ipso! Bildung seit 1876

Praxisarbeit: Entwicklung eines Labyrinth-Spiels in C

Konzeption, Implementierung und Dokumentation eines Konsolenspiels mit modularer Struktur und Diagrammen

Management Summary – Praxisarbeit Programmiertechnik (C)

In dieser Arbeit wurde ein kleines, textbasiertes Labyrinth-Spiel in C entwickelt. Ziel war es, ein Programm zu erstellen, das die Kerninhalte der Programmiertechnik praxisnah umsetzt: Datenstrukturen, Algorithmen, Speicherverwaltung, Modularisierung, Tests und Dokumentation.
Der Spieler (P) bewegt sich dabei mit den Tasten W, A, S, D durch ein zufällig generiertes Spielfeld zum Schatz (T). Hindernisse (O) blockieren den Weg. Das Projekt konnte zeigen, dass alle Kernfunktionen – Spiellogik, Eingabeverarbeitung und Siegesbedingung – zuverlässig umgesetzt wurden.

Vorgehen und Qualitätssicherung

- Zunächst wurde ein Datenmodell mit Schnittstellen definiert und schrittweise implementiert.
- Die Spiellogik und Eingaben wurden getestet, Randfälle wie ungültige Züge oder Randkollisionen früh berücksichtigt.
- Mit einem Makefile erfolgte der automatisierte Build und Test.
- Zur besseren Übersicht wurden Diagramme zur Architektur und Spiellogik erstellt.

Praxistauglichkeit und Nutzen

Das Spiel ist leicht reproduzierbar (make, ./labyrinth). Durch die modulare Struktur ist es wartbar und erweiterbar. Es eignet sich außerdem als Übungs- und Prüfungsgrundlage, da es die zentralen Aspekte der Programmierung in C verständlich darstellt.

Grenzen und Risiken

- Das Spiel ist bewusst textbasiert gehalten (keine grafische Oberfläche).
- Bei sehr hoher Hindernisdichte kann das Spielfeld unübersichtlich wirken.
- Erweiterte Funktionen wie mehrere Level oder ein Punktesystem wurden nicht umgesetzt.

Diagramme (Überblick)

Zur Verdeutlichung wurden mehrere Diagramme erstellt, die den Aufbau und Ablauf erklären:

1. Modul-Überblick (Dateien & Abhängigkeiten)
2. Datenmodell (C-Structs)
3. Spiellogik (Ablauf eines Spielzugs)
4. Übersicht Eingaben
5. Übersicht Spielfeld
6. Übersicht Renderer
7. Übersicht Zugauswertung

(GitHub-Repository: Diagramme auf GitHub)

Nächste Schritte (Roadmap)

- Einführung von Schwierigkeitsstufen und Punktesystem
- Erweiterung um eine grafische Benutzeroberfläche
- Zusätzliche Tests zur Verbesserung von Stabilität und Robustheit

Fazit

Das Projektziel – ein funktionierendes Konsolenspiel mit klarer Spiellogik, zuverlässiger Eingabeverarbeitung und verständlicher Dokumentation – wurde erreicht.
Die Arbeit zeigt, dass sich mit systematischem Vorgehen, Einsatz von Diagrammen und schrittweiser Umsetzung ein vollständiges Programm entwickeln lässt, das sowohl technisch korrekt als auch nachvollziehbar dokumentiert ist.

ipso! Bildung seit 1876

Inhaltsverzeichnis

1 Einleitung

Die vorliegende Praxisarbeit beschäftigt sich mit der Entwicklung eines einfachen Labyrinth-Spiels in der Programmiersprache C. Ziel des Projekts war es, ein lauffähiges Konsolenspiel zu programmieren, das übersichtlich aufgebaut ist und die im Unterricht behandelten Inhalte praktisch anwendet. Dabei standen vor allem die Strukturierung des Codes, die Verwendung von Header- und Quellcodedateien sowie das Arbeiten mit Modulen im Vordergrund.
Das Projekt eignet sich besonders gut, um zentrale Konzepte der Programmierung zu üben: Variablen und Datenstrukturen, Steuerung von Abläufen mit Bedingungen und Schleifen sowie die Eingabe und Ausgabe über die Konsole. Zudem sollte der Umgang mit Versionsverwaltung und Dokumentation in einem Repository geübt werden, da dies in der heutigen Arbeitswelt zu den grundlegenden Kompetenzen gehört.
Im Rahmen der Arbeit wurden neben dem Programmcode auch verschiedene Diagramme erstellt, welche den Aufbau und die Logik des Spiels veranschaulichen. Diese helfen dabei, den Entwicklungsprozess verständlich darzustellen und die Zusammenhänge zwischen den Modulen sichtbar zu machen.
Das Ziel der Arbeit ist es, ein funktionsfähiges und nachvollziehbar dokumentiertes Spiel vorzustellen, das die geforderten Kriterien erfüllt und gleichzeitig die erworbenen Kenntnisse in einer praktischen Umsetzung demonstriert.

2 Projektbeschreibung

In diesem Kapitel wird das Projekt näher beschrieben. Zuerst werden die Ziele und Anforderungen festgelegt, welche die Grundlage für die Umsetzung bilden. Danach wird auf die Arbeitsschritte eingegangen, die während der Entwicklung durchgeführt wurden. So entsteht ein klarer Überblick über die Idee und die Vorgehensweise.

2.1 Ziel und Anforderungen

Das Ziel des Projekts war die Entwicklung eines einfachen Konsolenspiels in der Programmiersprache C. Das Spiel sollte ein Labyrinth darstellen, in dem sich eine Spielfigur mit den Tasten **W, A, S, D** bewegen lässt. Zusätzlich musste eine Möglichkeit eingebaut werden, das Spiel jederzeit mit der Taste **Q** zu beenden.
Wichtige Anforderungen waren außerdem:

- Das Spielfeld wird zufällig generiert und enthält Hindernisse sowie einen Schatz.
- Der Spieler darf sich nur auf freie Felder bewegen. Kollisionen mit Wänden oder Hindernissen müssen verhindert werden.
- Die Position von Spieler und Schatz muss klar dargestellt werden.
- Sobald der Schatz gefunden wurde, soll eine eindeutige Siegesmeldung erscheinen.

Neben der reinen Funktionalität war auch die saubere Strukturierung des Codes eine zentrale Anforderung. Dazu gehört die Trennung in Header-Dateien, Implementierung und die Nutzung eines Makefiles zur einfachen Kompilierung. Ergänzend sollten Diagramme erstellt werden, um die Logik des Spiels und den Aufbau des Programms verständlich darzustellen.

2.2 Vorgehensweise / Planung

Zu Beginn stand die Frage, wie das Spiel grundsätzlich aufgebaut sein soll. Ich habe mir zuerst überlegt, welche Funktionen unbedingt nötig sind: ein Spielfeld, eine Spielfigur, die Bewegung über Tasten und ein Schatz, der gefunden werden muss. Daraus entstand eine einfache Planung, welche Module das Programm enthalten soll.
Der nächste Schritt war die Aufteilung in Dateien. Es sollte eine klare Trennung zwischen der Hauptdatei main.c, der Header-Datei labyrinth.h und der Implementierung in labyrinth.c geben. Auf diese Weise konnte ich die einzelnen Teile übersichtlicher halten. Zusätzlich habe ich ein **Makefile** eingeplant, damit das Programm später mit nur einem Befehl kompiliert werden kann.
Bei der Planung habe ich außerdem berücksichtigt, dass Tests notwendig sind. Deshalb wurde ein eigener Ordner für Testdateien erstellt, in dem einfache Überprüfungen der Spiellogik durchgeführt werden können. So konnte ich Fehler schneller finden und korrigieren.
Ein wichtiger Teil war auch die Dokumentation. Schon während der Planung habe ich festgelegt, dass ich Diagramme einsetzen werde, um die Struktur und Abläufe darzustellen. Dadurch sollte es einfacher sein, das Projekt für andere verständlich zu machen.
Insgesamt habe ich die Vorgehensweise so gestaltet, dass immer zuerst ein kleiner Teil programmiert und getestet wurde, bevor ich mit dem nächsten weitergemacht habe. Das hat geholfen, Fehler frühzeitig zu erkennen und das Projekt Schritt für Schritt aufzubauen.

3 Technische Umsetzung

In diesem Kapitel wird beschrieben, wie das Projekt technisch umgesetzt wurde. Dabei stehen vor allem die Struktur des Codes, die Aufteilung in einzelne Dateien sowie die verwendeten Hilfsmittel im Vordergrund. Zusätzlich wird erläutert, wie die Diagramme zur besseren Verständlichkeit beigetragen haben und welche Überlegungen während der Programmierung gemacht wurden.

