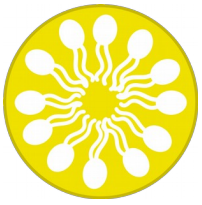
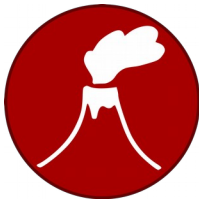
JuniorAkademie Adelsheim

15. Science Academy Baden-Württemberg 2017

Astronomie

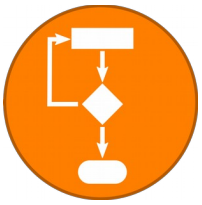


Chemie



Geophysik

Geschichte/ Amerikanistik



Informatik

TheoPrax

HOPP FOUNDATION for computer literacy & informatics

Regierungspräsidium Karlsruhe
Abteilung 7 – Schule und Bildung

Stiftung Bildung und Jugend

Dokumentation der JuniorAkademie Adelsheim 2017

15. Science Academy Baden-Württemberg

Veranstalter der JuniorAkademie Adelsheim 2017:
Regierungspräsidium Karlsruhe
Abteilung 7 –Schule und Bildung–
Hebelstr. 2
76133 Karlsruhe
Tel.: (0721) 926 4245
Fax.: (0721) 933 40270
www.scienceacademy.de
E-Mail: joerg.richter@scienceacademy.de
monika.jakob@scienceacademy.de
rico.lippold@scienceacademy.de

Die in dieser Dokumentation enthaltenen Texte wurden von den Kurs- und Akademieleitern sowie den Teilnehmern der 15. JuniorAkademie Adelsheim 2017 erstellt. Anschließend wurde das Dokument mit Hilfe von LaTeX gesetzt.

Gesamtredaktion und Layout: Jörg Richter

Vorwort

Seit nunmehr 15 Jahre findet die Junior Akademie Adelsheim im Landesschulzentrum für Umwelterziehung (LSZU) in Adelsheim statt. Der offizielle Startschuss für die diesjährige Akademie fiel schon mit dem Initiationstreffen im Januar, doch das erste Zusammentreffen der Kursleiter, Schülermentoren, Leiter der kursübergreifenden Angebote (KüAs) und 72 Teilnehmerinnen und Teilnehmern fand im Juni statt: Gemeinsam legten sie am Eröffnungswochenende den Grundstein für die erfolgreiche Kursarbeit der zweiwöchigen Sommerakademie. Im Oktober wurden Ergebnisse und Erlebnisse am Dokumentationswochenende festgehalten, das zugleich ein schönes Wiedersehen und ein Abschluss der Junior Akademie Adelsheim war.

In den Kursen, die im Rahmen der JuniorAkademie angeboten werden, bekommen die Jugendlichen die Möglichkeit, wissenschaftlichen Fragestellungen auf den Grund zu gehen, eigenständig zu arbeiten und die aufgeworfenen Fragen zu beantworten. Die Teilnehmer profitieren nicht nur von einem fachlichen Wissenszuwachs und neu erlerntem Methodenwissen, sondern entwickeln sich auch auf persönlicher Ebene weiter. Neue Freundschaften gehören ebenso zu den Dingen, die die Teilnehmerinnen und Teilnehmer von der Akademie mitnehmen, wie auch ein gestärktes Selbstbewusstsein und das Erlangen einer neuen Entwicklungsperspektive. Dies reflektieren die Jugendlichen auch für sich in der abschließenden Evaluation.

Das Erlangen eines neuen Standpunktes spiegelte sich auch in dem diesjährigen Motto der Akademie „Horizonte“ wider: Seit dem Eröffnungswochenende begleitete uns dieses Motto in Form von wunderschönen von vielen Teilnehmern eingesendeten Bildern zu diesem Thema, passenden philosophischen Gedanken und Sprüchen sowie gemeinsamen Aktionen durch die Akademiezeit.

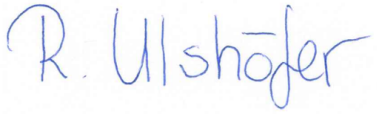
In jedem Kurs fand sich das Thema „Horizonte“ wieder: Der Astronomiekurs blickte nachts regelmäßig an den Himmelshorizont, die Geophysiker klärten darüber auf, dass es Gesteinshorizonte auch unter der Erdoberfläche gibt, und die Chemiker beschäftigten sich mit einem Horizont, der viel kleinere Dimensionen hat, den Nanopartikeln. Im Kurs Geschichte-/Amerikanistik setzten die Kursteilnehmer sich damit auseinander, wie eingeschränkt der geistige Horizont beispielsweise während der Hexenverfolgung war, in nur wenigen Disziplinen verändert sich der Horizont so schnell wie in der Informatik und im TheoPrax-Kurs vermischten sich zwei Horizonte zu einem Gesamtkonzept.

Das Motto sollte Anlass zum Reflektieren und Nachdenken sein, aber auch besonders die schönen Momente der Akademiezeit und das ganz besondere „Akademiegefühl“ begleiten und einrahmen. Wie sich der Horizont aller Teilnehmer während der Akademie verändert hat, wird jeder für sich selbst herausfinden, aber obwohl mit der Dokumentation die Akademie zu Ende gegangen ist, geht es hinterm Horizont ja immer weiter, und wir hoffen, dass für alle an der Akademie Beteiligten die Zeit in Adelsheim noch lange in guter Erinnerung bleiben wird.

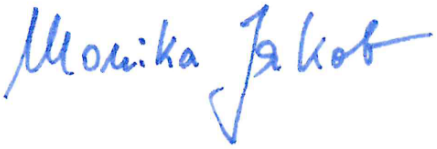
Aber jetzt wünschen wir euch viel Spaß beim Lesen, Schmökern und Erinnern!

Eure/Ihre Akademieleitung

Johanna Kroll (Assistenz)



Rebecca Ulshöfer (Assistenz)

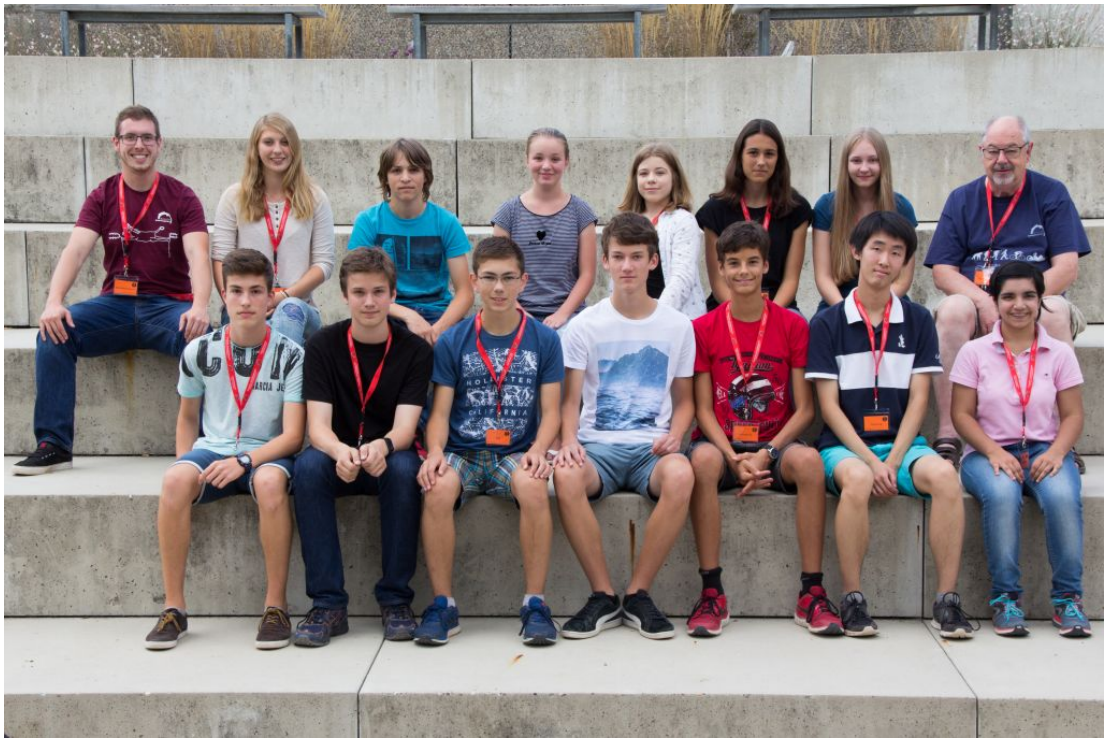


Dr. Monika Jakob

Jörg Richter

Inhaltsverzeichnis

Kurs 5 – Algorithmen zum Lösen von Problemen



Vorwort

Bernhard Petzold, Jochen Reder, Viktoria Scherb

Dass das Lösen von mathematischen Problemen viel Schreibarbeit verursachen kann, war den meisten Teilnehmern bereits aus dem Schulalltag bekannt. Ein Glück, dass wir heutzutage den Computer als Hilfsmittel benutzen können. Bevor dieser jedoch sinnvolle Ergebnisse liefert, muss das Problem von uns, den Programmierern, verstanden, in kleine Teilprobleme aufgedröselt und in einer dem Computer bekannten Sprache in Form von Algorithmen formuliert werden.

