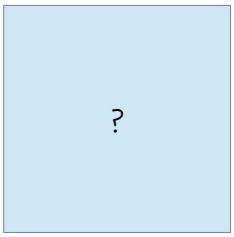
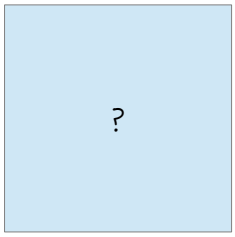
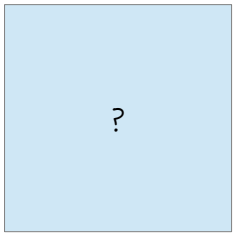
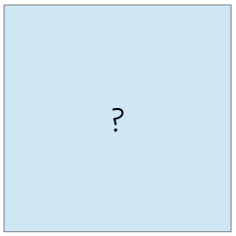
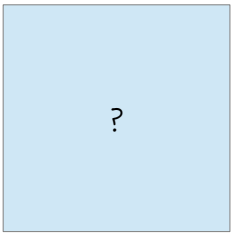
Name:_____ Datum:_____

Die Plus-Maschine

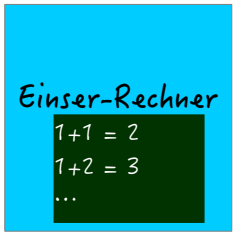
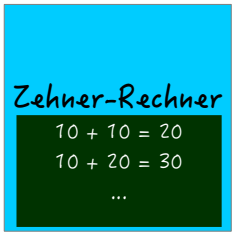
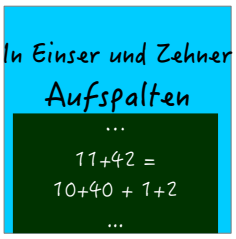
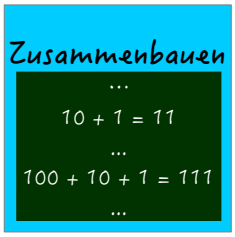
Zusatz-Arbeitsblatt Plus-Maschine

In diesem Arbeitsblatt erweitert ihr die bisherige Plus-Maschine so, dass sie mit allen zweistelligen Eingaben funktioniert.

Schneidet die vier Teile der Plus-Maschine unten aus und legt sie auf das passende freie Feld mit einem ?. Geht die Schritte anhand des Beispiels durch und probiert so aus, ob die Maschine das Richtige tut.

1. 	Beispiel: 96 + 35 wird zu 90 + 30 + 6 + 5	Beispiel: 100 + 20 + 11 wird zu 100 + 20 + 10 + 0 + 1
2. 	Beispiel: 90 + 30 + 6 + 5 wird zu 90 + 30 + 11	Beispiel: 100 + 20 + 10 + 0 + 1 wird zu 100 + 20 + 10 + 1
3. 	Beispiel: 90 + 30 + 11 wird zu 120 + 11	Beispiel: 100 + 20 + 10 + 1 wird zu 100 + 30 + 1
4. **FALLS** nur Einser, Zehner, 100er:  **SONST** 1.  2. BEGINNE VON VORNE.	Beispiel: 120 + 11 wird zu 100 + 20 + 11 Dann beginne von vorne. (siehe rechts oben)	Beispiel: 100 + 30 + 1 wird zu 131

Name:_____ Datum:_____

 Einser-Rechner 1+1 = 2 1+2 = 3 ...	 Zehner-Rechner 10 + 10 = 20 10 + 20 = 30 ...	
 In Einser und Zehner Aufspalten ... 11+42 = 10+40 + 1+2 ...	 Zusammenbauen ... 10 + 1 = 11 ... 100 + 10 + 1 = 111 ...	 100er abspalten ... 150 = 100 + 50 ...

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Fachbereich Informatik

Programmiersprachen und Softwaretechnik

Prof. Dr. Klaus Ostermann

Julian Jabs
julian.jabs@uni-tuebingen.de

DENKEN VERSTEHEN LERNEN
Lehrerskript

Einheit M2P – WS 16

Tübingen, 8. Dezember 2016

Übersicht

In der vorherigen Stunde haben die Schüler einen Additionsalgorithmus “gebastelt”. Das gleiche können sie jetzt am Rechner durch das Zusammenstecken von Blöcken tun. Anhand einer virtuellen Tafel können sie dann nachvollziehen, was in den einzelnen Schritten des Algorithmus mit den Zahlen passiert.