3.1 Struktur des Codes

Der Code wurde bewusst in mehrere Dateien aufgeteilt, um Übersicht und Wartbarkeit sicherzustellen. Die zentrale Datei ist **main.c**, welche den Einstiegspunkt des Programms enthält. Dort wird das Labyrinth gestartet und die Steuerung des Spiels umgesetzt.
Die Datei **labyrinth.h** enthält die Funktionsdeklarationen und stellt damit die Schnittstelle zwischen Hauptprogramm und Implementierung dar. Dadurch wird klar geregelt, welche Funktionen von außen sichtbar sind.
Die eigentliche Logik des Spiels ist in **labyrinth.c** implementiert. Dort befinden sich die Funktionen zur Erstellung des Labyrinths, zur Bewegung der Spielfigur sowie die Abfrage, ob der Schatz gefunden wurde oder nicht.
Zusätzlich wurde ein **Makefile** erstellt, mit dem das Programm einfach kompiliert werden kann. So genügt ein einziger Befehl, um das Spiel zu bauen, ohne dass alle Dateien manuell angegeben werden müssen.
Für die Überprüfung der Spiellogik wurde ein eigener **Ordner für Tests** angelegt. Darin können kleinere Testprogramme abgelegt werden, um einzelne Funktionen unabhängig vom Hauptprogramm zu prüfen.
Die Struktur entspricht damit einer sauberen Trennung zwischen **Benutzeroberfläche (main.c)**, **Schnittstelle (labyrinth.h)** und **Implementierung (labyrinth.c)**. Dadurch bleibt das Projekt auch bei Erweiterungen verständlich und gut wartbar.

3.2 Diagramme

Um die Abläufe und die Struktur des Programms besser verständlich darzustellen, habe ich zu den wichtigsten Teilen des Projekts verschiedene Diagramme erstellt. Diese helfen dabei, die Logik des Spiels sowie die Abhängigkeiten zwischen den Modulen und Daten klar aufzuzeigen.

In **Abbildung 1** ist ein Überblick über die Dateien und deren Abhängigkeiten dargestellt. Man erkennt, wie die Hauptdateien main.c, labyrinth.h und labyrinth.c miteinander verbunden sind und welche externen Bibliotheken (stdio.h, stdlib.h, rand()) eingebunden wurden. Auch die Anbindung der Tests ist hier sichtbar.

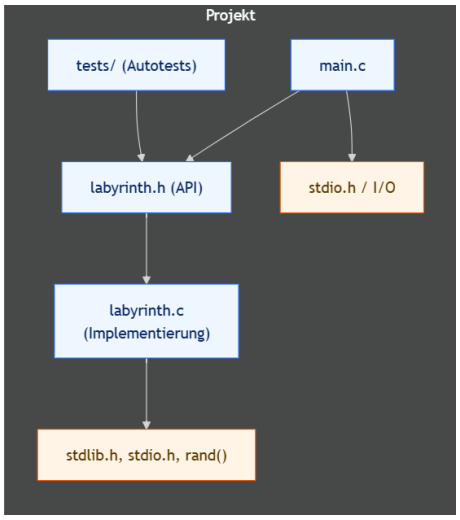


Abb.1: Modul-Überblick
(Dateien & Abhängigkeiten)

Abbildung 2 zeigt das Datenmodell in Form der verwendeten C-Structs. Damit wird deutlich, wie die Spielfeld-Daten, die Positionen von Spieler und Schatz sowie die Hindernisse in Strukturen zusammengefasst sind. So wird die Organisation der Daten übersichtlich gelöst.

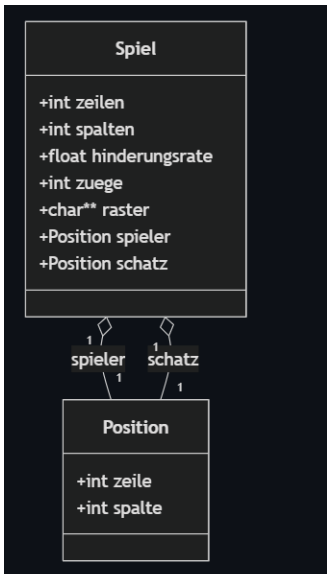


Abb.2: Datenmodell (C-Structs)

Der grundlegende Ablauf des Spiels ist in **Abbildung 3** dargestellt. Dort ist zu sehen, wie das Programm startet, das Spielfeld erstellt wird und die Steuerung über Eingaben erfolgt. Ebenfalls wird geprüft, ob ein gültiger Zug möglich ist oder ob das Spielende erreicht wurde.

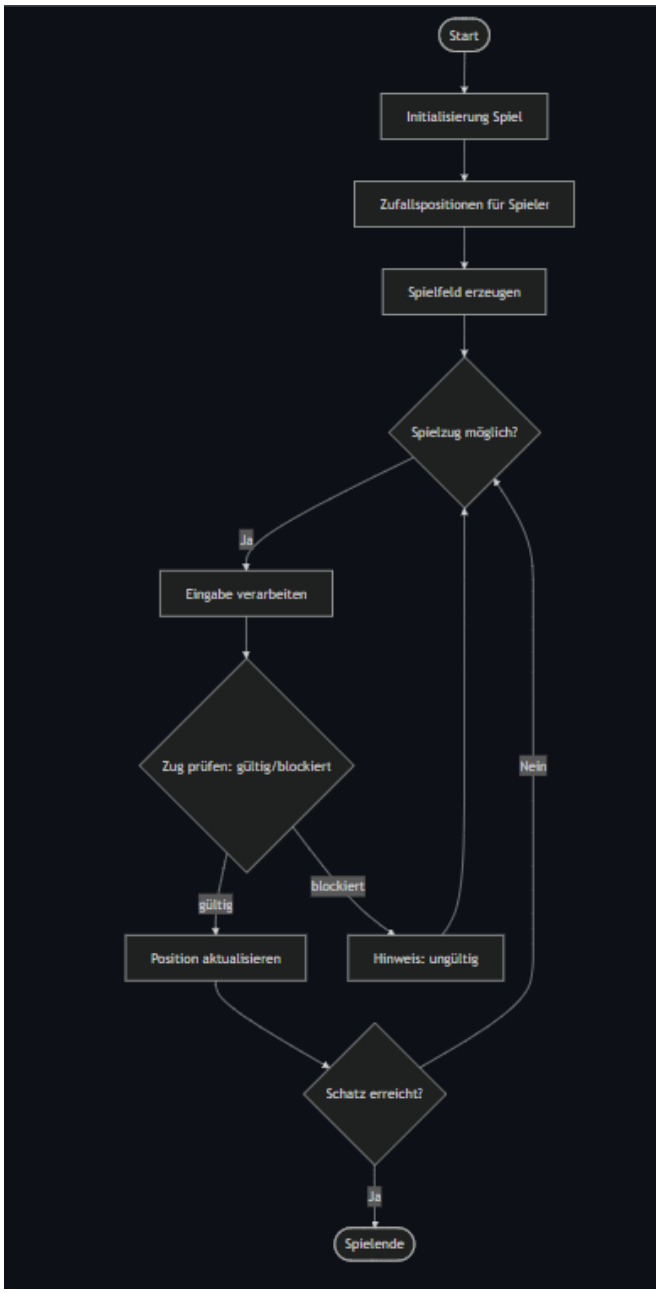


Abb.3: Übersicht Spiellogik

In **Abbildung 4** ist die Eingabelogik dargestellt. Das Diagramm zeigt, wie Tasteneingaben wie W, A, S und D für Bewegungen verarbeitet werden. Bei ungültigen Eingaben wird eine Fehlermeldung ausgegeben, und mit Q kann das Spiel beendet werden.

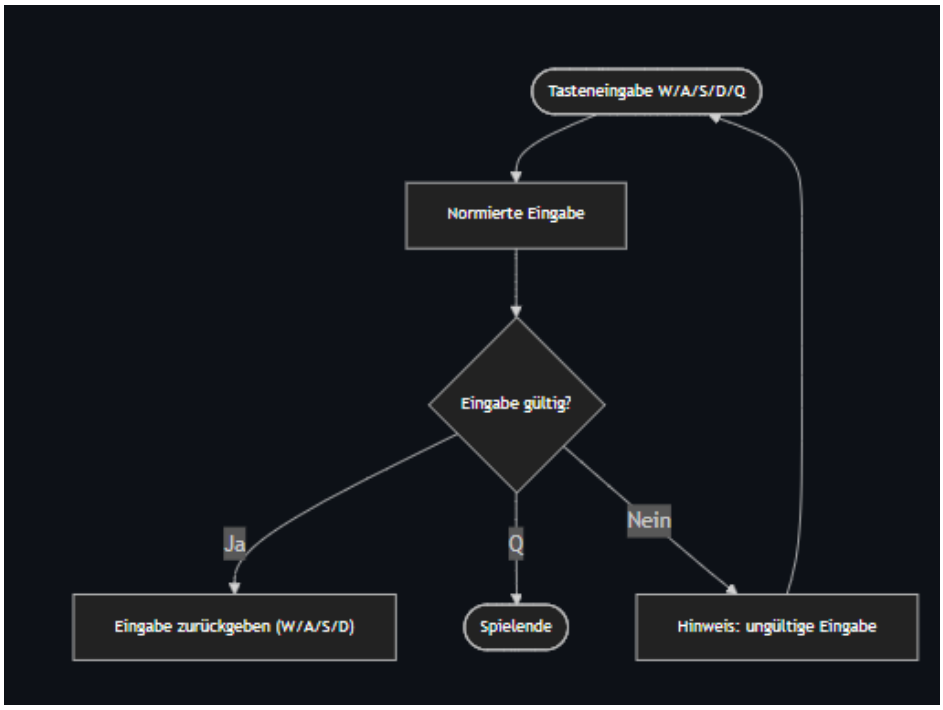


Abb.4: Übersicht Eingabe

Wie das Spielfeld selbst aufgebaut wird, zeigt **Abbildung 5**. Hier ist der Ablauf vom Anlegen des Gitters über das Platzieren von Hindernissen bis hin zur Positionierung des Spielers und des Schatzes zu sehen. Am Ende steht das vollständige Spielfeld bereit.

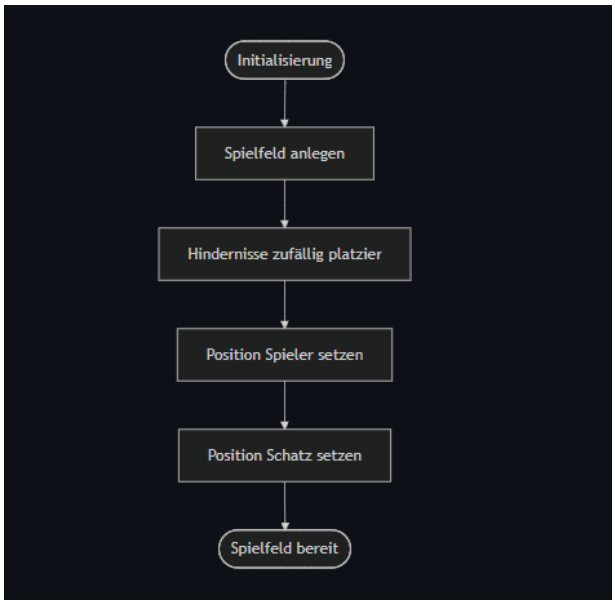


Abb.5: Übersicht Spielfeld

Die Darstellung des Spielfelds auf der Konsole ist in **Abbildung 6** beschrieben. Dabei geht es um die Initialisierung des Renderers, das Laden der Daten, das Zeichnen des Spielfelds und schließlich die Ausgabe in der Konsole.

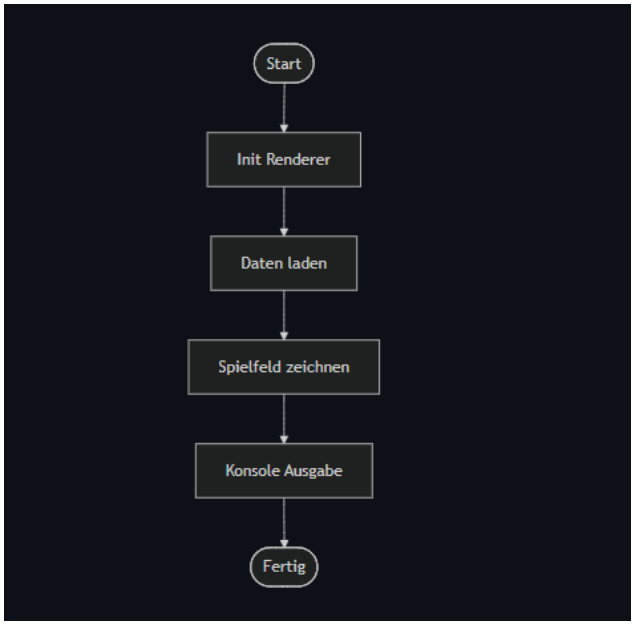


Abb.6: Übersicht Renderer

Zum Schluss zeigt **Abbildung 7** die Zugauswertung. Dort wird geprüft, ob ein Zug möglich ist oder ob der Spieler blockiert ist. Bei einem gültigen Zug wird die Position aktualisiert. Wenn der Schatz erreicht wurde, erscheint die Meldung „Herzlichen Glückwunsch – alle Schätze gefunden“ und das Spiel wird beendet.