Die Sorge, die wir uns zu Beginn wegen der unterschiedlichen Vorkenntnisse der Teilnehmer gemacht hatten, waren völlig unbegründet. Ganz im Gegenteil, denn die Vielfalt und unterschiedlichen Herangehensweisen brachten völlig neue Ansätze zutage. So entstand eine ganz wundervolle Eigendynamik im Kurs, durch die unser Horizont für gänzlich neue Lösungsansätze erweitert wurde.

Mit Stolz präsentieren wir nun die Ergebnisse dieser zwölf klugen Köpfe.

Personen

Amrei war im Kurs stets motiviert, ausgeglichen und hilfsbereit. Sie fand als Erste die größte Collatz-Zahl in einem gewissen Bereich. Zudem machte es viel Spaß, mit ihr über Lösungsansätze zu diskutieren. Sie beleuchtete ein Problem oftmals von einer ganz anderen Perspektive, an die wir anderen nicht gedacht hatten.

Bianca hatte immer gute Laune und gab nie auf, sondern versuchte tapfer weiter, eine

Lösung zu finden, obwohl sie erst wenig Programmiererfahrung hatte. Sie arbeitete unter anderem mit Lukas an den Folien für die Präsentationen. Außerdem konnte man mit Bianca viel Spaß haben und wunderbar plaudern. Das Orchester bereicherte sie mit ihrer Bratsche.

Erik war immer mit voller Begeisterung dabei. Er war sogar so begeistert, dass er unseren Schlachtruf für das Sportfest immer viermal anstatt dreimal schrie, was den restlichen Kurs irritierte, aber auch amüsierte.

Außerdem hatte Erik zu allen Themen gute Lösungsvorschläge, die er sofort ausprobierte und in ein Programm implementierte. Er konnte sein großes Wissen gut vermitteln und denen helfen, die weniger Erfahrung im Programmieren hatten. In seiner Freizeit spielt Erik gerne Schach und bot deswegen auch an einem Nachmittag die Schach-KüA an.

Fynn zeigte im Kurs immer großes Interesse, vor allem als es um das Programmieren des „Game of Life“ ging, für das er eine grafische Oberfläche entwickelte. Er war nicht nur ein guter Programmierer und beherrschte vier Programmiersprachen, man konnte sich auch immer gut mit ihm unterhalten und tiefgründige Gespräche führen. Außerdem opferte Fynn seine gesamte Freizeit für die Film-KüA, um den mega coolen Akademiefilm zu drehen.

Gregor, dessen Computer man am violetten Bildschirm erkennen konnte, war immer locker drauf und brachte uns mit seinen witzigen Shirts täglich zum Schmunzeln. Er war der Hauptverbraucher der Colorado-Mischungen, die uns Jochen mitbrachte, und ein großer Verfechter des Informatik-Bibers.

Außerdem berührte er uns alle, als er mit der Theater-KüA „Die Welle“ aufführte und seine Schauspielkünste unter Beweis stellte. Zudem bot er die Pen-and-Paper-Rollenspiel-KüA an.

Konstantin, der Lakritze verabscheut, arbeitete zusammen mit Lennart an der Entwicklung einer Grünen Welle. Er war im Kurs jederzeit gut gelaunt und ließ sich von nichts entmutigen – selbst Programmierfehler sah er immer mit Humor. Außerhalb des Kurses engagierte er sich im Orchester als Posaunist und war ein fantastischer Pen-and-Paper-Spieler.

Lennart beschäftigte sich mit großem Eifer am Programm der Grünen Welle. Er gestaltete seine Programme äußerst übersichtlich, war immer mit voller Konzentration dabei und hatte einige gute Lösungsansätze. Sogar kleine Fehler, die zum Scheitern eines Programms führten, bemerkte Lennart sofort. Auch er hatte viel Freude bei der Pen-and-Paper-KüA.

Lucie ist sehr mathematikbegeistert, deswegen bereitete ihr vor allem das Collatz-Problem viel Freude. Ihr mathematisches Wissen konnte sie sehr gut beim Programmieren anwenden. Außerdem war Lucie sehr kreativ, sie dachte sich die Schlachtrufe unseres Kurses für das Sportfest aus. Nicht nur im Kurs war sie engagiert, sondern auch im Orchester und in einem Ensemble mit ihrer Geige.

Lukas war zu allen immer sehr nett und höflich. Er konnte Sachverhalte sehr gut erklären, und es hat stets Spaß gemacht, mit ihm in einem Team zu arbeiten, denn er war immer zuverlässig und hilfsbereit. Sein Perfektionismus, vor allem beim der Gestaltung der PowerPoint-Folien für die Abschlusspräsentation, trug dazu bei, dass die Folien aber wirklich gut wurden.

Natalia arbeitete sich immer gerne in neue Themen ein. Obwohl sie ein Programmierneuling war, wurde sie bald zu einem wichtiges Mitglied der Arbeitsgruppe. Sie war der Meinung, die Einführung von Einbahnstraßen und Kreisverkehren oder die Enteignung der Hausbesitzer sei die optimale Lösung für das Verkehrsproblem auf der Rohrbacher Straße.

Udani, deren ununterbrochene gute Laune uns alle ansteckte, war immer interessiert, aufgeschlossen und hatte viel Spaß dabei, Neues zu lernen. Udani war wohl der größte Fan der Tanz-KüA, sie versuchte aber auch, möglichst viele verschiedene andere KüAs zu besuchen.

Youcheng, der zu seinem Namen kein passenderes Adjektiv als „YOLO“ fand, hatte immer einen witzigen Spruch auf Lager. Im Kurs sorgte er aber nicht nur für Spaß. Er beschäftigte sich intensiv mit dem Collatz-Problem und entdeckte die „Collatz-Zwillinge“. Außerdem ist Youcheng ein hervorragender Schach- und Minesweeper-Spieler.

Viktoria, unsere Schülermentorin, war für jeden Spaß zu haben. Sie spielte mit uns witzige Spiele, wie „Gordischer Knoten“ oder „Hallo, ich bin der Hannes“. Außerdem konnte man sich mit ihr super über die verschiedensten Themen unterhalten. Auch bei der Vorbereitung der Rotationspräsentation (in der sie keinen Biber haben wollte), der Abschlusspräsentation, der Gestaltung und Organisation der Kurs-T-Shirts und beim Programmieren stand sie uns stets zur Seite. Des Weiteren veranstaltete sie an einigen Abenden die Tanz-KüA, die bei allen rege Begeisterung hervorrief.

Bernhard bezeichnet sich selbst als „Harmoniemensch“. Als Nemo verkleidet brachte er uns beim Abschlussabend alle zum Lachen. Außerdem erklärte er uns anschaulich die Java-Grundlagen, war sofort zur Stelle, wenn jemand beim Programmieren nicht weiter kam, und beantwortete unermüdlich unsere Fragen. Als Dank dafür widmeten wir ihm sogar einen Schlachtruf.

Jochen, der begeisterte Mathematiker und „Lehrer in Freiheit“, war vor allem für den mathematischen Teil im Kurs verantwortlich und veranstaltete auch die Mathe-für-Physiker-KüA. Zudem organisierte er vieles wie beispielsweise unser „Hirnfutter“ und gab uns viele nützliche Ratschläge bei der Vorbereitung der Abschlusspräsentation.

Eröffnungswochenende und seine Inhalte

Youcheng Weng, Fynn Kiwitt

Gespannt und auch etwas nervös, was uns erwarten würde, kamen wir am 30. Juni in Adelsheim zum Eröffnungswochenende an und bezogen zunächst unsere Zimmer. Nach dem Abendessen und dem Eröffnungsplenum trafen wir uns zum ersten Mal in den Kursen. In einem Klassenzimmer des Eckenberg-Gymnasiums spielten wir verschiedene Spiele zum Kennenlernen. Eines der Spiele funktionierte wie „Ich packe meinen Koffer“, nur dass wir zu unseren Namen Adjektive fanden.

Im Anschluss gaben unsere Kursleiter einen Ausblick über Themenbereiche, die wir während unserer Akadmiezeit in den Sommerferien behandeln würden.

Nun stiegen wir in das Finden von Lösungen mit Hilfe von Algorithmen ein. Unsere erste Aufgabe bestand darin, die Primzahlen aus dem Bereich 1 bis 200 zu finden. Nach kurzem Probieren fiel uns eine Möglichkeit ein, mit der man die Zahlen systematisch prüfen kann:

Unser erster Ansatz bestand darin, jede Zahl auf ihre Teilbarkeit zu prüfen. Findet man einen Teiler, so ist die Zahl keine Primzahl und wird durchgestrichen.

