

Problemset01

Adrianus Kleemans

Rafael Breu

Erste Ideen

- 2 Strings, **n** für den Dateinamen, **f** für den Filter
- Jedes Zeichen iterativ prüfen auf
 - `f[i] == n[i]` oder
 - `f[i] == '?'`
- Problem: Sterne (*)
 - ==> nicht jedes Zeichen aus dem String `n` muss im Filter `f` vorhanden sein
 - ==> `n.length()` nicht zwingend gleich wie `f.length()`

Lösungsidee

- Vorherige Idee kapseln in Funktion

```
boolean compare(string a, string b)
```

- wahr: falls Strings gleiche Länge und gleicher Inhalt (``?` toleriert)
- falsch: falls eine Bedingung nicht erfüllt

- Schleife:

Prüfe Zeichen für Zeichen auf Stern, iteriere `i`.

Wenn Stern vorhanden:

Am Ende: `compare(f bis i, n bis i)`

An anderer Stelle: `compare(f bis i, n bis i) &&
compare(f ab i, n ab i)`

Sonst: `compare(f, n)`

Beispiel: Stern in der Mitte

```
f = „f*.txt“
```

```
n = „fname.txt“
```

Diagram illustrating the first part of the string comparison for `f` and `n`. The string `f` is `f*.txt`. A box highlights the character `f` at index 1, with an arrow labeled 1 pointing to it. Another box highlights the substring `.txt` starting at index 4, with an arrow labeled 4 pointing to its start.

Diagram illustrating the second part of the string comparison for `f` and `n`. The string `n` is `fname.txt`. A box highlights the character `f` at index 1, with an arrow labeled 1 pointing to it. Another box highlights the substring `.txt` starting at index 4, with an arrow labeled 4 pointing to its start.

```
compare(f.substring(0,1), n.substring(0,1)) &&
```

```
compare(f.substring(2,6), n.substring(5,9))
```



Einschränkungen

- Nicht anwendbar auf Filter, welche mehr als 1 Stern enthalten
- Ausnahmen müssen separat abgefangen werden (Nullstrings, `compare()` mit ungleich grossen Eingaben)