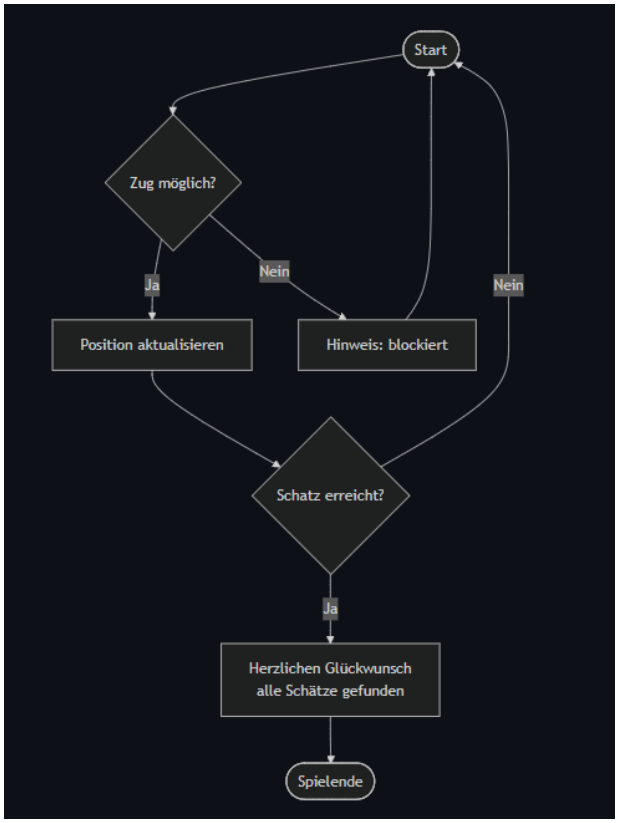


Abb.7: Übersicht Zugauswertung

3.3 Besonderheiten & Herausforderungen

Während der Umsetzung des Projekts sind einige Schwierigkeiten aufgetreten, die ich erst lösen musste, bevor das Programm richtig funktioniert hat. Eine Herausforderung war am Anfang das **Makefile**. Ich hatte es zwar erstellt, aber es lief nicht sofort so, wie ich wollte. Erst nach einigen Anpassungen an den Pfaden und den Compiler-Optionen hat das Kompilieren zuverlässig funktioniert.
Auch bei der **Spiellogik** gab es Hürden. Vor allem die Abfrage, ob ein Zug erlaubt ist oder nicht, hat mir anfangs Probleme bereitet. Zuerst konnte die Spielfigur auch durch Wände laufen, was natürlich falsch war. Das habe ich gelöst, indem ich eine genaue Überprüfung eingebaut habe, ob das Zielfeld frei ist.
Ein weiterer Punkt war die **Darstellung auf der Konsole**. Teilweise wurde das Spielfeld nicht sauber angezeigt oder die Spielfigur wurde nicht korrekt aktualisiert. Hier musste ich die Ausgabe Schritt für Schritt debuggen, bis es richtig funktionierte.
Auch das Einbauen der **Tests** war nicht ganz einfach, weil ich vorher noch nicht viel mit eigenen Testdateien gearbeitet hatte. Nach ein paar Versuchen konnte ich aber einfache Tests schreiben, die die wichtigsten Funktionen prüfen.
Am Ende haben diese Schwierigkeiten aber auch dazu geführt, dass ich den Code besser verstanden habe. Gerade durch die Fehler habe ich viel gelernt, wie man Probleme systematisch angeht und löst.

4 Ergebnisse & Tests

Nach der Umsetzung wurde das Programm mehrfach getestet, um sicherzustellen, dass alle Anforderungen erfüllt sind. Dabei habe ich verschiedene Testfälle durchgespielt und auch bewusst falsche Eingaben ausprobiert, um zu sehen, wie das Spiel reagiert.

4.1 Funktionstest

Beim Start des Programms wird das Labyrinth wie geplant erstellt und auf der Konsole dargestellt. Die Spielfigur erscheint an der vorgesehenen Startposition und der Schatz wird zufällig im Spielfeld platziert. Die Steuerung mit den Tasten **W, A, S, D** funktioniert zuverlässig. Bei einem gültigen Zug wird die Spielfigur korrekt verschoben, bei einem ungültigen Zug (z. B. in eine Wand) bleibt sie stehen. Wenn eine falsche Taste gedrückt wird, gibt das Programm eine entsprechende Meldung aus und fordert zur erneuten Eingabe auf. Mit **Q** kann das Spiel jederzeit beendet werden.
Besonders wichtig war die Überprüfung der **Schatzsuche**. Sobald der Spieler den Schatz erreicht, erscheint die vorgesehene Siegesmeldung: *„Herzlichen Glückwunsch – alle Schätze gefunden“*. Das Programm endet danach wie vorgesehen.

