ZEITREISE(N) WIE FRÜHER

Modus: Teamarbeit in Kleingruppen

Denkweisen: Scientific Thinking, Computational Thinking

Beschreibung

In dieser Challenge haben Sie die Möglichkeit, Ihre Programmierfähigkeiten in fünf verschiedenen historischen Programmiersprachen zu beweisen. Gemeinsam finden Sie sich in der Zukunft wieder, in diesem Setting finden sie eine Zeitmaschine, die Sie mit einem alten Terminal offenbar programmieren können. Sie programmieren zwei Algorithmen in jeweils sechs unüblichen Programmiersprachen und führen diese Programme aus. Mindestens eine der Sprachen muss bereits älter als 50 Jahre sein, dafür stehen Ihnen Online-Umgebungen zur Verfügung. Finden Sie die richtige Programmiersprache um Zeitschleifen in Ihrer Realität aufzudecken? Viel Glück!

Einleitung

Das Ziel dieser Übungsarbeit ist, dass Sie sich mit Aspekten der *Denkweisen der Informatik* auseinandersetzen und ein besseres Verständnis entwickeln. Der Fokus liegt bei dieser Arbeit auf *Scientific Thinking* und *Computational Thinking.* Das Projekt wird ihr Verständnis dafür schärfen, wie unterschiedliche Herangehensweisen in verschiedenen Programmiersprachen Lösungswege vorgeben und teilweise erzwingen. Sie erkennen dabei wie wichtig es ist sich vor dem Programmieren bereits einen Ansatz zurecht zu legen und diesen Ansatz selbstständig zu einem fertigen Werkzeug zu integrieren.

Ablauf

Bedenken Sie während dieses Projekts, dass Sie am Ende eine etwa 5-min Präsentation halten müssen. Dokumentieren Sie Ihre Arbeit während des Projekts in geeigneter Form, damit Sie ausreichend Material für Ihre Präsentation haben.

1. Suchen Sie mittels einer Suchmaschine Ihrer Wahl nach Informationen über so genannte Zeitschleifen. Kennen Sie Bücher, Filme oder Serien in denen Zeitschleifen eine wichtige Rolle spielen, dokumentieren Sie Ihren Suchprozess und Ihre Erfahrungen. Gemeinsam überlegen Sie sich eine Definition und machen sich Gedanken wie man eine Zeitschleife in einer großen Menge an

Messwerten potentiell entdecken kann. Formulieren Sie gemeinsam eine Hypothese die man mit einem Algorithmus testen kann.

2. Stellen Sie sich vor Sie finden sich in einer fiktiven Zukunft wieder, dort stehen Sie neben einem Seismograph der seine Ausgaben in Zahlenwerten von 0 - 9 speichert. Und das schon seit vielen vielen Jahren. Lassen Sie mit einer Software Ihrer Wahl zufällige Zahlenreihen erstellen. Versuchen Sie mittels einem Algorithmus eine zufällig generierte Zahlenreihe zu erstellen die mindestens 1000 Zahlen zwischen 0 und 9 enthält. Diese Zahlen repräsentieren die Ausgabe des Seismographen. Dokumentieren Sie diesen Prozess, haben Sie eine der beiden Denkweisen bewusst genutzt? Beschreiben Sie wie und wieso.

3. Jetzt wird es spannend, in Ihrem Setting beginnt das Terminal zu blinken, Sie wissen, dass Sie damit endlich beweisen können ob Ihr Team in einer Zeitschleife gefangen ist. Eine Zeitschleife würde sich zeigen wenn eine Serie an Zahlenwerten öfter vorkommt, das wäre der Anfang einer Wiederholung! Überlegen Sie gemeinsam wie Sie diesen Algorithmus bearbeiten können. Nutzen Sie dafür Pseudocode und versuchen Sie über mehrere Versionen einen guten Ansatz zu erarbeiten bevor Sie an den Programmcode gehen. Überlegen Sie gemeinsam wie Ihnen Informatisches Denken dabei hilft, dokumentieren Sie diesen Prozess ausführlich.

