



C Programmierkurs

8. Stunde: Denken in Algorithmen

Johannes Hayeß & Mirko Seibt

13. Dezember 2018

Technische Universität Dresden

Wie arbeitet der CSV-Parser von letzter
Stunde?

CSV-Parser - Initialisierung

```
CsvParser_new(...);
```

↪ ***CsvParser**

zeigt auf alles was der Parser zum Funktionieren braucht

CSV-Parser - Initialisierung

```
CsvParser_new(...);
```

↪ `*CsvParser`

zeigt auf alles was der Parser zum Funktionieren braucht

```
typedef struct CsvParser {  
    char *filePath_;  
    char delimiter_;  
    int firstLineIsHeader_;  
    char *errMsg_;  
    CsvRow *header_;  
    FILE *fileHandler_;  
    int fromString_;  
    char *csvString_;  
    int csvStringIter_;  
} CsvParser;
```

CSV-Parser - Zeile für Zeile



CSV-Parser - Zeile für Zeile



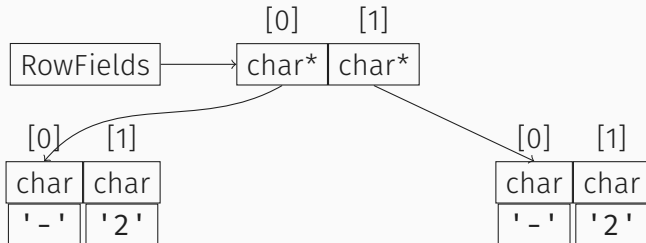
Die Zeile ist somit "-2,-2".

Der Parser bricht diese nun an der Kommastelle auseinander:

{"-2", "-2"}

CSV-Parser - Speicherorganisation von Zeilen

```
char **RowFields = CsvParser_getFields(csvparser);
```



Shortest path problem

Dijkstra's Algorithm

```
1  function Dijkstra(Graph, source, target):
2      create vertex set Q
3      for each vertex v in Graph:           // Initialization
4          dist[v] ← INFINITY                // Unknown distance from source to v
5          prev[v] ← UNDEFINED               // Previous node in optimal path from source
6          add v to Q                        // All nodes initially in Q (unvisited nodes)
7          dist[source] ← 0                  // Distance from source to source
8
9      while Q is not empty:
10         u ← vertex in Q with min dist[u]  // Node with the least distance
11
12         remove u from Q
13         for each neighbor v of u:         // where v is still in Q.
14             alt ← dist[u] + length(u, v)
15             if alt < dist[v]:              // A shorter path to v has been found
16                 dist[v] ← alt
17                 prev[v] ← u
18
19     S ← empty sequence                    // Start building shortest path
20     u ← target
21     if prev[u] is defined or u = source:  // Do something only if the vertex is reachable
22         while u is defined:               // Construct the shortest path with a stack S
23             insert u at the beginning of S // Push the vertex onto the stack
24             u ← prev[u]
25
26     return S
```



Wikipedia. *Dijkstra's Algorithm*. Lizenz: CC BY-SA 3.0. URL:
[https://en.wikipedia.org/wiki/Dijkstra%27s_](https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm)
[algorithm.](https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm)

Diese Präsentation ist lizenziert unter der **Creative Commons Attribution-ShareAlike 4.0 International** Lizenz.