4.2 Bewertung der Resultate

Die Tests haben gezeigt, dass das Spiel die definierten Anforderungen erfüllt. Sowohl die Spiellogik als auch die Abfragen funktionieren zuverlässig. Das Spielfeld wird korrekt dargestellt, die Bewegungen sind nachvollziehbar und es gibt keine Möglichkeit mehr, durch Wände zu laufen.
Ein kleiner Punkt, der noch verbessert werden könnte, ist die Darstellung in der Konsole. Bei sehr vielen Hindernissen wird das Spielfeld teilweise etwas unübersichtlich. Für die geforderten Funktionen spielt das aber keine Rolle und die Kernanforderungen sind erfüllt.
Insgesamt zeigt das Ergebnis, dass das Projektziel erreicht wurde: ein funktionierendes Konsolenspiel mit zufälligem Labyrinth, Bewegungslogik und klarer Siegbedingung.
Zusammenfassend lässt sich sagen, dass die im Kapitel 2 beschriebenen Anforderungen vollständig erfüllt wurden. Das Spiel erzeugt ein zufälliges Labyrinth, ermöglicht eine funktionierende Steuerung, erkennt ungültige Eingaben zuverlässig und zeigt beim Erreichen des Schatzes die vorgesehene Siegesmeldung an. Damit sind alle Kernziele des Projekts erreicht worden.

5 Schluss / Reflexion

5.1 Zusammenfassung

In dieser Arbeit wurde ein Konsolenspiel in C entwickelt, bei dem der Spieler in einem Labyrinth einen Schatz finden muss. Das Projekt umfasste die Planung, die Umsetzung in mehrere Dateien (*main.c*, *labyrinth.h*, *labyrinth.c*), das Erstellen eines *Makefiles* sowie die Dokumentation mit Diagrammen.
Durch die Tests konnte gezeigt werden, dass alle definierten Anforderungen erfüllt sind: Steuerung über die Tasten W, A, S, D, Spielende mit Q und die Ausgabe einer Siegesmeldung beim Finden des Schatzes. Damit wurde das Projektziel vollständig erreicht.

5.2 Reflexion des Lernprozesses

Rückblickend war das Projekt eine wertvolle Gelegenheit, die Arbeit mit C und die Strukturierung größerer Programme besser zu verstehen. Zu Beginn gab es Schwierigkeiten, insbesondere beim *Makefile* und bei der Umsetzung der Spiellogik. Gerade diese Herausforderungen haben jedoch dazu beigetragen, viel zu lernen. Ich habe erkannt, wie wichtig es ist, Probleme Schritt für Schritt zu lösen und den Code regelmäßig zu testen.
Eine weitere wichtige Erkenntnis war, dass eine klare Trennung des Codes und die Verwendung von Diagrammen die Nachvollziehbarkeit deutlich verbessern. Dadurch ist das Projekt nicht nur für mich verständlicher, sondern auch für andere Personen leichter nachvollziehbar. Insgesamt konnte ich durch diese Arbeit meine Fähigkeit stärken, systematisch vorzugehen und komplexere Aufgaben methodisch zu bearbeiten.

5.3 Ausblick

Das Spiel erfüllt die grundlegenden Anforderungen, könnte jedoch in Zukunft noch erweitert werden. So wäre es denkbar, mehrere Level mit unterschiedlichen Schwierigkeitsgraden einzubauen oder eine grafische Benutzeroberfläche zu entwickeln. Auch die Testabdeckung könnte vergrößert werden, um die Stabilität des Programms weiter zu erhöhen.
Insgesamt hat mir das Projekt gezeigt, dass sich mit strukturiertem Vorgehen und etwas Geduld ein komplettes Programm von der Planung bis zur Fertigstellung erfolgreich umsetzen lässt. Gleichzeitig habe ich Ideen gewonnen, wie das Projekt praxisnäher und benutzerfreundlicher weiterentwickelt werden könnte.

