Spiel NIM

# Einleitung:

NIM ist ein Strategiespiel, das von zwei Spielern gespielt wird. Es werden Streichhölzer in mehreren Reihen angeordnet. Abwechslungsweise nehmen jetzt die Spieler Hölzchen weg. Ein Spieler muss ein oder mehrere Hölzchen nehmen. Es dürfen aber nur Hölzchen aus einer einzelnen Reihe genommen werden.

Derjenige Spieler, welcher das oder die letzten Hölzchen nehmen kann, hat gewonnen.

# Aufgabe:

Ein NIM Spiel soll als Console-Applikation entwickelt werden. Dabei sollen diese Anforderungen berücksichtigt werden:

## Anforderungen:

* **Initialisierung des Spiels**: Die Anzahl Reihen und Anzahl Hölzchen können entweder vom Benutzer eingegeben werden oder vom Programm zufällig erzeugt werden. Beide Varianten sind zulässig.
* **Anzeige**: Der aktuelle Stand (Anzahl Hölzchen pro Reihe) muss in verständlicher Form auf der Konsole angezeigt werden. Wenn das Spiel zu Ende ist, soll der Gewinner angezeigt werden.
* **Spielverlauf**: Der Spieler wechselt sich ab mit dem Computer. Wenn der Spieler an der Reihe ist, wird er über eine Meldung aufgefordert, seinen Zug zu machen.
* **Benutzer-Spielzug**: Der Benutzer kann eingeben, in welcher Zeile, wie viele Hölzchen entfernt werden sollen. Fehlerhafte Eingaben des Benutzers sollen das Programm nicht zum Abstürzen bringen.
* **Computer**-**Spielzug**: Der Computer kann einen gültigen Spielzug durchführen. Es kann (muss aber nicht, gibt aber Zusatzpunkte) eine Gewinn-Strategie implementiert werden. Vergl. <https://www.hep-verlag.ch/pub/media/import/public/6486/mathematikaufgaben.pdf>. Auch verschieden «Levels» sind denkbar.
* **Korrektheit**: Die Spielzüge müssen korrekt angewendet werden.
* **Usability**: Mit Hilfe der angezeigten Meldungen soll dem Benutzer klar sein, wie das Spiel zu benutzen ist.

# Vorgaben:

## Header:

Für die Implementierung des Spiels brauchen Sie vermutlich diese Header-Includes:

#include <stdio.h> // rpintf, scanf

#include <stdint.h> // int32\_t

#include <stdlib.h> // rand()

#include <time.h> // initialisierung des Zufallsgenerators

## User Input:

Für das Lesen von Benutzereingaben können Sie die Funktion [scanf\_s](https://docs.microsoft.com/en-us/cpp/c-runtime-library/reference/scanf-s-scanf-s-l-wscanf-s-wscanf-s-l?view=msvc-160) verwenden. Ein Beispiel, wie diese Funktion verwendet wird, finden Sie in der nachfolgenden Abbildung.

int32\_t ReadNumber()

{

int32\_t number;

scanf\_s("%i", &number);

while (getchar() != '\n'); // this is needed to clear the buffer!

return number;

}

Wenn Sie nur eine einzelne Zahl lesen wollen, können Sie diese Funktion verwenden.

# Vorgehen:

## Analyse:

* Zeichnen Sie ein UML Aktivitäts-Diagramm, welches den Ablauf und die einzelnen Schritte des Spiels zeigt. Überlegen Sie sich, welche Aktionen sie in weitere Aktionen aufteilen möchten.
* Überlegen Sie sich, welche Daten Sie brauchen, um den Zustand des Spiels festzuhalten.
* Definieren Sie eine Reihenfolge, in welcher Reihenfolge Sie die Funktionen implementieren und testen.
* Definieren Sie, wie/ob Sie die einzelnen Funktionen testen (z.B. Bei der Anzeigefunktion sehen sie sofort, wenn etwas nicht stimmt. Das könne Sie so festhalten und sagen, dass sie das nicht ausdrücklich testen)

## Implementierung:

* Definieren Sie für die alle Aktionen c-Funktionen
* Definieren Sie geeignete Datenstrukturen
* Definieren Sie die entsprechenden Variablen, um den Spielstatus zu speichern.
* Implementieren und testen Sie die Funktionen gemäss Ihres Plans.

## Source Code Verwaltung und Hilfestellungen:

* Speichern Sie das Projekt mit git (github oder gitlab) und geben Sie mir Zugriff auf dieses Repository. Ich werde Kommentare, Anmerkungen u.U. Implementierungsvorschläge in einem Branch zu diesem Projekt machen.
* Teilen Sie mir mit, wenn Sie die Analyse abgeschlossen habe. Ich kann ihnen dann eine Rückmeldung dazu geben. Es ist aber normal, dass man während der Implementierung feststellt, dass eine Überlegung aus der Analyse so nicht funktioniert. Dennoch macht es Sinn, mit einer Analyse zu beginnen.
* Ich werde Sie bei der Lösung dieser Aufgabe unterstützen. Es sollen aber von ihnen klare eindeutige Fragen kommen.

# Bewertung:

Bewertet wird die Erfüllung der Anforderungen 70% sowie die Analyse 30%. Die gesamte Arbeit ergibt eine Note mit Gewicht 40% .