Junioraufgabe 2

Von Wem?

Vor- und Nachname: Anian Barthel

Team-Name: BitPioneer

Team-ID: 00473

Wann: 21.10.2024

Lösungsidee

Beliebig viele Objekte springen um die Wette immer einer nach dem andren. Der gewinnende Runner unterbricht und schreibt seinen Namen in die Konsole.

Umsetzung

Erst werden alle Buchstaben zu Kleinbuchstaben konvertiert und alle analphabetischen (= alles außer Buchstaben) Zeichen entfernt.

```
let characters: Vec<char> = text.
    to_lowercase() : String
    .chars().filter(|c| c.is_alphabetic()).collect();
```

Danach beginnt das Texthopsen.

Dann erstellt das Programm 2 Runner Objekte, diese speichern die aktuelle Position, Spielernamen und den „sauberen“ Text.

```
let mut runners: Vec<Runner> = vec![
    Runner::new(0, "Bela", &characters),
    Runner::new(1, "Amira", &characters),
];
```

Diese Runner Objekte dürfen dann nacheinander Springen.

```
loop {
    for runner in &mut runners {
        // hop the winner will exit the program and print his name
        runner.hop();
    }
}
```

Der Sprung funktioniert wie folgt: Erst wird die aktuelle Position überprüft in dem der aktuelle Character aus dem Array gelesen wird. Ist der Index zu groß wird das Programm beendet und der Spielername in die Konsole geschrieben.

```rust
pub fn hop(&mut self) {
    let new_pos = match &self.characters.get(self.pos) {
        Some(c) => calc_jump_width(c),
        None => {
            print!("The Winner is: {}", &self.player_name);
            exit(0);
        }
    };

    self.pos += new_pos;
}
```

Ist der Index passend wird eine neue Position berrechet in dem die ascii Zahl - 96 machen ist die Zahl ein Umlaut kann sie nicht berrechnet werden und wird einfach durch ein match ausgegeben.

```rust
pub fn calc_jump_width(c: &char) -> usize {  2 usages
    // umlauts are not in the ascii table so we have to handle them separately
    match c {
        'ä' => 27,
        'ö' => 28,
        'ü' => 29,
        'ß' => 30,
        _=> c.to_ascii_lowercase() as usize - 96
    }
}
```

Dann wird zu der alten Position die neue hinzugefügt und der Zyklus beginnt erneut.

Beispiele

1.

Text: „AbC“

1. Alles wird kleingeschrieben und analphabetische Zeichen entfernt neuer Text: „abc“
2. Runner 1:
    - Position 1
    - Buchstabe „a“ ergibt Sprungweite 1
    - Sprung an Position 1 + 1 = 2

3. Runner 2:
    - Position 2
    - Buchstabe „b“ ergibt Sprungweite 2
    - Sprung an Position 2 + 2 = 4
    -
4. Runner 1:
    - Position 2
    - Buchstabe „b“ ergibt Sprungweite 2
    - Sprung an Position 2 + 2 = 4
5. Runner 2:
    - Position 4
    - Text enthält keinen Buchstaben an der Position 4, deswegen wird das Programm beendet und schreibt den Gewinnernamen in die Konsole

2.

Text: „%d@G&Ge.zBhtß“

1. Alles wird kleingeschrieben und analphabetische Zeichen entfernt neuer Text: „dggezbhtß“
2. Runner 1:
    - Position 1
    - Buchstabe „d“ ergibt Sprungweite 4
    - Sprung an Position 1 + 4 = 5
3. Runner 2:
    - Position 2
    - Buchstabe „g“ ergibt Sprungweite 7
    - Sprung an Position 2 + 7 = 9
4. Runner 1:
    - Position 5
    - Buchstabe „z“ ergibt Sprungweite 26
    - Sprung an Position 5 + 26 = 31
5. Runner 2:
    - Position 9
    - Buchstabe „ß“ ergibt Sprungweite 30
    - Sprung an Position 9 + 30 = 39
6. Runner 1:
    - Position 31
    - Text enthält keinen Buchstaben an der Position 31, deswegen wird das Programm beendet und schreibt den Gewinnernamen in die Konsole

3. „hopsen1.txt“

   Bela gewinnt.

4. „hopsen2.txt“

   Bela gewinnt.

5. „hopsen3.txt“

   Bela gewinnt.

6. „hopsen4.txt“

   Amira gewinnt.

7. „hopsen5.txt“

   Bela gewinnt.

Quellcode

Siehe Umsetzung.