5.4 Fazit

Zusammenfassend lässt sich festhalten, dass das Projekt erfolgreich umgesetzt wurde und die definierten Anforderungen erfüllt sind. Es konnte ein funktionierendes Konsolenspiel erstellt werden, das alle Kernfunktionen wie Spielfigur-Bewegung, Schatzsuche, Eingabeprüfung und Siegesmeldung zuverlässig abdeckt.
Darüber hinaus bot die Arbeit die Gelegenheit, den Umgang mit der Programmiersprache C sowie die Nutzung von Modulen, Header-Dateien und Makefiles praxisnah zu üben. Besonders wertvoll war die Erfahrung, systematisch vorzugehen, auftretende Probleme Schritt für Schritt zu lösen und die Ergebnisse regelmäßig zu testen.
Das Projekt hat gezeigt, dass sich mit klarer Strukturierung, Geduld und kontinuierlichem Arbeiten auch ein größeres Vorhaben erfolgreich realisieren lässt.

6 Quellenverzeichnis

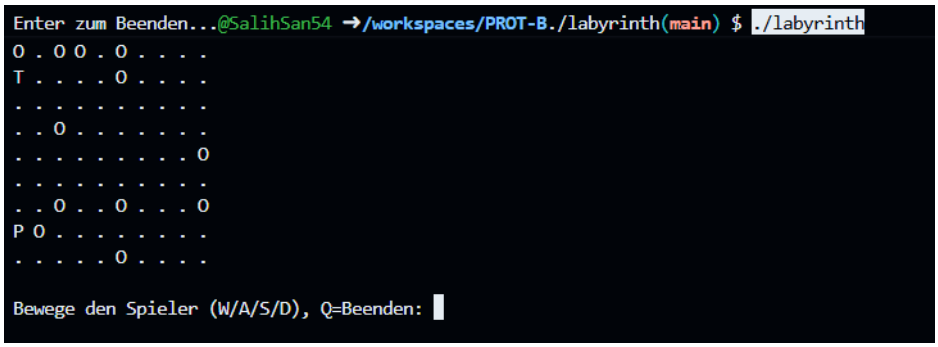
- Goll, Joachim; Stamm, Tobias (2023): *C als erste Programmiersprache – Nach den Standards C11 und C23*. 9. Auflage. Springer Vieweg, Wiesbaden.
- Unterrichtsmaterialien der ipso! Bildung AG, Modul Programmiertechnik A und B (2025).
- GitHub Repository: INF-EFZ / TE-PROA – https://github.com/INFEFZ/TE-PROA
- GitHub Repository: INF-EFZ / TE-PROB – https://github.com/INFEFZ/TE-PROB

7 Anhang

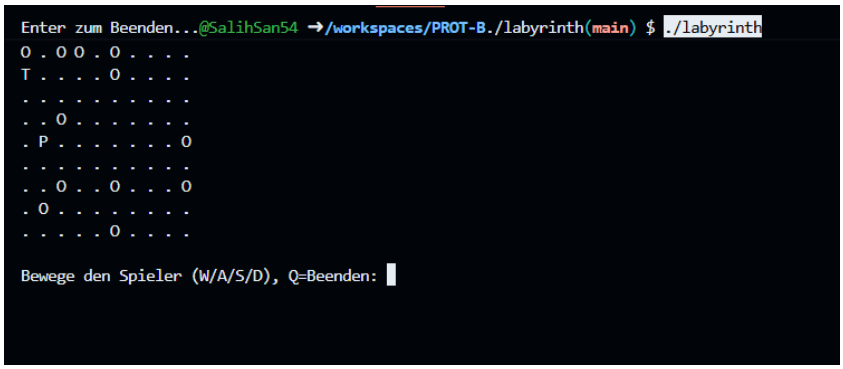
7.1 Screenshots

Nachfolgend sind Beispiel-Screenshots des Spiels in der Konsole dargestellt. Diese zeigen den Ablauf von Start bis Ziel:

1. **Spielstart:**
   Das Labyrinth wird zufällig generiert. Der Spieler **P** startet an einer Position, das Ziel **T** (Schatz) ist an einer anderen Stelle platziert. Hindernisse sind mit **O** gekennzeichnet.



2. **Bewegung im Spiel:**
   Der Spieler bewegt sich mit den Tasten **W (hoch)**, **A (links)**, **S (runter)**, **D (rechts)**.
   Im Beispiel wurde der Spieler ein paar Felder bewegt.



3. **Spielende / Schatz erreicht:**
   Sobald der Spieler **T** erreicht, wird das Spiel beendet und eine Siegmeldung ausgegeben.