Aufgabe 1: Störung

Team-ID: 00100

Team-Name: Team

Bearbeiter/-innen dieser Aufgabe:
Arne de Borman

21. November 2022

Inhaltsverzeichnis

1 Lösungsidee

Das Programm soll die Stellen im Text finden, die mit einem bestimmten Lückensatz übereinstimmen. Um diese Stellen zu finden, könnte man jede mögliche Stelle im Text mit dem gesuchten Satz abgleichen. Passende Stellen werden dann ausgegeben.

2 Umsetzung

Die Lösungsidee wird in rust implementiert. Der Text sowie der gesuchte Lückensatz werden in einen char-Vektor (so ähnlich wie ein Array) umgewandelt, da verschiedene Unicode Zeichen unterschiedlich viele Bytes groß sind. So braucht man 2 Bytes, um zum Beispiel 'é' darzustellen, aber nur ein Byte für 'a'. Mit den Bytes direkt zu arbeiten, würde also Zeichen 'teilen', was zu Problemen führen könnte. Es ist also einfacher mit rusts abstrakter Repräsentation für Zeichen zu arbeiten - dem char. Außerdem werden der Lückensatz sowie der Text in Kleinschreibung umgewandelt, um das Programm Groß-/Kleinschreibung ignorieren zu lassen. Jetzt wird jede Stelle im Text mit dem Lückensatz verglichen. Um eine Stelle mit dem Lückensatz zu vergleichen, wird jedes Zeichen im Lückentext mit dem entsprechendem Zeichen der Stelle verglichen. Handelt es sich bei dem Zeichen um '_', dann werden Zeichen im Text übersprungen, solange es sich bei den Zeichen im Text um Buchstaben handelt. Wenn die Zeichen alle übereinstimmen, so passt die Stelle und sie wird in der results-Liste gespeichert. Wurden alle Stellen abgeglichen, so werden passende ausgegeben.

3 Beispiele

Wir rufen das rust-Programm mit den verschiedenen BWINF-Eingabedateien auf. Diese Dateien liegen im selben Ordner wie das Programm. Die Textdatei liegt in "./data/Alice_im_Wunderland.txt". Das

Programm wird mit Hilfe des ”cargo run” Befehls ausgeführt. Das Terminal muss sich währenddessen im selben Verzeichnis wie die Cargo.toml Datei befinden. Hier wird vorausgesetzt, dass rust installiert ist. Alternativ kann man auch das vorkompilierte Programm für das Betriebsystem ausführen (z.B. stoerung-windows.exe).

```
~/bwinf/stoerung $ cargo run stoerung0.txt
gesucht: 'das _ mir _ _ _ vor'
Passende Stelle in Zeile 440 gefunden: das kommt mir gar nicht richtig vor

~/bwinf/stoerung $ cargo run stoerung1.txt
gesucht: 'ich muß _ clara _'
Passende Stelle in Zeile 425 gefunden: ich muß in clara verwandelt
Passende Stelle in Zeile 441 gefunden: ich muß doch clara sein,
```

Wenn keine Datei als Argument angegeben wird, so kann ein Lückensatz als Eingabe gegeben werden.

```
~/bwinf/stoerung $ cargo run
bitte den gesuchten Lückensatz eingeben:
sie _ _ Zeit _
gesucht: 'sie _ _ Zeit _'
Passende Stelle in Zeile 997 gefunden: sie wartete eine zeit lang,

~/bwinf/stoerung $ cargo run
bitte den gesuchten Lückensatz eingeben:
diesen Satz gibt es nicht
gesucht: 'diesen Satz gibt es nicht'
keine Passende Stelle für 'diesen Satz gibt es nicht' gefunden!
```

4 Quellcode

```rust
fn find_pattern(text: &str, searched: &str) -> Vec<(usize, String)> {
    //text und searched wird in Vec<char> umgewandelt
    let text: Vec<char> = text.to_lowercase().chars().collect();
    let searched: Vec<char> = searched.to_lowercase().chars().collect();
    let mut results = vec![];
    let mut line_n = 1;
    'text_loop: for text_i in 0..text.len() {
        if text[text_i] == '\n' {
            line_n += 1;
        }
        let mut offset = 0;
        'searched_loop: for searched_i in 0..searched.len() {
            if let Some(c) = text.get(text_i + searched_i + offset) {
                //
                match searched[searched_i] {
                    '_' => {
                        //ein wort überspringen
                        while let Some(c) = text.get(text_i + searched_i + offset) {
                            if c.is_whitespace() {
                                //ende des Wortes
                                offset -= 1; //whitespace ausschließen
                                break;
                            }
                            offset += 1;
                        }
                        continue 'searched_loop;
                    }
                    searched_c => {
                        if c == &searched_c {
                            continue 'searched_loop;
                        }
                    }
                }
            }
            //die Stelle passt nicht zum Lückensatz
            continue 'text_loop;
        }
        //Die Stelle passt zum Lückensatz
        let result_str = String::from_iter(&text[text_i..text_i + searched.len() + offset]);
        results.push((line_n, result_str));
    }
    results
}
```