Der Kurs beim Auflockerungsspiel im Freien

Bald merkten wir, dass die Suche nach den Primzahlen effizienter zu lösen ist. Wir strichen die Zahl 1, da sie keine Primzahl ist. Die Zahl 2 ist eine Primzahl, ihre Vielfachen jedoch nicht mehr. Sie konnten gestrichen werden. Die nächste Zahl in der Reihe war die 3, eine Primzahl. Alle Vielfachen von 3 wurden gestrichen. Danach kam die Primzahl 5, alle Vielfachen wurden gestrichen. Ebenso verfuhren wir mit den weiteren nicht-gestrichenen Zahlen. Am Ende bleiben nur Primzahlen übrig. Dieses Verfahren nennt man das Sieb des Eratosthenes.

Um die Suche effizienter zu gestalten, überlegten wir, ob wirklich jede Zahl betrachtet

werden muss oder einige nicht schon im Voraus aussortiert werden können. Wir kamen zu dem Schluss, dass diese Schritte nur bis zu der Zahl durchgeführt werden müssen, die größer oder gleich der Quadratwurzel des zu überprüfenden Bereiches ist. In unserem Fall der Primzahlen bis 200 ist das gerundet $\sqrt{200} \rightarrow 14$. Da alle niedrigeren Zahlen bereits gestrichen oder als Primzahlen erkannt wurden müsste sie mit einer Zahl größer 14 mulitpliziert werden und würde daher aus unserem Wertebereich herausfallen.

Die Regeln lauten somit:

1. Ist die aktuell betrachtete Zahl größer als die Wurzel aus der größten gesuchten Primzahl höre auf.
2. Ist die aktuell betrachtete Zahl noch nicht gestrichen ist, streiche alle ihre Vielfachen.
3. Betrachte die nächste Zahl und gehe zu Schritt 1.

Auf diese Weise haben wir in kurzer Zeit die Primzahlen unter 200 gefunden.

Am zweiten Tag wurde uns das Collatz-Problem vorgestellt:

Erzeuge eine Zahlenfolge nach den folgenden Regeln:

1. Beginne mit einer natürlichen Zahl N
2. Ist N = 1 dann Stopp
3. Ist N gerade halbiere N und gehe zu 1.
4. Ist N ungerade multipliziere mit 3 und addiere 1 gehe zu 1.

Wir sollten nun systematisch alle Zahlen von 1 bis 50 untersuchen. Hierbei stellten wir fest:

- Die Collatz-Folgen bis zur Zahl 27 waren relativ kurz.
- Bei der Startzahl 27 entstand eine Folge mit 111 Zahlen.

Außerdem stellten wir fest, dass manche Folgen andere Folgen enthielten. Zum Verdeutlichen hier ein Beispiel: $4 \rightarrow 2 \rightarrow 1$

$5 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$

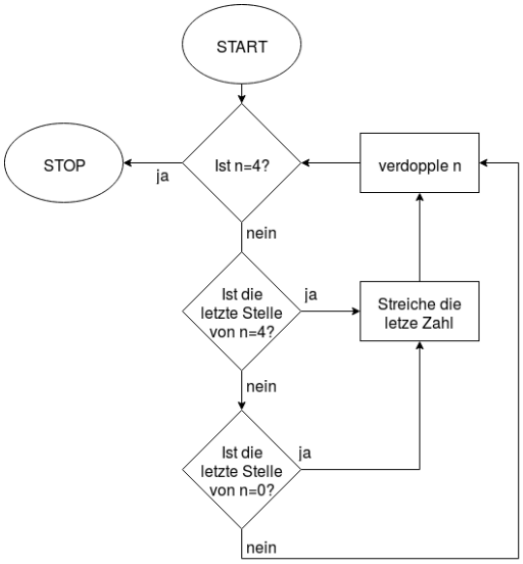
Wie man sieht, enthält die Folge der Startzahl 5 die Folge der Startzahl 4. Diese Erkenntnis ersparte uns eine Menge Arbeit, da wir so nicht bei jeder Folge bis zum Ende rechnen mussten, sondern auf vorherige Folgen zurückgreifen konnten. Alle entstandenen Folgen endeten auf 1.

Wir untersuchten noch Folgen, bei denen 3N+1 durch 5N+1 ersetzt wurde. Schon bei der Startzahl 5 stellten wir fest, dass die entstehende Folge nicht bei 1 endet.

Am Sonntag schrieben wir alle zusammen unser erstes Programm in Java. Dazu betrachteten wir eine alternative Variante, die dem Collatz-Problem ähnelt:

1. Nimm eine beliebige Zahl n
2. Wenn n = 4, STOP
3. Wenn n auf 4 endet, streiche die 4 und wiederhole ab 2
4. Wenn n auf 0 endet, streiche die 0 und wiederhole ab 2
5. Verdopple n, wiederhole ab 2

In folgender Abbildung wurde dieser Algorithmus als Flussdiagramm dargestellt.



Flussdiagramm der Collatz-Abwandlung

Wie wir programmieren lernten

Bianca Elkeries

Bei unserem ersten Kurstreffen am Eröffnungswochenende stellte sich schnell heraus, dass viele von uns noch nie programmiert hatten, andere kannten sich schon mit verschiedenen

Programmiersprachen aus, aber kaum einer hatte Erfahrung mit der Programmiersprache Java. So sah ein Großteil von uns am letzten Tag des Eröffnungswochenendes zum ersten Mal einen Programmcode, als Bernhard uns ein einfaches Programm zum Collatz Problem vorstellte. Unser Interesse war sofort geweckt, und wieder zuhause angekommen, konnten wir nicht anders, als sofort die Java Entwicklungsumgebung Eclipse herunterzuladen und selbst zu programmieren – so gut es eben ging. Bis zur Sommerakademie hatte sich der eine oder andere dann auch schon selbst etwas Wissen und Erfahrung angeeignet, und wir hatten mittlerweile alle keine Schwierigkeiten mehr, simple Programme zu schreiben. Wenn uns doch einmal Wissen fehlte oder unser Programm nicht laufen wollte, war immer jemand zur Stelle, der uns helfen konnte; oft halfen wir uns auch gegenseitig.

Grundlegende Anweisungen zum Schreiben eines Programmes sind zum Beispiel:

if-Anweisungen, um sicherzustellen, dass eine bestimmte Bedingung erfüllt ist und das Programm dementsprechend Befehle ausführen zu lassen:

```
if(Bedingung) {
    // Befehl
}
```

while-Schleifen, die einen Befehl oder eine Befehlskette wiederholen, solang eine bestimmte Bedingung erfüllt ist:

```
while(Bedingung) {
    // Befehl
}
```

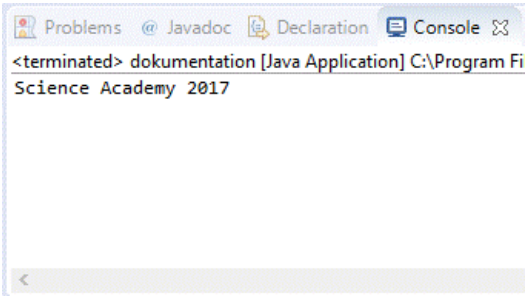
for-Schleifen durch die festgelegt werden kann, mit wie vielen Wiederholungen ein oder mehrere Befehle ausgeführt werden sollen:

```
for(Startwert; Endwert; Variable erhöhen) {
    // Befehl
}
```

System.out.println(),um etwas in der Konsole ausgeben zu lassen, wobei das „ln“ für „line“ steht, und einen Zeilensprung nach der Ausgabe einfügt:

```
System.out.println("Science
                    Academy 2017");
```

Die Ausgabe in der Konsole sieht dann aus wie folgt:



Konsolenausgabe des println-Beispiels

Am Ende der Akademie hatte dann jeder von uns an einem Programm zum „Collatz-Problem“ und zum „Game of Life“ gearbeitet sowie ein Programm zur „Grünen Welle“ nach seinen eigenen Ideen und Ansätzen umgeschrieben.

Rückblickend finde ich es unglaublich, wie viel jeder von uns in so kurzer Zeit dazulernte, obwohl wir anfangs so unterschiedlich viel – beziehungsweise wenig – Erfahrung mit Informatik hatten.

Selbst schreiben, ausprobieren und fragen – so erwarben wir die meisten Programmierkenntnisse. Ab und zu erklärte uns Bernhard nützliches zum aktuellen Thema, wie zum Beispiel Klassen und Methoden oder Arrays (siehe Programmierung des Game of Life).

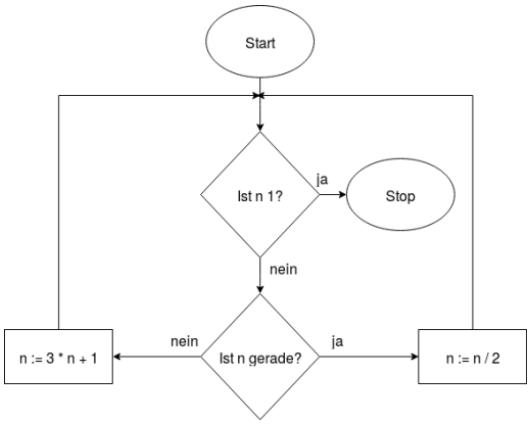
Kaum hatten wir etwas Neues gelernt, durften wir es auch schon in unseren eigenen Programmen umsetzen.

