Rust@Wellcome

Mission

Create a community of Rust developers on campus for learning, teaching, and discussing the Rust programming language and its use in bioinformatics and anywhere else

Conduct

Privacy

Code of Conduct

Be wonderful to each other, contribute, enjoy yourself

Coming up

Thursday	Jan 19	C3-03	16:30
Thursday	Feb 2	C3-02	16:30
Thursday	Feb 16	C3-02	16:30

From Feb 2, we will meet in-person/virtually every 2 weeks

Speaker	About	Activity
Kat Figueroa	First Rust project	Lightning Talk
Will Burden	Solving adventofcode problem w/ Rust	Lightning Talk

perfect first project

Kat

Problem

Program to run checks on large FASTQ files

Checks

- Read headers end with expected “/1” or “/2”
- Read headers are the same for both FASTQs
- If reads in the R1 FASTQ are the same as the R2 FASTQ

```
@SRR12281346.1 1/1
GTGCAGGAATTTGCTAGCAGTAAGCGACCGATAATGTGTTTTNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNTCATTTTTCTTTCAAATACGCACACGGAAAACATTAGGAAATGAG
CGTATT
+
GGGGGIIIIIIIIIIIIIIIIIIIIIIIIGGIIIIIG<A.<.###################
############################################################
############################################################
################..<......7<..<<AGA77.GAAGGAAAGAAGAGGGGIGGGGGI
GGG<7G
@SRR12281346.2 2/1
TGTTTTTCGTAGATGCGTTGGGGATCGGTCTGGCAAATACGCACACGGAAAGCATTAGGAA
ATGGGCATATTTTCTCGTGGTATGCTGGCATAAATCGGTGCAGGAATTTGCTAGCATACCA
CGAGAAAATACGCCCATTTCC
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
IIIIIIIGIIIIIIIIIIIIIIIIIIIIIIIIIIIIIGIIIIGIIIIIIIIIIIIIIIIIII
IIIIIIIIIIIIIGIIGIIGI
```

```
@SRR12281346.1 1/2
GTGCAGGAATTTGCTAGCAGTAAGCGACCGATAATGTGTTTTNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNTCATTTTTCTTTCAAATACGCACACGGAAAACATTAGGAAATGAG
CGTATT
+
GGGGGIIIIIIIIIIIIIIIIIIIIIIIIGGIIIIIG<A.<.###################
############################################################
############################################################
################..<......7<..<<AGA77.GAAGGAAAGAAGAGGGGIGGGGGI
GGG<7G
@SRR12281346.2 2/2
TGTTTTTCGTAGATGCGTTGGGGATCGGTCTGGCAAATACGCACACGGAAAGCATTAGGAA
ATGGGCATATTTTCTCGTGGTATGCTGGCATAAATCGGTGCAGGAATTTGCTAGCATACCA
CGAGAAAATACGCCCATTTCC
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
IIIIIIIGIIIIIIIIIIIIIIIIIIIIIIIIIIIIIGIIIIGIIIIIIIIIIIIIIIIIII
IIIIIIIIIIIIIGIIGIIGI
```

Solution

Forked someone else’s work

- Bio-streams github repo

Built off of an example provided on the github repo

Added the checks I wanted

- Asked for help and practiced

Getting started

Compiling local packages

```
cargo build --example fqcheck --release
```

Run the program

```
target/release/examples/fqcheck ../ReadsBySample/test_SRR_1.fastq.gz ../ReadsBySample/test_SRR_2.fastq.gz
```

Highlights

Code is concise and easy to read

Clap crate allows for command line arg parsing

- If using incorrect args the script fails and the errors explain the problems
- All this based off of the *Cli* struct

```
use clap::Parser;

#[derive(Parser)]
struct Cli {
    r1: PathBuf,
    r2: PathBuf,
}
```

Self-contained executable

- When I ran **cargo build** the executable was created and that same executable I can use on any other machine running the same operating system I used to create it

Solving AoC Day 1 with Iterators!

Problem Statement

Given a list of numbers, grouped by empty lines:

src/input.txt

```
1000
2000
3000

4000

5000
6000

7000
8000
9000

10000
```

1) Find the sum of each group.
2) Find the three largest of those sums.
3) Sum them.