Zusammenfassung

1. Einführung
2. Programmieren: Die Plus-Maschine
3. Zusätzliche Lernangebote

Lernziele

SchülerInnen werden...

- ...den Additionsalgorithmus aus Einheit M2 nachprogrammieren.
- ...das Programm schrittweise ablaufen lassen und die Veränderung der Zahlen nachvollziehen.
- ...anhand der schrittweisen Ausführung mögliche Fehler erkennen und beheben.

1 Einführung

Wiederholen Sie die Übung aus der letzten Stunde (“Die Plus-Maschine”):

- Was haben wir letztes Mal für eine “Maschine” gebaut?
- Was gab es für Bestandteile der Maschine?

- Wie haben wir die Zahlen aufgeteilt?

Erläutern Sie: Jetzt werden wir das mit einigen Blöcken am Rechner programmieren. Im Fenster links, wo ihr bei den bisherigen Aufgaben eine Figur bewegen solltet, ist jetzt eine Tafel zu sehen. Wenn ihr ein paar Blöcke zusammengesteckt habt (im rechten Fenster), könnt ihr auf “Nächster Schritt” klicken und sehen, was mit den Zahlen, die an die Tafel geschrieben sind, passiert. Stellt euch vor, dass ihr mit eurem Programm quasi einen Lehrer anweist, das an der Tafel stehende umzuschreiben.

2 Programmieren: Die Plus-Maschine

siehe bereitgestelltes zip-Archiv addpw-standalone

Fordern Sie die Schüler auf, sich nochmal das Arbeitsblatt vom letzten Mal anzuschauen, wenn sie Probleme damit haben die Blöcke zusammenzusetzen. Ermuntern Sie sie, zu überprüfen, ob das Ergebnis der Addition richtig ist und falls nicht, welcher Schritt fehlen könnte.

3 Zusätzliche Lernangebote

TODO?

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Fachbereich Informatik

Programmiersprachen und Softwaretechnik

Prof. Dr. Klaus Ostermann

Julian Jabs
julian.jabs@uni-tuebingen.de

DENKEN VERSTEHEN LERNEN
Lehrerskript

Einheit C7 – WS 16

Tübingen, 9. Dezember 2016

Übersicht

Ereignisse sind eine tolle Möglichkeit um einen vorher geschriebenen Algorithmus abwechslungsreicher zu machen. Manchmal wollen wir, dass unser Programm in der Lage ist, genau dann auf den Benutzer zu reagieren wenn der Benutzer es will. Dafür gibt es Ereignisse.

Zusammenfassung

1. Einstieg (15 Min.)
    a) Wiederholung
    b) Neue Wörter
    c) Eine Reihe von Ereignissen
2. Aktivität: Das große Ereignis (15 Min.)
3. Abschluss (5 Min.)
    a) Kurzgespräch: Was haben wir gelernt?
4. Test: Das große Ereignis (10 Min.)
5. Zusätzliche Lernangebote

Lernziele

Schüler werden...

- ...Anweisungen des Lehrers wiederholen.
- ...Aktionen des Lehrers als Signale dafür erkennen, Anweisungen zu geben.
- ...üben, vorher definierte Aktionen und ereignisgetriebene Aktionen zu unterscheiden.

Materialien

Die Schüler benötigen Stift(e) und das Testblatt (siehe Materialanhang), das Sie am Ende vor dem Test austeilen.

Als Lehrer brauchen Sie das Blatt “Ereignis-Steuerung” (siehe Materialanhang).

1 Einstieg (15 Min.)

1.1 Wiederholung

Rekapitulieren Sie zusammen mit den Schülern die letzte Stunde. Sie können dabei abwechselnd Fragen stellen und die Schüler ihre Antworten in kleinen Gruppen diskutieren lassen. Mögliche Fragen sind:

- Was haben wir letztes Mal gemacht?
- Was hättet ihr gerne noch gemacht?
- Sind euch nach der vorherigen Stunde noch Fragen eingefallen?
- Was hat euch an der vorherigen Stunde am besten gefallen?

1.2 Neue Wörter

Schreiben Sie den Begriff “Ereignis” und seine Bedeutung an die Tafel. Sprechen Sie den Begriff vor und lassen Sie die Schüler wiederholen.

- **Ereignis**: Er - eig - nis
  “Ein Ereignis ist eine Aktion, die auslöst, dass etwas passiert”

1.3 Eine Reihe von Ereignissen

- Bereiten Sie die Klasse darauf vor, eine Frage zu beantworten:
  - “Ich werde euch jetzt eine Frage stellen. Ich möchte dass ihr eure Hand hebt wenn ich euch für die Antwort aufrufen soll.”
  - Stellen Sie eine einfache Frage die die meisten Ihrer Schüler beantworten können sollten, z.B.:
    - Wie viele Daumen habe ich?
    - Was ist größer, ein Vogel oder ein Pferd?
  - Rufen Sie einen Schüler auf, der seine Hand gehoben hat, und lassen Sie ihn die Antwort geben.
  - Fragen Sie danach die Klasse, wie Sie wussten dass der Schüler von Ihnen aufgerufen werden wollte.
    - Ihre Klasse wird wahrscheinlich das Heben der Hand erwähnen.
  - Erklären Sie allen: Wenn Schüler ihre Hand heben, ist das ein “Ereignis”, das auslöst, dass Sie wissen dass die Schüler aufgerufen werden wollen.
- Fragen Sie die Klasse ob ihnen noch andere Ereignisse einfallen, die etwas signalisieren.

- Sie müssen die Schüler vielleicht daran erinnern, dass Sie nicht über ein Ereignis wie eine Geburtstagsparty oder einen Ausflug sprechen.
- Wenn sie Schwierigkeiten haben, können Sie sie daran erinnern, dass ein Ereignis eine Aktion ist, die auslöst, dass etwas passiert. Beispiele:
  * Was wenn der Wecker losgeht? Was löst das aus?
  * Was wenn man bei der Mikrowelle auf "Start" drückt? Was bewirkt das?
  * Was wenn man den "An"-Knopf auf der Fernbedienung des Fernsehers drückt?
- Leiten Sie über zur Aktivität: "Heute üben wir, Programme zu verändern, indem wir Ereignisse einführen."

2 Aktivität: Das große Ereignis (15 Min.)

Erinnern Sie die Klasse an Einheit C1 (Programmieren auf Kästchenpapier) am Anfang des Kurses und wie die Schüler sich gegenseitig angewiesen haben, ein Bild aus Quadraten auszumalen:

- In dieser Aufgabe wusstet ihr von vornherein genau, was ihr eure Freunde malen lassen wolltet. So konntet ihr ein Programm schreiben, dass sie vom Anfang zum Ende führte, ohne irgendwelchen Unterbrechungen.
- In den meisten echten Programmen können wir das nicht tun, da wir verschiedene Möglichkeiten haben wollen, je nachdem was der Benutzer braucht.
  - Sagen wir, ich will, dass sich meine Spielfigur nur bewegt, wenn mein Finger auf dem Bildschirm des Smartphones ist. Ich müsste die Spielfigur so programmieren, dass sie sich *nur* bewegt, wenn ich meinen Finger auf dem Bildschirm habe.
  - Meinen Finger auf dem Bildschirm zu haben wäre dann ein "Ereignis", das meiner Spielfigur sagt, dass sie sich bewegen soll.

Erklären Sie: In den Stunden bisher haben wir Algorithmen entwickelt, um einen Freund oder Flurb zu kontrollieren, und zwar für mehrere Schritte auf einmal. Das hat Spaß gemacht und war auch nützlich, aber was ist wenn wir nicht im Voraus alles wüssten, was unser Freund tun soll? Da kommen Ereignisse ins Spiel!

Hinweis: Wenn die Schüler verwirrt sind, sprechen Sie mit ihnen über ihre Lieblingsspiele und über all die Wege, wie sie den Spielfiguren mitteilen was diese tun sollen. Weisen Sie darauf hin, dass die Spiele sehr langweilig wären, wenn sie ohne Ereignisse einfach vom Anfang zum Ende liefen.

Beginnen Sie dann mit der eigentlichen Aktivität:

1. Projizieren Sie die Ereignis-Steuerung an die Wand.
2. Entscheiden Sie mit der Klasse, was jeder einzelne Knopf bewirken soll. Vorschläge:
   - Rosa Knopf: Sagt "Wuuuu!"
   - Türkiser Knopf: Sagt "Jaaa!"
   - Purpurner Drehknopf: "Bumm!"
   - Grüner Knopf: Klatschen
   - Oranger Knopf: Stampfen

3. Üben Sie mit der Klasse: Berühren Sie die Knöpfe auf dem Overhead-Projektor, worauf die Klasse entsprechend reagieren soll.
4. Machen Sie mit ein paar Sequenzen von Knöpfen weiter. Die Schüler sollen versuchen mit ihren Geräuschen hinterherzukommen.
5. Lassen Sie die Klasse wissen, dass jedes Mal wenn Sie einen Knopf drücken ein Ereignis ist, dass sie wissen lässt, was sie als nächstes tun sollen.
6. Lassen Sie die Klasse mit einer vorher geplanten Aufgabe beginnen, bevor Sie sie dann wieder per Knopfdruck unterbrechen. Vorschläge:
   - Bis 10 zählen
   - Ein Lied singen
7. Sobald die Klasse damit angefangen hat, drücken Sie hin und wieder den Knopf.
8. Macht weiter mit dieser Mischung, bis die Klasse den Unterschied zwischen vorher geplanten Handlungen und ereignisgetriebenen Handlungen versteht.

3 Abschluss (5 Min.)

3.1 Kurzgespräch: Was haben wir gelernt?

Diskutieren Sie mit den Schülern:

- Warum müssen wir in einem Programm mit Ereignissen umgehen können?
- Fallen euch noch andere Arten von Ereignissen ein?

4 Test: Das große Ereignis (10 Min.)

Teilen Sie die Tests für diese Einheit aus (siehe Materialanhang). Die Schüler haben 10 Minuten, das Blatt zu bearbeiten.

5 Zusätzliche Lernangebote

Mein Ereignis, deine Reaktion

- Weisen Sie jedem Schüler ein Ereignis zu, auf das er/sie achten kann, und eine passende Reaktion darauf. Verketten Sie dabei die Aktionen so, dass die Reaktion jedes Schülers ein Ereignis ist, das wiederum die Reaktion eines anderen auslöst.

Ereignischaos

- Teilen Sie die Klasse in Gruppen ein. Weisen Sie jeder Gruppe eine andere Reaktion für den selben Knopf zu. Machen Sie das für alle Knöpfe der Ereignissteuerung und sehen Sie sich das Chaos an!