Das Collatz-Problem Einführung

Fynn Kiwitt

Mit dem Collatz-Problem hatten wir uns schon am Eröffnungswochenende beschäftigt. Lothar Collatz stellte 1937 die Vermutung auf, dass man nach der Ausführung bestimmter Regeln immer eine Zahlenfolge erhält, die mit 1 endet, egal welche natürliche Zahl man als Anfangszahl nimmt.

Diese Regeln sind in folgendem Flussdiagramm visualisiert.



Collatz-Problem als Flussdiagramm dargestellt

Ein Beispiel mit der Startzahl 7 sieht wie folgt aus:

$7 \to 22 \to 11 \to 34 \to 17 \to 52 \to 26 \to 13 \to 40 \to 20 \to 10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1$

Auch wenn es uns gelang, diese Vermutung dank unseres Programms mit mehreren Millionen Startzahlen zu überprüfen, und wir dabei keine Startzahl finden konnten, für die die Folge nicht auf 1 endete, steht ein mathematischer Beweis dieses Problems immer noch aus. Im Jahre 2011 gab es einen deutschen Mathematiker, der von sich behauptete, dieses Problem bewiesen zu haben, bei näherer Betrachtung erwies sich dieser Beweis jedoch als falsch und wurde zurückgezogen. Der Mathematiker Paul Erdős soll einmal sogar gesagt haben, dass er glaube, die Mathematik sei noch nicht bereit, um Probleme wie diese zu lösen.

Zweierpotenzen, größte Zahlen, Iterationen

Udani Melissa Klose

Als Einstieg in die Programmierung nutzten wir das zuvor erklärte Collatz-Problem. Da es von der Komplexität der einzelnen Schritte sehr simpel ist, eignete es sich hervorragen um die Grundlagen der Programmiersprache Java zu erlernen. Um das Collatz-Problem etwas genauer zu untersuchen, beschäftigten wir uns mit Besonderheiten, die in den Folgen auftreten.

Zunächst suchten wir nach Zweierpotenzen in den untersuchten Folgen, da Folgen, die eine Zweierpotenz enthalten immer auf eins enden.

Zur besseren Veranschaulichung hier ein Beispiel:

$2^5/2 = 2^4$

$\to 2^4/2 = 2^3$

$\to 2^3/2 = 2^2$

$\to 2^2/2 = 2^1$

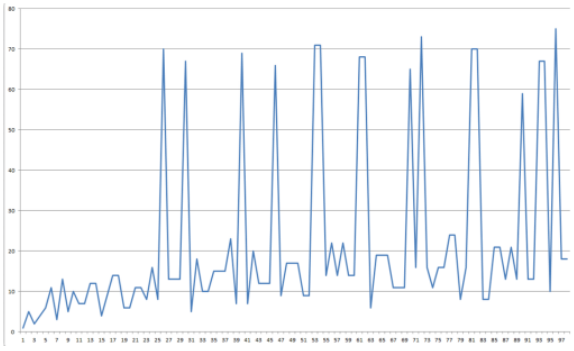
$\to 2^1/2 = 2^0$

$\to 2^0 = 1$

Diese Feststellung könnte wichtig für einen möglichen Beweis sein, dass tatsächlich, wie Lothar Collatz behauptete, jede Folge auf eins endet.

Danach ermittelten wir systematisch die Anzahl der Interaktionen, die die untersuchten Folgen durchlaufen, bis sie auf eins enden. Um diese Anzahl der Iterationen für ein beliebiges N herauszufinden, schrieben wir ein Programm, dass uns die Anzahl der Iterationen für eine beliebige Startzahl ausgab.

Aus den Daten, die unser Programm hervorbrachte, erstellten wir ein Diagramm. Bei diesem trugen wir auf der x-Achse die Startzahlen und auf der y-Achse die Anzahl der Iterationen ein. Wir ließen uns die Anzahl der Iterationen für die Startzahlen n=1 bis n=100 ausgeben.



Anzahl der Iterationen über den Startzahlen

Auf dem Diagramm sieht man, dass die Folgen mit Startzahlen bis 26 sehr wenige Iterationen benötigen, um mit eins zu enden. Bei der Startzahl 27 gibt es plötzlich einen großen Anstieg, 111 Schritte sind erforderlich. Insgesamt zeigt sich, dass bei höheren Zahlen die Anzahl der

Iterationen steigt und somit die Zahlenfolgen länger werden.

Interessant fanden wir es auch, die größte in den Collatz-Folgen erreichte Zahl zu ermitteln. Hierzu mussten wir unser Programm nur an wenigen Stellen anpassen, und es lieferte uns außer den Iterationen auch die größte Zahl der einzelnen Folgen.

„Collatz-Zwillinge“

Lucie Gatzmaga

Betrachten wir noch einmal das Diagramm, so fällt auf, das außergewöhnlich viele benachbarte Startzahlen die gleiche Anzahl an Iterationen vorweisen, allerdings erst von der Startzahl 12 aufwärts. Derartige Zahlen, entdeckt von Youcheng, nannten wir „Collatz-Zwillinge“. Der erste „Collatz-Zwillinge“ eines Zwillingspaares in dem von uns untersuchten Bereich ist immer eine gerade Zahl, bisher fanden wir dafür aber keine Erklärung.

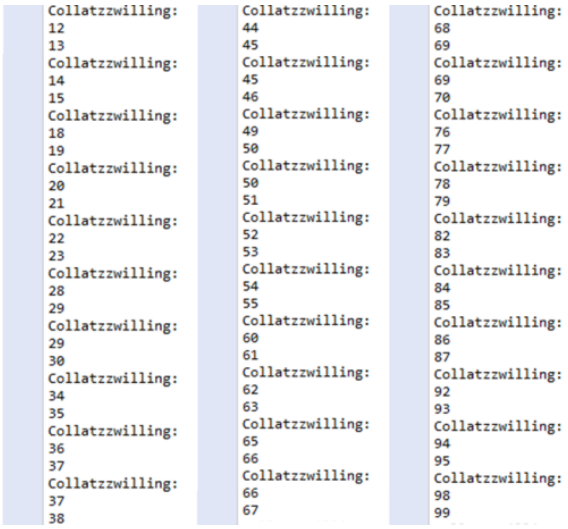
Weitere wichtige Fragen sind, ob es unendlich viele dieser „Collatz-Zwillinge“ gibt und ob der erste „Collatz-Zwillinge“ eines Zwillingspaares immer eine gerade Zahl ist. Um die Collatz-Zwillinge genauer untersuchen zu können, erweiterten wir unser Programm noch einmal so, dass es alle Collatz-Zwillinge in einem vorgegebenen Bereich ausgibt.

```
public static int schritte(int n) {

    int b = n;
    int steps = 0;

    while(b != 1) {

        if (b % 2 == 1) {
            b = 3 * b + 1;
        } else {
            b = b / 2;
        }

        steps++;
    }

    return steps;
}
```

```
public static void main(String[] args) {
  int a;

 for (int index = 1; index < 100; index++) {
    a = index;

    if (schritte(a) == schritte(a + 1)) {
     System.out.println("Collatzzwilling: ");
    }

    System.out.println(a);
    System.out.println(a+1);
  }

}
```

In der ersten Methode namens *schritte()* wird die Anzahl der Iterationen für eine Startzahl im Collatz-Algorithmus hochgezählt.

Später im Programmcode wird die Variable *a* als Startzahl definiert, um die „Collatz-Zwillinge“ mithilfe der oben angelegten Methode bestimmen zu können. Durch die for-Schleife kann der zu untersuchende Bereich festgelegt werden. In der Konsole werden schließlich alle „Collatz-Zwillinge“ geordnet aufgelistet.



Ausgabe der „Collatz-Zwillinge“ in der Konsole

Im Bereich von 1 bis 100 gibt es 30 „Collatz-Zwillinge“ und 8 „Collatz-Drillinge“. Dies sind drei aufeinanderfolgende Zahlen mit der gleichen Anzahl an Iterationen im Collatz-Algorithmus. Durch ein paar kleine Änderungen des obigen Programms kann man diese auch ganz einfach ausgeben lassen.

```
public static void main(String[] args) {
  int a;

  for(int index = 1; index < 100; index++) {
    a = index;

    if(schritte(a) == schritte(a + 1)
       && schritte(a) == schritte(a+1)) {
       System.out.println("Collatz-
                           Drilling:");
       System.out.println(A);
       System.out.println(A+1);
       System.out.println(A+2);
    }
  }
}
```

Die Ausgabe sieht so aus:

```
Collatzdrilling:
28
29
30
Collatzdrilling:
36
37
38
Collatzdrilling:
44
45
46
Collatzdrilling:
49
50
51
```

```
Collatzdrilling:
65
66
67
Collatzdrilling:
68
69
70
Collatzdrilling:
98
99
100
Collatzdrilling:
99
100
101
```

Ausgabe der „Collatz-Drillinge“ in der Konsole

Der erste „Collatz-Drilling“ ist nicht immer gerade. „Collatz-Vierlinge“ sind noch einmal seltener, im Bereich bis 100 kommen sie nur zweimal vor, im Bereich bis 500 gibt es 14 „Collatz-Vierlinge“.