4. Suchen Sie jetzt jeweils drei Programmiersprachen, von denen Sie noch nie zuvor gehört haben, oder die es bereits seit mindestens 1974 gibt. Diese Sprachen müssen die folgenden Eigenschaften erfüllen:

› Die Sprache muss in der Liste von Programmiersprachen auf Wikipedia enthalten sein.

https://de.wikipedia.org/wiki/Liste_von_Programmiersprachen

› Für die Programmiersprache muss es eine Beschreibung/Dokumentation geben, die ausreicht, um damit einfache Programme zu schreiben.

› Es muss geeignete Umgebung zur Ausführung geben. Für viele sehr alte Programmiersprachen gibt es Online-Emulatoren, oft in Javascript geschrieben; je nachdem, wieviel Erfahrung und Grundkenntnisse Sie mitbringen, können Sie auch Compiler bzw. Interpreter auf Ihrem Computer installieren. Ein weiterer Weg wäre, einen Emulator für einen alten Computer zu verwenden und eine dort vorhandene oder installierbare Programmiersprache zu nutzen.

› Mindestens eine der Programmiersprachen sollte schon vor 1974 existiert haben.

5. Beschreiben Sie Ihre Suche nach Programmiersprache und Emulator bzw. Laufzeitumgebung in Ihrer Dokumentation, und halten Sie das Ergebnis, also die sechs Sprachen, jeweils mit einer Quellenangabe für die Dokumentation und eine für Laufzeitumgebung/Emulation/Compiler/etc. fest

6. Implementieren Sie in diesen sechs Programmiersprachen den Algorithmus, der Zeitschleifen erkennen kann. Nutzen Sie dafür auch entsprechende Kommentare im Code um ihn einfacher verständlich zu machen. Lassen Sie Ihre neuen Werkzeuge über die zufälligen Zahlen laufen, finden Sie dort vermeintliche Zeitschleifen? Inkludieren Sie alle sechs Codestücke in Ihrer Dokumentation, und beschreiben Sie Ihre Erfahrungen während der Implementierung. Womit hatten Sie Probleme, und was ist Ihnen leicht gefallen? Welche unerwarteten Sprachelemente oder -strukturen haben Sie gefunden? Wie gefällt Ihnen die jeweilige Programmiersprache?

7. Sie haben jetzt Ihre Werkzeuge den ersten Tests unterzogen, wir brauchen dazu aber mehr Testdaten. Schreiben Sie in fünf der sechs Programmiersprachen auch einen Algorithmus der zufällige Zahlenreihen erzeugt. In Ihrer letzten Programmiersprache erarbeiten Sie einen Algorithmus der Ihre eigenen Erkennungstools austrickst. Welche Anpassungen müssen Sie vornehmen, dass Ihr Algorithmus ausschlägt und eine vermeintliche Zeitschleife erkennt? Würden Sie ohne Hilfe des Algorithmus in der Datenmenge die Zeitschleife erkennen?

8. Ein wesentlicher Teil Ihrer Endabgabe ist der Abschnitt *Reflexion & Feedback*. Beantworten Sie dabei die folgenden Fragen für die finale Abgabe, also nachdem Sie die Reviews geschrieben/ bekommen haben, und ergänzen Sie Ihr PDF um einen entsprechenden Abschnitt:

- Wie wurde Ihr Verständnis der gewählten Denkweise durch diese Übungsarbeit verändert?
- Inwiefern kann ein nachhaltiges Verständnis der gewählten Denkweise Ihnen im Studium oder danach im Beruf helfen?
- Welche Teile dieser Arbeit fanden Sie besonders schwer, welche zu einfach? (Bitte begründen Sie)
- Welche Aspekte dieser Arbeit haben Ihnen gut gefallen, welche würden Sie wie ändern?
- Sind Sie mit Ihrer Arbeit zufrieden?

Beachten Sie: Die Antworten auf die Fragen im Abschnitt *Reflexion und Feedback* gehen **nicht** in die Beurteilung Ihrer Arbeit ein!

Abgabe

Ihre Abgabe in TUWEL besteht aus zwei PDFs[1]:

- ein PDF mit der Präsentation. Dieses Dokument wird von Ihren Tutor_innen für Ihre Präsentation in der Workshopgruppe heruntergeladen und vorbereitet.
- ein PDF mit ausführlichen Antworten auf die Fragen zu *Reflexion und Feedback.*

Präsentation

Sie werden Ihr Projekt in einer der Workshop-Gruppen kurz präsentieren. Für diese Präsentation stehen Ihnen fünf Minuten zur Verfügung. Teilen Sie sich die Präsentation auf: Jedes Gruppenmitglied sollte einen Teil der Präsentation übernehmen.