← answer:

10000 + 11000 + 24000 = 45000

The Iterator Trait

```rust
pub trait Iterator {
  type Item;

  fn next(&mut self) -> Option<Self::Item>;
}
```

```rust
let _: Vec<u32> = (0..) // RangeFrom<u32> (implements Iterator)
    .map(|x| x ^ 2)     // Map<RangeFrom<u32>, |u32| -> u32>
    .take(10)           // Take<Map<RangeFrom<u32>, |u32| -> u32>>
    .collect();         // Vec<u32> (inferred, implements FromIterator)
```

Summing a list of numbers

src/main.rs

```rust
fn main() {
    let input = include_str!("input.txt");
    let mut sum = 0;
    // str::lines returns an iterator!
    for (i, line) in input.lines().enumerate() {
        sum += line.parse::<u32>()
            .unwrap_or_else(|err| panic!("failed to parse line {i}: {err}"));
    }


    println!("Sum: {sum}");
}
```

Summing a list of numbers

src/main.rs

```rust
fn main() {
    let input = include_str!("input.txt");

    // str::lines returns an iterator!
    let sum: u32 = input
        .lines()
        .enumerate()
        .map(|(i, line)| line.parse::<u32>()
            .unwrap_or_else(|err| panic!("failed to parse line {i}: {err}")))
        .sum();
    println!("Sum: {sum}");
}
```

Solution

src/main.rs

```rust
fn main() {
    let mut lines = include_str!("input.txt").lines().enumerate().peekable();
    let mut sums = Vec::new();

    // Stop when there are no more lines
    while lines.peek().is_some() {
        let mut sum = 0;
        // Iterate through the lines until we reach an empty line.
        for (i, line) in &mut lines {
            if line.is_empty() {
                break;
            }
            sum += line.parse::<u32>()
                .unwrap_or_else(|err| panic!("failed to parse line {i}: {err}"));
        }
        sums.push(sum);
    }

    // Sort in ascending order.
    sums.sort_unstable();

    // Sum the last (i.e. top) three.
    let total: u32 = sums.into_iter().rev().take(3).sum();
    println!("Total of top three: {total}");
}
```

It works!

```
➜ cargo run
   Compiling day1 v0.1.0 (/home/will/advent-of-code/day1)
    Finished dev [unoptimized + debuginfo] target(s) in 0.44s
     Running `target/debug/day1`
Total of top three: 45000
```

Solution

src/main.rs

```rust
fn main() {
    let mut lines = include_str!("input.txt").lines().enumerate().peekable();
    let mut sums = Vec::new();

    // Stop when there are no more lines
    while lines.peek().is_some() {
        let mut sum = 0;
        // Iterate through the lines until we reach an empty line.
        for (i, line) in &mut lines {
            if line.is_empty() {
                break;
            }
            sum += line.parse::<u32>()
                .unwrap_or_else(|err| panic!("failed to parse line {i}: {err}"));
        }
        sums.push(sum);
    }

    // Sort in ascending order.
    sums.sort_unstable();

    // Sum the last (i.e. top) three.
    let total: u32 = sums.into_iter().rev().take(3).sum();
    println!("Total of top three: {total}");
}
```

src/main.rs

```rust
fn main() {
    let mut lines = include_str!("input.txt").lines().enumerate().peekable();
    let mut sums = Vec::new();

    // Stop when there are no more lines
    while lines.peek().is_some() {
        let mut sum = 0;
        // Iterate through the lines until we reach an empty line.
        for (i, line) in lines.by_ref().take_while(|(_, line)| !line.is_empty()) {
            sum += line.parse::<u32>()
                .unwrap_or_else(|err| panic!("failed to parse line {i}: {err}"));
        }
        sums.push(sum);
    }

    // Sort in ascending order.
    sums.sort_unstable();

    // Sum the last (i.e. top) three.
    let total: u32 = sums.into_iter().rev().take(3).sum();
    println!("Total of top three: {total}");
}
```

Solution

src/main.rs

```rust
fn main() {
    let mut lines = include_str!("input.txt").lines().enumerate().peekable();
    let mut sums = Vec::new();

    // Stop when there are no more lines
    while lines.peek().is_some() {
        let mut sum = 0;
        // Iterate through the lines until we reach an empty line.
        for (i, line) in lines.by_ref().take_while(|(_, line)| !line.is_empty()) {
            sum += line.parse::<u32>()
                .unwrap_or_else(|err| panic!("failed to parse line {i}: {err}"));
        }
        sums.push(sum);
    }

    // Sort in ascending order.
    sums.sort_unstable();

    // Sum the last (i.e. top) three.
    let total: u32 = sums.into_iter().rev().take(3).sum();
    println!("Total of top three: {total}");
}
```

src/main.rs

```rust
fn main() {
    let mut lines = include_str!("input.txt").lines().enumerate().peekable();
    let mut sums = Vec::new();

    // Stop when there are no more lines
    while lines.peek().is_some() {
        let sum: u32 = lines
            .by_ref()
            .take_while(|(_, line)| !line.is_empty())
            .map(|(i, line)| line.parse::<u32>()
                .unwrap_or_else(|err| panic!("failed to parse line {i}: {err}")))
            .sum();
        sums.push(sum);
    }

    // Sort in ascending order.
    sums.sort_unstable();

    // Sum the last (i.e. top) three.
    let total: u32 = sums.into_iter().rev().take(3).sum();
    println!("Total of top three: {total}");
}
```

Solution

src/main.rs

```rust
fn main() {
    let mut lines = include_str!("input.txt").lines().enumerate().peekable();
    let mut sums = Vec::new();

    // Stop when there are no more lines
    while lines.peek().is_some() {
        let sum: u32 = lines
            .by_ref()
            .take_while(|(_, line)| !line.is_empty())
            .map(|(i, line)| line.parse::<u32>()
                .unwrap_or_else(|err| panic!("failed to parse line {i}: {err}")))
            .sum();
        sums.push(sum);
    }

    // Sort in ascending order.
    sums.sort_unstable();

    // Sum the last (i.e. top) three.
    let total: u32 = sums.into_iter().rev().take(3).sum();
    println!("Total of top three: {total}");
}
```

src/main.rs

```rust
fn main() {
    let mut lines = include_str!("input.txt").lines().enumerate();

    // Get the sum of each group of numbers.
    let mut sums: Vec<u32> = std::iter::from_fn(|| {
        lines
            .by_ref()
            .take_while(|(_, line)| !line.is_empty())
            .map(|(i, line)| line.parse::<u32>()
                .unwrap_or_else(|err| panic!("failed to parse line {i}: {err}")))
            .reduce(|sum, num| sum + num)
    })
    .collect();

    // Sort in ascending order.
    sums.sort_unstable();

    // Sum the last (i.e. top) three.
    let total: u32 = sums.into_iter().rev().take(3).sum();
    println!("Total of top three: {total}");
}
```

Going further: itertools

```rust
use std::cmp::Reverse;
use itertools::Itertools;

fn main() {
    let total: u32 = include_str!("input.txt")
        .lines()
        .enumerate()
        .group_by(|(_, line)| line.is_empty())
        .into_iter()
        .filter_map(|(is_empty, group)| Some(group).filter(|_| !is_empty))
        .map(|group| group
            .map(|(i, line)| line.parse::<u32>()
                .unwrap_or_else(|err| panic!("failed to parse line {i}: {err}")))
            .sum::<u32>())
        .sorted_unstable_by_key(|&sum| Reverse(sum))
        .take(3)
        .sum();
    println!("Total of top three: {total}");
}
```

It still works!

```
➜ cargo run
   Compiling either v1.8.0
   Compiling itertools v0.10.5
   Compiling day1 v0.1.0 (/home/will/advent-of-code/day1)
    Finished dev [unoptimized + debuginfo] target(s) in 1.63s
     Running `target/debug/day1`
Total of top three: 45000
```

Connect with us

Mailing list

https://wsi-lists.sanger.ac.uk/postorius/lists/rust-lang.sanger.ac.uk/

GitHub

https://github.com/Rust-Wellcome

Slack

mirroring Slack channel soon