Das Spiel des Lebens – Einführung

Konstantin Zeck

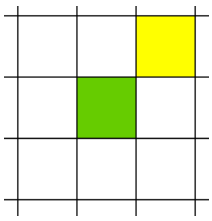
Das „Game of Life“, auch „Spiel des Lebens“ genannt, ist ein zellulärer Automat (eine Computersimulation), bei dem Zell-Lebenszyklen nach bestimmten Regeln mithilfe eines Algorithmus simuliert werden. Dadurch können unterschiedliche „Zivilisationen“ entstehen. Systeme dieser Art werden in der Biologie oder Medizin verwendet, um Zellansammlungen nachzubilden.

Das „Game of Life“ wurde 1970 von dem Mathematiker John Horton Conway entworfen. Er stellte die Aufgabe zu beweisen, dass durch endloses Wiederholen seiner Regeln theoretisch unbegrenztes Wachstum möglich ist.

Das Spielfeld ist kariert, wobei jedes Gitterfeld eine Zelle darstellt. Das Gesamtfeld kann dabei entweder begrenzt oder unbegrenzt sein. Ein „begrenztes“ Feld bedeutet, dass am Rand keine Zellen existieren. Im Gegensatz dazu befinden sich die Nachbarn der Randzellen bei einem „unbegrenzten“ Feld auf der gegenüberliegenden Seite.

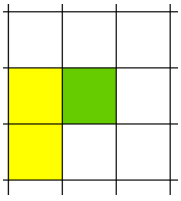
Eine Zell-Zivilisation wird in Generationen simuliert. Um die jeweils nächste Generation zu berechnen, werden zunächst alle Zellen überprüft. Die Änderungen geschehen dann alle gleichzeitig, wodurch die nächste Generation entsteht. Dabei kann der Zustand einer Einzelzelle entweder „tot“ oder „lebendig“ sein. Er wird von dem Zustand der acht Nachbarzellen beeinflusst. Je nach Anzahl lebender Nachbarn stirbt eine Zelle, bleibt am Leben oder wird lebendig.

Die erste Regel ist die Regel der „Einsamkeit“: Hat eine lebende Zelle weniger als 2 lebende Nachbarn, stirbt sie an Unterbevölkerung in der nächsten Generation.



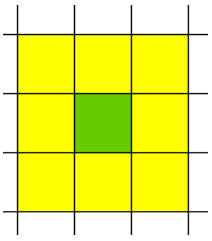
1. Regel der Einsamkeit – grüne Zelle stirbt durch Unterbevölkerung

Eine weitere Regel ist die „Überlebensregel“ und gilt für Zellen mit 2 oder 3 Nachbarn: diese bleiben in der nächsten Generation am Leben.



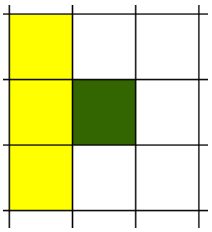
2. Regel des Überlebens – grüne Zelle überlebt diese Generation

Hat eine Zelle mehr als 3 Nachbarn, stirbt sie gemäß der dritten Regel, der „Überbevölkerungsregel“.



3. Regel der Überbevölkerung – grüne Zelle stirbt durch Überbevölkerung

Die vierte Regel besagt, dass tote Zellen mit genau 3 lebenden Nachbarn in der nächsten Generation lebendig werden.



4. Regel der Geburt – dunkelgrüne Zelle wird in der nächsten Generation geboren

Mithilfe dieser Regeln entstehen unterschiedliche Konstellationen. Die meisten Zellen sterben nach wiederholtem Anwenden dieser Regeln. Ein paar Konstellationen sind jedoch in der Lage, unendlich lange am Leben zu bleiben, sich zu bewegen oder sich zu reproduzieren.

Programmierung des „Game of Life“

Erik Klein

Mit der Programmierung des „Collatz-Problems“ erlernten wir die Java-Grundkonzepte und -konstrukte wie Schleifen, Verzweigungen und Datenstrukturen. Mit dem „Game of Life“ stiegen wir in die objektorientierte Programmierung ein. Diese ist sehr nützlich bei komplexeren Problemen. Wir lernten, Klassen aus den Java-Standardbibliotheken zu nutzen, was uns sehr viel Arbeit ersparte, und konnten auch selbst Klassen und Methoden programmieren.

Klassen und Methoden werden benötigt, wenn Teile des Programms häufiger auftauchen, um diese zu kürzen und übersichtlicher zu gestalten. Als einführendes Beispiel überlegten wir uns wiederkehrende Funktionen eines Getränkeautomaten:

Jeder Getränkeautomat kann eine unterschiedliche Anzahl an Colas haben, funktioniert aber trotzdem gleich und stammt von derselben Klasse, die sozusagen sein Bauplan ist, ab. Außerdem hat der Getränkeautomat Methoden wie *colasKaufen(anzahlColas)* oder *fantasKaufen(anzahlFantas)*, die die Anzahl der Fantas um *anzahlFantas* verringern und seine Geldmenge um *anzahlFantas*preisFanta* erhöhen. Im Programm kann jeder Automat einzeln angesprochen werden: zum Beispiel *automat1.colasKaufen(4);*, sodass beim Getränkeautomaten *automat1* vier Colas gekauft werden.

Konzentrierte Diskussion um die optimale Lösung.

Die Anwendung unserer neuen Fähigkeiten übten wir beim „Game of Life“, für das wir die Klassen *Bewohner* und *Spielfeld* programmierten. Die Klasse *Bewohner* hat die Variablen *positionAufDemSpielfeld*, *lebendZuT0* und *lebendZuT1*. Das Spielfeld hat eine Größe und eine zweidimensionale Liste (im Programm durch „[][]“ erkennbar) mit seinen einzelnen Bewohnern.

```
Bewohner[][] Welt = new Bewohner[10][10];
Arrays.fill(Welt, new Bewohner());
```

Wir erzeugten ein 10x10 großes Spielfeld mit Bewohnern.

Zu Beginn werden die lebenden und toten Bewohner festgelegt. Dann wird bis zur Beendigung des Programms das Spielfeld nach den vier Regeln neu berechnet und simuliert.

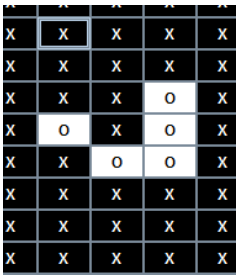
Dafür implementierten wir die Methoden *zeichnen()* und *nextIteration()*, welche nach den Regeln die nächste Iteration berechnen und ausgeben:

```
public static void nextIteration() {
  // Jeder Bewohner des Spielfelds
  for (int x = 0; x < width; x++) {
    for (int y = 0; y < height; y++) {

      // Nachbarn werden gezählt
      if (nachbarnZaehlen(x, y) < 2
      || nachbarnZaehlen(x, y) > 3) {

        // Die Zelle "stirbt"
        Welt[x][y].lebendt1 = false;

    } else if (nachbarnZaehlen(x, y) == 3) {

        // Die Zelle wird geboren
        Welt[x][y].lebendt1 = true;
      }
    }
  }
}
```

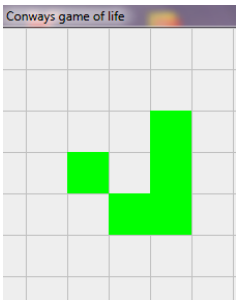
Die *nachbarnZaehlen(x , y)*-Methode prüft, wie viele Nachbarn des Bewohners auf der Position(x, y) leben und zählt sie. Je nachdem, ob bei Bewohnern am Rand des Feldes die Bewohner auf der anderen Seite mitgezählt werden oder nicht, ist das Spielfeld begrenzt oder unbegrenzt.

Jetzt wollten wir das Spielfeld noch grafisch darstellen, was einige in der Konsole und andere in einem neuen Fenster mithilfe eines „Grafical User Interface“ (GUI) umsetzten.

Hier zwei verschiedene Implementierungen:



1. Beispiel eines GUIs des Game of Life



2. Beispiel eines GUIs des Game of Life

Die Grüne Welle

Lukas Weber

Unser Ziel war, anhand des „Collatz-Problems“ und des „Game of Life“ das Programmieren zu erlernen beziehungsweise die vorhandenen Programmierkenntnisse zu erweitern, um uns anschließend an ein reales und praxisnahes Problem zu wagen. Dafür nahmen wir uns vor, eine Grüne Welle in der Rohrbacher Straße in Heidelberg zu simulieren.

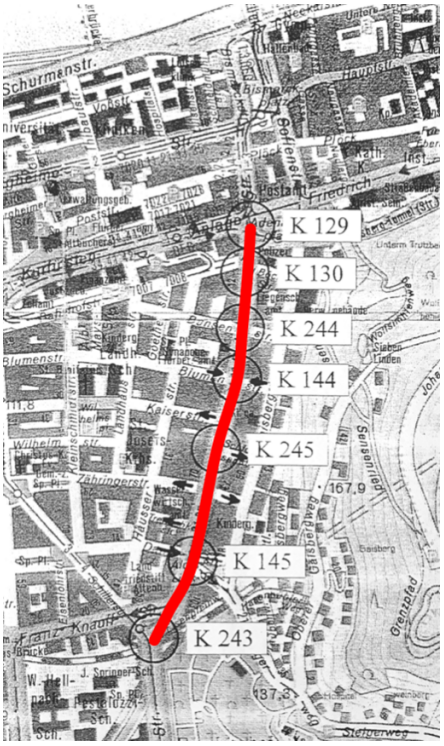
Exkursion nach Heidelberg

Da einige aus unserem Kurs in Heidelberg und Umgebung wohnen und diese Straße und das dortige Verkehrsaufkommen kennen, war diesen Teilnehmern die Rohrbacher Straße bekannt. Die grundlegende Problematik beinhaltet die Länge der Umlaufzeiten, das heißt die Länge der Ampelphasen und eine eventuelle Verschiebung der Grünphasen beziehungsweise die Anpassung an die jeweiligen Kreuzungssituation. Unser Ziel bezüglich dieses Problems war eine Verbesserung des Verkehrsflusses mithilfe von verschiedenen Ansätzen unter anderem auch für besondere Situationen wie zum Beispiel für den morgendlichen Berufsverkehr.

Die Rohrbacher Straße

Bereits vor der Akademie trafen sich die Teilnehmer unseres Kurses aus Heidelberg und Umgebung an der Ecke Bahnhofstraße/Rohrbacher Straße. Unsere Absicht an diesem Mittwochnachmittag war es, ein klares Bild von der Lage zu bekommen und vor allem wichtige Details für ein mögliches Programm herauszufinden. Zuerst legten wir uns einen Plan für die

nächsten Stunden zurecht, an dessen Anfang das Erkunden der Straße, also das Vertrautmachen mit der Umgebung stand. Hierbei berücksichtigten wir die vom Verkehrsamt festgelegten Knotenpunkte, welche die Problemstellen beschreiben.

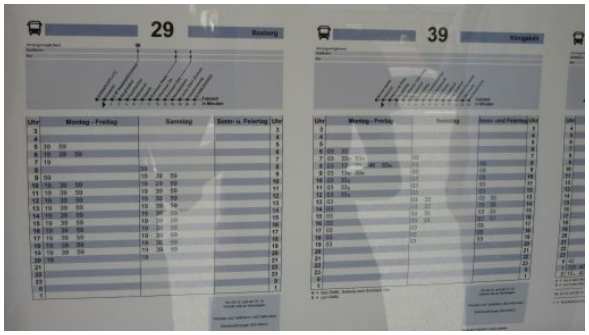


Übersichtsplan über die Knotenpunkte des untersuchten Straßenverlaufs.

Wir starteten bei Knotenpunkt K130 und gingen bis Knotenpunkt K145. Dabei notierten wir an jeder Kreuzung Auffälligkeiten, Besonderheiten, Fragen und den grundlegenden Aufbau, das heißt Fußgängerüberwege, Abbiegerichtungen und vorhandene Ampeln. Hier fielen uns zum Beispiel Induktionsschleifen an verschiedenen Stellen auf.

Eine wissenswerte Information waren Induktionsschleifen in Zufahrten

Als wir die Bushaltestelle Kaiserstraße erreichten, bemerkten wir, dass auch die Busse zu beachten sind und wir notierten ihre Taktung.



Die zwei Buslienien 29 und 39 verkehren jeweils im 20 Minutentakt beziehungsweise 30–60 Minutentakt

Nachdem wir die Straße abgelaufen waren, zählten wir 15 Minuten lang am K145 die Autos in beide Richtungen und auf die Abbiegerstraßen aufgeteilt. Natürlich kann ein Sommerferiennachmittag wohl kaum als Normalzustand gelten. Trotzdem half es uns, die Problematik besser zu durchdringen, weil wir dadurch über einen etwas längeren Zeitraum an einer komplexen Kreuzung die möglichen Fahrtwege nachvollziehen und Konfliktpunkte erkennen konnten. Als Belohnung spendierte uns Jochen ein Eis der „besten Eisdiele Heidelbergs“.

Diskussion mit Herr Bollian

Unsere Fragen, die zum Beispiel mit vorhandenen Induktionsschleifen zu tun hatten oder sich mit der aktuellen Steuerung bei der Ankunft eines Busses beschäftigten, wollten wir klären. Dazu verabredeten wir uns zu einem Gespräch mit dem Leiter des Amts für Verkehrsmanagement, Herrn Bollian, der uns sogar seine Diplomarbeit über die Rohrbacher Straße zur Verfügung stellte. An dieser Stelle nochmal ein großes Dankeschön an ihn!

Trotz einer netten Begrüßung durch Herr Bollian waren die meisten von uns erst etwas unsicher, aber schon nach kurzer Zeit entstand ein angeregtes Gespräch. Herr Bollian zeigte uns zunächst, wie man eine Straße normalerweise in der Verkehrstechnik darstellt, was für unsere weitere Arbeit essentiell war. Anhand dessen formulierten wir dann unsere Fragen

und machten erste Vorschläge zur besseren Koordinierung des Verkehrs.

Interessant war hierbei zum Beispiel der Verknüpfungsaspekt. Unsere Idee war, in Zeiten von drahtlosen Netzwerken auch die Ampeln drahtlos miteinander kommunizieren zu lassen, was laut Herr Bollian aus rein sicherheitstechnischen Gründen nicht möglich sei.

Auch über das Alter der Anlage wurden wir informiert, was dann einige unserer beobachteten Verzögerungen durch ineffiziente Ampelschaltung erklärte. Wir diskutierten noch weitere Ansätze und grundlegende Probleme, wie die nur in eine Richtung mögliche „Grüne Welle“, die uns anschaulich von Herr Bollian erklärt wurde.

Diskussionsrunde mit Herrn Bollian

An Informationen und Gesprächsstoff für unseren Kurs fehlte es uns also nach dieser Diskussion nicht. Glücklich, aber doch etwas müde vom anstrengenden Tag verabschiedeten wir uns von Herrn Bollian.

Fazit

Vor dem Amt für Verkehrsmanagement stellte sich die Gruppe dann noch einmal zum Foto auf.

Wir fassten die groben Erkenntnisse des Tags kurz zusammen, um damit eine Grundlage für die Diskussion im Kurs zu haben. Mit Vorfreude auf die Akademiezeit verabschiedeten sich alle in die unterschiedlichen Richtungen, und jeder hat seine eigenen Anregungen für die kommende Arbeit im Kurs mit nach Hause genommen.

Lösungsansätze für die Grüne Welle in der Rohrbacher Straße

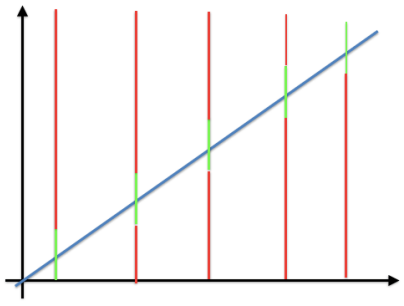
Amrei Mihan

Im Kurs während der Science Academy versuchten wir dann, eine Lösungsidee zu erarbeiten, die den Verkehr auf der Rohrbacher Straße besser regelt.

Unter Einbeziehung der Informationen, welche wir durch Herr Bollian und seine Diplomarbeit erhielten, kristallisierten sich während unseren ersten Diskussionsrunden im Kurs zur „Grünen Welle“ ziemlich schnell zwei verschiedene Lösungsansätze heraus. Diese wurden anschließend in kleineren Gruppen weiter ausgearbeitet. Im Folgenden werde ich auf die beiden unterschiedlichen Vorschläge genauer eingehen:

1. Die Grüne Welle

Die Grüne Welle ist eine immer populärer werdende, umweltfreundliche Form der Ampelschaltung. Bei einer Grünen Welle bekommt ein Auto, wenn es mit einer bestimmten konstanten Geschwindigkeit durch eine Straße fährt, an jeder Ampel grün.



x-Achse: Strecke, y-Achse: Zeit

Das Diagramm stellt eine Grüne Welle grafisch da. Die roten Striche stehen für die Rotphasen und die grünen Striche für die Grünphasen der Ampeln. Die blaue Linie stellt ein Auto dar. Die Grünphasen dauern bei jeder einzelnen Ampel gleich lang und sind zeitlich versetzt.

Um eine Grüne Welle zu konzipieren, muss man zuerst die erforderliche Dauer einer Grünphase $t_{Grünphase}$ ermitteln. Sie ergibt sich aus der Zeit, die eine Autokolonne mit einer vorher festgelegten Länge braucht, um die grüne Ampel zu passieren. Für diese Berechnung haben wir folgende Formel verwendet:

$$t_{Grünphase} = \frac{(L_{Auto} + SD) \cdot N_{Autos}}{v_{Autos}},$$

wobei L_{Auto} die Länge eines Autos ist, SD der Sicherheitsabstand, N_{Autos} die Anzahl der Autos und v_{Autos} die Geschwindigkeit der Autos. Das Ergebnis zeigt, eine wie lange Grünphase die Kolonne benötigt, um die Kreuzung zu passieren.

