VERKETTETE LISTEN

Ihre erste Aufgabe ist die Implementierung einer Datenstruktur zum effizienten Speichern beliebig großer Matrizen im Hauptspeicher. Die Datenstruktur soll mittels verketteter Listen (nicht mit Arrays) implementiert werden, d.h. jedes Element der Datenstruktur soll einen Zeiger auf das (oder die) folgenden Elemente enthalten, so dass eine Manipulation von Matrizen durch Einfügen oder Löschen von Zeilen und Spalten keine globale Modifikation von Daten, sondern lediglich die Änderung weniger Zeiger erfordert.

Erste Teilaufgabe: Erstellen Sie eine Funktion `grid_init(num_columns, num_rows)`, die solch eine Matrix mit der gegebenen Anzahl von Spalten und Zeilen erstellt. Die Inhalte können frei vorbelegt werden.

Zweite Teilaufgabe: Erstellen Sie eine Funktion `grid_print(grid)`, die mit beliebigen Parametern erstellte Matrizen in folgender Form ausgibt:

```
| 100 101 102 103 104 105 106 |
| 110 111 112 113 114 115 116 |
| 120 121 122 123 124 125 126 |
| 130 131 132 133 134 135 136 |
| 140 141 142 143 144 145 146 |
```

Dritte Teilaufgabe: Erstellen Sie eine Funktion `grid_print_dot(grid)`, die die Beziehungen in der Datenstruktur verwendeter Zeiger in der Form „M<Adresse> -> M<Adresse>“ in folgender Form ausgibt:

```
digraph {
  M0x669010 -> M0x669030;
  M0x669010 -> M0x669110;
  M0x669110 -> M0x669130;
  M0x669130 -> M0x669150;
}
```

Optional: Sie können das Programm „dot“ der *graphviz*-Sammlung verwenden, um die Ausgabe von `grid_print_dot(g)` in ein Diagramm umzuwandeln. Ein Aufruf von `dot pointers.txt -Tpng > pointers.png` erzeugt z.B. ein Diagramm folgender Art: