

# Übung zum C/C++-Praktikum Fachgebiet Echtzeitsysteme



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

## Übungen für den 6. Tag

### Aktivierung von C++11

In diesem und den folgenden Blättern wirst du Hinweise darauf finden, dass bestimmter Code (bspw. das Schlüsselwort `nullptr`) nur unter C++11 verfügbar sind.

C++11 muss explizit in den Projekteinstellungen wie folgt aktiviert werden:

**Rechtsklick auf der Projekt -> Properties -> C/C++ Build / Settings -> GCC C++ Compiler / Dialect -> Language Standard: ISO C++11 (-std=c++0x).**

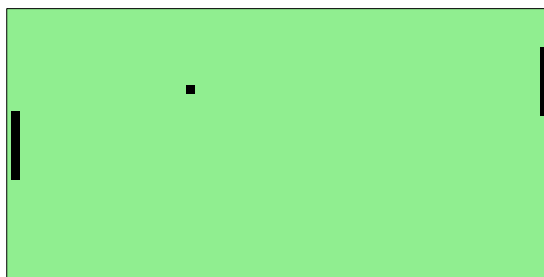
### Aufgabe 1 Implementation eines beliebigen Programms

Nachdem du alle Ein- und Ausgabemöglichkeiten des Boards kennengelernt hast, besteht die Aufgabe in diesem Teil darin, ein beliebiges Programm, zum Beispiel ein Spiel, zu implementieren. Du hast hierbei die freie Wahl, die folgenden Vorschläge sollen nur als Anregung dienen. Um die Aufgabe leichter zu gestalten, empfiehlt es sich die beiliegende 16FXlib zu verwenden. Die Dokumentation dazu findest du unter <http://echtzeitsysteme.github.io/tud-cpp-16FXlib/documentation/index.html>.

#### Vorschlag: Pong

Zwei Gegner sollen je einen Balken (Rechteck) am linken oder rechten Rand des Spielfeldes mit den Schiebereglern steuern können, um einen Ball (ein Quadrat) im Spiel zu halten. Erreicht der Ball den linken oder rechten Rand des Spielfeldes, so bekommt der Spieler auf der anderen Seite einen Punkt und der Ball wird an seine Anfangsposition (die Mitte des Spielfeldes) zurückversetzt. Erreicht der Ball den oberen oder unteren Rand sowie einen der Balken der Spieler, so wird der Ball reflektiert - verlässt also niemals das Spielfeld.

Gewonnen hat der Spieler, der zuerst eine definierte Anzahl an Punkten erreicht. Die aktuelle Punktzahl beider Spieler könnte auf der Siebensegmentanzeige ausgegeben werden.



#### Vorschlag: Game of Life

„Game of Life“<sup>1</sup> besteht aus einem zweidimensionalen Spielfeld. Jedes Feld steht für eine Zelle, die *tot* (grün) oder *lebendig* (schwarz) ist. Jede Zelle hat acht Nachbarzellen, die ebenso tot oder lebendig sein können. Zu Beginn gibt es eine vordefinierte Anfangsgeneration.

Durch festgelegte Regeln wird die nachfolgende Generation ermittelt:

- Eine **lebende Zelle** ...
  - mit 1 oder 0 lebenden Nachbarn stirbt aus Einsamkeit.

<sup>1</sup> siehe auch [http://de.wikipedia.org/wiki/Conways\\_Spiel\\_des\\_Lebens](http://de.wikipedia.org/wiki/Conways_Spiel_des_Lebens)

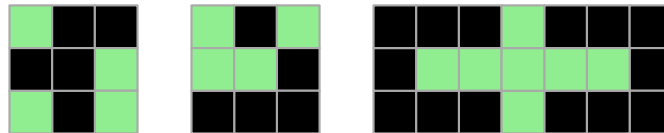
---

## Übung zum C/C++-Praktikum - Tag 6

---

- mit 4 oder mehr lebenden Nachbarn stirbt wegen Übervölkerung.
- mit 2 oder 3 lebenden Nachbarn bleibt am Leben.
- Eine **tote Zelle** mit genau 3 lebenden Nachbarn wird in der nächsten Generation geboren werden, andernfalls bleibt sie tot.

Als Anfangsgeneration eignen sich zufällige Populationen oder eine der folgenden Figuren:



### Hinweise

- Da das Spielfeld begrenzt ist, soll es torusförmig aufgebaut werden. Das heißt: Alles, was am unteren Rand des Spielfelds verschwindet, kommt oben wieder heraus – das Gleiche gilt für den linken und rechten Rand.
- Verwende als Spielfeld ein mehrdimensionales Array
- Ein weiteres mehrdimensionales Array bietet sich an, um die zukünftige Generation erzeugen zu können.
- Achte beim torusförmigen Feld unbedingt darauf, dass du nicht über die Grenzen des Spielfelds hinaus zugreifst! Das kann zu unvorhersehbarem und schwer zu debuggen Verhalten des ganzen Displays führen!

---

### Weitere Vorschläge

---

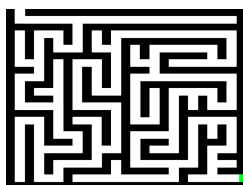
Asteroids



Pacman



Labyrinth



Ausweichspiele à la Hugo



Moorhuhn



Snake