Als zweiten Schritt betrachtet man, wann die nächste Ampel auf Grün schalten muss. Die Versetzung der Grünphase $t_{Versetzung}$ wird mit folgender Formel berechnet:

$$t_{Versetzung} = \frac{D_{Ampel}}{v_{Autos}},$$

wobei D_{Ampel} die Distanz zur ersten Ampel ist und v_{Autos} die Geschwindigkeit der Autos.

Damit erhält man die Zeit, in der die Autokolonne die nächste Ampel erreicht, also wann die zweite Ampel nach der ersten auf Grün schalten muss.

Eine Grüne Welle ist fußgängerfreundlich, da zwischen den Grünphasen sehr viel Zeit ist, in der die Fußgänger die Straße überqueren können. Natürlich wirkt sich die Grüne Welle auch positiv für die Autofahrer aus, da diese höchstens einmal beim Durchfahren der Straße anhalten müssen.

Das gilt jedoch nur für die Autofahrer aus der einen Richtung. Die Autofahrer, die in die entgegengesetzte Richtung der grünen Welle fahren wollen, müssen wegen den sehr kurzen Grünphasen, die für sie ungünstig gesetzt sind, häufiger anhalten. Aus diesem Grund sollte man eine Grüne Welle auch nur dann anwenden, wenn in der einen Richtung deutlich weniger Autos die Straße passieren wollen. Ist das Verkehrsaufkommen gleich oder ähnlich groß, kann es in der Gegenrichtung zu Staus kommen.

Die Grüne Welle birgt auch einen weiteren Nachteil: Die Autos kommen zwar schnell durch die Straße selbst, stauen sich aber vorher am Straßenanfang, da die kurzen Grünphasen immer nur begrenzt viele Autos durch die Straße lassen und somit wegen der dann folgenden langen Rotphase viele Autos warten müssen.

2. Ampeln gleichschalten

Bei diesem Vorschlag werden zuerst alle Ampeln gleich lang auf Rot und dann gleich lang auf Grün geschaltet. Der Hauptverkehr wird also einmal komplett unterbrochen, um den Seitenverkehr und Fußgänger über die Straße zu lassen. Bei dieser Variante muss der Autofahrer genau einmal in der Straße anhalten. Das gilt für beide Richtungen und ist daher für großes Verkehrsaufkommen auf beiden Seiten gut geeignet.

Auch das Problem mit den Staus am Straßenanfang ist hier gelöst, da sich wegen der längeren Grünphase geringere Wartezeiten am Beginn der Straße ergeben. Je nachdem, wann man in die Straße fährt, muss man an unterschiedlichen Ampeln anhalten. Der Stau am Straßenanfang bei der Grünen Welle wird so also auf die komplette Straße verteilt.

Ein Nachteil bei der Gleichschaltung ist jedoch, dass den Fußgängern und dem Seitenverkehr weniger Zeit bleibt, um die Straße zu überqueren. Das könnte dann zu Staus in den Seitenstraßen führen.

Fazit

Beide Varianten haben ihre Vor- und Nachteile. Deswegen überlegten wir, beide Lösungsansätze zu kombinieren, also je nach Tageszeit und dem damit verbundenen Verkehrsaufkommen eine der beiden Lösungen anzuwenden.

Das Programm zum Erzeugen einer Grünen Welle

Lennart Leuchte

Nach der Planungsphase versuchten wir, ein Programm zur Simulation der entwickelten Lösungsansätze zu schreiben. Dabei betrachteten wir in unserem Programm mehrere Faktoren, so zum Beispiel die Fußgänger, welche die Rohrbacher Straße überqueren wollen und damit die Grüne Welle unterbrechen würden. Ebenso mussten wir den Verkehr aus den Seitenstraßen berücksichtigen, um unnötige Wartezeiten und zu kurze Grünzeiten zu vermeiden. Zur Realisierung unseres Programmes bauten wir die vorher entwickelten allgemein gültigen Formeln in das Programm ein.

Vereinte Fehlersuche im Programmcode

Zur Abschätzung des Verkehrsaufkommens zu einer bestimmten Tageszeit und der daraus resultierenden Anzahl von Autos in einer Kolonne müssen bei weiterer Forschung Analysen und Statistiken von Verkehrszählungen hinzugezogen werden. In unserem Programm schätzten wir diese jedoch grob. Unser Programm bestand aus vier Hauptbestandteilen:

- der *Straße*, die das Grundgerüst für die graphische Ausgabe bildete
- das *Auto*, das speziell dem Erstellen von Punkten diente, die wir stellvertretend für Autos benutzten und denen wir die Möglichkeit gaben, zu fahren und Ampeln zu erkennen.
- eine *Ampel*-Klasse, die uns half, die Autos zu gezielten Zeitpunkten an bestimmten Orten anhalten zu lassen.
- das *Programm*, das dazu diente, alle Klassen und Funktionen zusammenzuführen, die Grünphasen und Versetzungen zu berechnen und an die Ampeln und Autos zu übermitteln.
- die Hauptklasse mit einer *main*-Methode dient dazu, das Programm zu starten und komplette Kreuzungen zu erstellen.

Programmausschnitt, um Autos zu erzeugen:

```
for (int zautos = 0; zautos < 50; zautos++) {

  // Autos werden generiert (50)
  Auto a = new Auto (
    rohrbacherStrasse,
    - (zautos * (Auto.LAENGE
    + Auto.MIN_ABSTAND)));
  rohrbacherStrasse.fuegeAutoHinzu(a);
}
```

Außerdem erstellten wir fünf Ampeln, die sich nur in der Schaltung und dem Abstand voneinander unterschieden, sonst jedoch alle genau identisch waren:

```
Ampel K145 = new Ampel (true, 100);
erzeugeBedarfsAmpelKnopf(K145);
K145.setFussgaengerAmpel(bedarfSchalter);
rohrbacherStrasse.fuegeAmpelHinzu(K145);
```

Als nächstes berechneten wir die Länge der Grünzeiten mit der oben genannten Formel. Bei zehn Autos (je sechs Meter lang und immer mindestens drei Meter Sicherheitsabstand), die pro Grünphase durchgelassen werden sollten, wird eine Grünzeit von 7,7 Sekunden benötigt. Die Rotzeit, die nötig ist, damit alle Autos aus den Nebenstraßen passieren können, schätzten wir auf 18 Sekunden:

```
K145.addSequenzElement(7700, true);
K145.addSequenzElement(18000, false);
K145.starteSequenz();
```

„true“ bedeutete in diesem Fall grün, „false“ verkörperte eine Rotphase.

Danach führten wir Berechnungen durch, um die Länge der Versetzung herauszufinden:

```
double versetzung = (160 / 11.6666) * 1000;
```

Die Variable *versetzung1* legten wir in diesem Fall an, um die Länge der Versetzung zu berechnen. Da die Abstände der Ampeln variierten, mussten wir den Wert für jede Ampel speziell eingeben, in diesem Beispiel der Wert 160.

```
public void addSequenzElement(
                int sekunden, boolean gruen)
```

Hier ist die Funktion zu sehen, die wir einrichteten, um die Grün- und Rotphasen an die Ampeln zu übermitteln. Der Wert „sekunden“ gibt die Länge der Ampelphase an und „gruen“ gibt an, ob die Sequenz eine grüne oder eine rote Phase wird.

Rotation- und Abschlusspräsentation

Gregor Freudenberger

Nach der ersten Woche der Science Academy standen die Rotationspräsentationen an. Dabei stellte jeder Kurs der restlichen Akademie vor, was er bisher erarbeitet hatte und was er noch vorhatte.

Unser Vortrag hatte, wie auch unser Kurs, drei Themenbereiche: Das „Collatz-Problem“, „Conway's Game of Life“ und die „Grüne Welle“ in der Rohrbacher Straße in Heidelberg.

Um diese Themen für die Präsentation vorzubereiten, teilten wir uns in drei Expertengruppen mit jeweils vier Personen auf. Die Expertengruppen bearbeiteten jetzt jeweils eins der drei Themen und erstellten eine PowerPoint-Präsentation. Dann wurden die drei Präsentationen von zwei der Kursteilnehmer und unserer Schülermentorin zusammengefügt und formatiert.

Je ein Teilnehmer einer Expertengruppe bildete mit je einem weiteren aus den anderen beiden Gruppen ein Team, das zusammen die Präsentation hielt. Die Dreiergruppen begannen, die Vorträge zu üben, indem sie die Präsentation vor dem gesamten Kurs hielten. Dabei machten unsere Kursleiter Verbesserungsvorschläge und erklärten uns, dass auch Leute, die sich nicht mit solchen Themen auseinandergesetzt hatten, verstehen sollten, was wir im Kurs bisher erlebt hatten.

