Dokumentation DES PRAKTIKUMSPROJEKTES

Informatik 3

Inhalt

[**Erste Sitzung - Einführung in Linux (UBUNTU)** 1](#_Toc117689025)

[Zweite Sitzung - Aufgabenstellung und Erklärung 3](#_Toc117689026)

[Dritte Sitzung - erste Programmierung 5](#_Toc117689027)

**Erste Sitzung - Einführung in Linux (UBUNTU)**

Zunächst wurde uns in der ersten Sitzung eine umfassende Einführung für Linux in der Ubuntu Umgebung gegeben.

Allgemeine Befehle wurden uns erklärt und demonstriert.

**Befehle von Ubuntu**

cd steht für change directory und führt uns zu dem gewünschten Ziel.

z.B. cd 10\_Workspace/03\_github/Info3Prak und dann befinden wir uns in diesem Ordner.

-rm steht für Remove und löscht Elemente.

**Für das Hochladen mit Terminal**

* cd 10\_WORKSPACES/03\_github/
* git clone
* ls
* cd Konstruktor\_Schachbrett/
* cp chess1.cpp
* cp chess1.cpp ./../..
* git clone
* cp chess1.cpp ./ChessInf3Pra/
* cd ChessInf3Pra/
* git status
* git add chess1.cpp
* git commit -m "VERSIONSNAME"
* git config --global user.email "EMAILADRESSE"
* git status
* git commit -m "VERSIONSNAME"
* git push INDIVIDUELLEADRESSE@github.com/novalidcode/ChessInf3Pra.git

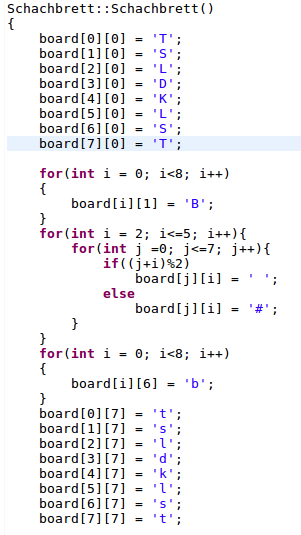
# Zweite Sitzung

# In der zweiten Sitzung haben wir uns mit der allgemeinen Gestalt des Programmes auseinandergesetzt. Hierbei haben wir einen Strukturdiagramm erstellt um uns einen Leitfaden für das Gesamte Projekt zu erstellen.

# C:\Users\amalek-sup\Downloads\c65832e0-d767-4ee0-b8c9-a1fe2591f09c.jfifg - Aufgabenstellung und Erklärung

# 

# Dritte Sitzung - erste Programmierung



Zunächst wird dafür eine Klasse Schachbrett definiert.

Der Konstruktor dieser Klasse wird initialisiert und gerufen: Der "Board Array" soll so angeordnet sein, dass die Figuren auf der Linken Array Platziert werden es werden 8 Zeilen erstellt mit dem Ziel, 8 Spalten für das Spielbrett zu erzeugen.

Auf der rechten Seite des Board Arrays wird zunächst Standardgemäß [0] eingegeben denn dieser sei zunächst die Startlinie des Brettes, also das Erste Feld für den Spieler [Auf der Konsole im Terminal die UNTERSTE Zeile]. Die erste Row ist mit „T“ für Turm, die zweite mit „S“ für Springer, die dritte mit „L“ für Läufer, die vierte mit „D“ für Dame, die fünfte mit „K“ für König und dann die sechste bis achte sind die Doppelfiguren wie im Default Anordnung des Spieles vorgesehen.

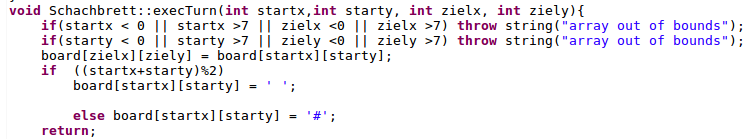
Der Code wiederholt dann jede der Acht Rows und erzeugt für jede Row eine Spalte pro Iteration.

Für alle Spalten in zweiter Zeile werden „B“ für Buben eingesetzt.

Danach versucht man mit Modulo „der Restlose Verteiler, die schwarzen Felder zu markieren. Das geschieht in dem alle leerstehende Felder mit Leer bezeichnet aber sollte die Zahl der Spalte und Zeile nicht restlos Teilbar sein dann wird die angesprochene Zeile/Spalte Feld mit „#“ bezeichnet.

Dieser wird dann für alle Spalten wiederholt solange wir nicht i+j haben denn diese sollen später mit # bezeichnet werden.

Wir setzen auch für den Gegenspieler dieselben Figuren aber in Kleinbuchstaben. Das ist im Grunde nur Wiederholung der obengenannten Strategie.



Dieser Funktion beginnt an der Stelle der Aktuellen Figur des Spielers und bewegt ihn an die Position seines vermeintlichen Zieles.

Der Code startet mit der Kontrolle ob „Startx und Starty außerhalb des zulässigen Bereichs sind. Die Zulässige Bereich ist definiert innerhalb von 0 und 7.

Sollten die gegebenen Werte NICHT im zulässigen Bereich sein wird „Array out of bound“ als Fehlermeldung angezeigt.

Des Weiteren wird geprüft ob auch die „Zielx“ und „ziely“ ebenfalls im zulässigen Bereich von 0 bis 7 liegen und die gleiche Fehlermeldung taucht auf, wenn das nicht der Fall ist.

Sollte die Fehler nicht auftreten dann werden die neue zielx und ziely Werte für Startx und Starty werte eingesetzt. Danach wird nochmal geschaut ob die Summe von Startx plus Starty restlos durch zwei teilbar ist um die Farbe der Felder nach einem Spielzug nochmal zu korrigieren.

Der Code versucht festzustellen, ob sich eine Figur auf dem Brett befindet. Wenn dies der Fall ist, wird wahr zurückgegeben; andernfalls wird false zurückgegeben. Die erste Codezeile prüft auf 'a' und 'z'. Wenn einer gefunden wird, gibt die Funktion false zurück. Andernfalls gibt die Funktion true zurück. Die zweite Codezeile prüft auf 'A' und 'Z'. Wenn einer gefunden wird, gibt die Funktion false zurück. Andernfalls gibt die Funktion true zurück. Der Code versucht festzustellen, ob eine bestimmte Figur auf dem Brett ein "W" ist oder nicht. Die erste if-Anweisung gibt false zurück, da piece kleiner als 'a' und größer als 'z' ist. Die zweite if-Anweisung gibt true zurück, da piece gleich 'A' und gleich 'Z' ist.