In der zur Verfügung stehenden Zeit sollten folgende Teile Ihrer Arbeit präsentiert werden:

- eine kurze Vorstellung des Projekts und der Ziele;
- ein Überblick über Ihren Arbeitsprozess;
- die wesentlichen Ergebnisse Ihres Projekts;
- ihre Schlussfolgerungen.

Konzentierten Sie sich bei der Präsentation auf Einsichten und Erkenntnisse (sowohl aus dem Projekt als auch aus dem Punkt *Reflexion und Feedback),* von denen Sie meinen, sie könnten für die anderen Studierenden in der Workshop-Gruppe interessant sein.

---

[1] Beachten Sie bitte, dass inzwischen alle aktuellen Betriebssysteme die Erzeugung von PDFs ohne zusätzliche Software erlauben. Geben Sie keine PDFs ab, bei denen Werbung oder Wasserzeichen von Gratis-Software eingebettet ist. Für Unterstützung befragen Sie bitte die allwissende Müllhalde (das Internet) bzw. https://www.wikihow.com/Convert-a-File-Into-PDF

Beachten Sie bitte die Richtlinie zur Verwendung von generativer AI, die im *readme.pdf* zu finden ist. Wesentliche Teile der Arbeit dürfen nicht durch generative AI-Systeme verfasst werden!

Anhang: wie man einen wissenschaftlichen Artikel liest

Wissenschaftliche Artikel sind meistens nicht dafür geschrieben, von vorne bis hinten gelesen zu werden. In Ihrem Studium werden Sie aber viele wiss. Publikationen lesen. Da hilft es oft, eine klare Strategie zu haben, wie man das angeht.

Ich habe hier für Sie die Ultrakurzversion zusammengeschrieben. Sie finden nach diesem kurzen Guide einige Links zu längeren Versionen. Dieser Guide gilt für »typische« wissenschaftliche Texte, also solche, die dem üblichen Aufbau folgen.

1. Überfliegen Sie das Abstract. Sie werden dann verstehen, um was es im Artikel geht, warum die Arbeit verfasst wurde, und in wenigen Worten üblicherweise auch, was das Ergebnis der Arbeit war. Das hilft Ihnen, den Rest besser einordnen zu können.

2. Lesen Sie jetzt den letzten Abschnitt des Papers, üblicherweise »Conclusions« oder »Discussion« genannt. Damit sollten Sie jetzt wissen, was die Autor_innen gemacht haben, und warum Sie es gemacht haben. Sie wissen auch, was dabei herausgekommen ist.

3. Der Abschnitt vor den Schlussfolgerungen sind üblicherweise »Results«. Überfliegen Sie diesen Teil, um zu sehen, wie relevant er für Sie ist.

4. Sehen Sie sich die Abbildungen an. In groben Zügen können Sie jetzt verstehen, um was es in diesem Paper geht, und was die Autor_innen gemacht haben. Zugegeben, das wird einfacher, je öfter Sie es machen.

5. Es sollte einen Abschnitt geben, der die Methodologie beschreibt, meistens »Methods« o.ä. Versuchen Sie grob zu verstehen, wie die Autor_innen gearbeitet haben (qualitativ, quantitativ, etc.).

Sie haben jetzt ein gutes Bild davon, um was es geht, und können entscheiden, ob Sie den Rest des Papers auch lesen wollen (zB. weil es relevant oder interessant ist). Eventuell ist aber auch nur noch der Abschnitt »Related Work« (o.ä.) für Sie spannend, weil Sie dort weitere Papers finden, die sich mit derselben oder einer ähnlichen Fragestellung beschäftigen – und vielleicht suchen Sie ja genau solche Arbeiten.

Weitere Guides:

- https://drewdennis.medium.com/how-to-read-scientific-papers-quickly-efficiently-e7030c4018fa
- https://www.bmj.com/about-bmj/resources-readers/publications/how-read-paper
- https://paperpile.com/g/read-scientific-paper/