Kurz vor der Präsentation waren alle ein bisschen aufgeregt. Doch dank der Präsentationstipps unserer Kursleiter, unserem Fachwissen und der guten Vorbereitung konnten alle eine gute Rotationspräsentation halten und anschließend den Abend beim Bergfest genießen.

Als die Akademie sich dem Ende zuneigte, mussten sich alle Kurse auf die Abschlusspräsentation vorbereiten. Als Grundgerüst für die Abschlusspräsentation nahmen wir die Rotationspräsentation, die allerdings noch erweitert und auf den neusten Stand gebracht werden musste.

Dafür fanden wir uns wieder in den Expertengruppen zusammen, die ihr Thema noch einmal überarbeiteten und neue Erkenntnisse und Fortschritte in die Präsentation einbauten. Die neuen Teilpräsentationen wurden wieder von zwei unserer Kursteilnehmern und Viktoria zusammengefügt.

Darauf folgte üben, üben, üben. Wieder und wieder hielten die Dreiergruppen der Rotation die Abschlusspräsentation vor dem Kurs und bekamen Ratschläge und Verbesserungsvorschläge vom Kurs und unseren Kursleitern.

Da wir die Abschlusspräsentation vor unseren Eltern, Geschwistern und auch wichtigen Sponsoren halten würden, lastete ein gewisser Druck auf uns, doch wieder einmal durch gute Vorbereitung, Fachkenntnisse und gegenseitige Hilfe gingen alle Präsentationen glatt über die Bühne, und wir alle konnten entspannt zum Abschlussfest gehen.

Nachwort

Lukas Weber

Die zwei Wochen haben zwar sehr viel neues Wissen in Sachen Informatik mit sich gebracht, aber auch der Zusammenhalt, der gute Mix aus Ernsthaftigkeit und Spaß und einzigartige Erfahrungen haben diese Zeit geprägt. Jeder hat von jedem gelernt, und alle haben ihren wichtigen Teil zu unserem Projekt beigetragen. Auch

die Kursleiter und unsere Schülermentorin haben dieses Erlebnis unglaublich bereichert und waren mit ihren unersetzlichen Erfahrungen und Ratschlägen immer für uns da. Rückblickend hat sich aus einer Gruppe motivierter Jugendlicher ein echt starkes Team entwickelt, das die Aufgaben gemeinsam und mit großem Interesse gelöst hat. Danke für eine unvergessliche Akademiezeit, durch die jeder vom Informatik-Fieber angesteckt wurde und sogar ein Jugend-forscht-Projekt entstanden ist!

Sprüche

Amrei Mihan, Gregor Freudenberger

Anonym: Elena kann besser programmieren als der gesamte Informatikkurs.

Lukas: Ja, Bianca hatte ja davor gar keine Ahnung.

Viktoria: Ich liebe Stilbrüche!

Viktoria: Ich hab was gegen Biber in seriösen Präsentationen.

Viktoria: NEIN, ihr könnt nicht alle auf einmal reden.

Konstantin: Hallo, ich bin Konstantin und komme nicht aus Konstantinopel.

Gregor: Viktoria, was kannst du eigentlich ... also informatisch?

Gregor: Ich unterstütze alle Biber, auch wenn sie technisch nicht so gut entwickelt sind oder ADHS haben.

Bernhard: Ich bin ein Harmoniemensch.

Jochen: Bist du Waage?

Bernhard: Ne, Fisch, ich schwimme um meine Probleme herum.

Bernhard: Das ist meine Glücksflosse, die ist von Geburt an so.

Jochen: Jetzt stellt euch mal vor, da sitzt ein Amerikanist, der hat gerade gar nix verstanden.

Jochen: Youcheng, du schaust grad so ins Leere, kannst du mal verständnisvoll schauen?

Alle: Ach so, da habe ich ein Semikolon vergessen.

Schlachtrufe: Alle: „Javadabadu“ Alle: „static, float, und double – WIR SCHLAGEN EUCH ZU WABBEL!!!“ Alle: „Bäääärn! – hard! Bäääärn! – hard! Bäääärn! – hard!“ Erik: „Bäärn! – hard!“

Danksagung

Wir möchten uns an dieser Stelle bei denjenigen herzlich bedanken, die die 15. JuniorAkademie Adelsheim / Science Academy Baden-Württemberg überhaupt möglich gemacht haben.

Finanziell wurde die Akademie in erster Linie durch die Stiftung Bildung und Jugend, die Hopp-Foundation, den Förderverein der Science Academy sowie durch den Fonds der Chemischen Industrie unterstützt. Dafür möchten wir an dieser Stelle allen Unterstützern ganz herzlich danken.

Die Science Academy Baden-Württemberg ist ein Projekt des Regierungspräsidiums Karlsruhe, das im Auftrag des Ministeriums für Kultus, Jugend und Sport Baden-Württemberg und mit Unterstützung der Bildung & Begabung gGmbH Bonn für Jugendliche aus dem ganzen Bundesland realisiert wird. Wir danken daher dem ehemaligen Leiter der Abteilung 7 des Regierungspräsidiums Karlsruhe, Herrn Abteilungspräsidenten Vittorio Lazaridis, der Leiterin des Referats 75 – allgemein bildende Gymnasien, Frau Leitende Regierungsschuldirektorin Dagmar Ruder-Aichelin, Herrn Dr. Hölz vom Ministerium für Kultus, Jugend und Sport Baden-Württemberg sowie dem Koordinator der Deutschen Schüler- und JuniorAkademien in Bonn, Herrn Volker Brandt, mit seinem Team.

Wie in jedem Jahr fanden die etwas über einhundert Gäste sowohl während des Eröffnungswochenendes und des Dokumentationswochenendes als auch während der zwei Wochen im Sommer eine liebevolle Rundumversorung am Eckenberg-Gymnasium mit dem Landesschulzentrum für Umwelterziehung (LSZU) in Adelsheim. Stellvertretend für alle Mitarbeiter möchten wir uns für die Mühen, den freundlichen Empfang und den offenen Umgang mit allen bei Herrn Oberstudiendirektor Meinolf Stendebach, dem Schulleiter des Eckenberg-Gymnasiums, besonders bedanken.

Zuletzt sind aber auch die Kurs- und KüA-Leiter gemeinsam mit den Schülermentoren und der Assistenz des Leitungsteams diejenigen, die mit ihrer hingebungsvollen Arbeit das Fundament der Akademie bilden.

Diejenigen aber, die die Akademie in jedem Jahr einzigartig werden lassen und die sie zum Leben erwecken, sind die Teilnehmerinnen und Teilnehmer. Deshalb möchten wir uns bei ihnen und ihren Eltern für ihr Engagement und Vertrauen ganz herzlich bedanken.

Bildnachweis

S. 12: Sternparallaxe
https://commons.wikimedia.org/wiki/File:ParallaxeV2.png
Wikimedia-User WikiStefan
CC-BY-SA (https://creativecommons.org/licenses/by-sa/3.0/legalcode)

S. 15: Doppelweiche
Bild basierend auf: https://commons.wikimedia.org/wiki/File:Plektita_trakforko_14.jpeg
Wikimedia-User Asiano
CC-BY-SA (https://creativecommons.org/licenses/by-sa/3.0/legalcode)

S. 19: Dr. Thomas Müller – mit freundlicher Genehmigung

S. 27: Dr. Thomas Müller – mit freundlicher Genehmigung

S. 56: Schnittbild durch einen Vulkan
https://commons.wikimedia.org/wiki/File:Submarine_Eruption-blank.svg
Wikimedia-Nutzer Sémhur
CC-BY-SA (https://creativecommons.org/licenses/by-sa/4.0/legalcode)

S. 56: Eyjafjallajökull first crater 20100329
https://commons.wikimedia.org/wiki/File:Eyjafjallajökull_first_crater_20100329.jpg
Wikimedia-Nutzer David Karnå
CC-BY (https://creativecommons.org/licenses/by/3.0/legalcode)

S. 56: Ruinen von Pompeji
https://commons.wikimedia.org/wiki/File:Vesuvius_from_Pompeii_(hires_version_2_scaled).png
Wikimedia-Nutzer Morn the Gorn
CC-BY-SA (https://creativecommons.org/licenses/by-sa/3.0/legalcode)

S. 65: Funktionsweise Geysir
https://commons.wikimedia.org/wiki/File:Funktionsweise_Geysir_de.svg
Wikimedia-Nutzer Huebi
CC-BY-SA (https://creativecommons.org/licenses/by-sa/2.5/legalcode)

S. 78: Arthur Miller – Hexenjagd: Mit freundlicher Genehmigung des S. Fischer Verlags

S. 116: Electromagnetic spectrum c
https://commons.wikimedia.org/wiki/File:Electromagnetic_spectrum_c.svg
Wikimedia-Nutzer Horst Frank / Phrood / Anony
CC-BY-SA (https://creativecommons.org/licenses/by-sa/3.0/legalcode)

Alle anderen Abbildungen sind entweder gemeinfrei oder eigene Werke.