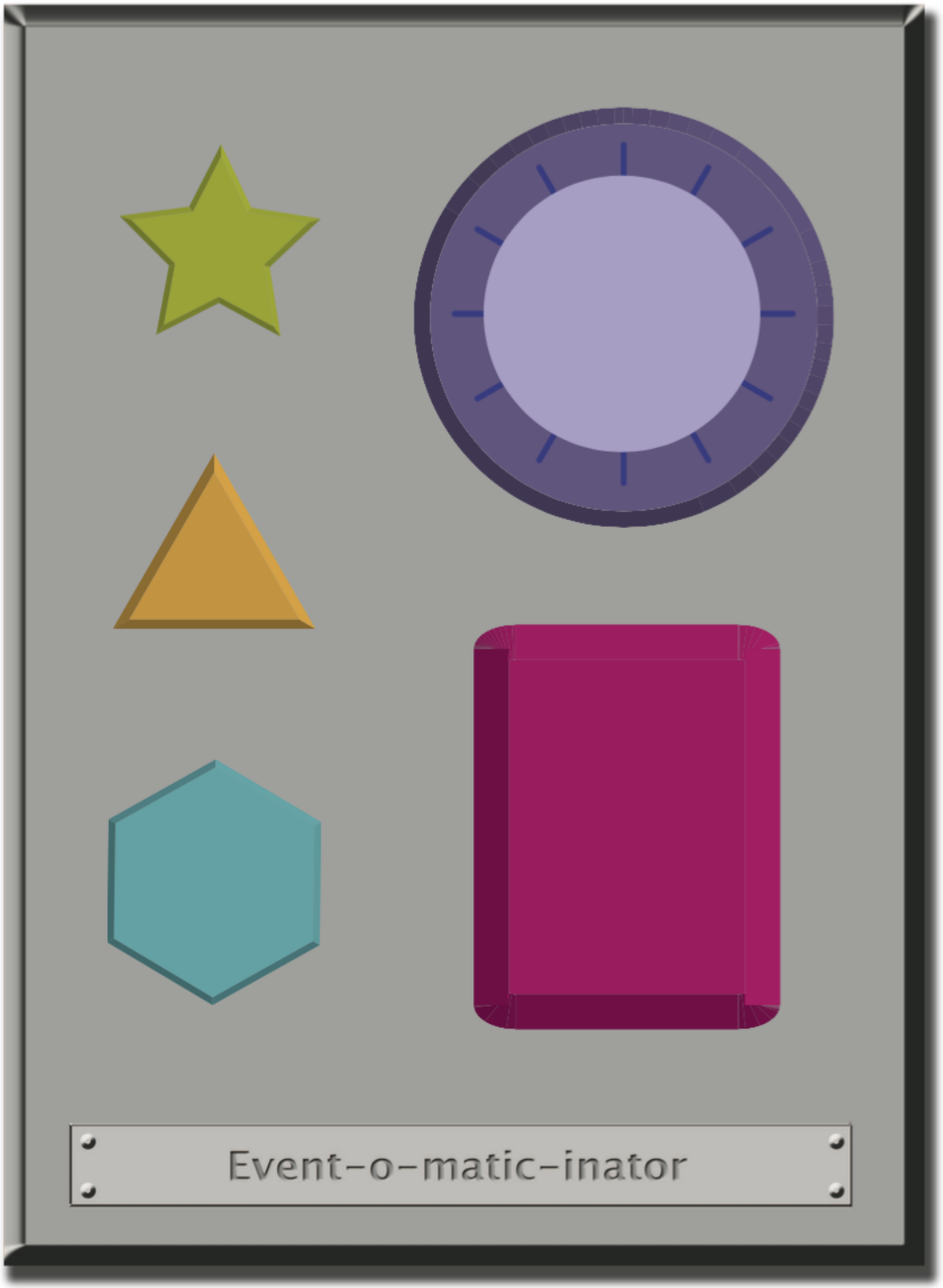
Materialanhang

Es folgen in Reihenfolge

- die “Ereignis-Steuerung”
- das Testblatt

Das große Ereignis

Ereignis-Steuerung



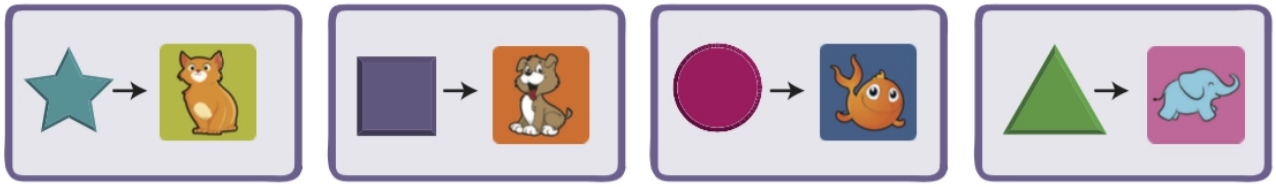
Name:_____ Datum:_____

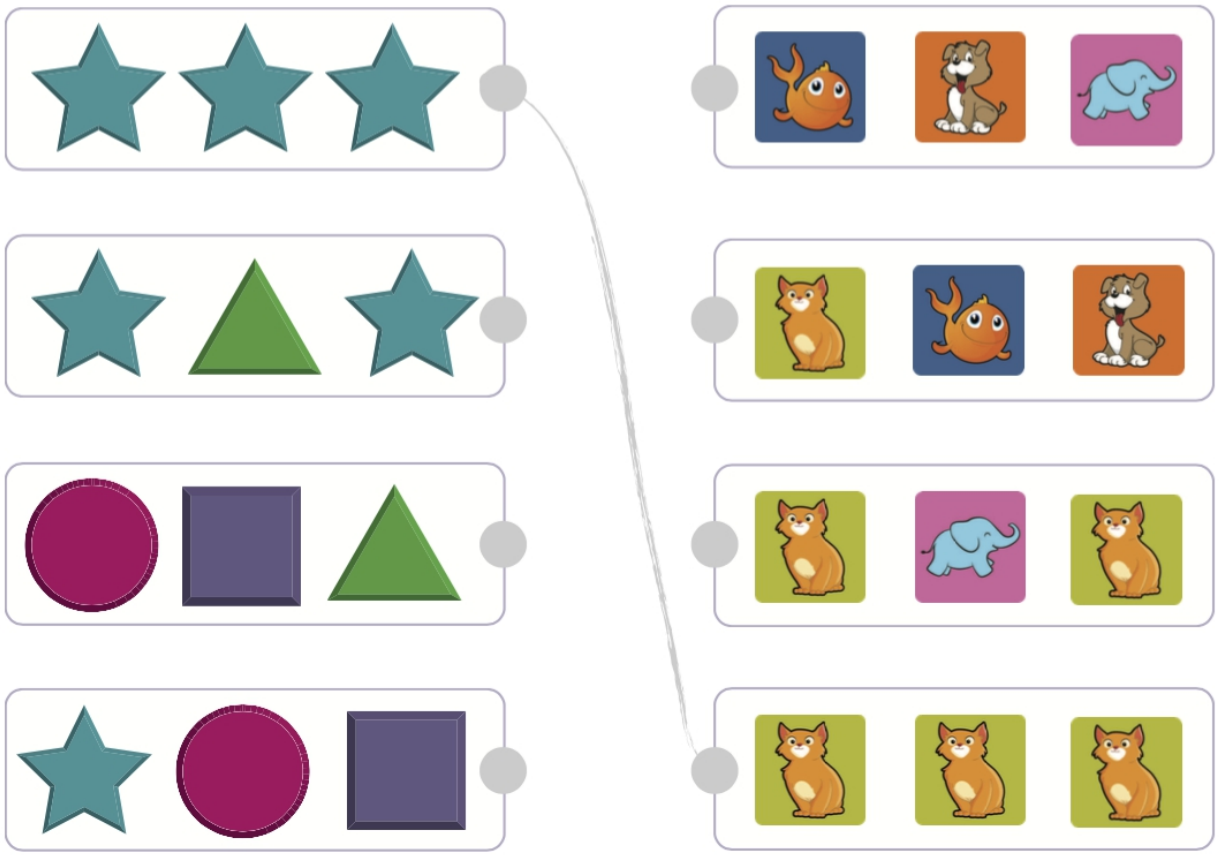
Das große Ereignis

Übungsblatt

Man hat dir eine magische Steuerung gegeben, dass das Bild im Rahmen auf deinem Tisch ändert.

Schau unten, was jeder Knopf macht. Kannst du herausfinden, welche Reihenfolge von Knopf-Ereignissen dazu führt, dass man im Rahmen die Bilder rechts sieht? Verbinde jede Reihe von Bildern mit der Reihe von Knöpfen, die diese anzeigen. Die erste wurde schon für dich gelöst.





EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Fachbereich Informatik

Programmiersprachen und Softwaretechnik

Prof. Dr. Klaus Ostermann

Julian Jabs
julian.jabs@uni-tuebingen.de

DENKEN VERSTEHEN LERNEN
Lehrerskript

Einheit C7P – WS 16

Tübingen, 9. Dezember 2016

Übersicht

In diesem Spezial-Level erschaffen die Schüler ihr eigenes “Flappy”-Spiel, indem sie Event-Handler verwenden um Mausklicks und Kollisionen von Objekten zu erkennen.

Zusammenfassung

1. Einführung
2. Programmieren: Flappy
3. Zusätzliche Lernangebote

Lernziele

SchülerInnen werden...

- ...Blöcke dem passenden Event-Handler zuordnen.
- ...ein Spiel unter Verwendung von Event-Handlern erschaffen.
- ...ein kreatives Artefakt mit anderen Schülern teilen.

1 Einführung

- Wiederholen Sie die Übung zu Ereignissen aus der letzten Stunde (Einheit C7: Das große Ereignis):
  - Wir haben Ereignisse für das Drücken der Knöpfe programmiert. Was sollte beim Drücken der Knöpfen zum Beispiel passieren?

