

# Praktikum Aufgabe 2 – Konsolenausgabe

Punktzahl gesamt: 100 Punkte

Abgabe der Lösungen bis spätestens 14.01.21 10:00 Uhr

## Aufgabe 1 – Ausgabe eines Zeichens

30 Punkte

Der erste Schritt in der Erstellung einer Konsolenausgabe ist die Ausgabe einzelner Zeichen. Implementieren Sie dazu die Methode **printC**, welche in der Klasse **cga\_screen.cc** im Paket **machine** deklariert ist. Verwenden Sie dazu die Anweisung:

```
CGA_START[pos] = (Cell){character, attrib};
```

Ein Besonderheit dabei ist, dass nur eine Position benötigt wird um in einem 2D-Array zu schreiben.

Nach der Implementierung der Methode **printC** können einzelne Zeichen ausgegeben werden. Um zu verhindern, dass die Zeichen außerhalb des sichtbaren Bereichs erstellt werden, implementieren Sie die Methode **checkBounce**. Diese Methode überprüft die Einhaltung der Konsolengrenzen und berechnet die Position, welche in **printC** benötigt wird.

## Aufgabe 2 – Object-Stream

30 Punkte

Um die Aufgabe auf der Konsole zu erleichtern bietet OOSTuBS den Stream **kout**:

```
kout << "Hallo Welt!" << endl;
```

Damit dieser Stream einfach genutzt werden kann, müssen der Operator **<<** für alle Datentypen überschrieben werden. Implementieren Sie dazu den Operator in der Klasse **o\_stream.cc** im Paket **object** für die folgenden Datentypen:

- char
- unsigned char
- const char
- bool (als *true* bzw. *false*)

Des Weiteren werden die folgenden Manipulatoren benötigt:

- endl (Einfügen eines Zeilenumbruchs)
- bin (Darstellung einer Zahl in binär)
- oct (Darstellung einer Zahl in oktal)
- dec (Darstellung einer Zahl in dezimal)
- hex (Darstellung einer Zahl in hexadezimal)

### Aufgabe 3 – Ausgabe auf der Konsole

30 Punkte

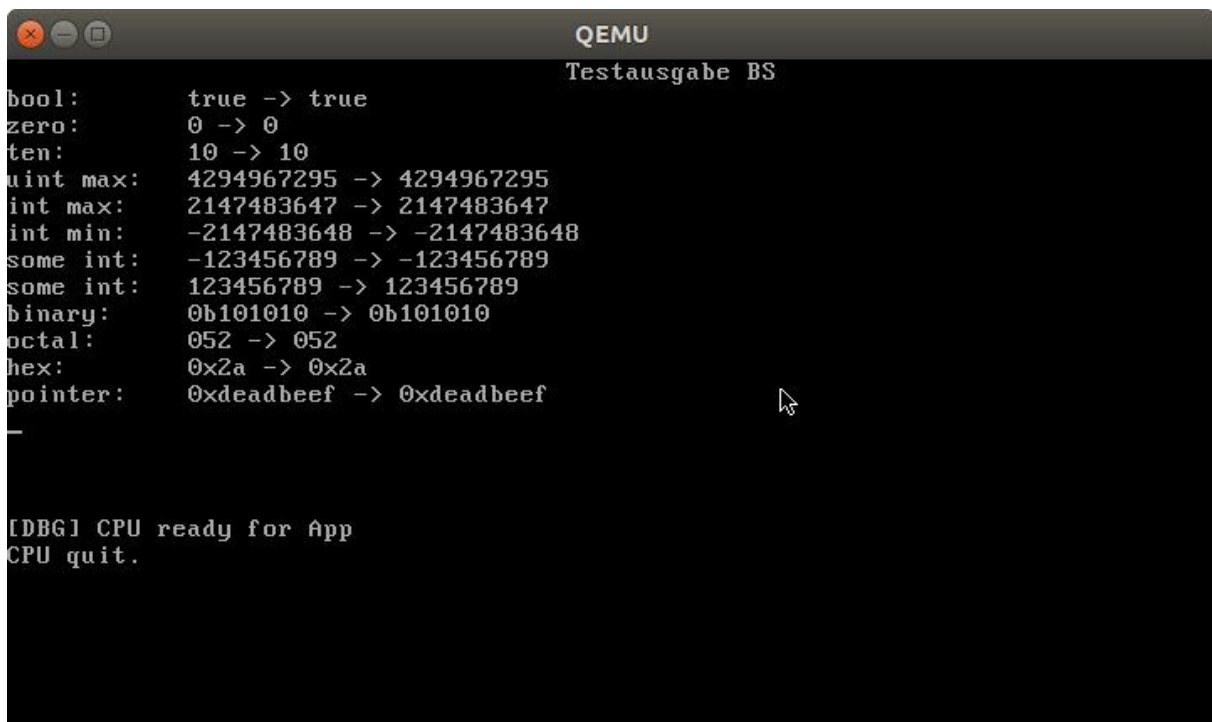
Implementieren Sie nun die Methode **print** in der Klasse **cga\_screen.cc** im Paket **machine**, die vom Betriebssystem aufgerufen wird um einen String mit fester Länge auf der Konsole auszugeben. Diese soll zunächst mit der Methode **getpos** die Position des Cursors speichern. Anschließend müssen alle Zeichen einzeln ausgegeben werden. Beachten Sie dabei, dass **\n** nicht automatisch einen Zeilenumbruch erzeugt und dies daher gesondert behandelt werden muss.

Implementieren Sie zusätzlich die Methode **writeToConsole**, die eine neue leere Zeile erzeugt, falls die Ausgabe über die untere Grenze der Konsole hinaus läuft. Der Inhalt muss dazu eine Zeile nach oben geschoben werden und eine die unterste Zeile mit Leerzeichen überschrieben werden. Rufe Sie diese Methode an der entsprechenden Stelle der **print** Methode auf.

### Aufgabe 4 – Testen, Testen, Testen

10 Punkte

Als letztes müssen Sie natürlich ihren Code auf korrekte Funktionalität prüfen. Dafür haben wir Ihnen eine Test Applikation zur Verfügung gestellt. Wenn Sie nun die Anwendung mit dem Konsolenbefehl **make qemu** starten sollte bei Ihnen die Ausgabe wie in der Abbildung unten erscheinen. Falls das nicht der Fall ist, sollten Sie noch einmal über die einzelnen Teilaufgaben schauen um eventuelle Fehler zu finden und zu beheben.



```
QEMU
Testausgabe BS

bool:      true -> true
zero:      0 -> 0
ten:       10 -> 10
uint max:  4294967295 -> 4294967295
int max:   2147483647 -> 2147483647
int min:   -2147483648 -> -2147483648
some int:  -123456789 -> -123456789
some int:  123456789 -> 123456789
binary:    0b101010 -> 0b101010
octal:     052 -> 052
hex:       0x2a -> 0x2a
pointer:   0xdeadbeef -> 0xdeadbeef

[DBG] CPU ready for App
CPU quit.
```