- Erläutern Sie: Jetzt werden wir Ereignisse zu unserem Code hinzufügen. Genauer gesagt werden wir ein Ereignis für Mausklicks schaffen, sowie eines wenn der Vogel ein Objekt berührt.
  - Beim Programmieren von Videospielen nennt man diese Art von Ereignis “collision detection”; damit können wir entscheiden was passieren soll, wenn ein Gegenstand mit einem anderen kollidiert oder ihn berührt.
  - Was habt ihr für Kollisionsereignisse in Spielen gesehen?

2 Programmieren: Flappy

Level 16 von Code.org Kurs 2

Im letzten Abschnitt können die Schüler an ihrem Spiel feilen, um es einzigartig zu machen. Ermuntern Sie sie auszuprobieren, wie unterschiedlich sie die Spiele innerhalb des vorgegebenen Rahmens gestalten können.

Hinweis: Dies ist eine gute Gelegenheit, um die von den Schülern entwickelten Spiele der Klasse und vielleicht, wenn die Schüler das möchten, auch der ganzen Schule zu zeigen.

3 Zusätzliche Lernangebote

Blick unter die Haube

- Lassen Sie die Schüler sich gegenseitig ihre von ihnen entwickelten Spiele zeigen.
- Sie sollen sich dabei auch den jeweiligen Code anschauen.
- Diskutiert in der Gruppe, auf was für verschiedene Weisen die Schüler ihre Spiele programmiert haben.
  - Was hat euch überrascht?
  - Was würdet ihr gerne ausprobieren?
- Jeder Schüler wählt sich das Spiel eines anderen aus und erweitert es.

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Fachbereich Informatik

Programmiersprachen und Softwaretechnik

Prof. Dr. Klaus Ostermann

Julian Jabs
julian.jabs@uni-tuebingen.de

DENKEN VERSTEHEN LERNEN
Lehrerskript

Einheit C*P – WS 16

Tübingen, 8. Dezember 2016

Übersicht

In dieser Programmieraktivität haben die Schüler die Möglichkeit, alle ihre bisher gelernten Fähigkeiten anzuwenden, um eine animierte Geschichte zu entwerfen. Es ist an der Zeit, kreativ zu werden und im Spiele-Labor eine Geschichte zu entwickeln.

Zusammenfassung

1. Einführung
2. Programmieren: Spiele-Labor: Entwirf eine Geschichte
3. Zusätzliche Lernangebote

Lernziele

SchülerInnen werden...

- ...Aktionen identifizieren, die zu Eingabeereignissen gehören.
- ...eine animierte, interaktive Geschichte entwerfen, unter Verwendung von Sequenz, Schleifen, und Event-Handlern.
- ...ein kreatives Artefakt mit anderen Schülern teilen.

1 Einführung

Wiederholen Sie die Ereignisbehandlung, die die Schüler für Flappy verwendet haben:

- Was habt ihr beim Programmieren von Flappy für Ereignisse verwendet?

- Jetzt werdet ihr mehrere Figuren animieren um eine Geschichte zu erzählen, wobei ihr Ereignisse verwendet, die von Pfeiltasten ausgelöst werden.
- Das ist eure Chance, wirklich kreativ zu werden!

2 Programmieren: Spiele-Labor

Level 17 von Code.org Kurs 2

Diese Aktivität lässt den Schülern fast völlig freie Hand. Die Schüler haben die Freiheit, ihre eigene Geschichte zu entwerfen. Sie möchten den Schülern vielleicht strukturierte Richtlinien mit an die Hand geben, bei denen es darum geht, welche Art Geschichte sie schreiben soll, insbesondere bei Schülern, die durch zu viele Möglichkeiten überfordert sind.

Hinweis: Dies ist eine gute Gelegenheit, um die von den Schülern entwickelten Spiele der Klasse und vielleicht, wenn die Schüler das möchten, auch der ganzen Schule zu zeigen.

3 Zusätzliche Lernangebote

Blick unter die Haube

- Lassen Sie die Schüler sich gegenseitig ihre von ihnen entwickelten Spiele zeigen.
- Sie sollen sich dabei auch den jeweiligen Code anschauen.
- Diskutiert in der Gruppe, auf was für verschiedene Weisen die Schüler ihre Spiele programmiert haben.
  - Was hat euch überrascht?
  - Was würdet ihr gerne ausprobieren?
- Jeder Schüler wählt sich das Spiel eines anderen aus und erweitert es.